Denken im Verborgenen — Reasoning im Latent Space

Kapitel online

Was passiert, wenn ein Modell seine Denkschritte gar nicht mehr in Worte fasst? Coconut, der Flaschenhals „ein Wort“ — und die ehrliche Debatte um Effizienz gegen Nachvollziehbarkeit.

Aktuell · Kapitel 36 von 50 · ca. 25 Min. · 300 XP

Einfach gesagt: Statt jeden Gedanken in ein Wort zu pressen, bleibt er hier ein Zahlenpunkt — schneller, aber du kannst nicht mehr mitlesen.

Im letzten Kapitel hast du gesehen, wie ein Modell „denkt“: Es schreibt seine Zwischenschritte als ganz normale Tokens auf und liest sie wieder ein. Denken ist Schreiben. Das hat einen wunderbaren Nebeneffekt — **wir können mitlesen.**

Jetzt die Frage, um die es heute geht: **Warum eigentlich in Wörtern?** Das Modell rechnet intern mit Vektoren. Es presst diese Vektoren mühsam in Sprache, nur um sie im nächsten Moment wieder in Vektoren zurückzuverwandeln. Ein bisschen so, als müsstest du beim Kopfrechnen jede Zwischenzahl laut aussprechen, aufschreiben und wieder ablesen — bevor du weiterrechnen darfst.

Was passiert, wenn man diesen Umweg einfach weglässt? Genau das haben Forscherinnen und Forscher Ende 2024 ausprobiert. Das Ergebnis ist elegant, effizient — und es macht ein Fenster zu, durch das wir bisher schauen konnten.

HerrKI-Tipp: Ein Satz für dieses Kapitel — **„Der Gedanke bleibt ein Vektor.“** Alles andere folgt daraus.

Was du hier lernst

- Was der **Flaschenhals „ein Wort“** ist — und wie viel Information dabei verloren geht
- Wie **Coconut** den Hidden-State direkt als nächsten Input zurückfüttert
- Warum ein einziger Vektor **mehrere Denkpfade gleichzeitig** tragen kann
- Warum das **weniger Tokens** braucht — und was es kostet
- Die ehrliche **Debatte**: Effizienz gegen Nachvollziehbarkeit

Für den Unterricht

Einstieg (5 Min): Kapitel gemeinsam lesen, Alltagsbeispiel aufgreifen.

Erkunden am Gerät (15 Min): die interaktiven Module auf

herrki.de/365_denken_im_latent_space ausprobieren.

Sichern mit Quiz (5 Min): „Kurzer Check" besprechen, „Das nimmst du mit" festhalten.

Woher das kommt: ein Paper von Ende 2024

„Training Large Language Models to Reason in a Continuous Latent Space“
Shibo Hao, Sainbayar Sukhbaatar u. a. — **Meta FAIR + UC San Diego**, Dezember 2024
arXiv:2412.06769 · Code: github.com/facebookresearch/coconut

Die Methode heißt **Coconut** — *Chain of Continuous Thought*, also „Kette kontinuierlicher Gedanken“ statt „Kette von Gedanken in Worten“.

⚠ **Bleiben wir ehrlich:** Das ist **aktive Forschung**, kein Produkt, das du heute im Browser anklickst. Es gibt erste experimentelle Modelle, die großen Labore erforschen die Richtung (2025/2026 ist daraus ein eigenes Forschungsfeld geworden, inklusive der Frage, ob man solche Modelle überhaupt noch interpretieren kann). **Niemand hier behauptet, dass ein bestimmtes kommerzielles Produkt so arbeitet.** Wir schauen uns eine Idee an, weil sie erklärt, wo die Reise hingehen könnte — und weil man an ihr wunderbar versteht, was ein Gedanke im Inneren eines Modells eigentlich ist.

Runter zum Tensor: der Flaschenhals „ein Wort“

Um zu verstehen, was Coconut weglässt, müssen wir uns anschauen, was beim *sichtbaren* Denken Schritt für Schritt passiert. Nimm einen einzigen Denkschritt aus Kapitel 360 unter die Lupe:

Hidden State

↓

`[0.41, −0.12, 0.87, …]`
[2048]

→

Wahrscheinlichkeiten

↓

`128 000 Werte`
Vokabular

→

EIN Token

↓

„Hafen“
1 Wort

→

Embedding

↓

`[0.33, 0.05, −0.61, …]`
[2048]

Mit Zahlen wird das Nadelöhr sichtbar. Wir verkleinern den Vektor zum Mitrechnen auf fünf Werte (in echt sind es 2048) und schauen uns die stärksten Kandidaten an:

```
// Ein Denkschritt, Schritt fuer Schritt
Hidden State [5]: 0.41 −0.12 0.87 0.33 −0.05 ← in echt 2048 Zahlen

// daraus die Wahrscheinlichkeiten ueber das Vokabular:
„Hafen“ 31.2 % ← Gewinner
„Ufer“ 28.7 % ← fast gleich stark … faellt trotzdem ersatzlos weg
„Kai“ 12.4 %
„Steg“ 9.8 %
„Bucht“ 7.1 % ← Rest: 10.8 % auf ~128.000 weitere Vokabeln

// Weitergereicht wird NUR das Wort „Hafen“ – nachgeschlagen als sein Embedding:
Embedding [5]: 0.33 0.05 −0.61 0.18 0.44 ← in echt 2048 Zahlen
```

Das „fast gleich stark“ zwischen *Hafen* und *Ufer* war im Hidden State noch vorhanden — nach dem Decodieren ist es weg. Genau diese Information rechnen wir gleich in Bits nach.

Merkst du, was da passiert? Der Zustand geht als **Vektor** rein und kommt als **Vektor** wieder raus — dazwischen wird er durch ein Nadelöhr gequetscht, das genau *ein einziges Wort* breit ist.

Wie eng ist dieses Nadelöhr? Rechnen wir grob nach.

- Ein Hidden-State mit 2048 Zahlen (16 Bit je Zahl) trägt **rund 32 000 Bit** an Zustand.
- Die Auswahl *eines* Tokens aus 128 000 möglichen entspricht $\log_2(128\,000) \approx$ **17 Bit**.

Das ist keine saubere informationstheoretische Rechnung — aber die Größenordnung stimmt und sie ist brutal: Beim Schritt „Vektor → Wort“ bleibt nur ein **winziger Bruchteil** des inneren Zustands übrig. Alles Feine, alles Unentschiedene, alles „vielleicht doch eher“ wird auf ein einziges hartes Wort gerundet. **Sprache ist für ein Modell eine Kompression — und zwar eine verlustbehaftete.**

Coconuts Idee: das Decodieren einfach weglassen

Und jetzt der ganze Trick, in einem Satz: **Der Hidden-State-Vektor wird direkt als nächster Input-Embedding-Tensor zurückgefüttert.** Kein Vokabular, kein Wort, keine Rundung.

Sichtbares Denken (Chain of Thought)

Hidden State *[2048]* ↓ decodieren Token „Hafen“ ≈ *17 Bit* ↓ nachschlagen Embedding *[2048]* ↓ nächster Durchlauf

Coconut (Chain of Continuous Thought)

Hidden State *[2048]* ↓ direkt weiterreichen nächster Durchlauf — kein Wort dazwischen —

Damit wandert das Modell nicht mehr von Wort zu Wort, sondern von **Punkt zu Punkt** — in genau dem Raum, den du in Kapitel 310 als Landkarte kennengelernt hast. Der Gedanke bleibt eine Koordinate. Und Koordinaten dürfen zwischen den Wörtern liegen.

> **Das muss man antrainieren.** Ein normales Modell kann das nicht einfach so. Coconut wird stufenweise umgewöhnt: Zuerst lernt es Aufgaben mit ganz normalen Sprach-Denkschritten. Dann wird der erste Denkschritt durch einen kontinuierlichen Vektor ersetzt, dann der zweite, dann der dritte — bis die Denkphase komplett ohne Worte auskommt. Ein Lehrplan, kein Schalter.

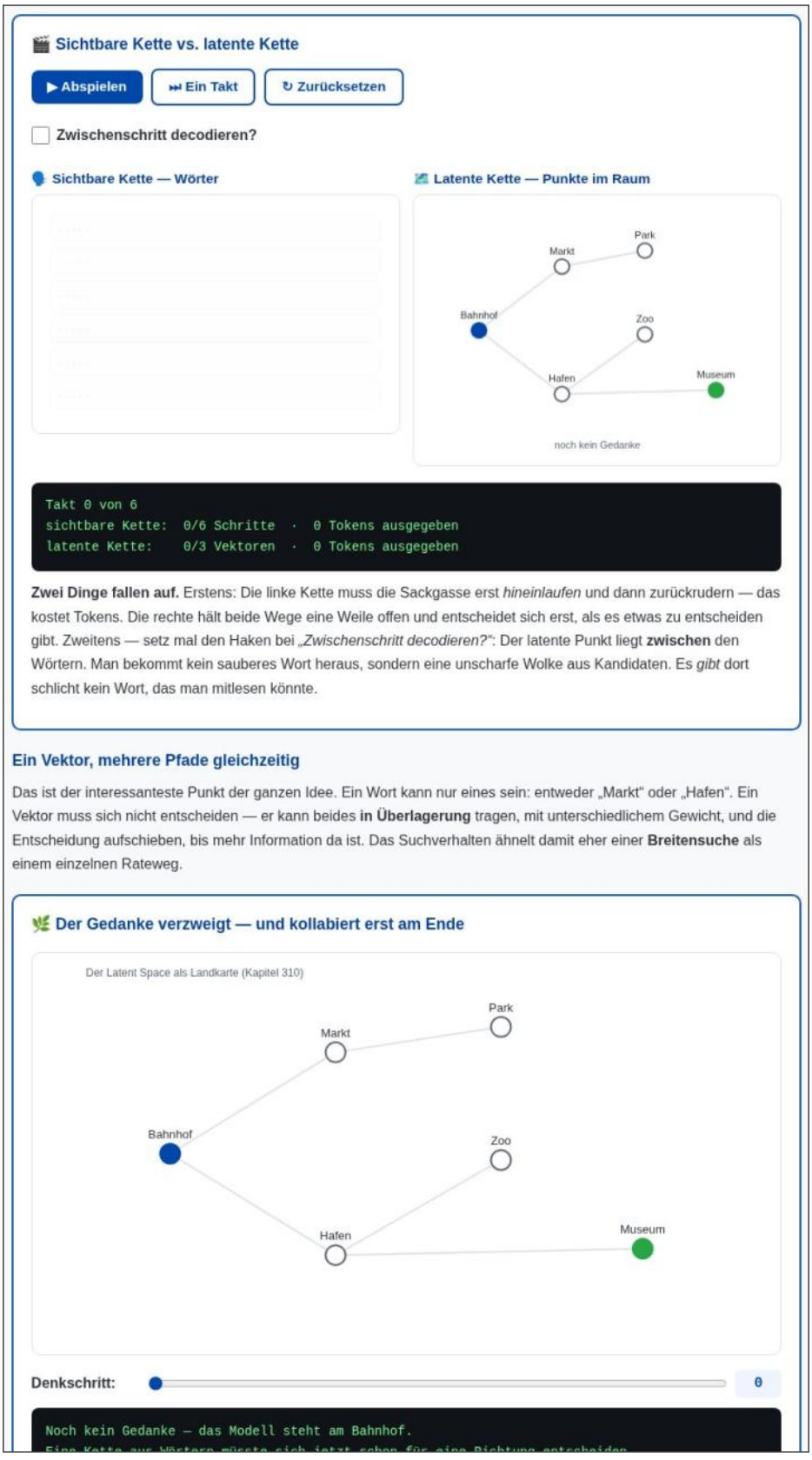
Sehen, was der Unterschied ist

Beide Seiten bekommen dieselbe Aufgabe — eine kleine Wegsuche, bei der man abwägen und Sackgassen verwerfen muss. Genau die Sorte Aufgabe, bei der sich die beiden Denkarten am stärksten unterscheiden:

> **Aufgabe:** Du stehst am **Bahnhof**. Wie kommst du zum **Museum**?
>
> - Vom **Bahnhof** geht es zum **Markt** oder zum **Hafen**.
> - Vom **Markt** geht es nur zum **Park** (Sackgasse).
> - Vom **Hafen** geht es zum **Zoo** oder zum **Museum**.

Am Gerät ausprobieren: herrki.de/365_denken_im_latent_space#abschnitt-4

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

Ein Vektor, mehrere Pfade gleichzeitig

Das ist der interessanteste Punkt der ganzen Idee. Ein Wort kann nur eines sein: entweder „Markt“ oder „Hafen“. Ein Vektor muss sich nicht entscheiden — er kann beides **in Überlagerung** tragen, mit unterschiedlichem Gewicht, und die Entscheidung aufschieben, bis mehr Information da ist. Das Suchverhalten ähnelt damit eher einer **Breitensuche** als einem einzelnen Rateweg.

Was das Paper misst — und was nicht

Ehrlich und ohne Marketing: Coconut ist kein Wundermittel, das alles besser macht. Das Paper berichtet vor allem zwei Dinge. Erstens verbessern sich **Aufgaben, bei denen man abwägen und Sackgassen verwerfen muss** (Logik- und Planungsaufgaben) — genau da, wo das Offenhalten mehrerer Möglichkeiten hilft. Zweitens braucht die latente Denkphase **deutlich weniger Schritte** als die ausgeschriebene. Die konkreten Messwerte findest du im Paper selbst; sie hier nachzuerzählen wäre Zahlenraten.

Was man ebenso ehrlich sagen muss: Bei manchen Aufgabentypen bringt das Verfahren keinen Vorteil, das Training ist aufwendiger als normales Chain-of-Thought-Training, und die Ergebnisse stammen aus Forschungs-Setups, nicht aus einem Produkt mit Millionen Nutzern.

Die Debatte — und die ist wirklich offen

Jetzt kommt der Teil, bei dem ich dich nicht mit einer Meinung abspeisen will, weil es in der Fachwelt selbst keine Einigkeit gibt. Beide Seiten haben starke Argumente:

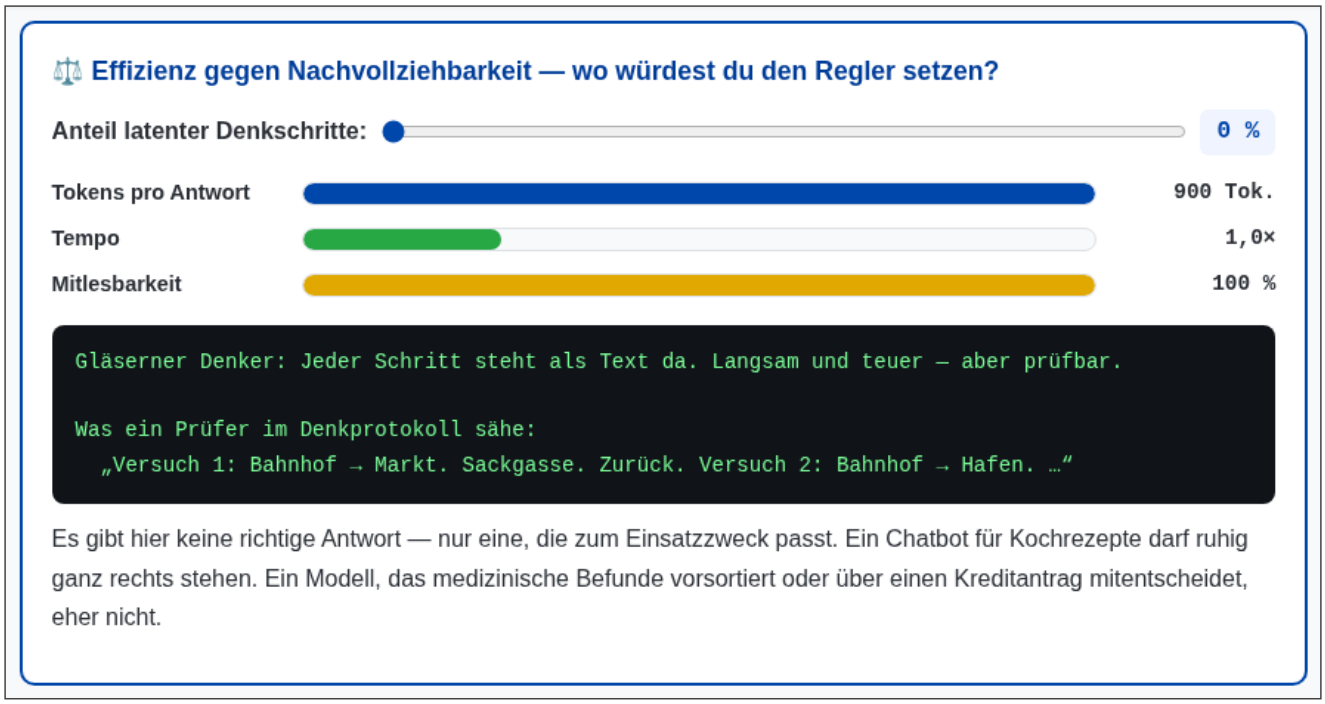
Dafür	Dagegen
Kein Informationsverlust am Nadelöhr „ein Wort“ — der ganze Zustand wird weitergereicht.	Wir können nicht mehr mitlesen. Der zentrale Kontrollmechanismus der letzten Jahre fällt weg.
Weniger Denkschritte, also weniger Tokens: schneller und billiger.	Fehlersuche wird zur Blackbox-Arbeit — kein Zwischenschritt, den man anzweifeln könnte.
Mehrere Möglichkeiten bleiben gleichzeitig offen (breitensuche-artig) statt früh zu raten.	Auditing und Regulierung setzen bislang genau auf nachvollziehbare Zwischenschritte.
Sprache ist eine Krücke fürs Denken — Menschen denken auch nicht ausschließlich in Sätzen.	Ein Modell könnte in den Vektoren Dinge „mitdenken“, die es in Worten nie sagen würde.

⚠ **Warum das mehr als Technikgeschmack ist.** Die sichtbare Denkkette war bisher eines der wenigen brauchbaren **Kontrollfenster** in ein Sprachmodell hinein: Man konnte einem Modell beim Planen zuschauen und im Zweifel eingreifen — bei riskanten Aufgaben genau der Punkt, an dem Sicherheitsforschung ansetzt. Wenn die Denkphase in Vektoren wandert, ist dieses Fenster zu. Genau deshalb ist die **Interpretierbarkeit latenter Denkketten** selbst zu einem aktiven Forschungsthema geworden — siehe etwa „Are Latent Reasoning Models Easily Interpretable?“ (arXiv:2604.04902). Die Frage ist nicht „ob man da noch reinschauen kann“, sondern „mit welchen Werkzeugen und wie zuverlässig“.

Und zur Fairness gehört auch das Gegenargument aus Kapitel 360: Eine sichtbare Denkkette ist ohnehin **kein Beweis**. Modelle formulieren Begründungen, die nicht die echte Ursache ihrer Antwort sind. Wir haben also nie ein perfektes Fenster gehabt — nur ein trübes. Die ehrliche Frage lautet deshalb: Ist ein trübes Fenster besser als gar keines? (Meine Vermutung: ja. Aber das ist eine Vermutung, kein Messwert.)

Am Gerät ausprobieren: herrki.de/365_denken_im_latent_space#abschnitt-6

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

Was heißt das für dich?

- **Kurzfristig: nichts.** Die Modelle, die du heute benutzt, denken in Tokens. Wenn dir ein Anbieter eine Denkkette zeigt, ist es echter Text — kein latenter Vektor.
- **Beim Lesen von Ankündigungen:** Achte auf die Formulierung. „Weniger Denk-Tokens bei gleicher Qualität“ ist genau die Richtung, aus der dieses Thema kommt.
- **Wenn dir Nachvollziehbarkeit wichtig ist:** Frag danach. Ob ein Modell seinen Denkweg offenlegt, ist eine Produktentscheidung — und Produktentscheidungen folgen der Nachfrage.
- **Fürs eigene Verständnis:** Du weißt jetzt, dass „Gedanke“ und „Wort“ im Modell zwei verschiedene Dinge sind. Das ist der Punkt, an dem KI aufhört, Magie zu sein — und anfängt, Ingenieursarbeit mit klaren Zielkonflikten zu sein.

Zum Weiterlesen

- **Das Paper:** Shibo Hao, Sainbayar Sukhbaatar u. a., *„Training Large Language Models to Reason in a Continuous Latent Space“* (Coconut), Meta FAIR + UC San Diego, Dezember 2024 — arxiv.org/abs/2412.06769 (PDF (arxiv.org/pdf/2412.06769))
- **Der Code:** github.com/facebookresearch/coconut
- **Interpretierbarkeit latenter Denkketten:** *„Are Latent Reasoning Models Easily Interpretable?“* — arxiv.org/abs/2604.04902
- **Auf HerrKI:** 310 — Latent Space (die Landkarte) und 360 — Erst denken, dann reden (das sichtbare Denken)

Kurzer Check – mit Lösung

Lösungen für Lehrkräfte – für Lernende die Online-Version nutzen.

Frage 1: Was genau lässt Coconut gegenüber normalem Chain-of-Thought weg?

- Die Transformer-Schichten — es rechnet mit einem kleineren Netz.
- **✓ Das Decodieren zum Token: Der Hidden-State wird direkt als nächstes Input-Embedding zurückgefüttert.**
- Das Training — die Fähigkeit entsteht ganz von allein beim Prompten.

Genau. Der Umweg „Vektor → Wort → Vektor“ entfällt. Und nein, das fällt nicht vom Himmel: Coconut wird stufenweise darauf trainiert, indem Sprach-Denkschritte nach und nach durch kontinuierliche Vektoren ersetzt werden.

Frage 2: Warum kann ein latenter Gedanke mehrere Denkpfade gleichzeitig tragen?

- Weil das Modell im Hintergrund mehrere Kopien von sich selbst startet.
- Weil der Latent Space unendlich viele Wörter enthält.
- **✓ Weil ein Vektor eine Überlagerung mit unterschiedlichen Gewichten sein darf — ein Wort dagegen muss sich für eine Bedeutung entscheiden.**

Richtig. Ein Punkt zwischen „Markt“ und „Hafen“ ist eine völlig legitime Koordinate. Das Wort „Markt“ hingegen ist eine Festlegung — und eine falsche Festlegung muss später mit weiteren Tokens zurückgenommen werden.

Frage 3: Was ist der wichtigste Einwand gegen latentes Reasoning?

- Es ist langsamer und teurer als sichtbares Denken.
- **✓ Die sichtbare Denkkette war ein Kontrollfenster — ohne sie werden Prüfung, Fehlersuche und Sicherheitsbewertung deutlich schwerer.**
- Es funktioniert grundsätzlich nur bei Rechenaufgaben.

Genau darum dreht sich die Debatte. Nebenbei: Das Fenster war noch nie perfekt — eine ausgeschriebene Begründung ist nicht automatisch die echte Ursache der Antwort. Aber ein trübes Fenster ist immer noch mehr als eine Wand.

Das nimmst du mit

- **Sprache ist ein Flaschenhals.** Aus tausenden Zahlen wird ein einziges Wort — der Rest fällt weg.
- **Coconut lässt das Decodieren weg.** Der Hidden-State wird direkt zum nächsten Input-Embedding.
- **Der Gedanke bleibt ein Punkt.** Und Punkte dürfen zwischen den Wörtern liegen — mehrere Wege gleichzeitig.
- **Weniger Tokens, mehr Blackbox.** Was an Effizienz gewonnen wird, geht an Mitlesbarkeit verloren.
- **Das ist Forschung, kein Produkt.** Spannend, offen, umstritten — genau richtig für die Kategorie „Aktuell“.

Und in „Prompt, Kontext oder Training?“ siehst du als Nächstes die andere Seite davon: nicht *wie* ein Modell denkt, sondern womit du sein Verhalten überhaupt beeinflusst — mit dem Prompt, mit mitgeliefertem Wissen oder mit Training.

So zitierst du dieses Kapitel

HerrKI: Denken im Verborgenen — Reasoning im Latent Space. herrki.de/365_denken_im_latent_space, ein privates Hobbyprojekt von Terragon.de, abgerufen am 09.09.2026.