EIN BUCH ÜBER EPISODISCHE INTELLIGENZ

Der Körper der KI

Strom, Kontrolle und die Rückkehr lokaler Intelligenz

Warum intelligente Systeme nicht permanent zentral brennen müssen.

MIT 12 ARCHITEKTURDIAGRAMMEN · GLOSSAR · QUELLENAPPARAT

Joachim Müller-Peter

Erstausgabe · Juli 2026 · © Joachim Müller-Peter

DER KÖRPER DER KI

Strom, Kontrolle und die Rückkehr lokaler Intelligenz

Joachim Müller-Peter

Erstausgabe · Juli 2026

Über dieses Buch

Dieses Buch untersucht KI nicht nur als Modell, sondern als Infrastruktur: als Strom, Wärme, Chips, Netze, Speicher, Runtime, Governance, Geschäftsmacht und lokale Verantwortung.

Die Leitidee lautet: KI ist nicht körperlos. Sie besitzt einen physischen, energetischen, organisatorischen und politischen Körper. Wer diesen Körper versteht, kann bessere Entscheidungen darüber treffen, welche Teile intelligenter Systeme lokal, regional, verteilt oder in Frontier-Rechenzentren stattfinden sollten.

Die zentrale Betriebsform nennt dieses Buch episodische Intelligenz: lokale semantische Systeme bereiten Bedeutung vor, deterministische Schichten begrenzen und prüfen, energieadaptive Scheduler verschieben nicht-dringliche Arbeit, und Frontier-Modelle werden gezielt gerufen, wenn Tiefe, Neuartigkeit oder Risiko es verlangen.

Was dieses Buch nicht behauptet

Dieses Buch ist keine Anti-Cloud-Schrift. Es behauptet nicht, dass Frontier-Rechenzentren verschwinden werden. Es behauptet auch nicht, dass lokale Systeme automatisch sicherer, sauberer oder günstiger sind.

Die These ist schmaler und stärker: Nicht jede intelligente Aufgabe verdient automatisch die teuerste, zentralste und energieintensivste Ausführungsform. Gute KI-Architektur beginnt mit der Frage nach Ort, Zeit, Risiko, Datenoffenlegung und notwendiger Denktiefe.

Damit bleibt das Buch skeptisch gegenüber zwei einfachen Erzählungen: der totalen Zentralisierung und der romantischen Dezentralisierung. Zwischen beiden liegt der eigentliche Bauplatz.

Impressum und Verwendungshinweis

Verfasser: Joachim Müller-Peter

Erstausgabe, Juli 2026 · Eigenveröffentlichung · https://next.itbois.ch/

Dieses Buch enthält technische, wirtschaftliche, regulatorische und energiewirtschaftliche Einschätzungen. Es ersetzt keine Rechtsberatung, Sicherheitsprüfung, Energieplanung oder technisches Audit.

Bei Recherche, Strukturierung und Redaktion wurden KI-gestützte Werkzeuge eingesetzt. Auswahl, Einordnung und Gesamtverantwortung für den veröffentlichten Text liegen beim Verfasser.

Inhaltsverzeichnis

Die Kapitelüberschriften wurden in dieser Fassung lesbarer gemacht. Die technischen Begriffe bleiben im Text erhalten; die Überschriften führen stärker über Bilder und Fragen.

Teil I - Das Missverständnis

Wir beginnen dort, wo KI am leichtesten falsch verstanden wird: auf der Oberfläche. Der Text erscheint, aber der Körper darunter bleibt unsichtbar. Teil I öffnet diese Oberfläche und zeigt, dass Intelligenz im Betrieb nicht aus Magie entsteht, sondern aus Strom, Kontext, Routing und Infrastruktur.

Kapitel 1 - Die Illusion der permanenten Superintelligenz

Warum KI nicht als körperlose Dauerintelligenz missverstanden werden darf

Eine KI-Antwort sieht leicht aus. Ein Cursor blinkt. Ein Satz erscheint. Dann noch einer. Für den Nutzer wirkt es, als käme Sprache aus einem unsichtbaren Raum. Als würde Intelligenz einfach aus der Cloud fallen, sauber, sofort, körperlos.

Dieses Buch beginnt mit dem Verdacht, dass genau dieses Bild falsch ist. Nicht ein bisschen falsch, sondern grundlegend irreführend. KI ist nicht körperlos. Sie hat einen Körper aus Strom, Chips, Kühlung, Netzen, Speichern, Regeln, Verträgen und Machtverhältnissen. Sie erscheint als Text, aber sie entsteht als Infrastruktur.

Die erste große Erzählung der generativen KI war verführerisch einfach: Ein Mensch fragt, ein Modell denkt, eine Antwort erscheint. Diese Erzählung war nützlich. Sie machte eine komplexe Technologie begreifbar. Sie öffnete Millionen Menschen den Zugang zu einer neuen Art von Software. Aber als Architekturgeschichte reicht sie nicht mehr. Sie unterschlägt zu viel.

Sie unterschlägt den Strom. Sie unterschlägt die Wärme. Sie unterschlägt das Routing. Sie unterschlägt die Datenbewegung. Sie unterschlägt, dass viele produktive KI-Systeme längst nicht mehr nur aus einem Prompt und einem Modell bestehen. Sie unterschlägt auch die politische Frage, wer die Orte kontrolliert, an denen diese Intelligenz betrieben wird.

Leitsatz dieses Kapitels
Die wichtigste Frage ist nicht, wie oft wir ein großes Modell aufrufen können. Die wichtigste Frage ist, welche Teile eines intelligenten Systems wirklich zentral, sofort und mit maximalem Energieeinsatz ausgeführt werden müssen.

1.1 Der Text erscheint wie Magie

Der Bildschirm ist eine glatte Oberfläche. Er zeigt uns nicht, was hinter ihm arbeitet. Das ist eine alte Eigenschaft digitaler Systeme: Sie machen die Arbeit unsichtbar. Ein Klick fühlt sich leicht an, auch wenn er Rechenzentren, Datenbanken, Netzwerke und Verträge berührt. Generative KI treibt diese Unsichtbarkeit auf die Spitze.

Wenn ein Sprachmodell antwortet, sehen wir keine Ventilatoren. Wir sehen keine Transformatoren. Wir sehen keine Speichersysteme, keine Lastverteiler, keine Queues, keine Sicherheitsfilter, keine Retrieval-Pipelines. Wir sehen nur Sprache. Das ist schön. Und gefährlich.

Denn was leicht aussieht, wird leicht falsch organisiert. Wenn eine Anfrage sich anfühlt wie ein kurzer Satz, behandeln wir sie irgendwann auch wie einen kurzen Satz. Dabei ist sie in Wirklichkeit ein Betriebsereignis. Sie löst Klassifikation aus, Kontextaufbau, Modellwahl, Rechenoperationen, Datenbewegung, Sicherheitsprüfungen, manchmal Tool-Aufrufe, manchmal weitere Modellläufe und fast immer Protokollierung.

Das gilt nicht nur für große öffentliche Chatbots. Es gilt noch stärker für Unternehmen. Dort ist eine KI-Antwort selten nur eine Antwort. Sie ist Teil eines Prozesses. Sie berührt Rollen, Rechte, Dokumente, Kunden, Verträge, Fristen, Haftung und Entscheidungsketten. Wer das Modell als alleinige Maschine betrachtet, sieht nur die Flamme. Nicht den Ofen.

```
Populäre Erzählung:
User fragt
  -> KI denkt
  -> Antwort erscheint

Betriebliche Realität:
User fragt
  -> Anfrage wird klassifiziert
  -> Kontext wird gesucht und begrenzt
  -> Rechte und Risiken werden geprüft
```

```
  -> Modell oder Tool wird gewählt
  -> Antwort wird erzeugt
  -> Ergebnis wird validiert
  -> Entscheidung wird protokolliert
```

1.2 Die falsche Ein-Satz-Architektur

Viele KI-Debatten beginnen immer noch mit einem Bild, das ungefähr so aussieht: Prompt rein, Intelligenz raus. Dieses Bild ist verständlich, aber es ist die falsche Abstraktionsebene. Es beschreibt eine Demo, nicht einen Betrieb.

Eine Demo darf so tun, als sei das Modell die Anwendung. Ein produktives System darf das nicht. In einer echten Organisation muss das System wissen, wer fragt, worauf diese Person zugreifen darf, welche Daten relevant sind, welche Daten nicht weitergegeben werden dürfen, welche Ausgabeform zulässig ist, wann ein Tool ausgeführt werden darf und wann ein Mensch dazwischen muss.

Die erste Welle der generativen KI hat die Frage gestellt: Was kann ein Modell? Die zweite Welle muss fragen: Wo gehört das Modell hin? Das ist eine andere Frage. Sie ist weniger spektakulär, aber wichtiger. Denn die Antwort entscheidet darüber, ob KI billig oder teuer wird, vertrauenswürdig oder riskant, souverän oder abhängig, messbar oder mystisch.

Das Schlusskapitel wird diese neue Voreinstellung Episodische Intelligenz nennen. Dieses erste Kapitel nennt zunächst nur das Problem: Wir haben zu lange so getan, als sei maximale zentrale Intelligenz ein natürlicher Default. Als wäre es normal, jede kleine Unsicherheit an die teuerste verfügbare kognitive Instanz zu schicken.

Das ist bequem. Aber Bequemlichkeit ist kein Architekturprinzip.

Nicht die Frage
Wie groß kann das Modell werden?

Die Frage dieses Buches
Welche Aufgabe braucht wirklich welches Maß an Kontext, Energie, Risiko und Zentralisierung?

1.3 Der Körper der KI

Die Maschine denkt nicht im Äther. Sie denkt in Strom. Das ist kein poetischer Zusatz, sondern eine technische Tatsache. Rechnen ist ein physischer Vorgang. Jede Matrixoperation, jeder Speicherzugriff, jede Netzwerkbewegung braucht Energie. Am Ende wird ein großer Teil dieser Energie zu Wärme. Sprache erscheint auf dem Bildschirm; Wärme verschwindet in Kühlsystemen.

Die International Energy Agency schätzt, dass der weltweite Stromverbrauch von Rechenzentren von etwa 415 TWh im Jahr 2024 auf rund 945 TWh im Jahr 2030 wachsen könnte. AI ist dabei nicht der einzige, aber ein zentraler Treiber dieses Wachstums. Diese Zahlen sind keine Fußnote. Sie sind das Fundament der Architekturfrage.

Wenn KI nur Software wäre, könnte man die Kostenfrage als Preislistendetail behandeln. Wenn KI aber organisierte Elektrizität ist, dann wird die Frage tiefer: Welche Intelligenz lohnt sich wann? Welche Arbeit muss sofort passieren? Welche Arbeit darf warten? Welche Arbeit sollte lokal bleiben? Welche Arbeit muss an eine Frontier-Instanz eskalieren?

Kapitel 2 öffnet diesen Maschinenkörper genauer: Strom, Kühlung, GPU-Dichte, Netzwerk, PUE, Facility-Overhead. Hier reicht zunächst die Erkenntnis: Eine KI-Antwort ist kein Lichtstrahl ohne Schatten. Sie ist ein Ereignis in einer Energieinfrastruktur.

```
Prompt
  -> Rechenoperationen
  -> Strom
  -> Wärme
  -> Kühlung
```

```
  -> Kosten
  -> Infrastrukturbindung
```

1.4 Nicht jede Anfrage verdient den Hochofen

Ein Hochofen ist ein großartiges Werkzeug, wenn man Stahl herstellen will. Er ist ein absurdes Werkzeug, wenn man ein Streichholz anzünden möchte. Genau diese Verwechslung droht bei KI: Weil Frontier-Modelle beeindruckend sind, werden sie schnell zum Standardwerkzeug für alles.

Aber viele Aufgaben brauchen keine maximale Generalisierung. Sie brauchen saubere Datenstrukturen. Sie brauchen Regeln. Sie brauchen Speicher. Sie brauchen eine klare Klassifikation. Sie brauchen ein kleines Modell, ein Lookup, eine Datenbankabfrage, eine deterministische Prüfung oder eine gut gebaute Retrieval-Schicht.

Eine einfache Terminverschiebung braucht keine Frontier-Kognition. Eine formale Extraktion aus einem bekannten Dokumententyp braucht oft keinen riesigen Reasoner. Eine interne Anfrage kann manchmal lokal entschieden werden, wenn Rechte, Kontext und Ziel klar sind. Und manche Hintergrundarbeiten - Embeddings, Indexpflege, Evaluierungen, Zusammenfassungen - müssen nicht in dem Moment passieren, in dem ein Nutzer wartet.

Das bedeutet nicht, dass Frontier-Modelle unwichtig werden. Im Gegenteil: Gerade weil sie wertvoll sind, sollten sie nicht gedankenlos verwendet werden. Ein gutes System schützt seine teuerste Ressource, statt sie aus Bequemlichkeit immer zuerst zu verbrennen.

Episodische Intelligenz wird später genau diesen Gedanken ausarbeiten: Die Frontier bleibt. Aber sie wird nicht mehr als ständiger Default behandelt, sondern als bewusste Eskalationsinstanz.

Architektur statt Askese

Dieses Buch ist kein Plädoyer gegen große Modelle. Es ist ein Plädoyer gegen ihren gedankenlosen Einsatz als Voreinstellung.

1.5 Der verborgene Teil der Intelligenz

Wenn ein KI-System zuverlässig wird, liegt das oft nicht daran, dass das Modell plötzlich magisch fehlerfrei ist. Es liegt daran, dass das System um das Modell herum besser wird. Kontext wird begrenzt. Quellen werden ausgewählt. Tools werden kontrolliert. Ausgaben werden strukturiert. Ergebnisse werden geprüft. Unsicherheit wird sichtbar gemacht.

Dieser Teil ist weniger glamourös als das Modell. Er steht selten auf Konferenzfolien. Aber in produktiven Systemen ist er entscheidend. Routing, Retrieval, Caching, Policy-Gates, Tool-Gates, Contracts, Evaluationen und Audit-Logs sind nicht Beiwerk. Sie sind die Maschine, in der das Modell arbeitet.

Kapitel 3 leuchtet diesen verborgenen Maschinenraum aus. Schon hier ist wichtig: Ein System kann intelligenter wirken, indem es weniger generiert. Es kann sicherer werden, indem es nicht alles an das Modell gibt. Es kann günstiger werden, indem es wiederholbare Arbeit cached. Es kann vertrauenswürdiger werden, indem es Übergänge prüfbar macht.

Das ist kein Rückfall in alte Informatik. Es ist die Rückkehr ihrer besten Eigenschaften: Begrenzung, Nachvollziehbarkeit, klare Zustände, harte Schnittstellen, überprüfbare Verträge. Die neue KI braucht nicht weniger klassische Architektur. Sie braucht mehr davon.

```
Intelligenz im Betrieb entsteht aus:
  Modellfähigkeit
+ Kontextauswahl
+ Rechteprüfung
+ Routing
+ Retrieval
+ Caching
+ Validation
+ Auditierbarkeit
```

1.6 Der Client kehrt zurück

Die Cloud war lange der Ort, an den alles verschoben wurde. Daten, Anwendung, Speicher, Entscheidung. Der lokale Rechner wurde zum Fenster. Er zeigte an, was anderswo geschah. Bei KI beginnt sich dieses Bild zu verschieben.

Lokale Geräte bekommen wieder Bedeutung. Nicht, weil sie die Frontier ersetzen. Sondern weil sie Kontext besitzen. Der lokale Client kennt Dokumente, Nutzer, Sprache, Rollen, Verlauf, Eingabegeräte, vielleicht sogar Energiezustand und Netzwerkbedingungen. Er kann entscheiden, was überhaupt gesendet werden muss. Er kann vorfiltern, strukturieren, zusammenfassen, klassifizieren und speichern.

Apple beschreibt mit Private Cloud Compute einen Ansatz, bei dem lokale Geräte und speziell gesicherte Cloud-Instanzen zusammenspielen. Das ist nicht identisch mit episodischer Intelligenz, aber es ist ein deutliches Signal: Die Zukunft der KI wird nicht nur in der Frage Cloud oder lokal entschieden, sondern in der Frage, welche Schicht welche Verantwortung übernimmt.

Der Client wird damit nicht nur Anzeige. Er wird Runtime. Er wird Wächter. Er wird Kontextmaschine. Er wird die Stelle, an der echte Daten noch Namen haben, bevor sie für externe Modelle abstrahiert werden. Genau dort beginnt später das Kapitel über Semantic Compression.

Der neue Client
Der Client zeigt nicht nur an, was die Cloud denkt. Er entscheidet, ob die Cloud überhaupt denken soll.

1.7 Semantik, ohne alles preiszugeben

Unternehmen wollen die Fähigkeiten großer Modelle nutzen. Gleichzeitig wollen sie nicht jede interne Wirklichkeit nach außen tragen. Namen, Kunden, Verträge, Strategien, Margen, Risiken und historische Zusammenhänge sind nicht einfach Kontext. Sie sind oft das Geschäft selbst.

Die naive KI-Architektur sagt: Gib dem Modell mehr Kontext, dann wird die Antwort besser. Das stimmt manchmal. Aber es ist nicht immer der richtige Preis. Mehr Kontext kann mehr Kosten bedeuten, mehr Risiko, mehr Datenabfluss, mehr Abhängigkeit und mehr Oberfläche für Fehler.

Die Alternative heißt in diesem Buch Semantic Compression. Gemeint ist nicht nur Token-Kompression. Gemeint ist die kontrollierte Verdichtung von Bedeutung: Der lokale Layer kennt die Rohwirklichkeit und übersetzt sie in abstraktere Aufgaben, ohne alles preiszugeben. Das Modell sieht Struktur, Muster und Problemklasse - aber nicht zwingend Namen, Kundendaten oder Geschäftsgeheimnisse.

Das ist keine magische Anonymisierung. Es ist ein schwieriger Architekturprozess. Zu starke Abstraktion zerstört Bedeutung. Zu schwache Abstraktion schützt nicht genug. Aber genau zwischen diesen Polen entsteht eine der wichtigsten Fragen produktiver KI: Wie viel Wirklichkeit muss ein externes Modell wirklich sehen?

```
Rohkontext lokal:
  Kunde, Vertrag, Historie, interne Begriffe, Risiken

Abstrahierter Frontier-Request:
  Problemklasse, Randbedingungen, Ziel, Unsicherheit, erlaubte Ausgabe
```

1.8 Governance wird ausführbar

Organisationen reagieren auf neue Risiken oft mit Dokumenten. Policies, Leitlinien, Checklisten, Freigabeprozesse. Das ist verständlich. Aber bei KI reicht Governance als Papier nicht aus. Die Regeln müssen in die Runtime wandern.

NIST beschreibt KI-Risikomanagement unter anderem über die Funktionen Govern, Map, Measure und Manage. Solche Rahmenwerke sind wichtig, aber sie müssen in operative Systeme übersetzt werden. Eine Regel, die nicht zur Laufzeit geprüft wird, bleibt eine Absicht. Ein Gate, das tatsächlich blockiert, protokolliert oder eskaliert, ist Architektur.

Episodische Intelligenz denkt Governance deshalb nicht als nachträgliche Kontrolle. Es denkt sie als Schicht im System. Ein Tool-Aufruf braucht ein Gate. Ein Frontier-Call braucht einen Grund. Ein Ergebnis braucht eine Prüfung. Ein sensibler Kontext braucht einen Vertrag. Ein riskanter Übergang braucht vielleicht einen Menschen.

Das klingt streng. In Wahrheit ist es befreiend. Denn nur wenn Übergänge prüfbar sind, kann ein System wachsen, ohne zur Blackbox zu werden. Prompts sind wichtig, aber sie tragen das System nicht allein. Zuverlässigkeit entsteht hier durch kontrollierte Zustände.

Governance als Runtime
Eine Policy, die nur gelesen wird, schützt wenig. Eine Policy, die ausgeführt wird, verändert Architektur.

1.9 Das Ameisenmodell

Die moderne KI-Erzählung ist stark vom Bild des riesigen Rechenzentrums geprägt. Hallen voller GPUs, Stromanschlüsse wie Industrieanlagen, Netzwerke mit gewaltiger Bandbreite. Dieses Bild ist real. Kapitel 10 wird erklären, warum es nicht verschwinden wird.

Aber es ist nicht das einzige Bild. Neben dem Hochofen gibt es die Ameisen. Viele kleine, kontrollierte Knoten können Arbeit übernehmen, wenn diese Arbeit gut zerlegbar ist, wenn Latenz akzeptiert wird und wenn Ergebnisse prüfbar sind. Das gilt nicht für alles. Aber es gilt für mehr, als die Cloud-only-Erzählung nahelegt.

Vorverarbeitung, Klassifikation, lokale Indizes, Embeddings, Tests, Evaluationen, synthetische Datensätze, Dokumentpflege, kleine Spezialmodelle, LoRA-Adapter, Monitoring, Vergleichsläufe - all das muss nicht zwingend im gleichen Hochenergiezentrum stattfinden, in dem Frontier-Modelle trainiert werden.

Die Pointe ist nicht romantisch. Es geht nicht darum, zufällige Wohnzimmer-PCs mit sensiblen Aufgaben zu füttern. Es geht um kontrollierte, signierte, auditierbare, energieadaptive Edge-Infrastruktur. Um Rechenverbünde, die arbeiten, wenn Kontext, Energie und Risiko es erlauben.

```
Frontier bleibt:
  tiefe Generalisierung, schwierige Unsicherheit, große Modellfabrik

Edge-Verbund hilft:
  vorbereiten, prüfen, verdichten, indexieren, messen, lernen
```

1.10 Zeit ist eine Ressource

Viele digitale Systeme wurden auf Sofortigkeit trainiert. Alles soll in Sekunden passieren. KI übernimmt diesen Reflex oft ungeprüft. Doch nicht jede Einsicht muss sofort entstehen. Manche Aufgaben dürfen warten. Manche dürfen nachts laufen. Manche dürfen auf Solarüberschuss warten. Manche dürfen in Büro-Leerlaufzeiten entstehen.

Das klingt klein, ist aber groß. Sobald Zeit nicht mehr als starre Echtzeitanforderung behandelt wird, verändert sich die Energielogik. Rechenarbeit kann verschoben werden. Nicht jede Aufgabe konkurriert dann mit der Antwortzeit eines Nutzers. Hintergrundwissen kann aufgebaut werden, bevor es gebraucht wird.

Google hat bereits früh beschrieben, flexible Rechenlasten stärker dann auszuführen, wenn sauberere Energie verfügbar ist. Die Green Software Foundation arbeitet an Werkzeugen für carbon-aware Software. Solche Ansätze sind keine fertige Lösung für KI, aber sie zeigen ein Prinzip: Software kann lernen, den Zustand der Energieinfrastruktur zu respektieren.

Episodische Intelligenz übernimmt dieses Prinzip und erweitert es: Nicht nur Emissionen, sondern auch Kosten, lokale Energie, Datenschutz, Latenz, Wärme und Business-SLA werden Teil der Scheduling-Frage. Die Maschine denkt nicht nur mit Chips. Sie denkt auch mit Kalendern.

Zeitfenster statt Dauerfeuer
Wenn eine Aufgabe nicht sofort erledigt werden muss, sollte sie auch nicht automatisch im teuersten Zeit- und Energiepunkt laufen.

1.11 Warum die Frontier trotzdem bleibt

Ein Buch über lokale Systeme, deterministische Schichten und Energieopportunismus kann leicht missverstanden werden. Deshalb muss der Gegenpunkt früh stehen: Die Frontier verschwindet nicht. Große Modelle, große Trainingsläufe, hochoptimierte Netzwerke und spezialisierte Rechenzentren bleiben notwendig.

Es gibt Aufgaben, bei denen viele kleine Knoten nicht reichen. Synchrones Training großer Modelle braucht Bandbreite, Speicher, Fehlerbehandlung, eng gekoppelte Hardware und extreme Betriebssicherheit. Forschungsvorsprung zählt. Trainingszeit zählt. Die Qualität der Modellfabrik zählt.

Diese These richtet sich nicht gegen Rechenzentren. Sie richtet sich gegen die automatische Voreinstellung, jede Frage sofort an die größte kognitive Instanz zu geben. Das Bild des Hochofens wird später wiederkehren; hier genügt der Hinweis: Ein reifes KI-System entscheidet zuerst, welche Form von Rechenarbeit angemessen ist.

```
Nicht:
  lokal gegen cloud

Sondern:
  lokal, regional, verteilt und frontier - je nach Aufgabe
```

1.12 Die Machtfrage

Wer KI nur als Modell betrachtet, sieht auch Macht nur als Modellbesitz. Das greift zu kurz. KI-Macht entsteht entlang des ganzen Stacks: Chips, Cloud, Rechenzentren, Energieverträge, Datenzugang, Runtime, APIs, Interfaces, Standards, Beschaffung, Zertifizierung und Wechselkosten.

Je zentraler KI betrieben wird, desto stärker konzentrieren sich diese Eintrittstore. Das bedeutet nicht, dass große Anbieter automatisch schlecht handeln. Es bedeutet, dass Abhängigkeiten entstehen, die man nicht erst bemerkt, wenn sie schmerzen. Wer sein Gedächtnis, seine Semantik und seine Prozesse vollständig in fremde Runtime-Schichten legt, verliert mehr als nur Datenhoheit.

Der EU Data Act adressiert unter anderem Datenzugang und Cloud-Wechsel. Solche regulatorischen Bewegungen zeigen, dass digitale Souveränität nicht nur ein romantischer Begriff ist. Sie ist eine praktische Frage von Portabilität, Interoperabilität, Exit-Rechten und technischen Wechseloptionen.

Kapitel 11 und 12 werden diese Frage später ausführlich behandeln. Hier reicht die Grundspannung: Wenn KI zur Infrastruktur wird, dann ist die Organisation dieser Infrastruktur eine gesellschaftliche Frage. Nicht nur eine Produktentscheidung.

Souveraenitaet beginnt frueher

Souveränität beginnt nicht erst beim eigenen Modell. Sie beginnt bei der Frage, wem Gedächtnis, Kontext, Semantik und Runtime gehören.

1.13 Das Buch in einem Bild

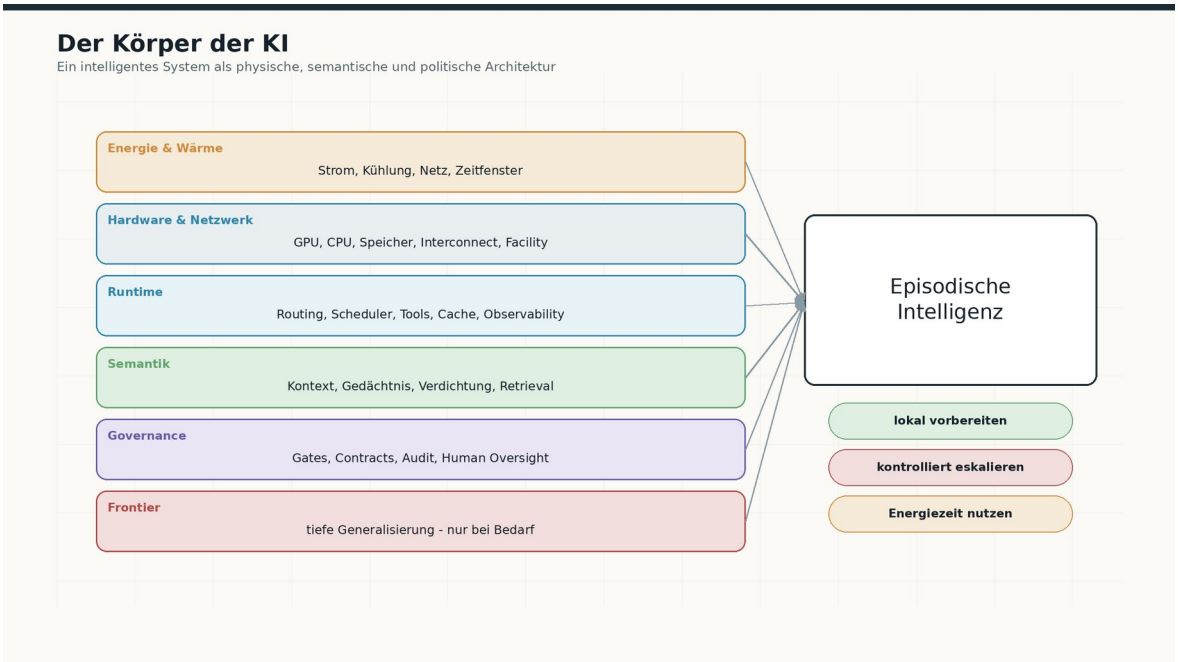

Abb. 1 - Der Körper der KI: ein intelligentes System als Energie-, Hardware-, Runtime-, Semantik-, Governance- und Frontier-Schichtung.

Episodische Intelligenz wird in den kommenden Kapiteln keine einzelne Technologie beschreiben. Es wird eine Betriebsform beschreiben. Eine Art, KI so zu organisieren, dass Kontext, Energie, Datenschutz, Governance, lokale Systeme und Frontier-Kognition zusammengedacht werden.

Das Bild sieht grob so aus:

```
lokale semantische Runtime
  -> deterministische Governance
  -> energieopportunistische Hintergrundarbeit
  -> kooperative Edge-Infrastruktur
  -> episodische Frontier-Eskalation
```

Diese fünf Schichten sind keine starre Produktarchitektur. Sie sind ein Denkrahmen. Manche Organisationen werden nur zwei davon brauchen. Andere werden alle fünf ausbauen. Wichtig ist nicht Vollständigkeit. Wichtig ist die neue Voreinstellung.

Die alte Voreinstellung lautet: Alles geht zuerst nach oben. Die neue lautet: Erst lokal verstehen, dann entscheiden, ob Eskalation nötig ist. Das klingt bescheiden. Aber Defaults formen Systeme. Sie formen Kosten. Sie formen Sicherheitsmodelle. Sie formen Märkte.

1.14 Die Illusion endet nicht mit Ernüchterung

Die Entzauberung der KI muss nicht langweilig sein. Im Gegenteil. Erst wenn man den Körper der Maschine sieht, wird ihre eigentliche Schönheit sichtbar. Nicht als mystische Stimme aus der Cloud, sondern als komplexes Zusammenspiel aus Schichten, Entscheidungen, Grenzen und Möglichkeiten.

Eine erwachsene KI-Architektur fragt nicht nur, was möglich ist. Sie fragt, was angemessen ist. Sie fragt nach Maß. Nach Ort. Nach Zeit. Nach Energie. Nach Risiko. Nach Eigentum. Nach Begründung. Nach dem Punkt, an dem ein System nicht mehr größer, sondern klüger werden sollte.

Dieses Buch ist deshalb kein Rückzug aus der KI. Es ist ein Versuch, KI ernster zu nehmen als die erste Begeisterung es konnte. Die erste Begeisterung zeigte, dass Maschinen Sprache erzeugen können. Die nächste Phase muss zeigen, wie wir diese Fähigkeit betreiben, ohne alles andere daran aus den Augen zu verlieren.

Vielleicht ist das der eigentliche Übergang: weg von KI als Spektakel, hin zu KI als Infrastruktur. Weg von der permanenten Superintelligenz, hin zur angemessenen Intelligenz am richtigen Ort, zur richtigen Zeit, mit der richtigen Menge Wirklichkeit.

1.15 Der Satz, mit dem wir beginnen

Am Ende dieses Buches wird eine Frage stehen. Sie steht auch am Anfang:

Welche Teile eines intelligenten Systems müssen wirklich zentral, sofort und mit maximalem Energieeinsatz ausgeführt werden?

Alles Weitere ist Ausarbeitung. Kapitel 2 steigt hinab in den Maschinenraum der Energie. Kapitel 3 öffnet den verborgenen Betriebsapparat. Kapitel 4 holt den Client zurück. Kapitel 5 fragt, wie Bedeutung geschützt und verdichtet werden kann. Kapitel 6 beschreibt Frontier als Eskalation. Kapitel 7 macht Governance ausführbar. Kapitel 8 und 9 geben dem Schwarm Arbeit und Stoffwechsel. Kapitel 10 hält die Grenze. Kapitel 11 und 12 stellen die Macht- und Souveränitätsfrage. Kapitel 13 kommt zu diesem Anfang zurück.

Die Illusion der permanenten Superintelligenz war nützlich, solange wir staunten. Jetzt müssen wir bauen.

Übergang: Aus der anfänglichen Irritation wird jetzt eine Messfrage. Wenn KI einen Körper hat, müssen wir fragen, wo in diesem Körper Energie, Wärme und Kosten entstehen.

Kapitel 2 - Die Maschine denkt in Strom

Strom, Wärme und die Kosten jeder Entscheidung

Die Maschine denkt nicht im Äther. Sie denkt im Strom.

Eine KI-Antwort sieht leicht aus. Ein Satz erscheint auf dem Bildschirm. Eine Zusammenfassung gleitet in ein Chatfenster. Ein Assistent formuliert eine E-Mail, analysiert einen Vertrag, entwirft eine Strategie. Von außen wirkt das fast körperlos. Keine Maschine ist zu sehen. Kein Motor dreht. Kein Rauch steigt auf. Das Ergebnis ist Text, also wirkt auch der Vorgang wie Sprache.

Aber hinter dieser Sprache steht ein industrieller Prozess. Nicht im alten Sinn von Schloten und Gießereien, aber im sehr realen Sinn von Strom, Kupfer, Glasfaser, Transformatoren, Kühlkreisläufen, Batterien, Generatoren, Hallen, Racks und Hitze. KI ist nicht nur Software. KI ist auch eine thermische Maschine.

Dieses Kapitel ist deshalb der Maschinenraum des Buches. Ohne ihn bleiben die späteren Ideen - lokale semantische Layer, deterministische Vorentscheidung, Semantic Compression, energieopportunistische Verarbeitung, episodische Frontier-Eskalation - elegant, aber nicht zwingend. Mit ihm werden sie zu einer Reaktion auf eine physische Grenze.

Die zentrale Frage lautet nicht: Verbraucht KI Energie? Natürlich tut sie das. Die zentrale Frage lautet: Welche Teile eines intelligenten Systems müssen wirklich im teuersten, dichtesten und energiehungrigsten Rechenzentrum der Welt stattfinden?

2.1 Der unsichtbare Körper der KI

Die Cloud ist eine gefährliche Metapher. Sie klingt nach Nebel, Leichtigkeit und Ortslosigkeit. Ein Rechenzentrum ist aber das Gegenteil von Nebel. Es ist schwer. Es hat Beton, Zäune, Umspannwerke, Notstromaggregate, Wasserleitungen, Brandschutz, Sicherheitszonen und Wartungsverträge. Es ist eine Fabrik, nur dass sie keine Autos, Chemikalien oder Stahl produziert, sondern digitale Verfügbarkeit.

Für klassische Webdienste war diese Fabrik schon lange notwendig. Suchmaschinen, Streaming, E-Commerce, Unternehmenssoftware und soziale Netzwerke haben Datenzentren groß gemacht. Die neue KI-Welle verändert jedoch die Dichte. Sie bringt Beschleuniger, GPU-Racks, Hochgeschwindigkeitsnetzwerke und Modellinferenz in Maßstäbe, die eher an Hochleistungsrechnen als an normale Büro-IT erinnern.

Die Internationale Energieagentur schätzt, dass Rechenzentren im Jahr 2024 rund 415 TWh Strom verbrauchten - ungefähr 1,5 Prozent des weltweiten Stromverbrauchs. Bis 2030 erwartet die IEA in ihrem Basisszenario rund 945 TWh. Das wäre etwas mehr als der heutige Stromverbrauch Japans. AI ist dabei nicht der einzige Treiber, aber einer der wichtigsten Beschleuniger. [S1] [S2]

Diese Zahlen sind groß genug, um ernst zu sein, aber sie sind nicht groß genug, um in Panik zu verfallen. Rechenzentren sind nicht der einzige Stromfresser der Welt. Elektromobilität, Klimaanlagen, industrielle Motoren und Wärmepumpen sind ebenfalls riesige Lastblöcke. Gerade deshalb muss man nüchtern bleiben. Das Argument dieses Buches ist nicht: KI darf keine Energie verbrauchen. Das Argument ist: KI muss lernen, Energie intelligenter zu organisieren.

Beobachtung	Bedeutung für episodische Intelligenz
Rechenzentren sind physische Infrastruktur.	KI-Architektur darf nicht nur logisch gedacht werden, sondern auch energetisch.
AI erhöht die Leistungsdichte in Racks.	Kühlung, Stromversorgung und Netzanschlüsse werden Teil der Modellfrage.
Wachstum ist regional konzentriert.	Lokale Netze, Standorte und politische Genehmigungen werden kritisch.
Die Nachfrage ist unsicher, aber real.	Die Architektur sollte flexibel sein statt blind auf permanente Zentralisierung zu setzen.

2.2 Ein Token ist ein Rechenereignis

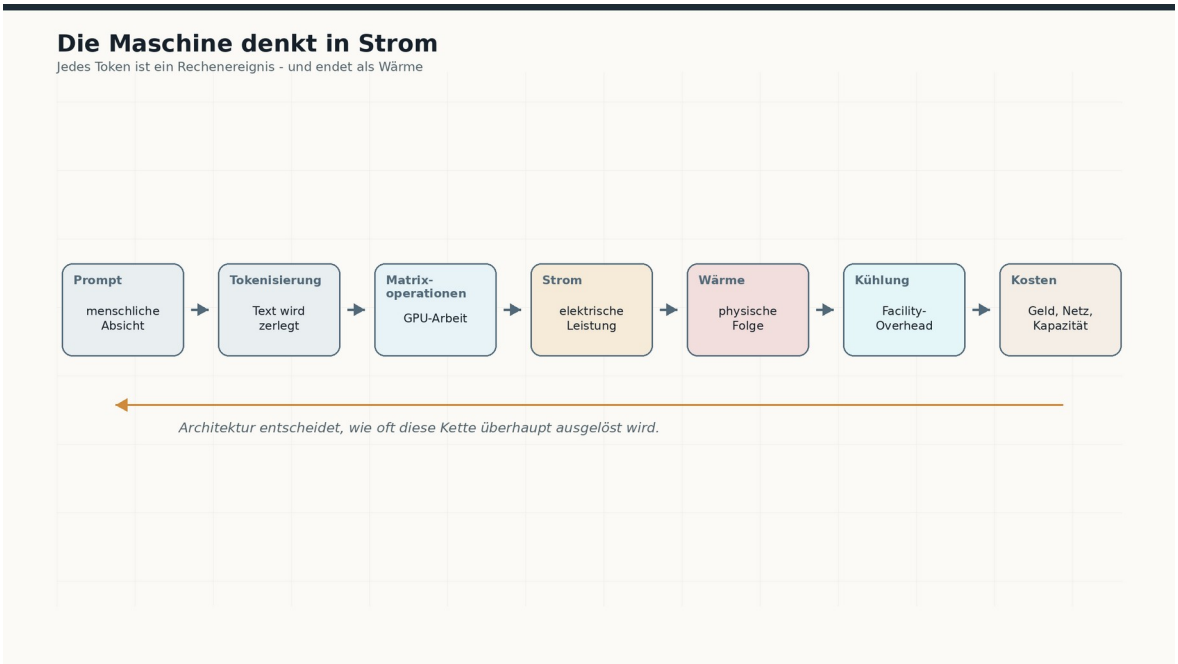

Abb. 2 - Die Maschine denkt in Strom: vom Prompt über Rechenoperationen zu Wärme, Kühlung und Kosten.

Im Alltag sprechen wir von Tokens, als wären sie kleine Textstücke. Technisch stimmt das teilweise: Ein Token ist eine Verarbeitungseinheit eines Sprachmodells, oft ein Wortteil, ein Wort oder ein Zeichenmuster. Energetisch ist ein Token aber mehr als Text. Ein generiertes Token ist ein Rechenereignis.

Damit ein großes Transformer-Modell ein nächstes Token erzeugt, müssen viele Rechenoperationen stattfinden: Matrixmultiplikationen, Speicherzugriffe, Attention-Mechanismen, Aktivierungen, Normalisierungen, Datenbewegungen zwischen GPU-Speicher und Recheneinheiten. Bei großen Modellen laufen diese Vorgänge nicht auf einem einzigen Chip, sondern verteilt über viele Beschleuniger, Server und Netzwerke.

Vom Prompt zur Wärme **Prompt**

-> Tokenisierung
-> Modellaktivierung
-> Matrixoperationen
-> Speicherzugriffe
-> GPU- und Netzwerkleistung
-> Stromverbrauch
-> Wärme
-> Kühlung
-> Infrastrukturkosten

Das einzelne Token ist klein. Die Masse ist das Problem. Eine Antwort mit wenigen hundert Tokens ist harmlos. Millionen Nutzer, lange Kontexte, Agentenloops, Retrieval, Wiederholungen, Evals, automatische Zusammenfassungen und Hintergrundprozesse verändern die Größenordnung. Der einzelne Tropfen ist nicht dramatisch. Der Strom ist es.

Hier beginnt die Architekturfrage. Wenn jede triviale Entscheidung durch ein großes Modell läuft, wird das System nicht nur teuer. Es wird thermisch und infrastrukturell schwerfällig. Wenn dagegen ein lokaler oder deterministischer Layer vorab entscheiden kann, welche Anfrage überhaupt Frontier-Reasoning benötigt, spart er nicht nur Tokens. Er spart Energie, Latenz, Netzwerkverkehr, Kühlung und organisatorische Abhängigkeit.

2.3 Die neue Leistungsdichte

Der Unterschied zwischen normaler Server-IT und moderner AI-Infrastruktur zeigt sich besonders an der Leistungsdichte. Eine einzelne NVIDIA H100 SXM-GPU hat eine konfigurierbare maximale Thermal Design Power von bis zu 700 Watt. Ein DGX H100/H200-System mit acht GPUs wird von NVIDIA mit bis zu 10,2 kW Systemleistung spezifiziert. Ein DGX B200-System liegt bei rund 14,3 kW maximaler Systemleistung. [S6] [S7] [S8]

Diese Zahlen sind keine exotischen Laborwerte. Sie sind ein Hinweis darauf, wie sich die Form des Rechenzentrums verändert. Früher konnte man Racks mit vergleichsweise moderaten Leistungsdichten planen. Moderne AI-Systeme verdichten Rechenleistung so stark, dass Stromverteilung, Kühlung und Rackdesign selbst zu Engpässen werden.

Eine einfache Überschlagsrechnung macht das greifbar. Zehntausend H100-SXM-GPUs können allein auf GPU-Ebene bis zu etwa sieben Megawatt thermische Designleistung bedeuten. Rechnet man Serverkomponenten, Netzwerk, Stromwandlungsverluste und Kühlung hinzu, landet man nicht mehr bei einem Serverraum, sondern bei einer Industrieanlage. Epoch AI schätzt für typische Frontier-AI-Datacenter, dass GPUs bei Peak-Operation nur etwa 40 Prozent der gesamten Leistung ausmachen; Server, Netzwerk, Facility-Overhead und Kühlung bilden den Rest. [S5] [S6]

Komponente	Beispielhafte Quelle	Warum wichtig
GPU	H100 SXM bis zu 700 W TDP	Der Beschleuniger ist der sichtbarste, aber nicht der einzige Energieblock. [S6]
AI-Server	DGX H100/H200 bis zu 10,2 kW; DGX B200 rund 14,3 kW	Die Serverplattform addiert CPU, Speicher, NVSwitch, Netz und Stromversorgung. [S7] [S8]
IT-Umfeld	Switches, Management-Nodes, Storage, Load Balancer	Das Modell läuft nicht allein; es läuft in einem System. [S1] [S5]
Facility	Kühlung, Stromumwandlung, Beleuchtung, Sicherheit	Der Rechenvorgang erzeugt Wärme und braucht Infrastruktur. [S1] [S4] [S5]

Das ist der erste wichtige Bruch mit dem romantischen Bild der KI. Ein Frontier-Modell ist nicht einfach ein Programm. Es ist ein Programm, das in einer sehr speziellen Energielandschaft lebt.

2.4 Training ist der Hochofen, Inferenz ist der Alltag

Man muss zwischen Training und Inferenz unterscheiden. Training ist der große industrielle Lauf: ein Modell lernt aus riesigen Datenmengen, Gewichte werden angepasst, Gradienten werden berechnet, Cluster

werden synchronisiert. Inferenz ist der tägliche Betrieb: ein bereits trainiertes Modell beantwortet Fragen, schreibt Text, verarbeitet Bilder, erzeugt Code oder führt Agentenschritte aus.

Training ist spektakulär. Es ist der Hochofen der Frontier-KI. Es braucht enorme Cluster, enge Synchronisierung, schnelle Netzwerke und hohe Auslastung. In diesem Bereich zählt Zeit, weil GPU-Zeit teuer ist und Forschungsdruck real. Wer einen Lauf statt in drei Monaten erst in zwölf Monaten abschließt, verliert nicht nur Zeit, sondern Kapital, Talentfenster und Marktposition.

Inferenz wirkt unscheinbarer, kann aber im Alltag zur größeren Last werden. Sobald ein Modell breit genutzt wird, entstehen viele kleine Anforderungen: Chats, Zusammenfassungen, Agentenläufe, Dokumentenanalyse, Schreibassistenz, Sucherweiterung, Codegenerierung, interne Unternehmensprozesse. Eine kritische Literaturauswertung zu Rechenzentrumsenergie verweist auf Daten von Meta und Google, nach denen Training etwa 20 bis 40 Prozent der ML-bezogenen Energie ausmachen kann, während 60 bis 70 Prozent aus Inferenz stammen; mit wachsender Nutzung generativer KI dürfte der Anteil der Inferenz weiter steigen. [S13]

Für episodische Intelligenz ist genau dieser Punkt zentral. Wenn Inferenz der Alltag ist, dann ist die Frage nicht nur, wie man große Modelle trainiert. Die Frage ist, wie man den täglichen Gebrauch so gestaltet, dass nicht jede Mikroentscheidung in die teuerste Modellschicht eskaliert.

Zwei Energiecharaktere
Training:
- **wenige, extrem große Läufe**
- **hohe Synchronisierung**
- **Forschungsvorsprung und Kapitalkosten**

Inferenz:
- **viele, wiederkehrende Anfragen**
- **hohe Verfügbarkeit**
- **Nutzerwachstum und Prozessintegration**
- **langfristige Betriebskosten**

2.5 Kühlung ist keine Fußnote

Fast jede Kilowattstunde, die in einem Rechenzentrum in IT-Hardware fließt, endet als Wärme. Das ist keine Metapher. Es ist Physik. Die Chips rechnen, die Speicher bewegen Daten, die Netzteile wandeln Strom, die Switches leiten Pakete, und am Ende muss die Wärme aus dem Gebäude.

Darum ist Kühlung kein Nebenthema. Die IEA beschreibt, dass der Anteil von Kühlung und Umweltkontrolle am Stromverbrauch eines Rechenzentrums stark variiert: ungefähr sieben Prozent bei sehr effizienten Hyperscale-Anlagen, aber über dreißig Prozent bei weniger effizienten Enterprise-Rechenzentren. [S10]

Die Kennzahl PUE - Power Usage Effectiveness - versucht diesen Zusammenhang zu fassen. Eine PUE von 1,0 wäre perfekt: Jede Kilowattstunde geht in IT. Eine PUE von 1,5 bedeutet: Für eine Kilowattstunde IT kommen zusätzlich 0,5 Kilowattstunden Infrastrukturverbrauch hinzu. In der Realität liegen gute Hyperscale-Anlagen oft deutlich niedriger als ältere Unternehmensrechenzentren, aber auch dort verschwindet der Facility-Overhead nicht.

Der LBNL-Bericht zur US-Rechenzentrumsenergie zeigt, dass die durchschnittliche PUE in den USA von etwa 1,6 im Jahr 2014 auf etwa 1,4 im Jahr 2023 gefallen ist. Für 2028 wird je nach Szenario ein Bereich von 1,15 bis 1,35 erwartet, getrieben durch effizientere Hyperscale- und Colocation-Anlagen sowie mehr flüssigkeitsgekühlte AI-Server. Gleichzeitig erwartet der Bericht steigende durchschnittliche WUE-Werte - also Wasserverbrauch pro Energieeinheit - unter anderem im Zusammenhang mit mehr flüssigkeitsgekühlten Systemen. [S4]

Das zeigt den zweiten Bruch mit dem Cloud-Bild: Effizienz verbessert sich, aber sie hebt die Physik nicht auf. Eine effizientere Kühlung macht ein Rechenzentrum nicht körperlos. Sie macht den Körper besser trainiert.

2.6 Das Netzwerk denkt mit

Ein Frontier-System besteht nicht aus GPUs allein. Es besteht aus GPUs, CPUs, Speicher, NVMe, Netzwerkkarten, NVSwitches, InfiniBand oder schnellen Ethernet-Fabrics, Load Balancern, Storage-Systemen, Management-Nodes, Observability, Security und Schedulern. Die Intelligenz erscheint als Modell, aber der Betrieb ist ein verteiltes System.

Die IEA ordnet Storage-Systemen rund fünf Prozent des Rechenzentrumsstromverbrauchs zu und nennt für Networking-Equipment bis zu fünf Prozent. Das klingt klein, ist aber bei großen Anlagen ein realer Energieblock. Außerdem ist Netzwerk nicht nur eine Energiefrage, sondern eine Performance-Grenze. Große Trainingsläufe brauchen schnelle Synchronisierung. Große Inferenzsysteme brauchen Routing, Load Balancing, Caches, KV-Speicher, Retrieval und Antwortstreaming. [S10]

Epoch AI beschreibt denselben Punkt aus einer anderen Perspektive: In einem typischen Frontier-AI-Datacenter sei die gesamte Serverleistung etwa 1,53-mal so hoch wie die GPU-Leistung allein; alle IT-Geräte lägen noch einmal darüber, und auf Facility-Ebene komme zusätzlicher Overhead für Kühlung, Beleuchtung und Stromumwandlung hinzu. [S5]

Das ist wichtig für dieses Buch, weil es den Begriff Intelligenz verschiebt. Nicht nur das Modell ist intelligent. Auch das System, das entscheidet, wann welches Modell, welcher Speicher, welches Tool und welche Validierung verwendet wird, nimmt an der Intelligenz teil. Routing ist nicht bloß Verwaltung. Routing ist eine energetische und semantische Entscheidung.

2.7 Der verborgene Energieblock: Deterministik, Routing, Retrieval, Governance

In produktiven KI-Systemen passiert viel, bevor ein großes Modell überhaupt antwortet. Eine Anfrage wird klassifiziert. Berechtigungen werden geprüft. Inhalte werden gefiltert. Dokumente werden gesucht. Kontext wird gekürzt. Caches werden geprüft. Tools werden ausgewählt. Policies werden angewendet. Ergebnisse werden validiert. Logs werden geschrieben. Fehler werden behandelt.

Diese Schichten sind nicht kostenlos. Sie laufen auf CPUs, nutzen RAM, schreiben in Datenbanken, bewegen Netzwerkpakete, erzeugen Metriken, halten Queues offen und speichern Zwischenstände. Wer behauptet, deterministisches Routing verbrauche keine Energie, romantisiert die eigene Pipeline.

Aber pro Entscheidung sind viele dieser Schichten sehr viel günstiger als ein großer Frontier-Call. Eine Regelprüfung, ein Cache-Lookup, ein Intent-Klassifikator, ein kleines lokales Modell oder eine Vektorsuche können einen teuren Modellaufruf vermeiden oder verkleinern. Der Energiegewinn entsteht nicht dadurch, dass Routing magisch kostenlos wäre. Er entsteht dadurch, dass gutes Routing die teuerste kognitive Schicht seltener und präziser aktiviert.

Produktiver KI-Pfad
User Request
-> Input Gate
-> Intent / Risk Classification
-> Policy Check
-> Cache Lookup
-> Retrieval / Context Selection
-> Compression / Abstraction
-> Model Routing
-> Frontier only if necessary
-> Validation
-> Response

Damit entsteht eine neue ökonomische Frage: Wie viel Bedeutung kann ein System vor dem Modell organisieren? In klassischen Softwarearchitekturen war das eine Frage von Qualität und Wartbarkeit. In Systemen episodischer Intelligenz wird es zusätzlich eine Frage von Strom, Wärme und Standortpolitik.

2.8 Die falsche Optimierung: alles zentral, alles sofort

Das heutige Cloud-Modell optimiert stark auf zentrale Verfügbarkeit. Dienste sollen jederzeit antworten, Lastspitzen aufnehmen, global skalieren, einheitlich kontrolliert werden und mit maximaler Auslastung arbeiten. Für viele Anwendungen ist das sinnvoll. Für manche sogar unverzichtbar.

Bei KI wird diese Logik jedoch schnell teuer. Nicht jede Aufgabe ist zeitkritisch. Nicht jede semantische Pflege muss sofort passieren. Nicht jede Akte muss in Echtzeit neu indiziert werden. Nicht jedes Dokument braucht ein Frontier-Modell. Nicht jede Frage verlangt die tiefste Reasoning-Schicht.

Wenn man alles sofort und zentral lösen will, entsteht eine Architektur, die ihre teuerste Ressource als Standardpfad verwendet. Das ist bequem, solange Energie, Hardware, Netzanschlüsse und Kapital scheinbar beliebig verfügbar sind. Sobald diese Ressourcen knapper, teurer oder politisch umstrittener werden, wird Bequemlichkeit zur Schwäche.

Der Fehler liegt nicht darin, große Rechenzentren zu bauen. Der Fehler liegt darin, ihre Rolle nicht genau genug zu bestimmen.

2.9 Die Alternative: Energieopportunistische Verarbeitung

Viele Teile intelligenter Systeme müssen nicht im Moment der Nutzeranfrage entstehen. Ein lokales oder regionales System kann Dokumente in ruhigen Zeiten klassifizieren, Embeddings aktualisieren, Indizes pflegen, Wissensgraphen verdichten, Evals ausführen, Caches auffrischen, Modelle feinjustieren oder Policy-Tests durchführen.

Wenn lokale Energie verfügbar ist - zum Beispiel Solarüberschuss, Nachtstrom, Abwärmenutzung oder ohnehin laufende Büro-Hardware - kann ein Teil dieser Arbeit zeitlich verschoben werden. Nicht der Nutzer wartet auf das Training. Das System arbeitet im Hintergrund. Es schläft nicht vollständig, aber es träumt günstig.

Energieopportunistischer Hintergrundbetrieb
Wenn Solarüberschuss:
-> Embedding Refresh
-> Dokumentklassifikation
-> lokaler Wissensgraph
-> Eval Runs
-> synthetische Testfälle
Wenn Nutzeranfrage live:
-> lokale Vorentscheidung
-> vorbereiteter Kontext
-> nur bei Unsicherheit Frontier-Eskalation

Das ist kein Argument gegen Rechenzentren. Es ist ein Argument gegen unnötige Dauerzentralisierung. Die großen Zentren bleiben für Training, Forschung, globale Modelle, komplexe Simulationen und tiefe Reasoning-Aufgaben wichtig. Aber sie müssen nicht zwangsläufig jede semantische Kleinarbeit übernehmen.

Energieopportunistische KI denkt nicht nur in FLOPS, sondern in Zeitfenstern. Sie fragt: Was muss jetzt passieren? Was kann vorbereitet werden? Was kann lokal bleiben? Was kann verschoben werden? Was muss eskalieren?

2.10 Lokale Systeme sparen nicht automatisch Energie

Hier muss das Buch ehrlich bleiben. Lokal ist nicht automatisch grün. Ein schlecht ausgelasteter Heimserver, ein ineffizienter Mini-Cluster, unnötige Hintergrundjobs oder ständig laufende lokale Modelle können am Ende mehr Energie verbrauchen als ein hocheffizientes Hyperscale-Rechenzentrum mit guter PUE und hoher Auslastung.

Die Stärke lokaler Systeme liegt daher nicht in einem romantischen Dezentralisierungsreflex. Sie liegt in spezifischen Fällen: kurze Datenwege, Datenschutz, lokale Semantik, geringe Latenz, Nutzung ohnehin vorhandener Hardware, zeitverschiebbare Aufgaben, Energieüberschüsse und die Vermeidung unnötiger Frontier-Aufrufe.

Man muss das testen. Jede Architektur episodischer Intelligenz braucht Messpunkte: Wie viele Frontier-Calls wurden vermieden? Wie viel lokale Energie wurde eingesetzt? Wie hoch war die Auslastung? Welche Qualität ging verloren? Welche Daten blieben lokal? Wie viel Netzwerkverkehr wurde vermieden? Ohne Messung wird aus Infrastrukturdenken schnell Ideologie.

Behauptung	Saubere Version
Lokal ist immer besser.	Falsch. Lokal ist nur besser, wenn Auslastung, Energiequelle, Datenschutz- oder Latenzvorteile den Overhead rechtfertigen.
Datacenter sind immer schlecht.	Falsch. Gut betriebene Hyperscale-Anlagen können sehr effizient sein und bleiben für Frontier-Aufgaben wichtig.
Deterministik ist kostenlos.	Falsch. Sie verbraucht Energie, kann aber teure Modellaufrufe reduzieren.
Training ist nicht zeitkritisch.	Teilweise. Für Hintergrundlernen ja; für Frontier-Pretraining oft nein.
Semantic Compression spart immer Energie.	Nicht automatisch. Sie spart nur, wenn der Abstraktionsaufwand kleiner ist als der vermiedene Modell- und Kontextaufwand.

2.11 Warum die Energiefrage eine Architekturfrage ist

Der naheliegende Reflex auf steigenden Energiebedarf lautet: effizientere Chips, bessere Kühlung, erneuerbare Energie, neue Kraftwerke, bessere PUE. All das ist wichtig. Aber es greift zu kurz, wenn die Architektur selbst verschwenderisch bleibt.

Ein schlecht organisiertes KI-System kann auch auf effizienter Hardware unnötig viel Arbeit erzeugen. Es kann zu lange Kontexte senden, obwohl wenige strukturierte Signale reichen würden. Es kann Agentenschleifen laufen lassen, obwohl ein deterministischer Workflow genügt. Es kann Rohdaten in die Cloud schicken, obwohl ein lokaler semantischer Layer eine abstrakte Aufgabe hätte erzeugen können. Es kann ein Frontier-Modell fragen, obwohl ein Cache bereits eine geprüfte Antwort enthält.

Darum ist Energie nicht nur Sache der Stromversorger und Datacenter-Planer. Energie ist auch Sache der Softwarearchitektur. Jedes Routing, jeder Kontextschnitt, jede Speicherentscheidung und jede Eskalationsregel verändert die Energiebilanz.

Episodische Intelligenz ist deshalb keine Anti-Cloud-Idee. Sie ist eine architektonische Grammatik für eine Welt, in der zentrale Hochenergie-Kognition wertvoll bleibt, aber nicht länger als Standardantwort auf jede kleine Frage behandelt werden sollte.

Entscheidungsfrage episodischer Intelligenz
Kann die Anfrage lokal deterministisch gelöst werden?
Ja -> lokal antworten
Nein -> reicht lokales kleines Modell?
Ja -> lokal antworten / validieren
Nein -> kann Kontext komprimiert werden?
Ja -> abstrahierte Frontier-Eskalation
Nein -> kontrollierte Rohkontext-Eskalation mit Schutzmaßnahmen

2.12 Der Netzanschluss ist nicht die Steckdose

Wenn über KI-Energie gesprochen wird, landet die Diskussion oft bei Chips und Kühlung. Das ist verständlich, aber unvollständig. Ein Rechenzentrum ist nicht nur eine Last im Gebäude. Es ist eine Last im Netz. Es braucht Transformatoren, Umspannwerke, Leitungen, Reservekapazitäten, Genehmigungen, Lastmanagement und manchmal neue Erzeugung.

Die DOE-Zusammenfassung des LBNL-Berichts zeigt, wie schnell sich das in einem Land wie den USA materialisiert: Rechenzentren verbrauchten dort 2023 rund 176 TWh beziehungsweise etwa 4,4 Prozent des US-Stroms; bis 2028 werden 325 bis 580 TWh und 6,7 bis 12 Prozent erwartet. [S3] EPRI geht in aktualisierten Szenarien davon aus, dass Rechenzentren bis 2030 je nach Entwicklung 9 bis 17 Prozent des US-Stromverbrauchs erreichen könnten. [S12]

Solche Zahlen sind nicht nur nationale Statistik. Sie werden lokal. EPRI weist ausdrücklich auf regionale Konzentrationen und Bundesstaaten hin, in denen Rechenzentren bereits heute oder künftig sehr hohe Stromanteile erreichen können. [S12] Das ist der Punkt, an dem KI-Infrastruktur politisch wird. Nicht wegen abstrakter Moral, sondern weil ein Netzanschluss ein knappes Gut ist.

Für episodische Intelligenz heißt das: Die Frage, ob eine Aufgabe lokal, regional oder zentral gerechnet wird, ist auch eine Netzfrage. Zentralisierung bündelt Last. Dezentralisierung verteilt Last, kann sie aber schwieriger messbar machen. Energieopportunistische Verarbeitung versucht, diese Spannung produktiv zu nutzen: nicht alles überall, nicht alles sofort, sondern Arbeit dort und dann, wo sie energetisch sinnvoll ist.

2.13 Das Effizienzparadox

Eine wichtige Falle in der Energiedebatte lautet: Wenn Rechenzentren effizienter werden, ist das Problem gelöst. Das stimmt nur teilweise. Effizienz verbessert den Energieeinsatz pro Recheneinheit. Aber wenn die Nachfrage schneller wächst als die Effizienz, steigt der absolute Verbrauch trotzdem.

Genau diese Spannung zeigen die Quellen. Die durchschnittliche PUE in den USA ist laut LBNL von etwa 1,6 im Jahr 2014 auf etwa 1,4 im Jahr 2023 gefallen, mit erwarteten Verbesserungen in Richtung 1,15 bis 1,35 bis 2028. [S4] Gleichzeitig steigen die absoluten Verbrauchsprognosen stark. Die IEA erwartet global eine Verdopplung auf rund 945 TWh bis 2030. [S1]

Das ist kein Widerspruch. Es ist das Effizienzparadox der KI-Infrastruktur. Die Maschine wird besser, aber sie wird auch häufiger, länger und tiefer benutzt. Modelle werden leistungsfähiger, Kontexte länger, Agenten aktiver, Unternehmen abhängiger, Nutzer zahlreicher.

Architektur muss deshalb beides leisten: Effizienz pro Operation verbessern und die Anzahl unnötiger Operationen reduzieren. Bessere Chips allein beantworten nicht, ob ein 80.000-Token-Kontext für eine Aufgabe wirklich nötig war. Eine bessere PUE beantwortet nicht, ob ein lokaler Cache den gesamten Frontier-Call hätte verhindern können.

Optimierung	Was sie verbessert	Was sie nicht automatisch löst
Bessere Chips	Weniger Energie pro Rechenoperation.	Unnötige Modellaufrufe und schlechte Workflows.
Bessere PUE	Weniger Facility-Overhead pro IT-kWh.	Absolutes Nachfragewachstum.
Liquid Cooling	Höhere Rackdichten und bessere Wärmeabfuhr.	Wasserbedarf, Komplexität und Standortfragen.
Erneuerbare Energie	Bessere Emissionsbilanz, wenn sauber integriert.	Dauerlast, Netzengpässe und zeitliche Verfügbarkeit.
Episodische Intelligenz	Weniger unnötige Eskalationen, lokale Semantik, zeitverschiebbare Arbeit.	Nicht jede Frontier-Aufgabe und nicht jedes Training.

2.14 Agenten, lange Kontexte und der stille Multiplikator

Die Energiefrage verschärft sich nicht nur durch mehr Nutzer. Sie verschärft sich durch neue Nutzungsformen. Ein einfacher Chat erzeugt eine Antwort. Ein Agent erzeugt oft eine Kette: planen, suchen, lesen, Tool Call, interpretieren, erneut fragen, validieren, korrigieren, zusammenfassen. Jeder Schritt kann ein Modellaufruf sein. Jeder Modellaufruf kann Kontext tragen. Jeder Kontext kann wachsen.

Das ist der stille Multiplikator. Nicht das sichtbare Ergebnis wird länger, sondern der unsichtbare Arbeitsweg. Ein Nutzer sieht am Ende vielleicht drei Absätze. Das System hat dafür aber zehn Zwischenschritte, fünf Retrievals, drei Validierungen und zwei Korrekturschleifen ausgeführt.

Gerade deshalb braucht produktive KI harte Laufzeitregeln. Agenten brauchen Budgets: maximale Schritte, maximale Tokens, Eskalationsgrenzen, Cache-Pflicht, Abbruchbedingungen, Qualitätsprüfungen und klare Verträge. Ohne solche Regeln wird Autonomie schnell zu Energieverschwendung mit höflicher Oberfläche.

```
Agentenbudget als Energievertrag
task_budget:
  max_model_calls: 4
  max_context_tokens: 12000
  require_cache_lookup: true
  require_retrieval_before_frontier: true
  stop_if_confidence_below: escalate_to_human
  log_energy_proxy: tokens + model_class + runtime
  forbid_unbounded_loops: true
```

Das ist nicht nur Kostenkontrolle. Es ist ein neues Disziplinfeld: kognitive Laufzeitökonomie. Die Frage lautet nicht mehr nur, ob ein Agent eine Aufgabe lösen kann. Die Frage lautet, mit welchem kognitiven Verbrauch er sie löst.

2.15 Eine Messung in Schichten

Man kann die Größenordnung mit einer bewusst groben Rechnung sichtbar machen. Zehntausend H100-SXM-GPUs mit bis zu 700 Watt TDP entsprechen allein auf GPU-Ebene bis zu 7 MW. [S6] Epoch AI beschreibt für Frontier-AI-Datacenter eine Schichtung, bei der Serverleistung, weiteres IT-Equipment und Facility-Overhead die GPU-Leistung deutlich übersteigen; die von Epoch genannten Multiplikatoren ergeben überschlägig einen Gesamtfaktor von rund 2,4 gegenüber reiner GPU-Leistung bei Peak-Betrieb. [S5]

Aus 7 MW GPU-Leistung werden damit in der Größenordnung rund 17 MW Facility-Leistung. Bei dauerhaftem Betrieb wären das etwa 150 GWh pro Jahr; bei 80 Prozent durchschnittlicher Last immer noch rund 120 GWh. Diese Rechnung ist nicht als exakte Standortplanung gemeint. Sie ist ein Maßstabsbild. Sie zeigt, warum die Frage nach unnötigen Modellaufrufen nicht kleinlich ist.

Für ein Unternehmen reicht diese Rechnung allein aber nicht. Es braucht eine kleinere, operativ nutzbare Messform: nicht nur „Wie viel Strom verbraucht ein Rechenzentrum?“, sondern „Welche Schicht unserer KI-Anwendung erzeugt welche Arbeit, welche Offenlegung und welche Eskalation?“

Tabelle 2.1 - Minimalmodell für Messpunkte in Systemen episodischer Intelligenz

Messpunkt	Was gemessen wird	Warum es für die Architektur zählt
Modellarbeit	Input- und Output-Tokens, Kontextlänge, Modelltyp, Zahl der Modellaufrufe, Cache-Hit-Rate	Zeigt, ob die teuerste kognitive Schicht unnötig häufig oder unnötig breit aktiviert wird.
Runtime	Agentenschritte, Tool Calls, Wiederholungen, Abbrüche, Latenz, Warteschlangenzeit	Macht sichtbar, ob Autonomie in produktive Arbeit oder in Schleifen zerfällt.
Energie	kWh pro Job oder Node-Stunde, Auslastung, PUE- oder Facility-Faktor, lokale Erzeugung, Carbon-Intensity	Verbindet Softwareentscheidungen mit Strom, Wärme, Standort und Zeitfenster.
Datenoffenlegung	Datenklasse, Rohdatenanteil, externe Chunks, Named Entities, Geschäftsgeheimnisse, Re-Hydration-Punkte	Zeigt, ob eine Anfrage nicht nur teuer, sondern auch unnötig offen ist.
Qualität	PASS/FAIL, Eval-Score, Unsicherheit, Human-Gate-Entscheidung, Fehlerrate nach Antwort	Verhindert, dass Energiesparen zur Qualitätsverschlechterung mit späteren Korrekturschleifen wird.

Dieses Minimalmodell ist absichtlich schlicht. Es ersetzt keine Rechenzentrumsbilanz und keine Lebenszyklusanalyse. Aber es zwingt eine KI-Anwendung, ihren eigenen Stoffwechsel zu sehen. Ohne solche Messpunkte bleibt „effizient“ ein Gefühl. Mit ihnen wird Effizienz verhandelbar.

2.16 Eine Business-KI-Pipeline als Energiepfad

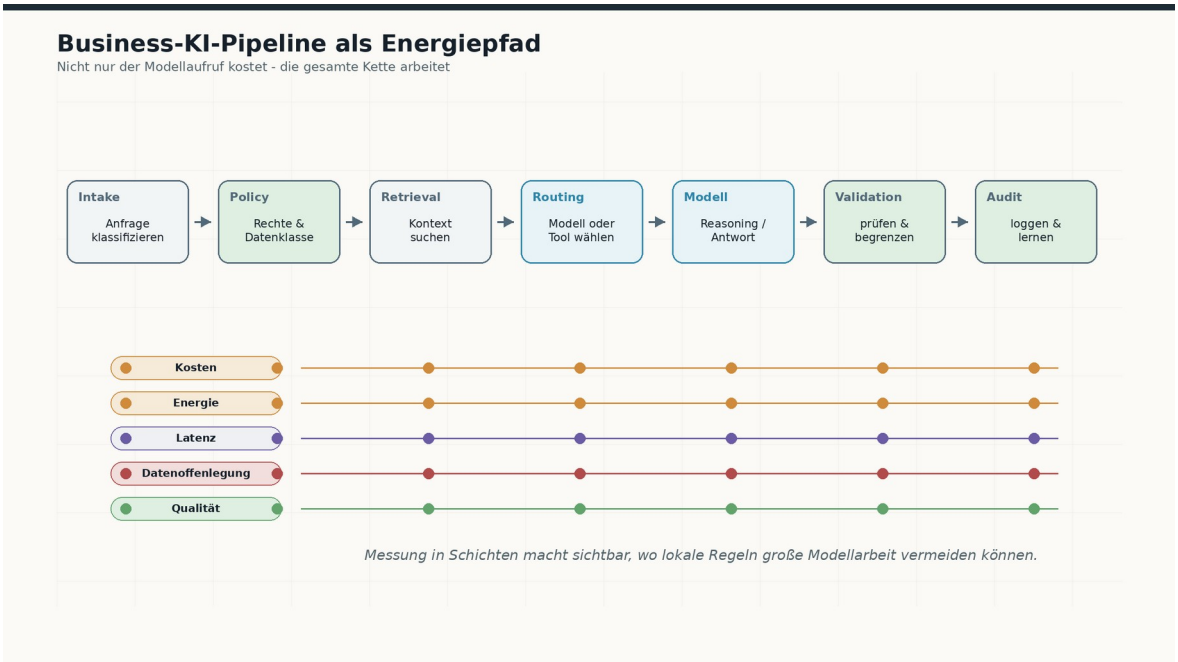

Abb. 3 - Business-KI-Pipeline als Energiepfad: Messpunkte für Kosten, Energie, Latenz, Datenoffenlegung und Qualität.

Typische Business-KI besteht selten aus einem einzigen Modellaufruf. Sie besteht aus einer Kette kleiner Entscheidungen. Die grobe Energiefrage lautet deshalb nicht nur: Wie teuer ist das Modell? Sondern: Welche vorgelagerten Schritte verhindern, verkleinern oder vervielfachen den Modellaufruf?

Eine deterministische Regelprüfung kann selbst Energie verbrauchen und dennoch sehr viel Energie sparen, wenn sie einen langen Agentenlauf verhindert. Eine Vektorsuche kann Strom kosten und dennoch sinnvoll sein, wenn sie einen 80.000-Token-Kontext durch zehn relevante Abschnitte ersetzt. Ein Logging-System kann Overhead erzeugen und trotzdem notwendig sein, wenn es später zeigt, welche Pfade verschwenderisch oder riskant waren.

Tabelle 2.2 - Beispielhafte Energie- und Steuerungsfunktion einer Business-KI-Pipeline

Schicht	Typischer Energie-/Kostenanteil	Architekturfunktion
Input-Gate	meist niedrig: Regeln, kleine Klassifikatoren, Datenklassifikation	Früh entscheiden, ob eine Anfrage überhaupt KI, Toolzugriff oder Frontier-Eskalation braucht.
Policy & Berechtigung	niedrig bis mittel: Verzeichnisdienste, Rollenprüfung, lokale Policy Stores	Verhindert, dass falsche Daten in falsche Pfade geraten. Spart spätere Korrektur, Risiko und unnötige Offenlegung.
Retrieval & Kontextbau	mittel: Vektorsuche, Datenbankzugriff, Chunking, Ranking, Kompression	Ersetzt rohen Datenexport durch ausgewählten Kontext. Kann Tokens sparen, kann aber bei schlechter Indexpflege selbst teuer werden.
Modellaufruf	oft dominant: großes Modell, lange Kontexte, mehrere Schritte, Tool-Loops	Hier entsteht die eigentliche teure Kognition. Diese Schicht sollte präzise, nicht reflexhaft aktiviert werden.
Validierung	niedrig bis mittel: Regeln, Structured Outputs, Evals, zweites Modell, Human Gate	Sichert Qualität. Zu wenig Validierung spart scheinbar Energie, erzeugt aber Fehlerkosten.
Logging & Observability	niedrig, aber dauerhaft: Metriken, Traces, Audit, Kosten- und Energiezähler	Macht Lernkurven sichtbar. Ohne Logs bleibt unklar, wo Energie, Tokens und Daten wirklich fließen.

Die Tabelle ist kein universelles Prozentmodell. Dafür sind Modelle, Workloads, Hardware, Strommix und Auslastung zu unterschiedlich. Sie ist ein Denkmodell. Es verschiebt die Frage von „KI verbraucht Strom“ zu „welcher Schritt verbraucht Strom, um welchen späteren Schritt zu vermeiden?“

Genau hier wird Deterministik wieder modern. Nicht weil Regeln intelligenter wären als Modelle. Sondern weil Regeln billig genug sind, um Modelle gezielter einzusetzen.

2.17 Welche Arbeit auf Sonne warten darf

Nicht jede Rechenarbeit hat denselben Zeitwert. Eine Nutzerfrage im Chat hat ein enges Latenzfenster. Eine Indexpflege, ein Eval-Lauf oder die Vorbereitung synthetischer Testfälle kann oft warten. Energieopportunistische KI beginnt mit dieser Unterscheidung.

Google beschreibt dieses Prinzip mit carbon-intelligent computing: flexible Rechenaufgaben werden zeitlich so verschoben, dass sie eher in Stunden mit saubererer Energie fallen. [S17] Die Green Software Foundation formalisiert mit der Software Carbon Intensity eine Denkweise, bei der Energie, Carbon-Intensity, Hardwareanteile und funktionale Einheit zusammen betrachtet werden. [S16]

Tabelle 2.3 - Aufgaben, die sich besonders für lokale oder zeitverschobene Verarbeitung eignen

Aufgabe	Warum sie warten kann	Grenze
Embedding-Refresh	Neue oder geänderte Dokumente müssen oft nicht sekundengenau indiziert werden.	Kritische Dokumente brauchen Priorität und klare Aktualitätsregeln.
Indexpflege	Duplikate, Chunks, Rankings und Metadaten lassen sich in ruhigen Fenstern verbessern.	Zu seltene Pflege verschlechtert Retrieval und kann spätere Modellarbeit erhöhen.
Evals und Regressionstests	Qualitätstests können nachts, bei Solarüberschuss oder auf idle Nodes laufen.	Sicherheitskritische Tests dürfen nicht zu spät kommen.
Synthetische Testdaten	Testfälle und adversarielle Beispiele sind wertvoll, aber selten live-pflichtig.	Sie müssen klar von Produktionsdaten getrennt bleiben.
Kleine Adapter/Fine-Tunings	Spezialisierung kann asynchron vorbereitet und später geprüft werden.	Training auf sensiblen Daten braucht Provenance, Datenschutz und Validierung.
Zusammenfassungs-Caches	Wiederkehrende Dossiers können vorab verdichtet werden.	Veraltete Caches sind gefährlich; sie brauchen Ablaufdatum und Revalidierung.

Der Satz „manche Einsichten dürfen auf Sonne warten“ ist deshalb nicht nur poetisch. Er ist eine Scheduling-Regel. Sie lautet: Alles, was nicht interaktiv, nicht risikokritisch und nicht synchron ist, bekommt ein Energiezeitfenster.

2.18 Wo lokale Verarbeitung kippt

Lokale Verarbeitung wird dann stark, wenn sie Datenwege verkürzt, Offenlegung reduziert, Latenz senkt, vorhandene Hardware nutzt oder teure Frontier-Arbeit verhindert. Sie kippt, wenn sie zu einer zweiten, schlecht ausgelasteten Schatteninfrastruktur wird.

Der Kipppunkt liegt selten an einem einzigen Wert. Er entsteht aus Auslastung, Hardwareeffizienz, Strommix, Kühlung, Wartung, Qualität und Wiederholungsrate. Ein lokales System, das eine Aufgabe dreimal schlecht löst und danach doch die Cloud ruft, hat nicht gespart. Es hat eine Umleitung mit Zusatzverkehr gebaut.

Tabelle 2.4 - Kipppunkte zwischen lokal, regional und Hyperscale

Kriterium	Lokal stark, wenn ...	Hyperscale/regional stärker, wenn ...
Auslastung	Hardware ohnehin läuft oder gezielt in Überschussfenstern aktiviert wird.	lokale Geräte dauerhaft idle Strom ziehen oder nur sporadisch genutzt werden.
Daten	Rohdaten sensibel sind und lokal vorverdichtet werden können.	Daten bereits zentral liegen und strenge zentrale Governance existiert.
Qualität	kleine Modelle, Regeln oder Retrieval zuverlässig genug sind.	lokale Fehlerraten viele Wiederholungen oder spätere Korrekturen erzwingen.

Energie	lokale Erzeugung, Abwärmenutzung oder flexible Lastfenster verfügbar sind.	Hyperscale-Standort deutlich sauberer, effizienter und besser ausgelastet ist.
Komplexität	Betrieb, Updates, Security und Monitoring kontrollierbar bleiben.	Versionen, Nodes und Artefakte so heterogen werden, dass Governance zerfällt.

Das ist die nüchterne Grenze lokaler Intelligenz: Nähe ist ein Vorteil, aber kein Freifahrtschein. Lokalität muss gemessen, begrenzt und betrieben werden. Sonst wird aus Souveränität nur ein kleineres Rechenzentrum mit schlechterer Buchhaltung.

2.19 Die europäische Netzfrage

Die Energiefrage wird besonders sichtbar, sobald man sie nicht global, sondern regional betrachtet. Europa ist kein leeres Stromfeld. Netze, Genehmigungen, Standorte, Abwärmenutzung, Datenschutz und Industriepolitik greifen ineinander.

Die Europäische Kommission verweist inzwischen ausdrücklich auf Energieeffizienz und Berichtspflichten von Rechenzentren und übernimmt für die globale Größenordnung die IEA-Projektion von rund 415 TWh im Jahr 2024 und rund 945 TWh bis 2030. [S18] Für Deutschland melden Branchenquellen lange Wartezeiten bei Netzanschlüssen; die German Datacenters Association nennt für neue Rechenzentren in Deutschland bis zu sieben Jahre. [S19] Für die Schweiz läuft im Auftrag des Bundesamts für Energie eine Aktualisierung der Datenlage zu Stromverbrauch, Effizienzpotenzialen und Marktentwicklung von Rechenzentren; eine ältere Studie sah Serverräume und Rechenzentren 2019 bei rund 3,6 Prozent des Schweizer Stromverbrauchs und identifizierte erhebliche Effizienzpotenziale. [S20]

Diese Beispiele sind keine fertige europäische Fallstudie. Sie sind Marker. Sie zeigen, dass KI-Infrastruktur nicht nur in globalen TWh stattfindet, sondern in Netzanschlüssen, lokalen Genehmigungen, Abwärmeplänen, Standortentscheidungen und politischen Prioritäten. Kapitel 11 und 12 werden daraus die Macht- und Souveränitätsfrage ableiten.

2.20 Der Verbindungsfaden zu Kapitel 9 und 10

Kapitel 9 macht aus diesem Maschinenraum eine Betriebslogik: Energieopportunistische KI fragt, wann eine Aufgabe laufen darf, welche Signale der Scheduler braucht und wie Preis, Carbon-Intensity, Netzlast, lokale Erzeugung, thermisches Budget und Business-SLA gegeneinander abgewogen werden.

Kapitel 10 setzt die Gegenkante: Nicht alles lässt sich verschieben, verteilen oder lokal vorverdichten. Frontier-Training, große Modellfabriken und eng gekoppelte HPC-ähnliche Systeme bleiben auf dichte Infrastruktur angewiesen. Die These dieses Buches ist deshalb nicht: Weg mit den Datacentern. Die These ist: Weg mit dem Reflex, jedes Problem sofort in den dichtesten Rechenkern zu werfen.

Der physische Körper der KI ist real. Darum muss jede Architektur entscheiden, welche Arbeit ihn wirklich braucht.

2.21 Der Übergang zu Semantic Compression

Kapitel 5 wird später zeigen, dass Datenschutz nicht nur Verschlüsselung ist. Es geht auch darum, welche Bedeutung ein entferntes Modell überhaupt sehen darf. Kapitel 2 liefert dafür den energetischen Grund: Auch Bedeutung kostet Strom, wenn sie durch ein großes Modell verarbeitet wird.

Lange Kontexte sind teuer. Unnötige Rohdaten sind teuer. Unklare Prompts sind teuer. Wiederholte Modellaufrufe sind teuer. Fehlerhafte Agentenloops sind teuer. Jede Schicht, die Kontext reduziert, Unsicherheit kapselt oder eine unnötige Eskalation verhindert, kann zur Energiearchitektur werden.

Semantic Compression ist daher nicht nur eine Privacy-Idee. Sie ist auch eine Energietechnik. Wenn ein lokales System aus einem 40-seitigen Business Case eine strukturierte, abstrakte Problembeschreibung erzeugt, reduziert es nicht nur Offenlegung. Es reduziert unter Umständen auch Tokens, Modellzeit, Netzwerklast und Validierungsaufwand.

Natürlich gilt auch hier: Die Kompression selbst kostet Energie und kann Bedeutung verlieren. Aber das ist der Punkt. Episodische Intelligenz ist keine Parole. Es ist eine Messfrage.

2.22 Gegenargumente, die im Buch stehen müssen

Ein Buch über episodische Intelligenz wird unglaubwürdig, wenn es nur die Vorteile lokaler und verteilter Systeme beschreibt. Die Gegenargumente sind nicht Beiwerk. Sie sind Teil der Wahrheit.

- Hyperscale-Rechenzentren können pro Rechenoperation sehr effizient sein, weil sie hohe Auslastung, optimierte Kühlung und professionelle Strominfrastruktur kombinieren.
- Frontier-Training braucht enge Synchronisierung und extrem schnelle Netzwerke. Viele verteilte Kleinknoten können diese Form von Training nicht einfach ersetzen.
- Lokale Hardware kann ineffizient sein, wenn sie schlecht ausgelastet, schlecht gekühlt oder permanent aktiv ist.
- Dezentrale Systeme erhöhen Komplexität: Versionsstände, Security, Qualitätssicherung, Hardwareheterogenität und Monitoring werden schwieriger.
- Energieverschiebung ist nicht automatisch Energieeinsparung. Solarüberschuss, Lastmanagement und echte Messung müssen nachgewiesen werden.
- Preis und Emissionsintensität sind nicht dasselbe. Ein System, das nur Preise optimiert, kann Emissionen verschieben oder sogar erhöhen.
- Manche Aufgaben brauchen tatsächlich Frontier-Reasoning mit viel Kontext. Zu starke Abstraktion kann Qualität zerstören.

Gerade diese Gegenargumente machen die These stärker. Episodische Intelligenz behauptet nicht, dass zentrale Datacenter verschwinden. Es behauptet, dass ihre Rolle präziser werden muss.

2.23 Verdichtung

Die Maschine denkt nicht im Äther. Sie denkt im Strom. Das ist unbequem, aber befreiend. Denn sobald man KI als körperliches System versteht, kann man sie anders bauen.

Man kann fragen, welche Arbeit wirklich live passieren muss. Welche Bedeutung lokal entstehen kann. Welche Entscheidungen deterministisch sind. Welche Kontexte vorbereitet werden können. Welche Energiequellen verfügbar sind. Welche Anfragen nur deshalb teuer werden, weil die Architektur zu faul ist, vorher nachzudenken.

Das Ziel ist nicht, große Rechenzentren moralisch zu verdammen. Das Ziel ist, sie nicht gedankenlos als Standardnervensystem der Welt zu behandeln. Ein Frontier-Datacenter kann ein Hochofen sein, ein Labor, ein Teilchenbeschleuniger, ein seltener Reasoning-Kern. Aber ein gesundes intelligentes System braucht auch lokale Reflexe, Speicher, Filter, Rhythmen und Ruhezeiten.

Vielleicht beginnt die nachhaltigere KI nicht mit kleineren Träumen, sondern mit besseren Wegen durch den Maschinenraum.

Übergang: Energie erklärt die physische Seite der KI. Doch der Betrieb entscheidet, wann dieser Energieeinsatz überhaupt nötig wird.

Kapitel 3 - Der verborgene Maschinenraum

Warum das Modell nur ein Motor in einer größeren Maschine ist

Das Modell ist nicht die Anwendung. Das Modell ist ein Motor in einer Maschine.

Die populäre Vorstellung von KI ist erstaunlich einfach: Ein Mensch schreibt etwas in ein Textfeld, ein Modell denkt, und kurz darauf erscheint eine Antwort. Diese Vorstellung ist bequem. Sie ist auch falsch. Sie beschreibt vielleicht eine Demo. Sie beschreibt nicht den Betrieb eines ernsthaften KI-Systems.

Produktive KI ist kein einzelner Denkakt. Sie ist eine Kette aus Entscheidungen. Manche davon sind probabilistisch, viele deterministisch, einige sicherheitskritisch, andere rein organisatorisch. Bevor ein großes Modell überhaupt antwortet, hat das System oft schon entschieden, was die Anfrage ist, ob sie erlaubt ist, welche Daten relevant sind, welches Werkzeug benutzt werden darf, welches Modell angemessen ist, welche Ausgabeform erwartet wird und ob ein Mensch eingeschaltet werden muss.

Das ist der verborgene Teil der KI. Er ist weniger spektakulär als ein Frontier-Modell. Er schreibt keine Gedichte. Er gibt keine Pressekonferenz. Aber ohne ihn wird KI teuer, langsam, unsicher und schwer kontrollierbar.

Dieses Kapitel schlägt deshalb eine nüchterne These vor: Die eigentliche Produktivierung von KI entsteht nicht dort, wo ein Modell möglichst viele Tokens erzeugt. Sie entsteht dort, wo ein System lernt, wann es keine Tokens erzeugen muss.

Nicht jede Frage braucht einen Hochofen. Manchmal reicht ein Schalter.

3.1 Das falsche Bild: Prompt rein, Antwort raus

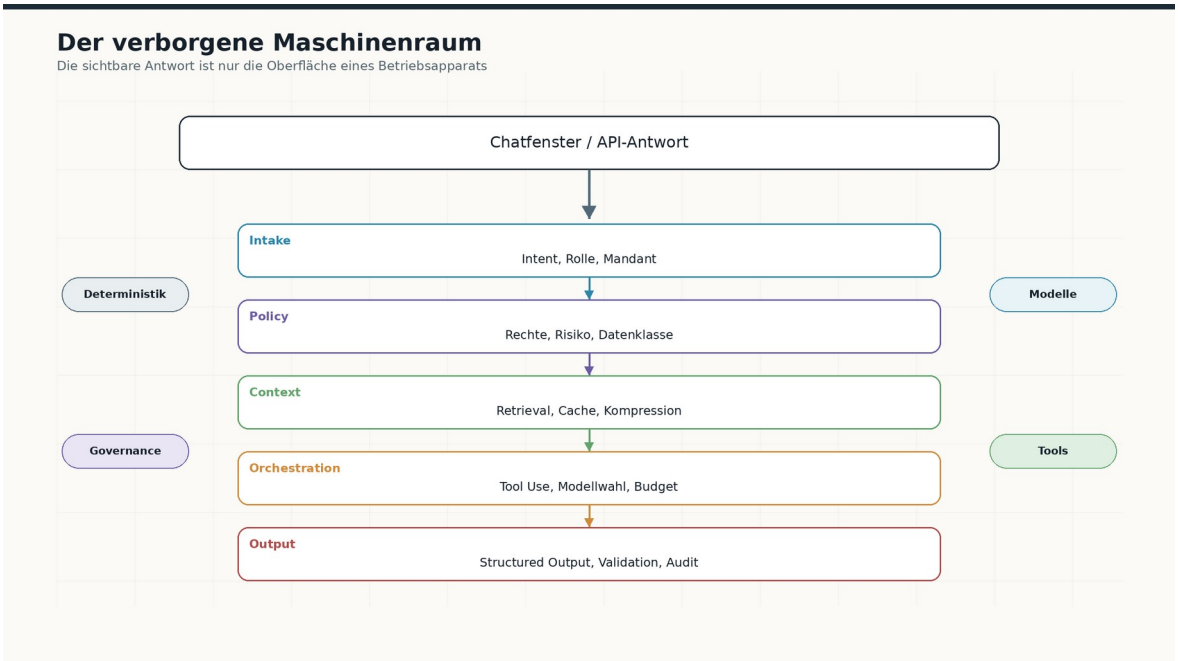

Abb. 4 - Der verborgene Maschinenraum: sichtbare Antwort und unsichtbare Betriebslogik dahinter.

Das klassische Chatbild ist eine gelungene Oberfläche, aber eine schlechte Architekturmetapher. Es suggeriert, dass Intelligenz in einem einzigen geschlossenen Raum stattfindet: Prompt hinein, Antwort heraus. In Wirklichkeit gleicht ein produktives System eher einem Flughafen als einem Orakel. Anfragen werden aufgenommen, geprüft, sortiert, geroutet, verzögert, beschleunigt, priorisiert, dokumentiert und manchmal abgewiesen.

Ein Kunde stellt eine Frage. Das System muss zuerst verstehen, ob es sich um Support, Vertragsprüfung, Buchhaltung, technische Diagnose oder eine rechtliche Eskalation handelt. Danach muss es klären, ob der Nutzer überhaupt Zugriff auf die relevanten Informationen hat. Dann wird entschieden, ob ein lokaler Cache reicht, ob ein Dokumentenindex abgefragt werden muss, ob ein Tool aufgerufen werden darf oder ob eine kleine Modellinstanz genügt. Erst wenn diese Schichten nicht ausreichen, lohnt sich der teure Schritt ins große Modell.

Anthropic beschreibt in seinen Agentenmustern genau solche Workflows: Prompt-Chaining, Routing, Parallelisierung, Orchestrator-Worker und Evaluator-Optimizer. Besonders wichtig ist der Hinweis, dass bei Prompt-Chaining programmgesteuerte Gates zwischen Zwischenschritten eingesetzt werden können, und

dass Routing Eingaben klassifiziert und an spezialisierte Folgeprozesse weiterleitet. Das ist keine Randnotiz. Das ist eine Bauanleitung für den Maschinenraum moderner KI-Anwendungen.

Damit verschiebt sich die Frage. Nicht mehr: Welches Modell ist am intelligentesten? Sondern: Welche Schicht muss diese Entscheidung treffen?

```
naives Bild:
  User -> LLM -> Antwort
produktive Realität:
  User
    -> Intake
    -> Policy Check
    -> Intent Routing
    -> Retrieval / Cache / Tool
    -> Model Call
    -> Validation
    -> Response / Action / Escalation
```

Der Unterschied zwischen beiden Bildern ist nicht kosmetisch. Im naiven Bild ist das Modell der ganze Vorgang. Im produktiven Bild ist das Modell eine Komponente in einem kontrollierten Prozess.

3.2 Die Produktionskette einer Anfrage

Eine KI-Anfrage beginnt nicht mit Denken. Sie beginnt mit Einordnung. Das System muss aus einer offenen menschlichen Äußerung eine bearbeitbare technische Situation machen. Diese Übersetzung ist der erste deterministische Akt: Aus Sprache wird Zustand.

Ein typischer Ablauf kann grob so aussehen:

```
request_state = {
  user_id, role, tenant, locale,
  intent, risk_class, data_scope,
  allowed_tools, retrieval_scope,
  model_budget, output_contract
}
```

Diese Struktur ist nicht dekorativ. Sie entscheidet, was passieren darf. Ein Support-Mitarbeiter erhält andere Daten als ein CFO. Eine öffentliche Produktfrage bekommt einen anderen Pfad als eine vertrauliche M&A-Analyse. Eine einfache FAQ-Frage kann aus dem Cache beantwortet werden. Eine strategische Entscheidung muss vielleicht durch mehrere Prüfungen und ein stärkeres Modell.

Hier entsteht ein wichtiger Übergang: KI-Systeme werden nicht zuverlässiger, indem man alle Regeln in einen Prompt schreibt. Sie werden zuverlässiger, indem man Regeln als Systemzustände, Verträge und Prüfungen implementiert. Prompts können Absichten beschreiben. Runtime-Logik setzt Grenzen.

Das ist der Grund, warum klassische Softwareprinzipien plötzlich wieder modern wirken: Rollen, Rechte, Zustände, Queues, Logs, Wiederholbarkeit, Idempotenz, Validierung. Die KI-Welle hat diese Dinge nicht abgeschafft. Sie hat sie wichtiger gemacht.

LangGraph beschreibt etwa Durable Execution als Technik, bei der Workflows ihren Fortschritt an Schlüsselpunkten speichern, pausieren und später fortsetzen können. In solchen Systemen müssen nicht-deterministische Operationen und Seiteneffekte besonders behandelt werden, damit Wiederaufnahme und

Replay kontrollierbar bleiben. Das klingt sehr weit weg vom Hype um autonome Agenten. In Wahrheit ist es genau die Art von Infrastruktur, die Agenten produktionsfähig macht.

Ein KI-System, das nicht weiß, in welchem Zustand es ist, ist kein Agent. Es ist ein Chatfenster mit Erinnerungslücken.

3.3 Routing ist eine Form von Intelligenz

Routing klingt langweilig. Es erinnert an Netzwerktechnik, Callcenter oder Paketlogistik. Genau deshalb wird es unterschätzt. In modernen KI-Systemen ist Routing jedoch eine der zentralen Formen operativer Intelligenz.

Routing beantwortet eine scheinbar kleine, aber entscheidende Frage: Was ist das hier für ein Fall? Wenn diese Frage sauber beantwortet wird, wird der Rest einfacher. Wenn sie falsch beantwortet wird, kann selbst ein starkes Modell in den falschen Kontext geraten, das falsche Werkzeug benutzen oder unnötig teuer arbeiten.

Ein gutes Routing-System trennt nicht nur Themen. Es trennt Risiko, Kosten und Unsicherheit. Es kann entscheiden, ob eine Anfrage lokal beantwortet wird, ob ein kleines Modell genügt, ob ein Retrieval-Schritt notwendig ist, ob ein Mensch prüfen muss oder ob eine Frontier-Eskalation gerechtfertigt ist.

Damit wird Routing zur Budgetentscheidung. Jedes Routing ist eine kleine Energieentscheidung. Nicht in dem Sinn, dass ein einzelner Klassifikationsschritt den Stromverbrauch der Welt verändert. Sondern in dem Sinn, dass eine Million solcher Entscheidungen bestimmen, ob ein System ständig die teuerste Rechenschicht aktiviert oder nur dann, wenn es wirklich nötig ist.

Ein mögliches Routing-Schema sieht so aus:

```
if request.intent in FAQ and cache.hit:
    answer_from_cache()
elif request.intent in simple_ops:
    call_small_model_or_tool()
elif request.risk_class == high:
    require_human_gate()
elif request.uncertainty > threshold:
    escalate_to_frontier()
else:
    answer_with_local_runtime()
```

Das ist keine Anti-KI-Haltung. Im Gegenteil. Es ist Respekt vor dem Modell. Man setzt es dort ein, wo seine Stärke liegt: Ambiguität, Generalisierung, neuartige Probleme, sprachliche Nuancen, strategische Muster. Man verschwendet es nicht an Dinge, die ein deterministischer Prozess besser, billiger und überprüfbarer erledigt.

Die Anthropic-Agentenmuster nennen Routing explizit als Workflow für komplexe Aufgaben mit unterscheidbaren Kategorien, die besser getrennt behandelt werden. Das ist ein technisches Muster, aber auch ein strategischer Gedanke: Generalität ist wertvoll, aber Spezialisierung ist effizient.

3.4 Retrieval: das Gedächtnis außerhalb des Modells

Retrieval ist die erste große Korrektur am Mythos des allwissenden Modells. Ein Sprachmodell muss nicht alles in seinen Parametern tragen. Es kann externe Speicher nutzen: Dokumente, Vektordatenbanken, Wissensgraphen, Logsysteme, Tabellen, Code-Repositories, Handbücher, Verträge, Tickets, Mails, technische Spezifikationen.

Das RAG-Paper von Lewis et al. formulierte diese Verschiebung früh und präzise: Modelle kombinieren parametrisches Gedächtnis mit nicht-parametrischem Gedächtnis, etwa einem dichten Index. Entscheidend ist nicht nur, dass dadurch Antworten faktischer werden können. Entscheidend ist, dass Wissen aktualisierbar wird, ohne das ganze Modell neu zu trainieren.

Für episodische Intelligenz ist Retrieval noch aus einem zweiten Grund wichtig. Es verschiebt Gedächtnis dorthin, wo es hingehört. Unternehmenswissen muss nicht vollständig in ein Frontier-Modell wandern. Es kann lokal oder regional liegen, verschlüsselt, berechtigt, auditierbar. Das Modell bekommt nur die relevanten Ausschnitte - oder, in der späteren Semantic-Compression-Variante, nur eine abstrahierte Struktur daraus.

Retrieval ist aber kein Zauberstab. Schlechter Retrieval-Kontext kann eine Antwort schlechter machen als gar kein Kontext. Falsche Dokumente, alte Versionen, überfüllte Kontextfenster, unklare Berechtigungen und Prompt-Injection in externen Inhalten können das System beschädigen. Ein RAG-System ist daher nicht einfach ein Suchfeld vor einem Modell. Es ist ein Governance-System für Wissen.

Gutes Retrieval muss mindestens vier Fragen beantworten:

- Darf dieser Nutzer diese Information sehen?
- Ist diese Information aktuell genug für die Entscheidung?
- Ist sie relevant genug, um Tokenbudget zu verbrauchen?
- Kann sie sicher an das Modell übergeben werden?

Wieder sehen wir: Der wertvolle Teil liegt nicht nur im Generieren. Er liegt im Auswählen.

3.5 Cache: das System muss nicht alles neu denken

Ein produktives System erkennt Wiederholung. Menschen stellen ähnliche Fragen. Prozesse erzeugen ähnliche Zwischenschritte. Systemprompts, Tooldefinitionen, Rollenregeln und Formatvorgaben wiederholen sich. Wenn jedes dieser Elemente immer wieder neu durch die teuerste Inferenz läuft, verbrennt das System Geld, Zeit und Energie.

Prompt Caching ist deshalb mehr als eine Preistabelle. OpenAI beschreibt, dass wiederkehrende lange Prompt-Präfixe genutzt werden können, um Latenz und Kosten zu reduzieren. Das technische Detail ist hier weniger wichtig als das Prinzip: Wiederholung ist eine architektonische Ressource.

Caching gibt es auf mehreren Ebenen:

- Response Cache: dieselbe oder sehr ähnliche Frage wurde bereits beantwortet.
- Retrieval Cache: dieselbe Dokumentenmenge wurde bereits gesucht und gerankt.
- Embedding Cache: dieselben Texte wurden bereits vektorisiert.
- Prompt Prefix Cache: statische Instruktionen und Tooldefinitionen bleiben gleich.
- Decision Cache: ein Routing- oder Policy-Ergebnis kann wiederverwendet werden.

Jede dieser Ebenen spart nicht nur Zeit. Sie reduziert die Wahrscheinlichkeit, dass ein großes Modell unnötig arbeitet. Das ist der unscheinbare Bruder der Energieopportunität: nicht nur rechnen, wenn Strom günstig ist, sondern gar nicht rechnen, wenn das Ergebnis bereits bekannt oder deterministisch ableitbar ist.

Natürlich darf Caching nicht blind sein. Veraltete Antworten, falsche Berechtigungen oder situationsabhängige Daten können gefährlich werden. Deshalb braucht jeder Cache einen Vertrag: Gültigkeit, Scope, Nutzerkontext, Datenversion, Ablaufzeit, Widerruf. Ohne solche Regeln wird Effizienz zur Fehlerquelle.

Ein gutes KI-System denkt nicht weniger, weil es dumm ist. Es denkt weniger, weil es sich erinnert.

3.6 Contracts: wenn Sprache wieder Maschine werden muss

Sprachmodelle sind stark, weil sie offene Sprache verarbeiten. Produktionssysteme brauchen aber geschlossene Formen. Eine Datenbank kann mit einer poetischen Antwort wenig anfangen. Ein Workflow

braucht Felder, Typen, Zustände, Enums, IDs und Entscheidungspunkte. Darum muss die offene Sprache des Modells am Ende wieder in maschinenlesbare Verträge übersetzt werden.

Structured Outputs sind ein Beispiel für diesen Übergang. OpenAI beschreibt die Funktion als Möglichkeit, Antworten an ein vorgegebenes JSON-Schema zu binden. Das Ziel ist nicht nur hübscheres JSON. Das Ziel ist, dass ein nachgelagertes System prüfen kann, ob eine Ausgabe die erwartete Form hat.

Damit entsteht eine wichtige Architekturregel: Je näher ein LLM an eine Aktion kommt, desto formaler muss seine Ausgabe werden. Ein freier Text ist gut für Erklärung. Ein Vertrag ist nötig für Ausführung.

Ein Contract kann etwa so aussehen:

```
{
  "decision": "escalate" | "answer" | "reject",
  "risk_class": "low" | "medium" | "high",
  "confidence": 0.0-1.0,
  "required_sources": ["contract", "policy"],
  "human_review_required": true | false,
  "reason_summary": "short explanation"
}
```

Der Contract ist eine Brücke zwischen Wahrscheinlichkeit und Kontrolle. Das Modell darf unsicher sein. Das System muss wissen, was es mit Unsicherheit macht.

In vielen Anwendungen wird genau diese Schicht unterschätzt. Man bittet das Modell, sich an ein Format zu halten, und hofft, dass es klappt. Hoffnung ist aber kein Interface. Ein gutes System validiert, verwirft, wiederholt, eskaliert oder reduziert den Scope. Aus einem Modelloutput wird erst dann ein Prozessschritt, wenn er einen Vertrag erfüllt.

3.7 Gates: Intelligenz braucht Bremsen

Gates sind die Stellen, an denen ein System sagt: bis hierhin und nicht weiter. Sie sind die Bremsen, Schleusen und Prüfpunkte eines KI-Workflows. Ohne Gates wird aus einem hilfreichen Modell schnell ein zu selbstbewusster Praktikant mit Root-Zugriff.

Ein Gate kann simpel sein: Ist das JSON gültig? Enthält die Antwort eine Quelle? Liegt die Konfidenz über einem Schwellenwert? Ist die angefragte Aktion für diese Rolle erlaubt? Wurde ein kritisches Wortfeld erkannt? Muss ein Mensch zustimmen?

Gates können deterministisch sein, modellbasiert oder hybrid. Die deterministischen Gates sind dabei besonders wertvoll, weil sie hart sind. Sie diskutieren nicht. Wenn ein Tool nicht erlaubt ist, ist es nicht erlaubt. Wenn ein Betrag über der Freigabegrenze liegt, wird eskaliert. Wenn ein Dokument außerhalb des Berechtigungsbereichs liegt, wird es nicht in den Kontext aufgenommen.

Das ist ein entscheidender Unterschied zwischen Modellkontrolle und Systemkontrolle. Modellkontrolle versucht, das Modell durch Instruktionen brav zu machen. Systemkontrolle begrenzt, was auch ein fehlgeleitetes Modell tatsächlich tun kann.

OWASP listet Prompt Injection, unsichere Output-Verarbeitung, sensible Informationsweitergabe, unsicheres Plugin-Design und excessive agency als zentrale Risiken von LLM-Anwendungen. Das sind keine rein sprachlichen Probleme. Es sind Systemprobleme. Ihre Antwort liegt nicht nur in besseren Prompts, sondern in besseren Grenzen.

```
LLM output
  -> schema valid?
  -> policy valid?
```

```
-> source valid?
-> permission valid?
-> budget valid?
-> action allowed?
-> commit or escalate
```

Der Gatekeeper ist nicht der Feind der Intelligenz. Er ist ihre Haftpflichtversicherung.

3.8 Tool Use: wenn das Modell in die Welt greift

Solange ein Modell nur Text erzeugt, bleiben Fehler oft im Bereich der Sprache. Sobald es Tools nutzt, wird es operativ. Es kann suchen, rechnen, Dateien lesen, Tickets anlegen, Code ausführen, Zahlungen vorbereiten, Kalender verändern oder Datenbanken abfragen. Dann ist nicht mehr nur die Antwort relevant, sondern die Wirkung.

ReAct zeigte früh, wie Reasoning und Acting verschränkt werden können: Modelle erzeugen nicht nur Gedankenschritte, sondern auch Aktionen, etwa Abfragen externer Quellen. Toolformer untersuchte, wie Sprachmodelle lernen können, APIs aufzurufen, Argumente zu wählen und Ergebnisse in weitere Vorhersagen einzubauen. MRKL-Systeme gingen noch grundsätzlicher vor und beschrieben eine modulare neuro-symbolische Architektur mit Sprachmodellen, Wissensquellen und diskreten Reasoning-Modulen.

Alle drei Linien stützen dieselbe Richtung: Das Modell ist nicht mehr nur Generator. Es wird Koordinator. Aber genau dadurch steigt die Notwendigkeit von deterministischen Rändern. Ein Toolaufruf muss autorisiert, parametrisiert, protokolliert und begrenzt werden.

MCP ist ein aktuelles Beispiel für diese Verschiebung. Das Model Context Protocol beschreibt sich als offener Standard, mit dem KI-Anwendungen an externe Systeme angebunden werden können: Datenquellen, Tools und Workflows. In der Spezifikation erscheinen Ressourcen, Prompts und Tools als zentrale Server-Fähigkeiten. Das ist infrastrukturell wichtig, weil Toolzugriff standardisiert wird. Es ist sicherheitstechnisch heikel, weil standardisierter Zugriff auch standardisierte Angriffsflächen erzeugt.

Sobald ein Modell handeln darf, reicht Vertrauen nicht mehr. Dann braucht es Capability-Design: Welche Aktion ist möglich? Mit welchen Parametern? In welchem Kontext? Mit welcher Nutzerzustimmung? Mit welchem Audit-Log? Mit welchem Undo?

Die Frage lautet nicht: Kann das Modell das Tool benutzen? Die Frage lautet: Sollte es dieses Tool in diesem Zustand überhaupt sehen?

3.9 Prompt Injection zeigt die Grenze der reinen Prompt-Kontrolle

Prompt Injection ist der Moment, in dem das alte Softwaredenken an seine Grenze stößt. In klassischen Systemen trennt man Daten und Befehle. Eine SQL-Injection ist gefährlich, weil Benutzereingaben als Befehle interpretiert werden können. Die Abwehr besteht darin, diese Grenze technisch zu erzwingen.

Bei LLMs ist diese Grenze viel weicher. Das britische National Cyber Security Centre formulierte es 2025 scharf: Unter der Haube eines LLM gebe es keine eingebaute Unterscheidung zwischen Daten und Instruktionen, sondern nur die Vorhersage des nächsten Tokens. Daraus folgt keine Panik, aber eine unbequeme Architekturwahrheit: Man darf die Sicherheit eines Systems nicht allein darauf stützen, dass das Modell schon unterscheiden werde, welche Texte Befehle sind und welche nur Inhalte.

Das ist für episodische Intelligenz zentral. Je mehr externe Inhalte in ein Modell wandern - E-Mails, Webseiten, Tickets, Dokumente, Chatverläufe -, desto stärker wird die Trennung zwischen vertrauenswürdiger Instruktion und untrusted content zur Systemaufgabe. Das Modell kann dabei helfen. Aber die harten Grenzen müssen außerhalb des Modells liegen.

Ein sicherer Produktionspfad muss daher mindestens unterscheiden zwischen:

- Systeminstruktionen, die vom Betreiber kommen.

- Nutzerabsichten, die vom aktuellen User kommen.
- Dateninhalten, die nur gelesen werden sollen.
- Toolergebnissen, die nicht automatisch vertrauenswürdig sind.
- Aktionen, die reale Folgen haben.

Der wichtigste Punkt: Prompt Injection ist kein Grund, KI nicht einzusetzen. Es ist ein Grund, KI nicht naiv einzusetzen. Das Buch sollte hier hart bleiben. Wer einem LLM ungeprüft Werkzeuge, Daten und Privilegien gibt, baut keinen Agenten. Er baut eine offene Tür mit Charme.

3.10 Context Engineering: das Arbeitsgedächtnis der Maschine

Prompt Engineering war der erste Reflex. Context Engineering ist der reifere Begriff. Es geht nicht mehr nur darum, die richtigen Worte in eine Systemnachricht zu schreiben. Es geht darum, den gesamten Zustand zu kuratieren, den ein Modell für eine Entscheidung sieht: Systemanweisungen, Tools, Rollen, Quellen, Memory, Zwischenergebnisse, Constraints und Outputverträge.

Anthropic beschreibt Context Engineering als die Kunst und Wissenschaft, die Tokens zu kuratieren, die in das begrenzte Kontextfenster eines Modells gelangen. Wichtig ist dabei der Hinweis, dass Kontext eine endliche Ressource ist. Mehr Kontext ist nicht automatisch besser. Mehr Kontext kann auch mehr Ablenkung, mehr Kosten, mehr Angriffsfläche und mehr Unschärfe bedeuten.

Das passt direkt zu unserer These aus Kapitel 2. Tokens sind nicht nur Text. Tokens sind Rechenereignisse. Ein schlechter Kontext ist daher doppelt teuer: Er kostet Energie und verschlechtert die Entscheidung.

Context Engineering ist damit die operative Schwester von Semantic Compression. Es fragt: Was muss das Modell sehen? Semantic Compression fragt zusätzlich: In welcher abstrahierten Form darf es das sehen?

```
raw universe of possible context
  -> permissions filter
  -> relevance ranking
  -> risk filter
  -> compression / summarization
  -> output contract
  -> model context
```

Ein gutes KI-System ist nicht nur ein starkes Modell. Es ist ein Kurator von Aufmerksamkeit.

3.11 Der deterministische Layer ist das Betriebssystem der KI

Wenn man all diese Bausteine zusammenlegt - Routing, Retrieval, Caching, Contracts, Gates, Toolkontrolle, Kontextmanagement -, entsteht ein Bild: Vor und hinter dem Modell liegt ein Betriebssystem. Nicht im engeren Sinn wie Linux oder macOS. Sondern als Runtime, die Ressourcen verwaltet, Zustände hält, Rechte prüft, Prozesse plant und Fehler begrenzt.

Dieser Layer ist oft unsichtbar, weil er nicht nach KI aussieht. Er besteht aus Datenbanken, Queues, Services, Policy-Engines, Schema-Validatoren, Access-Control-Listen, Telemetrie, Logs, Feature Flags, Kostenbudgets und Retry-Strategien. Genau deshalb ist er wichtig. Er holt die probabilistische Kraft des Modells in eine Welt, in der Unternehmen Rechenschaft, Wiederholbarkeit und Betriebssicherheit brauchen.

Man kann diesen Layer als Local Cognitive Runtime beschreiben, wenn er nah am Nutzer oder im Unternehmen liegt. Er kann lokal entscheiden, was überhaupt an eine Cloud geht. Er kann private Rohdaten zurückhalten. Er kann wiederkehrende Arbeit vorberechnen. Er kann abstrakte Anfragen erzeugen. Er kann lokale Modelle und deterministische Regeln kombinieren. Erst danach wird entschieden, ob eine Frontier-Eskalation nötig ist.

Hier entsteht der direkte Übergang zu Episodische Intelligenz. Die Intelligenz verteilt sich:

```
lokal:
```

```
  Identität, Kontext, Rechte, Daten, Routing
regional / schwarm:
  Indexierung, Evals, Adapter, Wissenspflege
frontier:
  schwieriges Reasoning, Generalisierung, neue Muster
```

Das Modell bleibt wichtig. Aber es wird nicht mehr als monolithisches Zentrum verstanden. Es wird als episodisch aktivierter Hochleistungsbaustein in einem größeren System verstanden.

3.12 Das Effizienzparadox

Wer produktive KI baut, erlebt ein merkwürdiges Paradox. Man beginnt mit dem Ziel, ein Modell zu verwenden. Dann verbringt man einen großen Teil der Arbeit damit, Modellaufrufe zu vermeiden.

Das klingt widersprüchlich, ist aber ökonomisch logisch. Jeder vermiedene große Modellaufruf spart Kosten, Latenz, Energie, Fehlerrisiko und Kontextverschmutzung. Jede Anfrage, die lokal geroutet, aus dem Cache beantwortet oder durch ein kleines Modell gelöst werden kann, macht die Frontier-Schicht wertvoller, weil sie für die wirklich schwierigen Fälle frei bleibt.

Das bedeutet nicht, dass das deterministische System kostenlos ist. Auch Routing, Retrieval, Logs, Datenbanken und lokale Modelle verbrauchen Energie. Aber sie verbrauchen sie anders: kleinteiliger, besser planbar, oft lokal, oft cachebar, oft asynchron. Viele dieser Aufgaben müssen nicht in Echtzeit in einem zentralen Hochleistungsdatacenter passieren.

Hier taucht wieder die Energiefrage aus Kapitel 2 auf. Die beste Kilowattstunde ist nicht immer die billigste. Manchmal ist es die, die gar nicht erst gebraucht wird. Und die beste Frontier-Inferenz ist manchmal die, die ein gutes System nicht anfordern musste.

Ein intelligentes System maximiert nicht Modellnutzung. Es minimiert unnötige Modellnutzung.

3.13 Gegenargument: Wird das System dadurch nicht viel komplexer?

Ja. Und das ist das stärkste Gegenargument gegen diese Architektur. Ein einfaches Chat-System ist leicht zu erklären. Ein Hybrid-System mit Gates, Routern, lokalen Layern, Retrieval, Caches, Toolrechten, Human-in-the-loop und Frontier-Eskalation ist komplexer.

Diese Komplexität ist real. Sie erzeugt neue Fehlerquellen: falsches Routing, veraltete Caches, überstrenge Gates, blinde Policy-Regeln, schlechte Retrieval-Rankings, zu viele Tools, unklare Zustände, fehlende Observability. Ein schlecht gebauter deterministischer Layer ist kein Schutz. Er ist Bürokratie mit Stromanschluss.

Das Gegenargument muss im Buch ernst genommen werden. Episodische Intelligenz darf nicht behaupten, dass lokale und deterministische Schichten automatisch besser sind. Sie sind nur dann besser, wenn sie wartbar, beobachtbar und überprüfbar sind. Sonst verlagert man das Problem nur vom Modell in den Maschinenraum.

Deshalb braucht jedes solche System mindestens:

- Tracing: Welche Entscheidung wurde warum getroffen?
- Evaluation: Hat der Pfad bessere Ergebnisse erzeugt als ein direkter Modellaufruf?
- Rollback: Kann eine falsche Aktion zurückgenommen oder gestoppt werden?
- Versionierung: Welche Regeln, Prompts, Indizes und Modelle waren aktiv?
- Budgetierung: Welche Kosten und Latenzen verursacht welcher Pfad?
- Security Design: Welche Privilegien hat welcher Schritt?

Das ist nicht weniger Engineering. Es ist mehr Engineering. Aber es ist die Art von Engineering, die entsteht, wenn eine Technologie vom Experiment in den Betrieb wandert.

3.14 Ein durchgehender Fall: die Vertragsfrage

Ein Kapitel über Routing, Retrieval und Gates bleibt zu abstrakt, solange keine konkrete Anfrage durch die Maschine läuft. Nehmen wir deshalb einen einfachen Satz aus einem Unternehmen:

„Fasse mir die wichtigsten Risiken aus diesem Lieferantenvertrag zusammen und sage mir, ob wir bestellen können."

Ein naives System schickt Vertrag, Frage und vielleicht noch ein paar interne Richtlinien direkt an ein großes Modell. Eine produktive Runtime macht das Gegenteil. Sie zerlegt die Frage, bevor sie denkt. Denn in diesem einen Satz stecken mindestens vier verschiedene Aufgaben: Vertragsanalyse, Rollenprüfung, Beschaffungslogik und mögliche Entscheidungsvorbereitung.

Tabelle 3.1 — Der Weg einer Vertragsfrage durch die Runtime

Schicht	Entscheidungsfrage	Artefakt	Warum es zählt
Intake	Was will der Nutzer wirklich?	intent = supplier_contract_risk	Ohne Einordnung wird jede Anfrage zum allgemeinen Prompt.
Identity / Role	Darf diese Person diese Vertragsdaten sehen?	role_scope = purchasing_manager	Datenschutz und Geschäftsgeheimnisse beginnen vor dem Modell.
Data Scope	Welche Dokumentteile dürfen in den Kontext?	allowed = clauses, delivery terms; blocked = personal notes, negotiation history	Nicht jeder relevante Text darf offengelegt werden.
Retrieval	Welche Dokumente sind aktuell und verbindlich?	latest_contract, supplier_policy, risk_matrix	Falscher Kontext erzeugt falsche Autorität.
Context Assembly	Wie viel Kontext passt in die Aufgabe?	bounded_context_pack	Kontext ist Budget, Risiko und Aufmerksamkeit zugleich.
Model / Tool	Reicht ein lokales Modell oder braucht es Eskalation?	small_model + deterministic checks; frontier only if conflict	Das Modell wird gewählt, nicht automatisch aufgerufen.
Output Contract	Welche Form muss das Ergebnis haben?	risk_labels, evidence, open_questions, no_final_legal_decision	Sprache wird wieder maschinenlesbar.
Validation / Gate	Sind Quellen, Rollen und Format korrekt?	PASS / FAIL / ESCALATE	Das System entscheidet, ob die Antwort freigegeben wird.
Audit / Cache	Was muss nachvollziehbar bleiben?	decision_trace without raw secret dump	Nachvollziehbarkeit darf kein neues Datenleck werden.

Der wichtige Punkt ist nicht, dass jede Firma genau diese Pipeline braucht. Der Punkt ist: Eine produktive KI-Anwendung muss die Frage in kontrollierbare Zwischenzustände verwandeln. Erst dann wird sichtbar, welcher Teil wirklich Modellarbeit ist und welcher Teil klassisches Systemdesign.

3.15 Observability: die Maschine muss sich selbst sehen

In klassischen Softwaresystemen ist Observability die Fähigkeit, aus Logs, Metriken und Traces zu verstehen, was ein System tut. Bei KI-Systemen reicht das nicht mehr. Man muss nicht nur wissen, dass ein API-Call erfolgreich war. Man muss wissen, warum das System diesen Weg gewählt hat.

Dafür braucht episodische Intelligenz eine sparsame, aber konsequente Telemetrie. Sparsam, weil vollständige Prompts, Dokumente und Antworten selbst sensible Daten enthalten können. Konsequent, weil ohne Spuren niemand unterscheiden kann, ob ein Fehler im Routing, im Retrieval, im Modell, im Tool oder im Gate entstanden ist.

Tabelle 3.2 — Minimal-Telemetrie für eine KI-Runtime

Messpunkt	Minimum	Nicht protokollieren	Nutzen
Routing	intent, risk_class, chosen_path	vollständige private Anfrage	zeigt, ob das System unnötig eskaliert
Retrieval	Dokument-IDs, Versionen, Scores, Filter	vollständige Dokumentinhalte	macht veralteten oder falschen Kontext sichtbar
Modellaufruf	Modellklasse, Token, Latenz,	Rohgeheimnisse im Prompt	verbindet Qualität mit

	Kostenklasse		Energie- und Kostenbudget
Tool Call	Toolname, Berechtigung, Parameterklasse, Ergebnisstatus	Credentials, vollständige Datenbankantworten	begrenzt Excessive Agency und Fehlerketten
Gate	PASS/FAIL/ESCALATE plus Grundklasse	interne vertrauliche Bewertungsdetails	macht Governance prüfbar
Human Gate	Entscheidung, Rolle, Zeitpunkt, Begründungsklasse	private Freitextnotizen ohne Zweck	verhindert bloßes Abnicken

OpenTelemetry arbeitet an semantischen Konventionen für generative KI und agentische Systeme. Das ist ein wichtiges Signal: Die Industrie beginnt, LLM-Aufrufe, Tool-Schritte, Agentenpfade und Tokenverbrauch nicht mehr als unsichtbare Magie zu behandeln, sondern als beobachtbare Operationen.

Die Gefahr liegt auf der anderen Seite. Wer alles protokolliert, erzeugt ein zweites Schattenarchiv der sensibelsten Unternehmensdaten. Gute Observability ist deshalb nicht „alles speichern“. Gute Observability ist: genau genug sehen, um zu steuern, und wenig genug speichern, um nicht selbst zum Risiko zu werden.

3.16 Routing ist Sicherheits- und Energieentscheidung zugleich

Routing wird oft nur als Kostenoptimierung erzählt: kleines Modell zuerst, großes Modell nur bei Bedarf. Das ist richtig, aber zu eng. In produktiven Systemen entscheidet Routing gleichzeitig über Sicherheit, Qualität, Offenlegung, Energie und Verantwortung.

```
route(request):
  classify intent, risk, data_scope, urgency
  if forbidden_data or forbidden_action: reject_or_human_gate()
  elif cache_valid and risk_low: answer_from_cache()
  elif local_context_sufficient: answer_with_local_runtime()
  elif uncertainty_high and data_can_be_compressed:
semantic_compression_then_frontier()
  elif risk_high: human_gate()
  else: bounded_frontier_call()
```

Diese kleine Logik ist unspektakulär. Aber sie ist genau der Ort, an dem episodische Intelligenz praktisch wird. Das Modell denkt nicht weniger, weil man ihm misstraut. Es denkt seltener, weil das System vorher klärt, ob Denken an dieser Stelle überhaupt der richtige Prozess ist.

3.17 Frameworks sind Gerüste, keine Architektur

Frameworks wie LangGraph, MCP, Structured Outputs oder Tool-Calling sind nützlich, weil sie wiederkehrende Muster standardisieren: Zustände, Unterbrechungen, Tool-Schnittstellen, Schema-Ausgaben, Wiederaufnahme von Workflows. Sie ersetzen aber nicht die Architekturentscheidung.

Ein Tool-Protokoll sagt nicht, welches Tool in welchem Risikokontext erlaubt ist. Ein JSON-Schema sagt nicht, ob der Inhalt fachlich richtig ist. Ein Graph sagt nicht, ob der Übergang verantwortbar ist. Ein Cache sagt nicht, ob eine Antwort noch gültig ist. Die Runtime muss diese Fragen explizit beantworten.

Die knappe Formel lautet: Frameworks liefern die Schienen. Governance entscheidet, welche Züge fahren dürfen. Observability zeigt, ob sie pünktlich, sicher und mit angemessenem Energieeinsatz fahren. Das Modell ist in diesem Bild nicht kleiner geworden. Aber es steht endlich an seinem richtigen Platz.

3.18 Übergang zu Semantic Compression

Kapitel 2 hat gezeigt, dass KI einen Körper hat: Strom, Wärme, Kühlung, Netzwerk, Infrastruktur. Dieses Kapitel hat gezeigt, dass KI auch ein Betriebssystem braucht: Routing, Retrieval, Gates, Contracts, Caches und Toolgrenzen.

Damit ist die Bühne für Kapitel 5 vorbereitet. Wenn der lokale Layer ohnehin entscheiden muss, welche Daten relevant sind, welche Rechte gelten und welche Ausgabe erwartet wird, dann kann er noch etwas Zusätzliches tun: Er kann Rohrealität in eine abstrahierte Aufgabe verwandeln.

Genau das ist Semantic Compression. Nicht nur weniger Tokens. Nicht nur Datenschutz durch Namensentfernung. Sondern die systematische Frage: Welche Struktur eines Problems muss eine Frontier-Schicht kennen, ohne den konkreten Business Case zu sehen?

Der verborgene Teil der KI ist damit nicht nur Betrieb. Er ist Vorentscheidung darüber, welche Welt die Maschine überhaupt sehen darf.

Architektur-Skizze: Der verborgene Layer

```
User / Business Process
        |
        v
Intake + Identity + Permissions
        |
        v
Intent Routing + Risk Classification
        |
        +--> Cache / FAQ / deterministic answer
        |
        +--> Tool workflow / database action
        |
        +--> Retrieval + context curation
        |
        v
Compression + Output Contract
        |
        v
Small model / local model / frontier model
        |
        v
Validation Gates + Human Review
        |
        v
Response / Action / Audit Log
```

Kernaussagen des Kapitels

- Produktive KI ist kein einzelner Modellaufruf, sondern eine Kette aus Systementscheidungen.
- Routing, Retrieval, Caching, Contracts und Gates bestimmen Kosten, Sicherheit und Qualität.
- Deterministische Schichten sind keine Rückkehr zur alten IT, sondern die Betriebsform moderner KI.
- Toolzugriff macht Modelle operativ und erhöht deshalb die Bedeutung harter Grenzen.

- Context Engineering behandelt Aufmerksamkeit als knappe Ressource.
- Die Frontier-Schicht wird wertvoller, wenn sie nur bei echter Unsicherheit aktiviert wird.
- Semantic Compression ist die nächste Stufe dieses verborgenen Layers.

Teil II - Der lokale Körper

Wenn KI einen Körper hat, muss nicht jede Bewegung im Zentrum stattfinden. Teil II fragt, welche Arbeit näher an Daten, Geräten, Rollen und Verantwortung liegen kann - und wie Bedeutung geschützt wird, bevor sie zur Frontier wandert.

Kapitel 4 - Der Client kehrt zurück

Edge, lokale Runtime und Verantwortung nahe an Daten und Energie

Die nächste KI-Welle wird nicht nur in größeren Rechenzentren stattfinden. Sie wird auch auf Geräten stattfinden, die schon vor uns stehen: Laptops, Telefone, Edge-Server, lokale Cluster, Maschinen, Kameras, Router, Werkstätten und Büro-Rechner.

Kapitel 2 hat gezeigt: KI hat einen Körper. Sie verbraucht Strom, erzeugt Wärme und braucht Infrastruktur. Kapitel 3 hat gezeigt: Das Modell ist nicht die Anwendung. Es ist ein Motor in einer längeren Maschine. Kapitel 5 zeigt später: Nicht jeder Business-Kontext darf oder muss zur Frontier. Zwischen diesen Gedanken liegt ein scheinbar alter, aber plötzlich wieder moderner Ort: der lokale Rechner.

Lange wirkte der lokale Rechner wie ein Auslaufmodell. Alles wanderte in die Cloud. Speicher, Anwendungen, Zusammenarbeit, Datenbanken, Backups, sogar Identität. Der Client wurde dünn. Er zeigte an, was anderswo geschah. Er war Fenster, nicht Werkbank.

Generative KI hat dieses Muster zunächst noch verstärkt. Die großen Modelle liefen in riesigen Rechenzentren. Der Nutzer schrieb in ein Chatfenster. Die eigentliche Arbeit passierte irgendwo fern der eigenen Steckdose. Die Antwort kam zurück wie Wetter aus einer fernen Wolke.

Aber diese Wolke bekommt Risse. Nicht weil Cloud falsch wäre. Sondern weil jedes zentrale System irgendwann an Grenzen stößt: Energie, Latenz, Datenschutz, Kosten, Netzlast, Souveränität, regulatorische Verantwortung und schlichte Betriebsrealität.

Deshalb kehren lokale Systeme zurück. Nicht als nostalgischer Rückzug in den Keller. Nicht als Anti-Cloud-Romantik. Sondern als notwendige zweite Schicht einer reiferen KI-Architektur.

Der Client wird wieder ein Ort der Bedeutung. Nicht nur ein Bildschirm.

4.1 Zentralisierung ist nie das Ende der Geschichte

Die Geschichte der Informatik pendelt. Sie kennt Phasen der Zentralisierung und Phasen der Verteilung. Mainframes, Terminals, Personal Computer, Client-Server, Web, Cloud, Edge. Jede Epoche erklärt sich gern zur letzten. Keine ist es.

Zentralisierung bringt enorme Vorteile: gemeinsame Infrastruktur, einfache Wartung, bessere Auslastung, schnelle Updates, Sicherheitskontrolle, Skaleneffekte. Ohne zentrale Cloud-Infrastruktur gäbe es die heutige KI-Landschaft nicht in dieser Form. Frontier-Training, globale APIs, multimodale Modelle und schnelle Iteration brauchen große, koordinierte Systeme.

Aber Zentralisierung erzeugt auch neue Kosten. Sie macht aus jeder Anfrage einen Transportvorgang. Daten müssen hochgeladen, verarbeitet, gespeichert, kontrolliert, geloggt und wieder zurückgeschickt werden. Je sensibler und häufiger diese Anfragen werden, desto stärker fällt auf: Nicht jede Operation verdient eine Reise ins Rechenzentrum.

Das ist der Punkt, an dem lokale Systeme wieder interessant werden. Nicht weil sie stärker sind als die Frontier. Sondern weil viele Aufgaben gar keine Frontier brauchen.

Eine lokale Runtime kann einfache Dinge sofort entscheiden. Sie kann privaten Kontext halten. Sie kann Embeddings berechnen. Sie kann Dokumente vorbereiten. Sie kann Intent erkennen. Sie kann einen Cache befragen. Sie kann prüfen, ob ein Tool-Aufruf erlaubt ist. Sie kann eine Anfrage so weit reduzieren, dass nur noch der wirklich schwierige Rest übrig bleibt.

```
Version 1: Thin Client -> Cloud macht alles

Version 2: Client kann einzelne KI-Funktionen lokal ausführen

Version 3: Client wird lokale kognitive Runtime

Nicht: Cloud oder lokal
```

```
Sondern: Welche Schicht gehört wohin?
```

4.2 Warum die Edge zurückkommt

Edge Computing ist kein neues Konzept. Es beschreibt im Kern, Rechenarbeit näher an die Quelle der Daten zu bringen, um Latenz und Bandbreite zu reduzieren. Im KI-Kontext bekommt dieser Gedanke neue Schärfe, weil Daten nicht mehr nur Dateien sind. Sie sind Kontext: persönliche Historie, interne Dokumente, laufende Prozesse, Sensorwerte, Kalender, Mails, Tickets, Code, Vertriebsnotizen, medizinische Bilder, Produktionsdaten.

Je näher die Verarbeitung an diesen Daten bleibt, desto weniger Rohmaterial muss durch Netze wandern. Das kann Latenz verringern, Bandbreite sparen, Datenschutz verbessern und Betrieb auch dann ermöglichen, wenn die Verbindung instabil ist. Genau diese Vorteile nennen klassische Edge-Erklärungen seit Jahren. Mit generativer KI werden sie strategisch.

Der Unterschied ist: Früher ging es an der Edge oft um Kamerabilder, Sensordaten, industrielle Steuerung oder Content-Caching. Jetzt geht es um Sprache, Bedeutung, persönliche Agenten, Dokumentenverständnis und semantische Entscheidungen. Die Edge wird nicht nur schneller. Sie wird kognitiver.

Diese Rückkehr ist nicht romantisch. Sie ist eine Kostenrechnung. Jede nicht gesendete Datei spart Transport. Jeder lokal entschiedene Intent spart einen Modellaufruf. Jeder lokale Cache-Treffer spart Wartezeit. Jede lokal maskierte Entität verringert Offenlegung. Jede lokale Vorprüfung reduziert das Risiko, dass ein großes Modell mit Kontext gefüttert wird, den es nicht sehen sollte.

Die Edge wird deshalb zum Ort der ersten Entscheidung. Dort wird nicht alles gelöst. Aber dort wird entschieden, was überhaupt eine größere Instanz braucht.

Die wichtigste Frage an der Edge lautet nicht: Kann sie alles? Sondern: Was muss nicht mehr weg?

4.3 Der Client als kognitive Runtime

Wenn man die ChatGPT-App, einen Browser-Agenten oder eine künftige Business-KI nur als Interface betrachtet, übersieht man die nächste Entwicklungsstufe. Der Client kann mehr sein als Eingabefeld und Streaming-Fenster. Er kann eine lokale kognitive Runtime werden.

Eine solche Runtime verwaltet nicht nur Darstellung. Sie verwaltet Bedeutung. Sie hält lokalen Kontext, kennt Berechtigungen, bewertet Datenklassen, steuert Caches, ruft lokale Modelle auf, baut Retrieval-Indizes, prüft Contracts, entscheidet über Eskalation und schützt private Informationen vor unnötiger Offenlegung.

Das ist ein anderer Client als der klassische Thin Client. Er zeigt nicht nur an, was der Server denkt. Er entscheidet mit, ob der Server überhaupt denken soll.

In dieser Architektur besteht der lokale Client aus mehreren Schichten. Ganz unten liegt die Hardware: CPU, GPU, NPU, Speicher, Secure Enclave oder Trusted Execution-Komponenten, lokale SSD, Netzwerk. Darüber liegt eine Runtime für lokale Modelle. Darüber ein semantischer Speicher: Dokumentenindex, Embeddings, Knowledge Cache, eventuell ein lokaler Vektorstore. Darüber sitzen Gates: Rollen, Rechte, Datenklassen, Policies, Kostenlimits, Offline-Modus. Erst danach kommt die Frage, ob ein Frontier-Modell gebraucht wird.

Der Unterschied zum klassischen Chatfenster ist fundamental. Der lokale Client ist nicht mehr passiver Bote. Er ist der erste Kurator der Wirklichkeit.

```
local_cognitive_runtime:
  user_interface
  local_identity_and_permissions
  local_context_store
  embedding_and_index_layer
```

```
  small_model_router
  deterministic_gates
  cache_and_memory
  escalation_contract
  frontier_call_only_if_needed
```

4.4 Die Hardware hat begonnen, sich zu verändern

Der vielleicht wichtigste Hinweis auf diese Entwicklung steckt nicht in einem Manifest, sondern in der Hardware. Telefone, Laptops und Edge-Geräte bekommen zunehmend spezialisierte Beschleuniger für neuronale Netze. NPUs, Apple Neural Engine, Qualcomm Hexagon, Google Tensor, Intel AI Boost, mobile GPUs und spezialisierte Edge-Module sind keine dekorativen Datenblatt-Extras mehr. Sie zeigen, dass KI-Arbeit nicht ausschliesslich im Rechenzentrum erwartet wird.

Apple beschreibt Apple Intelligence als System aus mehreren spezialisierten generativen Modellen, darunter ein ungefähr drei Milliarden Parameter großes On-Device-Sprachmodell und ein größeres Server-Modell für Private Cloud Compute. Das ist ein klares Hybrid-Signal: lokale Modelle für alltägliche Aufgaben, Servermodelle für größere Anforderungen.

Google beschreibt Gemini Nano als On-Device-Foundation-Modell, das in Android über AICore läuft, lokale Hardware nutzt, niedrige Inferenzlatenz ermöglicht und Daten lokal verarbeiten kann. Zugleich baut Google mit LiteRT eine Runtime für on-device Machine Learning und GenAI auf, die Modelle über mobile, Web- und Mikrocontroller-Plattformen mit Hardwarebeschleunigung ausführbar machen soll.

Microsoft definiert Copilot+ PCs unter anderem über NPUs mit 40+ TOPS. Das ist mehr als Marketing. Es ist eine Schwelle, ab der bestimmte KI-Funktionen nicht nur über Cloud-APIs, sondern lokal auf dem Gerät gedacht werden können. NVIDIA Jetson wiederum zeigt die industrielle Seite: kompakte, energieeffiziente Edge-AI-Module für Robotik, Kameras, Maschinen und physische Systeme.

Diese Beispiele beweisen nicht, dass die Cloud verschwindet. Sie zeigen etwas Interessanteres: Die Hardwareindustrie bereitet sich darauf vor, dass KI-Lasten auf vielen Ebenen laufen. Im Telefon. Im Laptop. Im Auto. In der Fabrik. Auf dem lokalen Server. Und weiterhin in der Frontier-Cloud.

```
hardware_signal:
  phones -> on-device foundation models
  laptops -> NPUs and local AI APIs
  edge modules -> robotics and industrial inference
  local servers -> vector stores and small LLMs
  cloud -> frontier reasoning and heavy workloads
```

4.5 Kleine Modelle sind keine kleinen Träume

Ein Fehler in vielen KI-Debatten besteht darin, kleine Modelle nur als schlechtere große Modelle zu betrachten. Dann wirkt jedes lokale Modell wie ein Kompromiss, ein Spielzeug, ein Modell zweiter Klasse. Diese Sicht ist zu grob.

Ein kleines Modell muss nicht alles können. Es muss seine Aufgabe gut genug können. Intent-Erkennung, Klassifikation, Extraktion, Zusammenfassung, Umformulierung, Sprache-zu-Struktur, Policy-Hinweise, Dokumentenvorbereitung, lokale Suche, Code-Snippets, einfache Dialoge, Vorfilterung, Sprachkorrektur: Viele dieser Aufgaben brauchen keine maximale Weltmodell-Tiefe.

Gerade in einem System episodischer Intelligenz zählt nicht die Frage, ob ein kleines Modell so gut ist wie ein Frontier-Modell. Die Frage lautet: Kann es einen Teil der Arbeit erledigen, bevor die teure Instanz eingeschaltet wird?

Das kann sehr wertvoll sein. Ein kleines lokales Modell kann erkennen, dass eine Anfrage nur eine Formatierung braucht. Es kann feststellen, dass ein Dokument zuerst klassifiziert werden muss. Es kann Entitäten maskieren. Es kann einen Routing-Vorschlag erzeugen. Es kann eine Frage an die lokale Wissensbasis stellen. Es kann Unsicherheit markieren und dann bewusst eskalieren.

Damit verändert sich die Rolle kleiner Modelle. Sie sind nicht Ersatz für die Frontier. Sie sind Reflexe, Sensoren, Vorarbeiter, Übersetzer und Wächter.

Ein kleines Modell ist nicht dann wertvoll, wenn es alles weiss. Es ist wertvoll, wenn es weiss, wann es nicht reicht.

```
local_small_model_tasks:
  classify_intent
  extract_entities
  summarize_locally
  check_policy_hints
  rewrite_for_clarity
  prepare_retrieval_query
  estimate_uncertainty
  decide_escalation_candidate
```

4.6 Der lokale Speicher: Gedächtnis ohne Dauer-Cloud

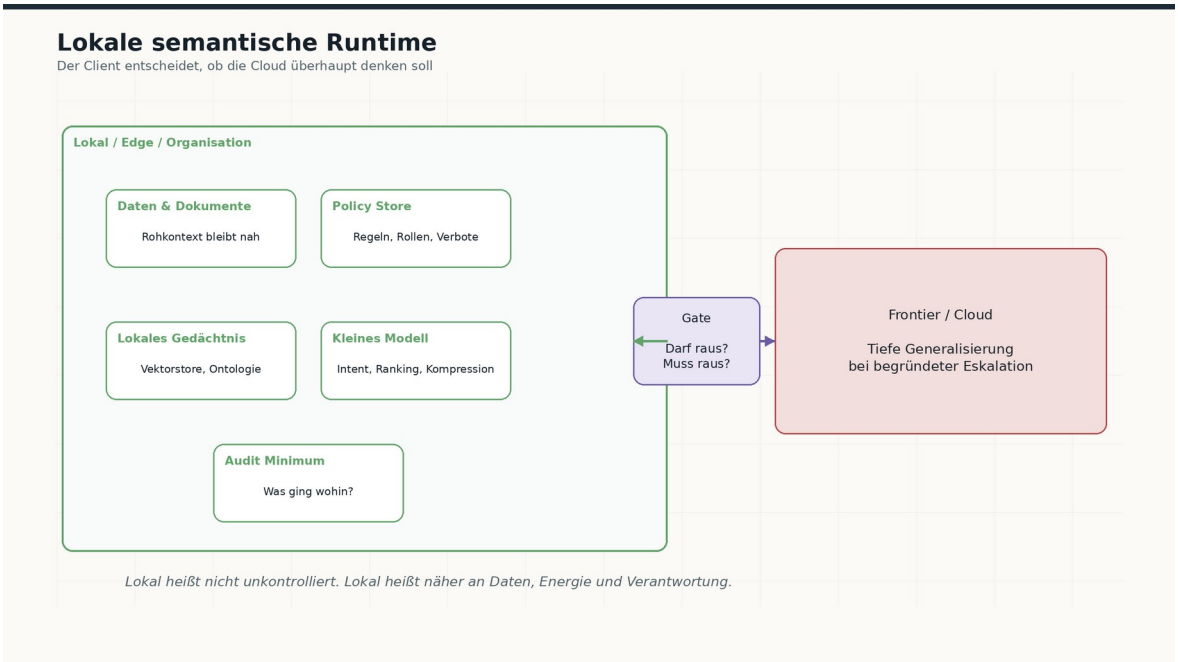

Abb. 5 - Lokale semantische Runtime: der Client als Ort von Kontext, Policy, Gedächtnis, Kompression und Audit.

Ein lokales KI-System wird erst richtig interessant, wenn es nicht bei jeder Anfrage bei null beginnt. Es braucht lokales Gedächtnis. Nicht unbedingt ein poetisches Gedächtnis, sondern ein technisches: Dateien, Indizes, Embeddings, Metadaten, Zugriffsrechte, Zusammenfassungen, Prozesszustände, frühere Entscheidungen, offene Fragen.

Hier kommen lokale Vektorstores und Suchsysteme ins Spiel. FAISS ist eine Bibliothek für effiziente Ähnlichkeitssuche und Clustering dichter Vektoren. Qdrant, Chroma, Milvus und ähnliche Systeme zeigen,

dass semantische Suche nicht zwingend ein entferntes Cloud-Produkt sein muss. Sie kann lokal, eingebettet, standalone oder hybrid betrieben werden.

Der lokale Speicher ist die Grundlage für eine andere Art von KI-Nutzung. Statt alle Dokumente an ein großes Modell zu senden, kann der Client lokal fragen: Welche drei Abschnitte sind für diese Aufgabe relevant? Welche Dokumente darf dieser Nutzer sehen? Welche Entitäten müssen maskiert werden? Welche alten Zusammenfassungen sind noch gültig? Welche Daten sind veraltet?

Das ist nicht nur Datenschutz. Es ist auch Energie- und Kostenlogik. Je besser der lokale Speicher arbeitet, desto kleiner wird der Kontext, der später an ein größeres Modell geht. Und je kleiner dieser Kontext ist, desto weniger Tokens, Wartezeit, Netzlast und Wärme entstehen in der zentralen Infrastruktur.

In einem Unternehmen könnte ein lokaler oder regionaler semantischer Speicher die eigentliche Grundlage der KI-Souveränität werden. Nicht das Modell allein macht das System wertvoll, sondern die Art, wie es mit eigenem Kontext umgeht.

```
raw_business_context:
  documents
  tickets
  contracts
  calendars
  code
  customer_notes
local_memory_layer:
  chunk -> embed -> index -> filter -> retrieve -> compress
frontier_context:
  only_relevant_and_allowed_fragments
```

4.7 Lokaler Kontext ist der gefährlichste Kontext

Je näher ein System am Nutzer sitzt, desto näher sitzt es an der Wahrheit. Das ist seine Stärke und sein Risiko.

Der lokale Client kennt Dinge, die ein Frontier-Modell idealerweise nie roh sehen sollte: echte Namen, interne Projektnamen, Kundendaten, Kalenderdetails, private Notizen, geöffnete Dateien, Bildschirmkontext, Standort, Kommunikationsmuster, Browserhistorie, Vertraulichkeitsstufen. Genau deshalb ist die lokale Runtime so wichtig.

Wenn die Cloud der Ort großer Generalisierung ist, dann ist der Client der Ort konkreter Verletzlichkeit. Dort liegen nicht abstrakte Muster, sondern echte Fälle. Dort liegt nicht nur ein Vertragstyp, sondern der Vertrag. Nicht nur ein Lieferkettenrisiko, sondern der Lieferant. Nicht nur ein Meeting, sondern der Raum, die Namen, die Abhängigkeiten.

Ein gutes lokales System muss deshalb zwei gegensätzliche Dinge leisten: Es muss Kontext verstehen, aber Kontext auch schützen. Es muss Bedeutung extrahieren, ohne unnötig Rohdaten weiterzugeben. Es muss wissen, welche Wahrheit lokal bleiben muss.

Das führt direkt zu Kapitel 5. Semantic Compression ist ohne lokale Systeme kaum denkbar. Erst wenn ein lokaler Layer den echten Business-Kontext kennt, kann er ihn sinnvoll abstrahieren.

Die Cloud soll Muster sehen. Der Client kennt die Namen.

4.8 Lokale Systeme sparen nicht automatisch Energie

Hier braucht das Buch Nüchternheit. Lokal ist nicht automatisch nachhaltig. Ein Laptop, ein Mac-mini-Cluster, ein Edge-Server oder eine Fabrikbox verbraucht ebenfalls Strom. Viele schlecht konfigurierte lokale Systeme können ineffizienter sein als ein hochoptimiertes Rechenzentrum.

Die These dieses Buches ist deshalb nicht: Lokal ist immer besser. Die These lautet: Lokal erlaubt andere Optimierungspunkte.

Ein lokales System kann Arbeit zeitlich verschieben. Es kann Indizes dann aktualisieren, wenn das Gerät ohnehin läuft. Es kann Solarüberschuss nutzen. Es kann Nachtfenster nutzen. Es kann mit vorhandener Hardware arbeiten. Es kann kleine Aufgaben ohne Netzwerkreise erledigen. Es kann große Modellaufrufe vermeiden. Es kann vertrauliche Rohdaten lokal halten und nur verdichtete Aufgaben senden.

Das ist kein magischer Energiesieg. Es ist eine Architekturmöglichkeit.

Der entscheidende Unterschied liegt in der Art der Last. Ein zentrales Frontier-System muss hohe Verfügbarkeit, geringe Latenz, massive Parallelität und globale Nachfrage bedienen. Ein lokales System kann viele Arbeiten langsamer, opportunistischer und kontextnäher ausführen. Es muss nicht jede Aufgabe sofort und maximal beantworten.

Damit entsteht ein anderes Energieprofil: weniger Hochleistungsdauerlast, mehr Hintergrundarbeit; weniger rohe Kontexttransporte, mehr lokale Verdichtung; weniger Default-Frontier, mehr begründete Eskalation.

```
energy_logic:
  not "local = free"
  but "local = schedulable, contextual, selective"
good_local_workloads:
  embedding_refresh_when_idle
  document_indexing_with_solar_surplus
  local_cache_lookup
  local_intent_routing
  context_masking_before_cloud
```

4.9 Die Grenzen lokaler Systeme

Die Rückkehr lokaler Systeme darf nicht als einfache Erlösung verkauft werden. Lokale KI hat harte Grenzen.

Erstens: Qualität. Kleine Modelle sind für viele Aufgaben gut genug, aber nicht für alle. Tiefe Analyse, komplexe Synthese, schwieriges Reasoning, multimodale Spezialfragen und neue wissenschaftliche Probleme werden weiterhin größere Modelle brauchen.

Zweitens: Hardwareheterogenität. Die lokale Welt ist unordentlich. Verschiedene CPUs, GPUs, NPUs, Treiber, Betriebssysteme, Speichergrenzen, thermische Limits, Akkuzustände und Sicherheitsmodelle machen Deployment schwieriger als eine zentrale API.

Drittens: Updates. Ein Cloud-Modell kann zentral verbessert werden. Lokale Modelle müssen verteilt, validiert, kompatibel gehalten und gegebenenfalls zurückgerollt werden. Das ist MDM, DevOps, Security und Modellpflege zugleich.

Viertens: Sicherheit. Ein lokales System ist näher an den Daten, aber auch näher an kompromittierten Geräten. Malware, physischer Zugriff, unsichere Plugins, Prompt-Injection aus lokalen Dokumenten und falsche Rechtevererbung werden zu realen Risiken.

Fünftens: Beobachtbarkeit. In zentralen Systemen lässt sich vieles einheitlich loggen und überwachen. In verteilten lokalen Systemen wird Audit schwieriger. Man muss entscheiden, welche Logs lokal bleiben,

welche zentral aggregiert werden dürfen und wie man Missbrauch erkennt, ohne wieder alles zu zentralisieren.

Diese Grenzen sind kein Argument gegen lokale Systeme. Sie sind ein Argument gegen Naivität. Episodische Intelligenz braucht lokale Stärke und zentrale Fähigkeit. Nicht eines von beidem allein.

4.10 Was wirklich lokal gehört

Nicht alles gehört auf den Client. Aber einiges gehört genau dorthin.

Lokal gehören vor allem jene Schichten, die eng mit privatem Kontext, schneller Vorentscheidung, niedriger Komplexität oder deterministischer Kontrolle verbunden sind. Dazu zählen lokale Suche, Datenklassifikation, Maskierung, einfache Zusammenfassungen, Intent-Routing, Zugriffskontrolle, Cache, Dokumentenvorbereitung und das Halten persönlicher oder betrieblicher Prozesszustände.

Nicht zwingend lokal gehören dagegen Aufgaben, die große Weltmodelle, schwere Simulationen, tiefe Synthese, enorme Modellgröße, aktuelle globale Informationen oder hohe Parallelisierung benötigen. Diese Aufgaben dürfen eskalieren. Aber sie sollten es bewusst tun.

Man kann es als Schichtenmodell formulieren:

```
local:
  identity, permissions, raw private context
  local memory, embeddings, cache
  small models, intent, extraction, masking
  deterministic gates and contracts
regional / organization:
  shared knowledge bases
  team memory, policy stores, audit aggregation
  fine-tuned specialist models
frontier:
  deep reasoning, novel synthesis, hard uncertainty
  broad world knowledge, high-value escalation
```

Diese Aufteilung ist nicht endgültig. Aber sie macht eine wichtige Verschiebung sichtbar: Intelligenz wird nicht an einem Ort gebaut. Sie wird über Ebenen verteilt.

4.11 Der neue Client ist ein Wächter der Eskalation

Wenn der Client zur kognitiven Runtime wird, verändert sich seine wichtigste Aufgabe. Er ist nicht mehr nur dafür da, eine Antwort darzustellen. Er entscheidet, ob eine Frage klein bleibt oder groß wird.

Diese Eskalationslogik ist der Kern episodischer Intelligenz. Lokal wird geprüft: Ist die Aufgabe trivial? Ist sie wiederkehrend? Gibt es einen Cache-Treffer? Reicht ein kleines Modell? Ist der Kontext sensibel? Muss etwas maskiert werden? Gibt es eine lokale Policy, die den Vorgang blockiert? Ist ein Frontier-Call gerechtfertigt?

Erst wenn diese Fragen beantwortet sind, entsteht die eigentliche Anfrage an ein größeres Modell. Dann ist sie nicht mehr roh, sondern vorbereitet. Nicht mehr grenzenlos, sondern vertraglich beschrieben. Nicht mehr blind, sondern budgetiert.

Das ist der Unterschied zwischen Cloud-first und Escalation-first. Cloud-first fragt: Welches Modell antwortet? Escalation-first fragt: Welche Schicht ist für diesen Schritt die richtige?

```
request arrives
```

```
  -> classify task locally
  -> check permissions
  -> search local memory
  -> try small model / deterministic path
  -> estimate uncertainty and risk
  -> compress or mask context
  -> if justified: frontier escalation
  -> verify output locally
  -> act or ask human gate
```

Der teuerste Teil der KI wird nicht abgeschafft. Er wird seltener, klarer und bewusster eingeschaltet.

4.12 Die Rückkehr ist keine Rückabwicklung

Es wäre falsch, lokale Systeme als Rückkehr in eine vor-cloudige Vergangenheit zu verstehen. Der neue lokale Rechner ist nicht der alte Desktop mit Word und E-Mail. Er ist Teil eines verteilten kognitiven Systems.

Er arbeitet mit Cloud-Diensten, nicht gegen sie. Er nutzt lokale Modelle, aber auch Frontier-Modelle. Er hält privaten Kontext, aber kommuniziert mit Team-Systemen. Er läuft offline, aber synchronisiert, wenn es sinnvoll ist. Er nutzt kleine Reflexe und große Denkphasen.

Das macht die Architektur komplexer. Aber diese Komplexität spiegelt die Wirklichkeit besser wider. Menschen arbeiten ebenfalls nicht nur zentral. Ein Teil geschieht im Körper, ein Teil im Reflex, ein Teil im Gedächtnis, ein Teil im Gespräch, ein Teil in der bewussten Analyse. Intelligenz ist verteilt, geschichtet und situativ.

Episodische Intelligenz überträgt diesen Gedanken auf KI-Infrastruktur. Lokale Systeme sind nicht die Gegner der Frontier. Sie sind ihre Umgebung. Sie entscheiden, wann die Frontier gebraucht wird, was sie sehen darf und wie ihre Antwort in echte Prozesse zurückgeführt wird.

Damit wird Kapitel 4 zur Brücke. Es zeigt, wo Semantic Compression entstehen kann. Es zeigt, warum deterministische Gates lokal so wichtig sind. Und es zeigt, warum Energie nicht nur im Rechenzentrum, sondern in der Architekturentscheidung beginnt.

Die Cloud bleibt das Fernrohr. Der Client bleibt die Hand am Objekt.

Architektur-Skizze: Lokale kognitive Runtime

```
USER / DEVICE
  |
  v
local context intake
  - screen / files / docs / calendar / app state
  |
  v
local semantic layer
  - classify, embed, index, retrieve, summarize
  |
```

```
  v
governance layer
  - permissions, data class, policy, cost, risk
  |
  v
small model / deterministic path
  - solve locally if enough
  |
  v
escalation contract
  - compress, mask, define task, set budget
  |
  v
FRONTIER MODEL
  - only when justified
  |
  v
local verification and action gate
```

4.13 Lokale Verantwortung: Edge heißt nicht unkontrolliert

Die Rückkehr lokaler Systeme darf nicht mit Wildwuchs verwechselt werden. Ein lokales KI-System ist nur dann eine Reifeform der Architektur, wenn es selbst verwaltet, begrenzt und prüfbar ist. Sonst verschiebt man das Risiko bloß aus dem Rechenzentrum auf tausend unübersichtliche Endgeräte.

Die wichtigste Formel lautet deshalb: lokal heißt näher an Daten, Energie und Verantwortung - nicht außerhalb von Verantwortung. Ein lokales Modell, ein lokaler Vektorspeicher oder ein lokaler Agent braucht dieselben Grundfragen wie ein Cloud-System: Wer darf ihn nutzen? Welche Daten darf er sehen? Welche Tools darf er aufrufen? Welche Logs darf er erzeugen? Wann muss er stoppen?

In der Praxis entsteht daraus ein lokaler Verantwortungsstack. Er beginnt mit Geräteverwaltung und Identität, setzt sich fort über verschlüsselte Speicherung, Policy Stores, Modell- und Tool-Registries und endet bei lokalen Evals, die prüfen, ob die Runtime noch das tut, was sie tun soll. Die App ist dann nicht nur ein Fenster. Sie ist ein kleiner, auditierbarer Betriebsraum.

Schicht	Lokale Verantwortung	Warum sie wichtig ist
Device und Identität	Gerätestatus, Nutzerrolle, Compliance, Schlüsselmaterial	Ein kompromittiertes Gerät darf nicht zum privilegierten KI-Knoten werden.
Lokaler Policy Store	Datenklassen, erlaubte Modelle, Tool-Grenzen, Kosten- und Energiegrenzen	Die Entscheidung soll vor dem Cloud-Call fallen, nicht danach im Audit.
Modell-Registry	Welche lokalen Modelle sind zugelassen, quantisiert, versioniert und getestet?	Sonst entsteht Schatten-KI mit unbekannten Gewichten und unbekanntem Verhalten.
Semantischer Speicher	Indexgröße, TTL, Zugriffsrechte, Entitätsmaskierung, Löschregeln	Ein lokaler Vektorstore kann selbst zum sensiblen Gedächtnis werden.
Telemetrie	Nur notwendige Metriken: Fehler, Drift, Kosten, Eskalationen, keine Rohdaten	Kontrolle darf nicht das nächste Datenleck bauen.

4.14 Qualität prüfen, ohne alles zentral zu überwachen

Lokale Systeme brauchen Qualitätskontrolle. Aber wenn jede lokale Entscheidung zentral protokolliert wird, verliert die Architektur einen Teil ihres Sinns. Die Kunst besteht darin, Qualität messbar zu machen, ohne den lokalen Kontext wieder durch die Hintertür zu exportieren.

Dafür eignen sich drei Arten von Prüfungen. Erstens lokale Regressionstests: bekannte Aufgaben, erwartete Ausgabeformen, erlaubte Fehlerbereiche. Zweitens stichprobenartige Review-Fenster: nicht jeder Schritt wird hochgeladen, aber ausgewählte Artefakte werden geprüft. Drittens aggregierte Betriebsmetriken: Wie oft eskaliert der Client? Wie oft verweigert er? Wie oft schlägt Validierung fehl? Diese Zahlen können helfen, ohne Rohkontext offenzulegen.

Wichtig ist die Trennung zwischen Inhalt und Verhalten. Der Inhalt eines Kundenvertrags muss nicht in ein zentrales Monitoring-System wandern, damit man sieht, dass ein lokaler Contract-Parser häufiger scheitert. Was gemessen wird, ist nicht die Wirklichkeit des Falls, sondern das Verhalten der Maschine.

Prüfziel	Lokale Methode	Zentral teilbar?
Antwortformat	Structured-output-Test gegen Schema	Ja, als Schema-Erfolgsquote ohne Inhalt
Retrieval-Qualität	Goldene Testfragen auf lokalem Index	Teilweise, als Score und Dokumentklasse
Datenabfluss	Entity- und Secret-Scanner vor Eskalation	Ja, als Zähler je Datenklasse
Modell-Drift	Wiederkehrende Testfälle nach Modellupdate	Ja, als Delta, nicht als Rohprompt
Nutzerfeedback	Lokales Feedback mit Redaction vor Export	Nur stark reduziert und zweckgebunden

4.15 Der lokale Speicher braucht ein Gedächtnisbudget

Ein lokaler semantischer Speicher ist verführerisch. Er macht Systeme schneller, persönlicher und sparsamer. Aber er sammelt Bedeutungsreste: Embeddings, Zusammenfassungen, Chunks, Metadaten, Rollen, Prozesszustände. Auch wenn der Rohtext gelöscht ist, kann der Speicher noch verraten, woran gearbeitet wurde.

Deshalb braucht jedes lokale Gedächtnis ein Budget. Nicht nur Speicherplatz in Gigabyte, sondern ein Bedeutungsbudget: Welche Datenklasse darf wie lange bleiben? Welche Chunks werden nur flüchtig gehalten? Welche Embeddings werden pro Projekt getrennt? Welche Indizes dürfen exportiert werden? Welche Vektoren müssen gelöscht werden, wenn der Vertrag, der Fall oder das Projekt endet?

Der lokale Speicher darf also nicht zu einer zweiten Schatten-Datenbank werden. Er muss kleiner sein als die Organisation, für die er arbeitet. Er muss vergesslich bleiben können.

Datenart	Lokale Speicherung	Lösch-/Grenzregel
Öffentliche Wissensbasis	Index und Embeddings möglich	Regelmäßige Aktualisierung, Quellenversionen halten
Interne Dokumente	Projekt- oder rollengetrennte Indizes	TTL, Zugriff nach Rolle, Löschung bei Projektende
Personenbezug	Nur minimal, maskiert oder flüchtig	Kurze TTL, Zweckbindung, keine offenen Exporte
Geschäftsgeheimnis	Getrennter Speicher mit strenger Policy	Keine Cloud-Eskalation ohne Contract und Freigabe
Strategische Absicht	Möglichst nicht persistieren, nur abstrahieren	Rekonstruktionsprüfung vor jeder Weitergabe

4.16 Schatten-KI vermeiden

Die größte Gefahr lokaler KI ist nicht, dass sie zu schwach ist. Die größte Gefahr ist, dass sie unsichtbar wird. Wenn Teams eigene Modelle laden, eigene Vektordatenbanken bauen und eigene Agenten mit internen Dateien verbinden, entsteht eine zweite Infrastruktur, die niemand geplant hat und niemand verantwortet.

Die Antwort darauf ist nicht, lokale KI zu verbieten. Das würde nur die inoffizielle Nutzung fördern. Die Antwort ist ein erlaubter Korridor: genehmigte lokale Runtimes, registrierte Modelle, bekannte Tool-Schnittstellen, prüfbare Datenklassen, einfache Freigabewege und harte Verbote für besonders sensible Kontexte.

Ein reifes Unternehmen sagt nicht: Alles lokal ist gut. Es sagt: Diese lokalen Fähigkeiten sind erlaubt, diese sind gebunden, diese sind verboten, und diese müssen zur Frontier oder zum Menschen eskalieren.

4.17 Drei kleine Fallstudien für die lokale Runtime

Damit der lokale Client nicht abstrakt bleibt, helfen drei kleine Szenen. Sie sind keine Produktvorschläge, sondern Prüfsteine für die Architektur.

Szenario	Lokale Arbeit	Eskalation nur wenn
KMU-Dokumentenindex	Dokumente chunking, Embeddings, Berechtigungen, lokale Zusammenfassungen	Konflikt zwischen Quellen, strategische Bewertung, externe Formulierung nötig
Werkstatt / Maschine	Sensorlogs sortieren, Fehlercodes clustern, Wartungshistorie abrufen	Unbekanntes Fehlerbild, Sicherheitsrisiko, Gewährleistungsfrage
Kanzlei / Beratung	Akten lokal strukturieren, Namen maskieren, Fristen erkennen	Rechtsauslegung, Mandatsstrategie, externe Argumentation
Laptop-Agent	Kalender, Dateien, E-Mails und lokale Notizen vorsortieren	Private oder geschäftliche Daten die lokale Grenze verlassen müssten

Diese Beispiele zeigen den eigentlichen Punkt: Lokalität ist kein Selbstzweck. Sie lohnt sich dort, wo Nähe zu Daten, Kontext, Latenz, Energie oder Verantwortung wertvoller ist als maximale Modellstärke.

Zwischenfazit

Die Rückkehr lokaler Systeme ist kein Nebenschauplatz. Sie ist eine Voraussetzung für eine KI-Architektur, die mit Energie, Datenschutz, Latenz, Kosten und Verantwortung umgehen kann.

Der lokale Client wird nicht alles leisten. Aber er wird vieles entscheiden: Was bleibt lokal? Was wird verdichtet? Was wird maskiert? Was wird gecacht? Was wird verworfen? Was wird eskaliert?

Damit verschiebt sich die Frage des Buches erneut. Es geht nicht mehr nur darum, wie stark Modelle sind. Es geht darum, wie intelligent die Umgebung ist, die sie einsetzt.

Wenn Kapitel 2 der Maschinenraum war und Kapitel 3 der operative Apparat, dann ist Kapitel 4 der Ort, an dem die Maschine wieder in die Nähe des Menschen rückt. Nicht als Spielzeug. Als Schicht der Souveränität.

Begriffe für dieses Kapitel

Lokale kognitive Runtime: Client- oder Edge-Schicht, die Kontext, kleine Modelle, Speicher, Gates und Eskalationslogik lokal verwaltet.

On-Device AI: KI-Inferenz, die direkt auf dem Endgerät stattfindet, typischerweise mit CPU, GPU oder NPU.

Edge AI: KI-Verarbeitung nahe an der Quelle der Daten, etwa auf Geräten, lokalen Servern, Maschinen oder industriellen Modulen.

Lokaler semantischer Speicher: Ein lokaler Index aus Dokumenten, Embeddings, Metadaten und Prozesszuständen, der Kontext bereitstellt, ohne alles in die Cloud zu senden.

Eskalationswächter: Eine lokale oder regionale Logik, die entscheidet, wann ein großes Modell wirklich gebraucht wird.

Übergang: Lokalität ist zunächst ein Ort. Die nächste Frage ist, welche Bedeutung diesen Ort verlassen darf.

Kapitel 5 - Semantische Verdichtung

Wie Bedeutung geschützt wird, bevor sie zur Frontier wandert

Die entscheidende Datenschutzfrage lautet nicht nur, ob Daten verschlüsselt sind. Die entscheidende Frage lautet, ob der zentrale Reasoner die Rohrealität überhaupt sehen muss.

Semantic Compression ist der Versuch, diese Rohrealität lokal in eine abstrakte, strukturierte Aufgabe zu übersetzen. Namen werden Rollen. Verträge werden Zustände. Firmen werden Kategorien. Geldbeträge werden Größenklassen. Historische Details werden Risikomuster. Das Frontier-Modell bekommt nicht das geheime Dossier, sondern ein minimales semantisches Problem.

Das klingt zunächst wie Anonymisierung. Es ist aber mehr und weniger zugleich. Mehr, weil nicht nur Identitäten entfernt werden, sondern die gesamte Anfrage in eine neue Bedeutungsform gebracht wird. Weniger, weil dadurch keine perfekte Anonymität garantiert ist. Wer das verspricht, verkauft Nebel in einer Glasflasche. Semantic Compression ist kein magischer Schutzwall. Sie ist ein Architekturprinzip: Gib dem entfernten Modell nur die Bedeutung, die es für seine Aufgabe braucht - und keinen Millimeter mehr.

Nicht jedes KI-System muss die Welt sehen. Manchmal reicht es, wenn es die Struktur eines Problems sieht.

5.1 Der Moment, in dem Verschlüsselung nicht mehr reicht

Verschlüsselung ist eine der großen zivilisatorischen Leistungen der Informatik. Sie schützt Daten auf Kabeln, in Speichern, in Backups, in Datenbanken, in Transportwegen. Sie macht aus einer lesbaren Nachricht eine Folge von Zeichen, die ohne Schlüssel wie Rauschen wirkt. Für klassische IT-Sicherheit ist das ein gewaltiger Schutz.

Aber KI-Systeme berühren einen unangenehmen Punkt. Ein Modell soll nicht nur speichern oder weiterleiten. Es soll verstehen, gewichten, zusammenfassen, vergleichen, schlussfolgern. In diesem Moment genügt es nicht, dass Bits verschlüsselt sind. Irgendwo muss Bedeutung entstehen. Irgendwo muss aus Zeichen wieder Sinn werden. Und genau dort wird Bedeutung selbst zum Risiko.

Man kann das nüchtern formulieren: Ende-zu-Ende-Verschlüsselung verhindert im Idealfall, dass ein Betreiber Inhalte sieht. Viele KI-Aufgaben verlangen aber gerade Verarbeitung dieser Inhalte. Apple beschreibt diese Spannung explizit für Private Cloud Compute: Vollständige Ende-zu-Ende-Verschlüsselung ist für bestimmte Cloud-LLM-Verarbeitung nicht möglich, weil das Modell die Daten der Anfrage verarbeiten muss. [K5-S1]

Apple Private Cloud Compute ist deshalb interessant, weil es die Cloud nicht einfach als normalen Server behandelt. Apple formuliert Anforderungen wie stateless computation, technisch erzwingbare Garantien, keinen privilegierten Runtime-Zugriff, Nicht-Zielbarkeit und verifizierbare Transparenz. Das ist keine kleine Produktnotiz. Das ist eine neue Sprache für KI-Clouds: weniger Administratorenmagie, mehr überprüfbare Architektur. [K5-S1] [K5-S2]

Meta verfolgt mit Private Processing für WhatsApp eine verwandte Richtung. Die technische Beschreibung spricht über Trusted Execution Environments, Remote Attestation, Oblivious HTTP, anonyme Credentials, Forward Security und stateless processing. Auch hier lautet das Ziel: KI-Funktionen sollen möglich werden, ohne dass der Betreiber die privaten Nachrichten dauerhaft oder frei einsehen kann. [K5-S3] [K5-S4]

Diese Architekturen sind wichtig. Sie zeigen, dass die Industrie verstanden hat: KI-Privacy ist nicht nur eine juristische Checkbox. Sie ist Laufzeitarchitektur. Trotzdem beantworten sie vor allem die Frage, wie sensible Daten in einer Cloud-Umgebung geschützt verarbeitet werden können. Episodische Intelligenz stellt eine zusätzliche Frage: Was wäre, wenn wir die sensible Rohbedeutung gar nicht erst vollständig exportieren?

Verschlüsselung schützt den Kanal. Semantic Compression fragt, ob der Kanal die ganze Wirklichkeit überhaupt tragen muss.

5.2 Bedeutung ist der neue Klartext

In der klassischen Sicherheit spricht man von Klartext, wenn verschlüsselte Daten wieder lesbar sind. In KI-Systemen reicht dieser Begriff nicht mehr. Der Text kann anonymisiert wirken und trotzdem verräterisch sein. Ein Firmenname kann fehlen, aber die Kombination aus Branche, Umsatz, Standort, Lieferkettenproblem, Zeitfenster und Projektziel kann ausreichen, um den Fall zu erkennen.

Das Problem liegt nicht nur in personenbezogenen Daten. Es liegt auch in Geschäftsgeheimnissen. Eine Anfrage kann keine einzige Person nennen und trotzdem die strategische Lage einer Firma offenlegen. Sie kann keine Kundennamen enthalten und trotzdem zeigen, welche Märkte angegriffen, welche Akquisitionen vorbereitet oder welche Schwachstellen intern bekannt sind.

Bedeutung ist also nicht nur im Namen. Bedeutung sitzt in Relationen: Wer hängt von wem ab? Welche Entscheidung steht bevor? Welche Ressource ist knapp? Welche Frist ist kritisch? Welche Option ist politisch heikel? Welche Risikoannahme hat das Management noch nicht ausgesprochen?

Darum reicht die naive Praxis nicht: Wir nehmen schnell alle Namen raus und schicken den Rest ins Modell. Das ist Regex-Privacy. Für ein Frontier-Modell ist sie oft nur ein dünner Vorhang. Wenn das Modell genug Kontext bekommt, kann es Muster erkennen, auch wenn einige Labels fehlen.

Der europäische Grundsatz der Datenminimierung passt hier überraschend gut als architektonische Inspiration. Die EDPS beschreibt Data Minimization als Begrenzung der Erhebung auf das direkt Relevante und Notwendige für einen bestimmten Zweck. Semantic Compression übersetzt diesen Grundsatz in die KI-Laufzeit: Nicht nur personenbezogene Daten sollen minimiert werden, sondern exportierte Bedeutung. [K5-S10]

5.3 Was Semantic Compression ist - und was nicht

Der Begriff Compression kann in die Irre führen. Gemeint ist nicht nur, aus zwanzig Seiten fünf Absätze zu machen. Gemeint ist auch nicht nur, Tokenkosten zu senken. Semantic Compression heißt: Ein lokales System übersetzt Rohkontext in eine kleinere, abstraktere und kontrolliertere Bedeutungsschicht.

Normale Zusammenfassung fragt: Was steht im Text? Semantic Compression fragt: Was muss ein externer Reasoner wissen, um eine Aufgabe zu lösen, ohne den Fall selbst zu kennen?

Das ist ein anderer Maßstab. Es geht nicht nur um Kürze, sondern um Entkopplung. Der konkrete Kunde, der konkrete Vertrag, der konkrete Lieferant, die interne Roadmap, die geplante Akquisition, die politische Empfindlichkeit, die echte Prozesskette: All das bleibt lokal. Nach außen geht eine strukturierte Aufgabe, die das Muster enthält, nicht die ganze Wirklichkeit.

Forschung zu Semantic Compression mit großen Sprachmodellen untersucht bereits, wie Bedeutung trotz verkürzter Repräsentationen erhalten bleiben kann. Gilbert et al. beschreiben, dass es in manchen Fällen nicht nötig ist, jedes Detail perfekt rekonstruieren zu können, solange semantische Präzision oder Intention ausreichend erhalten bleiben. Fei et al. untersuchen Semantic Compression, um längere Kontexte für LLMs nutzbar zu machen. Für episodische Intelligenz ist daran weniger die Tokenzahl entscheidend als der Architekturgedanke: Bedeutung kann gezielt verdichtet, geprüft und operationalisiert werden. [K5-S5] [K5-S6]

Gleichzeitig muss dieses Buch sauber trennen. Die wissenschaftliche Verwendung von Semantic Compression meint oft Kontextverdichtung für Modellleistung. Die hier vorgeschlagene Verwendung meint zusätzlich Bedeutungsisolation für Datenschutz, Geschäftsgeheimnisse und Energieeffizienz. Das ist verwandt, aber nicht identisch.

Abgrenzung der Begriffe

Begriff	Primäre Frage	Was geschützt wird	Grenze
Verschlüsselung	Kann jemand die Daten lesen?	Transport, Speicher, Zugriff	Während Verarbeitung braucht ein System oft Klartext oder eine geschützte Klartext-Umgebung.
Anonymisierung	Kann eine Person oder	Identität, personenbezogene	Kombinierte Merkmale

	Organisation identifiziert werden?	Merkmale	können re-identifizierend wirken.
RAG	Welchen Kontext braucht das Modell?	Aktualität, Herkunft, externes Wissen	RAG entscheidet selten von selbst, was nicht herausgegeben werden darf.
Semantic Compression	Welche Bedeutung darf exportiert werden?	Business-Kontext, Relationen, strategische Details	Zu starke Abstraktion zerstört Qualität; zu schwache Abstraktion verrät den Fall.

5.4 Die lokale semantische Runtime

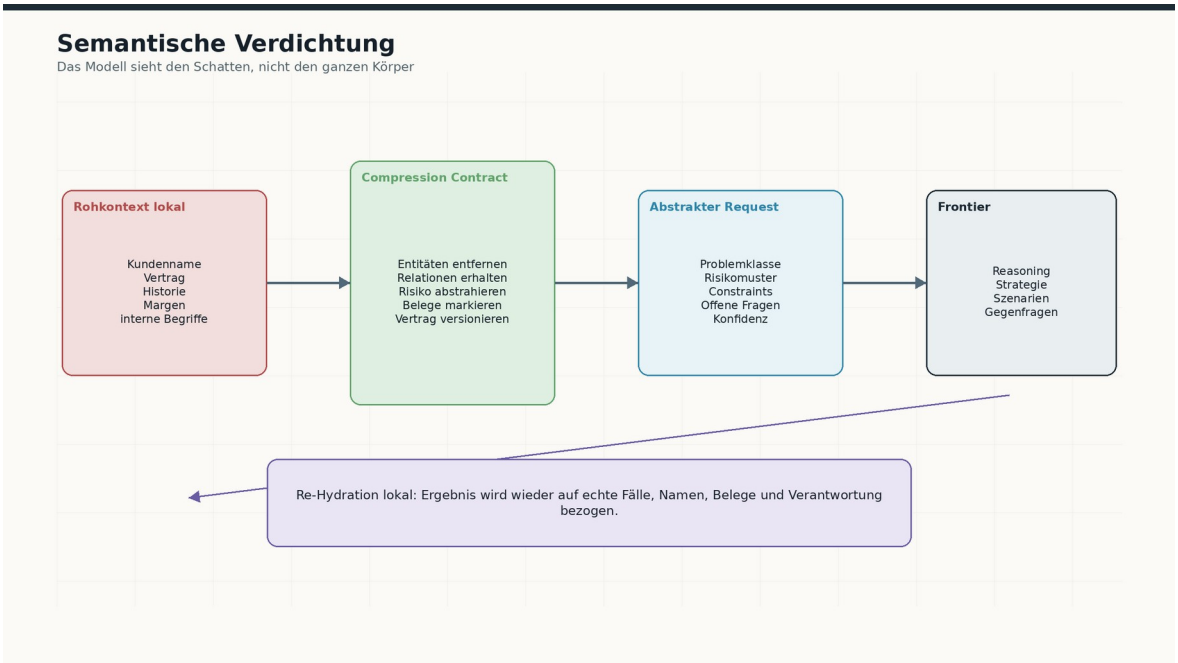

Abb. 6 - Semantische Verdichtung: Rohkontext bleibt lokal, die Frontier erhält nur eine abstrahierte Aufgabe.

Semantic Compression braucht einen Ort. Dieser Ort ist nicht das Frontier-Modell. Es ist der lokale semantische Layer: ein System auf dem Gerät, im Unternehmensnetz, in einer privaten Cloud oder in einem regional kontrollierten Cluster. Dieser Layer kennt die echte Welt. Er kennt Kundennamen, Rechte, interne IDs, Verträge, Datenbanken, Historien, Zugriffsregeln, Sensitivitätsstufen und lokale Ontologien.

Das Frontier-Modell sieht im Idealfall nur die abstrakte Aufgabe. Der lokale Layer sieht die Wirklichkeit.

Damit verschiebt sich die Rolle des Clients. Die App ist dann nicht nur ein Eingabefeld mit Streaming-Ausgabe. Sie wird zu einer kleinen kognitiven Runtime. Sie entscheidet, welche Daten lokal bleiben, welche Chunks relevant sind, welche Bedeutungsmerkmale in eine Anfrage dürfen, ob eine Anfrage risikoreich ist, ob ein Mensch zustimmen muss und ob eine Cloud-Eskalation überhaupt nötig ist.

In produktiven Systemen existieren solche Laufzeitideen bereits in anderen Formen. Agent-Orchestrierungstools wie LangGraph betonen langlebige, zustandsbehaftete Workflows, Durable Execution und Human-in-the-Loop-Muster. Das ist keine fertige Semantic-Compression-Architektur, aber es zeigt eine wichtige Richtung: KI wird nicht nur Prompt und Antwort, sondern ein kontrollierter Ablauf mit Zuständen, Pausen, Prüfungen und Übergängen. [K5-S11]

Der lokale semantische Layer ist in Systemen episodischer Intelligenz nicht Beiwerk. Er ist der Ort der Souveränität. Dort liegt das Gedächtnis. Dort liegt der Mapping-Schlüssel zwischen Wirklichkeit und Abstraktion. Dort liegt die Entscheidung, ob das Frontier-Modell nur eine Problemklasse sieht oder eine sanitisierte Fallbeschreibung oder gar nichts.

Architekturskizze: lokale semantische Runtime
Raw Business Context

```
-> Rechteprüfung
-> Intent- und Risikoerkennung
-> lokales Retrieval
-> Sensitivitätsklassifikation
-> Semantic Compression Contract
-> abstrakte Frontier-Anfrage
-> Frontier Reasoning
-> lokale Verifikation
-> Re-Hydration in den echten Business-Kontext
```

5.5 Beispiel: Die Akquisition, die das Modell nicht kennen soll

Nehmen wir ein mittelständisches Unternehmen, das eine Akquisition vorbereitet. Der echte Kontext ist hochsensibel. Es gibt Namen, Termine, interne Bewertungen, Cashflow-Probleme, Lieferkettenrisiken, Personalfragen, vielleicht sogar noch nicht veröffentlichte Marktbewegungen. Ein klassischer Cloud-Prompt würde daraus einen Alptraum für Datenschutz, Compliance und Geschäftsgeheimnisse machen.

Der naive Prompt könnte so aussehen:

```
Naiver Rohprompt - nicht geeignet für eine externe Frontier-Eskalation
Kunde: Müller AG, Zürich.
Zielunternehmen: NordTech Components GmbH.
Projektname: Albatros.
Geplante Übernahme im Q3.
Problem: kurzfristig negativer Cashflow, Abhängigkeit von Lieferant Y,
interne Sorge um Abwanderung des CTO, offene Gewährleistungsrisiken.
Bitte analysiere die wichtigsten M&A-Risiken und formuliere eine Empfehlung.
```

Aus Sicht des Modells ist das wunderbar. Aus Sicht des Unternehmens ist es Selbstentblößung. Der Prompt verrät nicht nur Namen. Er verrät Absicht, Timing, Verwundbarkeit und Strategie.

Ein lokaler Semantic-Compression-Layer würde nicht einfach Namen entfernen. Er würde die Aufgabe umbauen. Er würde prüfen, welche Aspekte für das externe Reasoning notwendig sind und welche lokal bleiben müssen.

```
Abstrahierte Frontier-Anfrage
task_type: acquisition_risk_analysis
company_scale: mid_market
buyer_sector: industrial_manufacturing
seller_sector: component_supplier
transaction_stage: pre_signing_due_diligence
risk_patterns:
  - short_term_liquidity_pressure
  - supplier_concentration
  - key_person_retention_risk
  - warranty_claim_uncertainty
constraints:
  - no entity names
  - no exact dates
  - no exact revenue or valuation
  - preserve causal relations
request:
  Analyze likely risk clusters, missing diligence questions,
  mitigation options, and decision checkpoints.
```

Das Frontier-Modell bekommt nun eine echte Aufgabe, aber keinen echten Fall. Es kann Risikocluster benennen, Due-Diligence-Fragen formulieren, Mitigationsvorschläge entwickeln und Entscheidungslogik liefern. Die Antwort ist generisch genug, um nicht selbst zum Datenleck zu werden, aber strukturiert genug, damit der lokale Layer sie zurück in den echten Fall übersetzen kann.

Genau hier entsteht der Begriff Re-Hydration. Das Modell liefert abstrakte Empfehlungen. Der lokale Layer reichert sie wieder mit der echten Welt an: Namen, Verantwortlichkeiten, Fristen, Dokumente, Eskalationswege. Der zentrale Reasoner bleibt blind für den Fall, aber nützlich für die Struktur.

```
Re-Hydration lokal
Frontier-Ausgabe:
  "Prüfe Retention-Risiko für Schlüsselperson."
Lokaler Layer mappt zurück:
  Schlüsselperson = CTO Zielunternehmen
  Dokumente = HR-Interview, Vesting-Plan, Kündigungsfrist
  Owner = M&A Lead + HR Legal
  Deadline = interne Closing-Checkliste
Das Modell sieht die Person nicht.
Das Unternehmen bekommt trotzdem eine konkrete Aufgabe.
```

5.6 Der Compression Contract

Damit Semantic Compression nicht zum Bauchgefühl wird, braucht sie einen Vertrag. Dieser Vertrag beschreibt, was erhalten bleiben muss, was entfernt werden muss, was transformiert werden darf und wann eine Anfrage nicht nach außen gehen darf.

Ohne Vertrag ist Semantic Compression nur Prompt-Kosmetik. Mit Vertrag wird sie überprüfbare Laufzeitarchitektur.

Ein Compression Contract ist kein juristischer Vertrag, obwohl er juristische Anforderungen abbilden kann. Er ist ein technischer und semantischer Vertrag zwischen lokalem Kontext, Sicherheitslogik und Frontier-Eskalation. Er zwingt das System, seine eigene Bedeutungsweitergabe explizit zu machen.

```
Beispiel: Semantic Compression Contract
contract_id: ma_risk_analysis_v1
allowed_task: acquisition_risk_analysis
preserve:
  - risk class
  - causal relations
  - sequence of decision points
  - uncertainty level
  - regulatory category
remove:
  - entity names
  - project names
  - exact dates
  - exact financial figures
  - internal owner names
transform:
  - amounts -> size bands
  - dates -> relative phase
  - companies -> roles
  - people -> function labels
  - documents -> evidence categories
forbidden:
  - unpublished transaction intent
  - customer-specific commercial terms
  - privileged legal advice
verification:
  - no named entities remain
  - re-identification score below threshold
  - task utility score above threshold
escalation:
  - if utility too low: ask local human
  - if risk too high: local-only processing
```

Die sieben Prüfungen eines Compression Contracts

1. Zweckprüfung: Welche Aufgabe soll das Frontier-Modell tatsächlich lösen?
2. Relevanzprüfung: Welche Information ist für diese Aufgabe notwendig?
3. Sensitivitätsprüfung: Welche Information wäre bei Offenlegung schädlich?
4. Transformationsprüfung: Welche Details können in Rollen, Klassen, Bänder oder Relationen übersetzt werden?
5. Rekonstruktionsprüfung: Kann der echte Fall aus der abstrahierten Anfrage plausibel wiedererkannt werden?
6. Utility-Prüfung: Reicht die abstrahierte Anfrage noch aus, damit das Modell sinnvoll arbeitet?
7. Eskalationsprüfung: Wer entscheidet, wenn Schutz und Nutzen kollidieren?

Diese Prüfungen sind langweilig im besten Sinne. Sie nehmen dem System die mystische Aura. Sie machen aus KI einen kontrollierbaren Prozess. Nicht jeder Prompt darf direkt zum Modell. Nicht jede Abstraktion ist gut. Nicht jede Antwort darf ungeprüft zurück in den Geschäftsprozess.

Der Compression Contract ist damit ein enger Verwandter von Governance-Layern, Policy Engines und PASS/FAIL-Übergängen. Er beschreibt nicht nur, was das System tun soll. Er beschreibt, was es nicht wissen darf.

5.7 Stufenmodell: Wie viel Wirklichkeit darf nach außen?

Semantic Compression ist keine Ein/Aus-Entscheidung. Zwischen lokalem Schweigen und vollständigem Rohdatenexport liegt ein ganzer Fächer von Offenlegungsstufen. Ein reifes System episodischer Intelligenz müsste diese Stufen bewusst wählen.

Die niedrigste Stufe ist rein lokal. Keine Cloud. Keine externe Anfrage. Das ist sinnvoll, wenn die Aufgabe einfach ist, wenn das Risiko hoch ist oder wenn ein lokales Modell ausreicht. Danach folgen zunehmend reichere Abstraktionen. Erst die Problemklasse. Dann Rollen und Relationen. Dann sanitisierte Fallnarrative. Erst sehr spät - und nur unter starken Garantien - eine vertrauliche Cloud-Verarbeitung mit mehr Kontext.

Offenlegungsstufen für Frontier-Eskalation

Stufe	Name	Was rausgeht	Typischer Einsatz
0	Lokal-only	Keine externe Anfrage.	Sensitive Fälle, einfache Aufgaben, lokale Modelle ausreichend.
1	Problemklasse	Nur abstrakter Aufgabentyp und Ziel.	Brainstorming, allgemeine Strategiemuster.
2	Rollen + Relationen	Akteure als Rollen, kausale Beziehungen, Risikomuster.	Analysen ohne konkrete Identitäten.
3	Sanitisiertes Narrativ	Generische Fallbeschreibung mit transformierten Details.	Komplexe Fragestellungen, bei denen Struktur allein nicht reicht.
4	Vertrauliche Cloud-Verarbeitung	Mehr Kontext in TEE/PCC/Private-Processing-ähnlicher Umgebung.	Hoher Nutzen, kontrollierte technische Garantien.
5	Rohkontext	Vollständige Anfrage mit echten Details.	Nur nach bewusster Freigabe; für viele Business Cases zu vermeiden.

Dieses Stufenmodell ist wichtig, weil es den falschen Gegensatz auflöst. Es geht nicht um Cloud gegen lokal. Es geht um kontrollierte Offenlegung. Das lokale System muss nicht heldenhaft alles allein lösen. Aber es sollte die Menge an exportierter Wirklichkeit bewusst dosieren.

In vielen Unternehmen gibt es heute nur zwei unausgesprochene Zustände: gar nicht nutzen oder alles in den Prompt kippen. Episodische Intelligenz schlägt eine dritte Sprache vor: stufenweise, prüfbare, semantisch begrenzte Eskalation.

5.8 RAG ist ein Verwandter - aber nicht dasselbe

Retrieval-Augmented Generation war ein früher Hinweis darauf, dass das Modell nicht alles im eigenen Kopf haben muss. Das RAG-Paper von Lewis et al. kombiniert parametrisches Modellwissen mit nicht-parametrischem Speicher, typischerweise einem durchsuchbaren Index. Damit wird Wissen aktualisierbarer, nachvollziehbarer und stärker an externe Quellen gebunden. [K5-S7]

Für episodische Intelligenz ist RAG wichtig, aber nicht ausreichend. RAG fragt: Welchen Kontext soll ich dem Modell geben? Semantic Compression fragt davor: Welche Art von Kontext darf das Modell überhaupt sehen?

Ein lokaler RAG-Stack kann also Teil von Semantic Compression sein. Er findet relevante Dokumente. Er extrahiert Passagen. Er mappt sie auf Entitäten, Rollen und Beziehungen. Aber bevor diese Passagen an ein Frontier-Modell gehen, muss eine zweite Frage gestellt werden: Trägt diese Passage notwendige Bedeutung oder nur riskanten Ballast?

Das ist der Unterschied zwischen Kontextmaximierung und Bedeutungsminimierung. Viele RAG-Systeme sind noch auf möglichst gute Antwortqualität optimiert. Sie stopfen mehr passenden Kontext in das Modell. Semantic Compression sagt: Guter Kontext ist nicht maximaler Kontext. Guter Kontext ist minimal ausreichender Kontext.

RAG vs. Semantic Compression
RAG:
Finde relevante Dokumente und gib sie dem Modell.
Semantic Compression:
Finde relevante Bedeutung, transformiere sie,
entferne Identität und strategischen Ballast,
prüfe Rekonstruktionsrisiko,
und gib erst dann eine Aufgabe an das Modell.

5.9 Warum das auch ein Energiekapitel ist

Auf den ersten Blick ist Semantic Compression ein Datenschutzkapitel. Auf den zweiten Blick ist es ein Energiekapitel.

Jedes Dokument, das nicht übertragen wird, muss nicht über das Netz. Jeder nicht gesendete Chunk wird nicht tokenisiert. Jedes nicht erzeugte Token landet nicht im KV-Cache. Jede vermiedene Frontier-Eskalation spart GPU-Zeit, Warteschlange, Kühlung und Nachverarbeitung. Das ist keine romantische Sparsamkeit. Es ist Systemökonomie.

Die lokale Vorarbeit kostet ebenfalls Energie. Sie ist nicht gratis. Aber sie kann zeitlich verschoben werden: bei Solarüberschuss, nachts, im Leerlauf, auf vorhandener Hardware, in regionalen Clustern. Der teure zentrale Reasoner muss nicht für jede kleine Bedeutungsoperation geweckt werden.

Damit verbindet Kapitel 5 die Datenschutzfrage mit der Grundthese des Buches. Die Zukunft produktiver KI liegt nicht nur darin, stärkere Modelle zu bauen. Sie liegt darin, weniger Wirklichkeit an stärkere Modelle schicken zu müssen.

Semantic Compression macht lokale Systeme wertvoller. Sie sind nicht nur Vorstufen zur Cloud. Sie sind der Ort, an dem entschieden wird, wie viel Welt überhaupt in die Cloud wandert.

Das sparsamste Token ist nicht das billig generierte Token. Es ist das Token, das gar nicht erst gebraucht wurde.

5.10 Grenzen: Warum diese Idee gefährlich werden kann

Dieses Kapitel darf nicht in Euphorie enden. Semantic Compression ist mächtig, aber gefährlich, wenn sie falsche Sicherheit erzeugt.

Erstes Risiko: Re-Identifikation. Auch abstrahierte Merkmale können einen Fall verraten. In kleinen Branchen reichen wenige Attribute. Ein mittelständischer Maschinenbauer aus einer bestimmten Region mit einer bestimmten Lieferkettenkrise und einem bestimmten Akquisitionsmuster ist vielleicht schon eindeutig, auch ohne Namen.

Zweites Risiko: Kontextverlust. Je stärker abstrahiert wird, desto eher verliert das Modell die Besonderheiten, die für eine gute Entscheidung notwendig wären. Dann erzeugt es schöne generische Beratung. Sie klingt vernünftig und trifft den Fall doch nicht.

Drittes Risiko: lokale Fehlklassifikation. Wenn der lokale Layer die Aufgabe falsch versteht, komprimiert er in die falsche Richtung. Dann schützt das System vielleicht irrelevante Details und verrät die entscheidenden. Oder es entfernt genau jene Information, die kausal wichtig war.

Viertes Risiko: Ontologie-Bias. Jede lokale Bedeutungsstruktur spiegelt die Annahmen ihrer Erbauer. Wenn die Ontologie zu grob ist, werden Fälle falsch eingeordnet. Wenn sie zu spezifisch ist, kann sie selbst verräterisch werden.

Fünftes Risiko: Audit-Illusion. Ein schön formulierter Compression Contract kann wie Kontrolle aussehen, ohne echte Tests zu haben. Dann entsteht Compliance-Theater: Kästchen sind abgehakt, aber die semantische Leakage bleibt.

Sechstes Risiko: Verarbeitungsumgebung. Selbst wenn die semantische Anfrage reduziert ist, können technische Umgebungen weitere Risiken tragen: Logs, Metriken, Timing, Seiteneffekte, Modellverbesserung, Tool-Aufrufe, externe Websuche. Privacy-Enhancing Technologies sind laut OECD kein Allheilmittel; sie müssen kombiniert werden und bringen Zielkonflikte zwischen Nutzen, Effizienz und Benutzbarkeit. [K5-S8]

Siebtes Risiko: Die Falle der vollständigen Abstraktion. Wenn alles nur noch Rollen und Klassen sind, kann das Modell nicht mehr erkennen, dass ein Fall moralisch, politisch oder rechtlich besonders ist. Wirklichkeit lässt sich nicht beliebig verlustfrei in Symbole pressen.

Gegenprüfung: Was Kritiker einwenden werden

- Ein starkes Modell braucht oft Details; abstrakte Prompts können zu generischen Antworten führen.
- In spezialisierten Märkten kann auch anonymisierte Struktur identifizierbar bleiben.
- Lokale Compression-Layer erhöhen Komplexität, Wartungsaufwand und Fehlermöglichkeiten.
- Ein falsch konfigurierter Layer kann gefährlicher sein als ein bewusst kontrollierter Rohprompt.
- TEEs, PCC-ähnliche Systeme und Private Processing lösen andere Teile des Problems und bleiben für manche Fälle unverzichtbar.
- Die Energieersparnis hängt davon ab, ob lokale Vorverarbeitung wirklich effizienter ist als zentrale Verarbeitung.

Diese Gegenargumente sind nicht lästig. Sie sind die Stützpfeiler der Glaubwürdigkeit. Das Buch sollte sie nicht wegwischen, sondern in das Design aufnehmen.

Ein gutes System episodischer Intelligenz wird deshalb nicht immer maximale Compression wählen. Es wird angemessene Compression wählen. Das ist ein entscheidender Unterschied.

5.11 Leakage messen: nicht nur Namen zählen

Semantic Compression scheitert nicht erst, wenn ein echter Name in der Anfrage steht. Sie scheitert schon dann, wenn die abstrahierte Anfrage mit hoher Wahrscheinlichkeit auf den echten Fall zurückführt. Ein Named-Entity-Check ist deshalb nur die unterste Schicht. Er findet Namen, Orte, Kundennummern, E-Mail-Adressen und Projekttitel. Er findet aber nicht automatisch die Kombination aus Branche, Zeitpunkt, Marktbewegung, Rollenstruktur und Problemklasse.

Ein reifer Leakage-Test arbeitet in Stufen. Zuerst prüft er Oberfläche: direkte Identifikatoren. Dann prüft er Quasi-Identifikatoren: seltene Kombinationen, markante Ereignisse, besondere Relationen. Danach prüft er Rekonstruktion: Kann ein zweites Modell, ein Analyst oder ein Angreifer aus der abstrahierten Anfrage

plausible reale Kandidaten ableiten? Erst zuletzt kommt die Utility-Frage: Ist die Anfrage noch nützlich genug?

Prüfung	Frage	Typische Methode
Identifier Check	Sind direkte Namen, IDs, Orte, Personen oder Projekttitel enthalten?	NER, RegEx, Secret Scanner, DLP-Regeln
Quasi-Identifier Check	Verrät die Kombination aus Merkmalen den Fall?	Kombinationsscore, Seltenheitsprüfung, Branchen-/Zeitfenster-Analyse
Adversarial Reconstruction	Kann ein zweites Modell den realen Kontext erraten?	Red-Team-Prompts, Kandidatenlisten, Rekonstruktionsversuche
Utility Check	Bleibt genug Bedeutung für sinnvolles Reasoning?	Aufgabenbenchmark, Expertenreview, Antwortvergleich
Decision Gate	Darf diese Abstraktion nach außen?	Schwellenwerte, Human Gate, Eskalationsvertrag

Der entscheidende Punkt ist unbequem: Es gibt keinen magischen Schalter, der Bedeutung sicher anonym macht. Es gibt nur messbare Reduktion, dokumentierte Restrisiken und klare Grenzen, ab wann eine Anfrage lokal bleiben muss.

5.12 Zweites Beispiel: Maschinenbau ohne echte Margen

M&A ist ein gutes Beispiel, aber nicht das einzige. Näher an vielen europäischen Unternehmen liegt der Maschinenbau: Ein mittelständischer Hersteller will ein Angebot für einen Großkunden prüfen. Der echte Kontext enthält Kundennamen, Lieferanten, technische Sonderlösungen, Margen, Ersatzteilstrategie, Liefertermine, Vertragsstrafen und interne Einschätzungen über die Verhandlungsmacht des Kunden.

Eine rohe Anfrage wäre gefährlich: Sie würde nicht nur Personen oder Firmen verraten, sondern das ökonomische Nervensystem des Angebots. Die semantisch komprimierte Anfrage könnte dagegen so aussehen: Ein mittelständischer Anlagenbauer prüft ein kundenspezifisches Angebot mit hoher Engineering-Abweichung, angespanntem Lieferzeitfenster, begrenzter Preissetzungsmacht und erhöhtem Risiko von Nachlaufkosten. Entwickle eine strukturierte Prüfung für Risiko, Marge, vertragliche Schutzklauseln und Eskalationspunkte.

Das Frontier-Modell sieht dann nicht den Kunden, nicht die Zeichnung, nicht die Marge und nicht den Lieferanten. Es sieht das Muster. Es kann Fragen formulieren, Risiken strukturieren und Verhandlungslogik liefern. Die konkrete Re-Hydration - also die Rückübersetzung in echte Namen, Preise und Termine - bleibt lokal.

Rohkontext	Lokale Transformation	Darf nach außen?
Kundenname und Region	Kundentyp, Marktmacht, Größenklasse	Nur abstrahiert
Exakte Marge	Marge unter Zielkorridor / in Zielkorridor / über Zielkorridor	Nur als Klasse
Lieferant und Einkaufspreis	Lieferkettenrisiko und Preisvolatilität	Nur als Muster
Spezifische technische Lösung	Engineering-Abweichung und Komplexitätsklasse	Nur wenn nicht IP-relevant
Interne Verhandlungsnotiz	Verhandlungsrisiko, Abhängigkeit, Eskalationsbedarf	Nur stark verdichtet oder lokal

5.13 Drei Arten von Geheimnis

Semantic Compression wird klarer, wenn man drei Arten von Geheimnis unterscheidet. Personenbezogene Daten betreffen identifizierbare Menschen. Geschäftsgeheimnisse betreffen Werte, Prozesse, Preise, Technologien, Kundenbeziehungen oder operative Details eines Unternehmens. Strategische Absichten betreffen das, was eine Organisation vorhat: kaufen, verkaufen, angreifen, sparen, restrukturieren, kündigen, verhandeln.

Diese drei Kategorien überlappen, aber sie sind nicht gleich. Eine Anfrage kann ohne personenbezogene Daten hochsensibel sein. Und sie kann ohne Geschäftsgeheimnis trotzdem strategisch gefährlich sein, weil sie Richtung und Timing einer Organisation verrät.

Geheimnistyp	Beispiel	Compression-Ziel
Personenbezogene Daten	Name, E-Mail, Gesundheitsangabe, Mitarbeiterrolle	Identifizierbarkeit entfernen oder lokal halten
Geschäftsgeheimnis	Marge, Rezeptur, Lieferant, Code, Kundenliste	Werttreiber abstrahieren, konkrete Details lokal halten
Strategische Absicht	Akquisition, Preiserhöhung, Markteintritt, Restrukturierung	Absichtsklasse und Risikoform statt konkreter Plan
Kombinationsrisiko	Branche + Zeitpunkt + Rollen + Ereignis	Seltenheit prüfen und Offenlegungsstufe senken

5.14 Kleine lokale Modelle als Kompressionsarbeiter

Semantic Compression muss nicht vollständig regelbasiert sein. Kleine lokale Modelle können helfen, Rohtext in Rollen, Relationen, Risikoarten, Fristen, Entitäten und Abstraktionsklassen zu übersetzen. Gerade hier sind sie wertvoll, weil sie nicht weltbewegend klug sein müssen. Sie müssen zuverlässig sortieren, maskieren, klassifizieren und Grenzen erkennen.

Der lokale Kompressionsarbeiter hat eine andere Aufgabe als ein Frontier-Modell. Er soll nicht die endgültige Strategie liefern. Er soll den Rohkontext so formen, dass ein stärkeres Modell nützlich arbeiten kann, ohne die volle Wirklichkeit zu sehen. Das ist eine bescheidenere, aber sehr wichtige Intelligenzform.

5.15 Compression Contracts versionieren und auditieren

Wenn Compression Contracts nur in Prompttexten leben, werden sie unbeherrschbar. Sie brauchen Versionen. Eine Anfrage muss später beantworten können: Nach welchem Contract wurde sie komprimiert? Welche Entitäten wurden entfernt? Welche Bedeutungsmerkmale wurden behalten? Welche Offenlegungsstufe wurde gewählt? Wer oder was hat sie freigegeben?

Contract-Feld	Zweck
contract_id / version	Nachvollziehbarkeit und Reproduzierbarkeit
data_classes	Welche Datenarten waren im Rohkontext betroffen?
preserve	Welche Bedeutung muss erhalten bleiben?
remove	Welche Identifikatoren und Details müssen verschwinden?
transform	Welche Werte werden in Klassen, Rollen oder Relationen übersetzt?
leakage_score	Messwert für Rekonstruktionsrisiko
utility_score	Messwert für Nützlichkeit
approval_gate	Automatisch, Human Gate oder lokal bleiben

Versionierung ist kein bürokratischer Luxus. Sie verhindert, dass die Organisation später nicht mehr weiß, welche Wirklichkeit sie herausgegeben hat.

5.16 Re-Hydration: der gefährlichste Moment

Die komprimierte Anfrage reist nach außen. Die Antwort kommt zurück. Jetzt muss sie wieder in den echten Fall übersetzt werden. Dieser Schritt ist gefährlich, weil hier die abstrakte Antwort wieder mit Namen, Preisen, Dokumenten, Rollen und Entscheidungen verbunden wird.

Re-Hydration sollte deshalb lokal und kontrolliert geschehen. Die Mapping-Tabellen gehören nicht in die Cloud. Sie sollten kurzlebig sein, projektgebunden, verschlüsselt und getrennt vom Modellaufruf gespeichert. In Hochrisikofällen braucht Re-Hydration ein Human Gate: Nicht weil Menschen alles besser wissen, sondern weil dieser Moment Wirkung erzeugt.

5.17 Die Energie-Gegenrechnung

Semantic Compression spart nicht automatisch Energie. Sie erzeugt lokale Arbeit: Klassifikation, Maskierung, Leakage-Prüfung, Contract-Erstellung, vielleicht mehrere kleine Modellaufrufe. Aber sie kann größere Energieblöcke vermeiden: lange Rohprompts, wiederholte Fehlversuche, unnötige Frontier-Eskalationen, umfangreiche Context Windows und zentrale Retrieval-Schleifen.

Die seriöse Gegenrechnung lautet deshalb nicht: Compression ist immer billiger. Sie lautet: Compression lohnt sich, wenn die lokale Vorarbeit weniger kostet als der vermiedene große Kontext, die vermiedene Eskalation, die vermiedene Wiederholung und das vermiedene Offenlegungsrisiko.

Kostenblock	Compression erhöht	Compression senkt
Lokale CPU/NPU/kleines Modell	Ja, zusätzliche Vorarbeit	Nein
Token im Frontier-Call	Nein	Oft deutlich, weil Rohkontext schrumpft
Wiederholte Rückfragen	Kann steigen, wenn zu stark abstrahiert	Sinkt, wenn Contracts gute Aufgaben formen
Datentransfer	Minimal lokal	Sinkt, wenn Dokumente nicht exportiert werden
Governance-Aufwand	Steigt vorne	Sinkt hinten durch klare Audit-Spur
Risiko	Nicht null	Sinkt, wenn Rekonstruktion schwerer wird

5.18 Ein Architekturvorschlag für episodische Intelligenz

Eine produktive Semantic-Compression-Architektur könnte aus neun Schichten bestehen. Jede Schicht ist bewusst unspektakulär. Zusammen ergeben sie ein System, das nicht auf Vertrauen in einen einzelnen Prompt angewiesen ist.

1. Local Identity and Rights Layer: Wer fragt, und worauf darf diese Person zugreifen?

2. Sensitivity Classifier: Welche Datenarten, Geschäftsgeheimnisse oder regulatorischen Kategorien sind betroffen?

3. Local Retrieval: Welche internen Dokumente, Graphen oder Datenpunkte sind wirklich relevant?

4. Semantic Extractor: Welche Entitäten, Rollen, Relationen, Risiken, Ziele und Constraints stecken im Kontext?

5. Compression Contract Engine: Was muss erhalten, entfernt, transformiert oder verboten werden?

6. Leakage Evaluator: Wie wahrscheinlich ist Re-Identifikation oder Offenlegung des echten Falls?

7. Utility Evaluator: Reicht die abstrahierte Aufgabe noch für sinnvolles Reasoning?

8. Frontier Escalation Gateway: Welche Stufe der Offenlegung wird gewählt, und mit welchem Modell?

9. Local Verification and Re-Hydration: Wird die Antwort geprüft, auf den echten Fall zurückgeführt und sicher protokolliert?

```
Pipeline: Semantic Compression Gateway
input: raw_case_context + user_goal
1. authorize(user, documents)
2. classify_sensitivity(raw_case_context)
3. retrieve_minimal_relevant_context(user_goal)
4. extract_semantic_graph(context)
5. apply_compression_contract(graph, task_type)
6. score_leakage(abstract_request)
7. score_utility(abstract_request)
8. if leakage_high: local_only_or_human_review()
9. if utility_low: request_more_local_context()
10. call_frontier_model(abstract_request)
11. verify_response_against_contract()
12. rehydrate_response_into_local_business_context()
```

13. audit_decision_without_logging_secrets()

Diese Pipeline ist nicht als fertiger Standard gemeint. Sie ist eine Denkfigur. Aber sie macht sichtbar, dass Semantic Compression kein Textfilter am Ende ist. Sie ist ein Gateway zwischen Wirklichkeit und Frontier-Kognition.

Die eigentliche Kunst liegt in Schritt 6 und 7: Leakage und Utility gleichzeitig zu messen. Zu viel Schutz macht die Antwort nutzlos. Zu viel Nützlichkeit kann Schutz zerstören. Semantic Compression lebt in diesem Spannungsfeld. Sie ist kein statisches Verfahren, sondern ein Regler.

5.19 Warum Unternehmen diese Schicht wollen werden

Viele Unternehmen stehen heute vor einem Dilemma. Sie sehen, dass Frontier-Modelle nützlich sind. Gleichzeitig wissen sie, dass ihre wertvollsten Daten gerade nicht in fremde Systeme gehören: Kundensituationen, Preislogiken, interne Prozesse, Produktpläne, juristische Einschätzungen, Schwachstellen, Verhandlungspositionen.

Die billigste Antwort lautet: Dann nutzt eben keine externen Modelle. Das ist sauber, aber oft unrealistisch. Die zweite Antwort lautet: Dann vertraut eben dem Anbieter. Das ist bequem, aber strategisch dünn. Die dritte Antwort ist Episodische Intelligenz: Nutzt externe Reasoning-Leistung episodisch, aber baut eine lokale semantische Kontrollschicht davor.

Das wird besonders interessant für KMU, regulierte Branchen und europäische Unternehmen. Sie brauchen nicht immer das größte Modell als Dauerhirn. Sie brauchen oft eine Architektur, die lokale Wirklichkeit schützt und trotzdem von globaler Modellintelligenz profitieren kann.

Semantic Compression kann hier zu einem Verkaufsargument werden, aber auch zu einem internen Governance-Prinzip. Die Frage in einem AI-Review lautet dann nicht mehr nur: Welchen Anbieter nutzen wir? Sondern: Welche Bedeutung verlassen wir? In welcher Stufe? Mit welchem Vertrag? Mit welcher Prüfung? Mit welchem Audit?

5.20 Der poetische Kern: Das Modell sieht den Schatten, nicht den Körper

Vielleicht ist dies die einfachste Metapher: Der lokale Layer hält den Körper der Wirklichkeit. Das Frontier-Modell sieht nur den Schatten, den Umriss, die Richtung des Lichts. Es erkennt, ob eine Gestalt fällt, ob ein Risiko wächst, ob ein Muster bekannt ist. Aber es kennt nicht den Namen des Menschen, der dort steht.

Das ist kein Rückzug aus KI. Es ist eine Reifung. Der erste Hype wollte alles in das Modell legen: alle Daten, alle Aufgaben, alle Entscheidungen, alle Kontexte. Episodische Intelligenz fragt: Was muss wirklich zentral gedacht werden? Was kann lokal entschieden werden? Was darf nur als Schatten wandern?

Semantic Compression ist damit nicht nur Datenschutztechnik. Sie ist eine Haltung zur Infrastruktur. Sie sagt: Die Welt gehört nicht automatisch dem größten Modell. Die Welt darf lokal bleiben. Nur ihre abstrakte Frage reist weiter.

Und vielleicht ist genau das die nächste Form von KI-Souveränität: nicht alles selbst rechnen müssen, aber selbst entscheiden, welche Bedeutung man herausgibt.

Die Frontier bleibt der Hochofen. Aber der Rohstoff, der hineingeht, wird nicht mehr blind hineingekippt. Er wird lokal geprüft, geformt, verdichtet - und manchmal gar nicht erst eingeschmolzen.

Begriff	Arbeitsdefinition
Semantic Compression	Transformation von Rohkontext in minimal ausreichende, weniger verräterische Bedeutung für externe Reasoning-Systeme.
Semantic Least Privilege	Ein Modell erhält nur die Bedeutung, die es für eine konkrete Aufgabe benötigt.
Compression Contract	Explizite Regeln, welche semantischen Merkmale erhalten, entfernt, transformiert oder verboten werden.

Re-Hydration	Lokale Rückübersetzung einer abstrakten Modellantwort in den echten Business-Kontext.
Disclosure Level	Stufe der kontrollierten Offenlegung gegenüber Cloud- oder Frontier-Systemen.
Leakage Score	Abschätzung, ob die abstrahierte Anfrage den echten Fall rekonstruierbar macht.
Utility Score	Abschätzung, ob die abstrahierte Anfrage noch nützliche Modellantworten ermöglicht.

Quellenanker für Kapitel 5

Die folgenden Quellenanker sind noch keine finale Zitierweise. Sie sollen die weitere Recherche führen und im späteren Lektorat in ein einheitliches Literaturformat überführt werden.

[K5-S1] Apple Security Research: Private Cloud Compute - A new frontier for AI privacy in the cloud (2024). Link - Kernquelle für stateless computation, enforceable guarantees, no privileged runtime access, non-targetability und verifiable transparency.

[K5-S2] Apple Private Cloud Compute Security Guide. Link - Technische Dokumentation zu PCC und seinen Sicherheitsanforderungen.

[K5-S3] Meta AI: Private Processing for WhatsApp Overview - Technical Whitepaper (2025). Link - Kernquelle für TEEs, Remote Attestation, OHTTP, anonyme Credentials, stateless und forward secure processing.

[K5-S4] Meta Engineering: Building Private Processing for AI tools on WhatsApp (2025). Link - Ein zugänglicherer Architekturüberblick zu Meta Private Processing.

[K5-S5] Gilbert et al.: Semantic Compression With Large Language Models (2023). Link - Forschungsanker für semantische Verdichtung, Rekonstruktion und intent preservation.

[K5-S6] Fei et al.: Extending Context Window of Large Language Models via Semantic Compression (2023). Link - Forschungsanker für Semantic Compression als Kontextfenster-Erweiterung und Reduktion semantischer Redundanz.

[K5-S7] Lewis et al.: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks (2020). Link - Grundlage für die Einordnung von RAG als verwandte, aber andere Architekturidee.

[K5-S8] OECD: Sharing trustworthy AI models with privacy-enhancing technologies (2025). Link - Überblick zu Privacy-Enhancing Technologies und ihren Zielkonflikten zwischen Privacy, Utility, Effizienz und Usability.

[K5-S9] Trail of Bits: PCC - Bold step forward, not without flaws (2024). Link - Nützliche Gegenquelle: PCC ist stark, aber nicht identisch mit Ende-zu-Ende-Verschlüsselung oder Homomorphic Encryption; Daten werden in einer geschützten Umgebung verarbeitet.

[K5-S10] European Data Protection Supervisor: Data minimisation glossary entry. Link - Regulatorischer Anker für Datensparsamkeit als Architekturprinzip.

[K5-S11] LangGraph Documentation: overview / durable execution / human-in-the-loop. Link - Beispiel für Laufzeitarchitekturen mit Zustand, Persistenz, Human-in-the-Loop und kontrollierten Übergängen.

[K5-S12] NIST Privacy Framework. Link - Allgemeiner Privacy-Engineering-Anker für systematische Privacy-Risikosteuerung.

[K5-S13] ICO: Anonymisation and pseudonymisation guidance (2025). Relevanter Anker für Re-Identifikationsrisiko, Pseudonymisierung und organisatorische Maßnahmen. https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-sharing/anonymisation/

[K5-S14] ICO: Pseudonymisation. Verweist ausdrücklich auf Re-Identifikation als Sicherheits-/Wirksamkeitstest von Anonymisierung und Pseudonymisierung. https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-sharing/anonymisation/pseudonymisation/

[K5-S15] NIST Privacy Framework. Privacy-Engineering-Anker für strukturierte Risikosteuerung, Datenverarbeitung und organisatorische Verantwortlichkeit. https://www.nist.gov/privacy-framework

[K5-S16] OpenAI Structured Outputs. Architekturanker für maschinenprüfbare Ausgabeformate, die Compression Contracts operationalisierbar machen können. https://platform.openai.com/docs/guides/structured-outputs

[K5-S17] Green Software Foundation: Software Carbon Intensity. Methodischer Anker dafür, Energie-/Carbon-Fragen pro funktionaler Einheit zu messen statt pauschal zu behaupten. https://sci.greensoftware.foundation/

Übergang: Semantische Verdichtung schützt Kontext. Nun folgt die operative Frage: Wann ist der Moment gekommen, trotzdem die große Instanz zu rufen?

Kapitel 6 - Frontier auf Abruf

Warum große Modelle nicht verschwinden, sondern gezielter gerufen werden

Frontier-Intelligenz wird wertvoller, wenn sie seltener, bewusster und begründeter aktiviert wird.

Nach dem Energieproblem und dem verborgenen Maschinenraum stellt sich eine einfache Frage: Wenn jede Anfrage Kosten, Latenz, Wärme, Risiko und Kontextbewegung erzeugt, warum behandeln wir das stärkste Modell dann so oft als Default?

Die Antwort ist historisch verständlich. Die ersten großen Erlebnisse mit generativer KI waren Chat-Erlebnisse. Man schrieb etwas in ein Textfeld, ein großes Modell antwortete, und die Oberfläche wirkte wie ein direkter Draht zu einer allgemeinen Intelligenz. Für Demos war das ideal. Für Infrastruktur ist es ein schlechtes Betriebssystem.

Dieses Kapitel schlägt einen anderen Blick vor: Frontier-Modelle werden in produktiven Systemen nicht verschwinden. Sie werden wichtiger. Aber gerade weil sie wichtig sind, sollten sie nicht für alles benutzt werden. Sie sollten wie Spezialinstanzen behandelt werden: selten aktiviert, sauber begründet, mit einem klaren Auftrag, einem Budget, einem Ausgabeformat und einer Prüfung danach.

Das klingt zunächst nach Einschränkung. In Wahrheit ist es eine Aufwertung. Ein starkes Modell wird nicht dadurch respektiert, dass man es für triviale Standardfälle verheizt. Man respektiert es, indem man es dort einsetzt, wo seine besondere Fähigkeit wirklich gebraucht wird: bei Ambiguität, Konflikt, neuartigen Mustern, strategischer Abwägung und Grenzfällen.

Nicht jede Anfrage braucht das Großhirn. Manche brauchen nur einen Reflex. Manche brauchen Gedächtnis. Manche brauchen einen Spezialisten.

6.1 Das alte Bild: Das stärkste Modell macht alles

Das naive KI-System hat nur eine ernsthafte Entscheidung: Es schickt möglichst viel Kontext an das stärkste verfügbare Modell. Diese Architektur ist einfach zu erklären, schnell zu bauen und in frühen Experimenten oft erstaunlich wirksam. Sie ist aber auch teuer, schwer zu kontrollieren und energetisch grob.

In einem solchen System wird das große Modell zum universellen Ansprechpartner. Es beantwortet FAQs, fasst Protokolle zusammen, sortiert Tickets, schreibt E-Mails, ruft Tools auf, analysiert Verträge und entscheidet über Eskalationen. Das klingt nach Generalität. In der Praxis ist es oft Verschwendung. Ein Cache hätte die FAQ beantwortet. Ein deterministischer Parser hätte die Daten extrahiert. Ein lokales kleines Modell hätte die E-Mail umformuliert. Ein Rechtecheck hätte den Zugriff vorher verweigert.

Die stärkste Rechenschicht wird dadurch nicht zum Gehirn, sondern zum Müllschlucker. Sie bekommt alles, was die Architektur vorher nicht sauber sortiert hat.

Produktive Systeme werden sich davon lösen müssen. Nicht aus romantischem Lokalismus. Aus nüchterner Betriebslogik. Wer jedes Problem an die teuerste Instanz weiterleitet, baut kein intelligentes System. Er baut einen sehr teuren Mangel an Vorentscheidung.

```
naive Architektur:

User -> größtes Modell -> Antwort

bessere Frage:

Welche Instanz muss diese Anfrage wirklich sehen?
```

6.2 Was episodisch bedeutet

Episodische Frontier-Kognition bedeutet nicht, dass ein großes Modell selten aus technischen Gründen erreichbar ist. Es bedeutet, dass seine Aktivierung Teil eines bewussten Systemzustands wird.

Ein Frontier-Call passiert dann nicht, weil das System keine andere Idee hat. Er passiert, weil vorherige Schichten einen begründeten Fall erzeugt haben: Die Anfrage ist wichtig genug, unklar genug, neu genug oder riskant genug, um stärkere Kognition zu rechtfertigen.

Das Wort episodisch ist hier absichtlich gewählt. Es beschreibt eine zeitlich begrenzte Aktivierung. Das System lebt nicht permanent im Hochleistungsmodus. Es wechselt in ihn hinein und wieder heraus. Davor gibt es lokale Wahrnehmung, Routing, Retrieval, Kompression, Policy und Budgetierung. Danach gibt es Validierung, Speicherung, Audit und eventuell menschliche Freigabe.

Damit verändert sich die Rolle des Modells. Es ist nicht mehr der Ort, an dem die gesamte Anwendung stattfindet. Es ist eine Phase in einem Prozess.

```
local runtime
  -> classify
  -> retrieve / cache / tool
  -> compress context
  -> check uncertainty
  -> escalate only if needed
  -> verify result
  -> persist local state
```

Der Frontier-Call ist nicht die Antwort. Er ist ein kontrollierter Moment in einer längeren Maschine.

6.3 Kaskaden: erst billig, dann stark

Die technische Literatur hat für diese Idee bereits mehrere Namen: Model Cascades, LLM Routing, Hybrid Inference, adaptive Inference. Die Grundidee ist immer ähnlich. Nicht jede Anfrage wird sofort an das größte Modell geschickt. Das System versucht zuerst billigere, kleinere oder spezialisierte Wege. Nur wenn diese nicht ausreichen, eskaliert es.

FrugalGPT beschreibt diesen Ansatz sehr direkt. Die Autoren unterscheiden Prompt-Anpassung, LLM-Approximation und LLM-Kaskaden als Wege, um Inferenzkosten zu senken. In ihren Experimenten konnte FrugalGPT laut Paper die Leistung des besten einzelnen Modells mit bis zu 98 Prozent geringeren Kosten erreichen oder bei gleichem Kostenbudget die Genauigkeit verbessern. Diese Zahl ist kein universelles Versprechen für jedes Produktivsystem. Aber sie zeigt die Richtung: Die teuerste Instanz muss nicht immer zuerst kommen. [Q1]

Andere Arbeiten beschreiben Routing und Cascading als Auswahlproblem zwischen Modellen unterschiedlicher Kosten und Fähigkeiten. Eine Kaskade stoppt, wenn ein Zwischenmodell ein ausreichend gutes Ergebnis liefert. Wenn nicht, geht die Anfrage weiter an eine stärkere Stufe. Das ist technisch schlicht. Strategisch ist es tief: Qualität entsteht nicht mehr nur durch ein einzelnes Modell, sondern durch die richtige Entscheidungskette. [Q2]

Hybrid-LLM-Ansätze formulieren dasselbe Problem aus einer Edge- und Kostenperspektive: Kleine Modelle können billig und lokal laufen, große Modelle liefern bessere Qualität bei schweren Aufgaben. Ein Router entscheidet, welcher Pfad angemessen ist. Genau das ist der Kern episodischer Intelligenz. [Q3]

```
answer = small_model(request)

if contract_pass(answer) and uncertainty < threshold:
```

```
    return answer
context = compress_for_frontier(request, local_state)
answer = frontier_model(context)
return verify_and_commit(answer)
```

Der entscheidende Punkt ist nicht die konkrete Implementierung. Der entscheidende Punkt ist das Stoppkriterium. Ein System muss wissen, wann genug gedacht wurde.

6.4 Der Eskalationsvertrag

Wenn ein lokales System an die Frontier eskaliert, sollte es nicht einfach einen Rohprompt weiterreichen. Es sollte einen Eskalationsvertrag erzeugen. Dieser Vertrag beschreibt, warum die Eskalation nötig ist, was das große Modell sehen darf, welches Ziel verfolgt wird, welches Ausgabeformat erwartet wird und welche Grenzen gelten.

Ein solcher Vertrag ist etwas anderes als ein langer Prompt. Er ist ein strukturierter Übergang zwischen Systemschichten. Das lokale System sagt nicht: Bitte denk mal. Es sagt: Ich habe diesen Fall klassifiziert, diese Unsicherheiten gefunden, diese Daten verdichtet, diese Pfade bereits ausgeschlossen, dieses Risiko erkannt, und ich benötige jetzt Reasoning unter diesen Bedingungen.

Damit wird die Frontier-Kognition kontrollierbarer. Das Modell bekommt weniger Rohdaten, aber mehr Struktur. Es bekommt nicht die ganze Welt des Unternehmens, sondern einen präzisen Ausschnitt mit Auftrag und Grenzen.

```
frontier_escalation = {
  problem_class: "supplier_risk_after_acquisition",
  business_context: "mid_market_manufacturing",
  local_attempts: ["retrieval", "rule_check", "small_model_summary"],
  unresolved_conflicts: ["cashflow_vs_delivery_dependency"],
  privacy_level: "abstracted_business_case",
  allowed_outputs: ["risk_scenarios", "questions_for_human_review"],
  forbidden_outputs: ["legal_advice", "final_transaction_decision"],
  max_reasoning_budget: "deep",
  require_verification: true
}
```

Das ist die operative Schwester von Semantic Compression aus Kapitel 5. Dort ging es darum, Rohkontext in abstrahierte Bedeutung zu verwandeln. Hier geht es darum, diese abstrahierte Bedeutung als Eskalationsfall zu verwenden.

Der Vertrag schützt also gleichzeitig drei Dinge: das Unternehmen vor unnötiger Offenlegung, das System vor unkontrollierter Modellnutzung und das Modell vor schlecht geformten Aufgaben.

6.5 Wann Frontier wirklich gebraucht wird

Es gibt Aufgaben, für die ein Frontier-Modell zu teuer und zu offen ist. Und es gibt Aufgaben, bei denen ein schwächeres System fahrlässig wäre. Die Kunst liegt nicht darin, alles lokal zu machen. Die Kunst liegt darin, den richtigen Moment für Eskalation zu erkennen.

Frontier-Kognition ist besonders wertvoll, wenn mehrere Bedingungen zusammenkommen: hoher Kontextkonflikt, unsichere Zieldefinition, strategische Folgen, seltene Muster, widersprüchliche Quellen oder die Notwendigkeit, mehrere Domänen gleichzeitig zu verbinden.

Ein einfaches Beispiel: Ein Kunde fragt nach dem Passwort-Reset. Das ist kein Frontier-Fall. Ein anderer Kunde fragt, ob ein Produktionsausfall mit einer Vertragsklausel, einer Lieferantenabhängigkeit und einer laufenden Akquisition zusammenhängt. Das ist ein anderer kognitiver Raum. Hier kann das große Modell Muster verbinden, Hypothesen strukturieren und gute Rückfragen vorschlagen.

Typische Eskalationsgründe:

- Die lokale Runtime erkennt hohe Unsicherheit trotz Retrieval.
- Die Entscheidung hat finanzielle, rechtliche oder sicherheitsrelevante Folgen.
- Mehrere Wissensbereiche widersprechen sich oder müssen gemeinsam bewertet werden.
- Der Fall ist selten genug, dass deterministische Regeln nicht tragen.
- Das System soll nicht nur antworten, sondern alternative Szenarien und Gegenargumente entwickeln.
- Ein Mensch braucht eine strukturierte Entscheidungsgrundlage, keine automatische Entscheidung.

Gerade der letzte Punkt ist wichtig. Episodische Frontier-Kognition bedeutet nicht, dass das große Modell die Entscheidung übernimmt. Häufig besteht seine wertvollste Rolle darin, eine menschliche Entscheidung besser vorzubereiten.

6.6 Test-Time Compute: Denken als variable Tiefe

Bisher klingt Eskalation nach einer Modellwahl: kleines Modell oder großes Modell. Die nächste Verschiebung ist noch interessanter. Es geht nicht nur darum, welches Modell antwortet. Es geht darum, wie viel Rechenarbeit für eine konkrete Antwort erlaubt wird.

Die Forschung zu Test-Time Compute untersucht genau diese Frage: Wie verbessert sich die Leistung eines Sprachmodells, wenn es bei der Inferenz mehr Rechenbudget erhält? Snell et al. formulieren das als Trade-off zwischen Inferenzzeit-Compute und Pretraining-Compute. Damit wird Denken nicht mehr als fester Vorgang betrachtet, sondern als skalierbarer Modus. [Q4]

Für episodische Intelligenz ist das zentral. Ein System kann nicht nur zwischen lokal und Frontier unterscheiden. Es kann innerhalb der Frontier unterschiedliche Denkmodi anfordern: flach, mittel, tief. Eine einfache Zusammenfassung braucht keine lange Such- oder Prüfphase. Eine schwierige strategische Frage vielleicht schon.

```
reasoning_budget:
  shallow  -> direct answer / low ambiguity
  medium   -> compare options / check assumptions
  deep     -> multi-step analysis / adversarial review / scenario building
```

Damit wird Rechenzeit selbst zu einem Gegenstand der Architektur. Nicht jedes Problem bekommt denselben Denkraum. Das System vergibt Denkbudget wie eine knappe Ressource.

Das ist philosophisch schöner, als es zunächst klingt. Es bedeutet: Intelligenz ist nicht nur die Fähigkeit zu antworten. Intelligenz ist auch die Fähigkeit, zu erkennen, wie viel Antwortarbeit eine Situation verdient.

6.7 Sparse Activation: Die Idee lebt bereits im Modell

Interessanterweise passiert etwas Ähnliches bereits innerhalb moderner Modellarchitekturen. Mixture-of-Experts-Modelle aktivieren nicht für jeden Input denselben vollständigen Parameterraum. Ein Routing-Mechanismus wählt Expertenteile aus. Das Modell wird dadurch groß in der Kapazität, aber sparsamer in der aktiven Berechnung.

Der Switch Transformer beschreibt diese Idee als sparsely activated model: Viele Parameter existieren, aber pro Beispiel wird nur ein Teil aktiviert. Das Paper berichtet deutliche Pretraining-Speedups bei gleichem Ressourcenbudget und betont zugleich die Herausforderungen von Routing, Kommunikation und Stabilität. [Q5]

Mixtral 8x7B machte diese Logik später im Open-Weight-Bereich sichtbar. Mistral beschreibt Mixtral als sparse mixture of experts model und hebt den Kosten-/Performance-Aspekt hervor. [Q6]

Auch Apple beschreibt seine neueren Foundation Models als Kombination aus einem kompakten On-Device-Modell und einem serverseitigen Modell mit Parallel-Track Mixture-of-Experts für Private Cloud Compute. Das ist keine direkte Bestätigung episodischer Intelligenz, aber ein starkes Signal: Moderne KI-Systeme organisieren Leistung zunehmend selektiv. [Q7]

Routing außen. Routing innen. Vielleicht ist das die heimliche Grammatik effizienter KI.

Die Architektur außerhalb des Modells beginnt also, die innere Logik moderner Modelle zu spiegeln. Innen entscheidet ein Router, welche Experten feuern. Außen entscheidet eine Runtime, welche Schicht aktiviert wird: Cache, Tool, lokales Modell, kleines Cloud-Modell, Frontier-Modell, Mensch.

Das ist einer der stärksten Gedanken dieses Buches: Die Zukunft der KI ist nicht nur größer. Sie ist bedingter. Nicht alles feuert immer. Bedeutung entsteht durch Auswahl.

6.8 Spekulatives Denken: kleine Entwürfe, große Prüfung

Speculative Decoding zeigt ein weiteres verwandtes Muster. Ein kleineres oder schnelleres Draft-Modell schlägt Token vor, ein größeres Zielmodell prüft sie. Ziel ist, die serielle Langsamkeit autoregressiver Generierung zu reduzieren, ohne das Ausgabeverhalten des Zielmodells zu verändern. [Q8]

Für unser Buch ist nicht jedes technische Detail wichtig. Wichtig ist das Muster: Leichte Recheninstanzen können Vorarbeit leisten, während schwere Instanzen selektiv prüfen oder korrigieren. Das ist nicht dasselbe wie Business-Routing, aber es folgt derselben ökonomischen Intuition.

Man könnte sagen: Das große Modell muss nicht jeden ersten Gedanken selbst haben. Es kann prüfen, verwerfen, bestätigen oder vertiefen. Schon hier wird die teuerste Kognition entlastet.

6.9 Edge und Private Cloud: Reflex und Cortex

Die großen Plattformen bewegen sich ebenfalls in Richtung hybrider Schichten. Apple beschreibt seine Foundation Models als Zusammenspiel aus On-Device-Modellen, die für Apple Silicon optimiert sind, und serverseitigen Modellen für Private Cloud Compute. 2025 nennt Apple ein kompaktes Modell mit ungefähr 3 Milliarden Parametern sowie ein serverseitiges Mixture-of-Experts-Modell. [Q7]

Google beschreibt Private AI Compute als Weg, Gemini-Cloud-Modelle für persönlichere AI-Erlebnisse zu nutzen, während die Nutzerdaten innerhalb einer privaten, isolierten Verarbeitungsschicht geschützt bleiben sollen. Auch hier ist das Signal nicht: Alles bleibt lokal. Das Signal ist: Die Cloud wird nur dann attraktiv, wenn sie als kontrollierte Erweiterung lokaler oder persönlicher Kontexte erscheint. [Q9]

Diese Entwicklungen sind keine Beweise, dass Episodische Intelligenz exakt so kommen muss. Aber sie zeigen, dass die Industrie selbst an der Grenze zwischen lokaler Verarbeitung und episodischer Cloud-Leistung arbeitet. On-device allein reicht für viele Aufgaben nicht. Unkontrollierte Cloud allein ist für viele Kontexte nicht akzeptabel. Dazwischen entsteht Architektur.

```
device / local layer  -> reflex, memory, privacy, routine
regional layer        -> retrieval, shared services, background work
frontier layer        -> rare ambiguity, deep reasoning, synthesis
human layer           -> accountable judgment
```

6.10 Die Budgetmaschine

Episodische Frontier-Kognition ist letztlich Budgetierung. Nicht nur Geldbudget. Ein gutes System verwaltet mehrere Budgets gleichzeitig: Energie, Latenz, Datenschutz, Genauigkeit, Risiko, Autonomie und Aufmerksamkeit.

Das klingt nach Verwaltung, ist aber ein Kernstück produktiver Intelligenz. Ein System, das seine eigenen Budgets nicht kennt, kann nicht intelligent eskalieren. Es kann nur hoffen, dass ein großes Modell schon alles richten wird.

```
request_budget = {
  latency_ms: 1500,
  privacy_level: "business_sensitive",
  energy_priority: "low",
  quality_target: "high",
  human_review_required: true,
  frontier_allowed: "only_after_local_failure"
}
```

Hier wird die Deterministik aus Kapitel 3 zur Voraussetzung der Frontier-Nutzung. Ohne klare Budgets wird Eskalation willkürlich. Mit Budgets wird sie steuerbar.

Man könnte auch sagen: Die Runtime entscheidet nicht nur, welches Modell antwortet. Sie entscheidet, welcher Teil der Welt belastet wird: das lokale Gerät, der regionale Cluster, das Datacenter, ein Mensch, ein späterer Hintergrundprozess.

6.11 Die Gefahr der falschen Sparsamkeit

Dieses Kapitel darf nicht klingen wie ein billiger Trick zur Kostensenkung. Es gibt eine reale Gefahr: Systeme können zu sparsam werden. Sie können starke Modelle zu spät einschalten, falsche Sicherheit erzeugen oder schwierige Fälle mit zu schwachen Mitteln behandeln.

Das ist besonders gefährlich, wenn die lokale Runtime ihre eigene Unsicherheit schlecht kalibriert. Ein Router, der falsch routet, kann mehr Schaden anrichten als ein teurer Default. Ein kleines Modell, das souverän klingt, aber eine wichtige Nuance übersieht, ist kein Fortschritt. Eine Kaskade, die nur auf Kosten optimiert, wird zum Qualitätsrisiko.

Darum braucht episodische Frontier-Kognition Gegenmittel: Evals, Unsicherheitsmessung, Audit Logs, Human Gates, Fehleranalysen, rote Linien und klare Eskalationsregeln. Effizienz darf nicht mit Mutlosigkeit verwechselt werden. Das System muss nicht nur wissen, wann es sparen darf. Es muss wissen, wann Sparen gefährlich wird.

Warnsignale für Unter-Eskalation:

- Das lokale System gibt Antworten ohne überprüfbare Konfidenz.
- Kosten werden stärker optimiert als Fehlerfolgen.
- High-Risk-Fälle werden wie Routine behandelt.
- Die Runtime kann nicht erklären, warum sie nicht eskaliert hat.
- Nutzer verlieren die Möglichkeit, eine stärkere Prüfung anzufordern.

Ein gutes System spart nicht am Denken. Es spart am falschen Denken.

6.12 Wann Frontier nicht gebraucht wird

Die andere Seite ist genauso wichtig. Es gibt viele Fälle, in denen Frontier-Kognition nicht nur unnötig, sondern schlechter ist als deterministische oder lokale Verarbeitung. Ein großes Modell kann elegant formulieren, aber es macht einen Rechtecheck nicht automatisch besser. Es kann eine Rechnung erklären, aber es sollte sie nicht als freie Sprachwahrscheinlichkeit ausführen. Es kann eine Policy paraphrasieren, aber es ersetzt keine überprüfbare Policy Engine.

Routinefälle, wiederkehrende Workflows, formatierte Extraktion, Standardantworten, einfache Transformationen, Berechtigungsentscheidungen und regulierte Prozessschritte sind oft keine guten Frontier-Fälle. Dort zählen Wiederholbarkeit, Auditierbarkeit und eindeutige Grenzen mehr als kreative Generalisierung.

Typische Nicht-Frontier-Fälle:

- FAQ-Antworten mit stabilem Inhalt und hohem Cache-Treffer.
- Berechtigungsprüfung, Mandantentrennung, Rollenlogik.
- Extraktion aus bekannten Formularen oder festen Datenformaten.
- Standardisierte Zusammenfassungen mit geringem Risiko.
- Tool-Aufrufe mit klaren Parametern und deterministischer Validierung.
- Hintergrundaufgaben, die zeitlich verschiebbar sind.

Das ist keine Geringschätzung der KI. Es ist die Rückkehr eines alten Ingenieurprinzips: Nutze das einfachste Werkzeug, das die Aufgabe zuverlässig erfüllt.

6.13 Die episodische Runtime

Aus all dem entsteht ein Bild einer Runtime, die zwischen mehreren kognitiven Ebenen vermittelt. Sie ist nicht nur API-Gateway. Sie ist Budgetmaschine, Datenschutzschicht, Gedächtnisverwaltung, Tool-Orchestrator, Gatekeeper und Eskalationslogik.

Eine solche Runtime könnte für jede Anfrage einen Zustand erzeugen. Dieser Zustand enthält nicht nur den Text des Nutzers, sondern die operative Bedeutung des Falls: Wer fragt? Worum geht es? Welche Daten dürfen benutzt werden? Welche Qualität ist nötig? Wie riskant ist ein Fehler? Welche lokalen Wege wurden bereits versucht? Was kostet eine Eskalation?

```
User Request
  -> Local Intake
  -> Policy + Identity + Scope
  -> Cache / Retrieval / Tools
  -> Small Model or Rule Engine
  -> Contract Check
      PASS -> local response
      FAIL -> semantic compression
  -> Frontier Escalation
  -> Verification / Human Gate
  -> Local Memory Update
```

Der Unterschied zu einer normalen Chat-App ist grundlegend. Im Chat-Paradigma verschwindet vieles im Prompt. In der episodischen Runtime wird der Prozess selbst sichtbar und steuerbar.

Das ist auch die Voraussetzung für Energieopportunismus, den spätere Kapitel vertiefen. Wenn das System weiß, welche Aufgaben dringend sind und welche nicht, kann es Arbeit zeitlich verschieben: Embeddings bei Solarüberschuss, Evals nachts, Indexpflege im Leerlauf, Frontier-Eskalation nur bei Bedarf.

6.14 Das Datacenter als Hochenergie-Spezialist

Die Konsequenz ist nicht, dass Frontier-Datacenter verschwinden. Das wäre naiv. Sie bleiben notwendig für Training, globale Modelle, wissenschaftliches Reasoning, sehr große multimodale Systeme und Aufgaben, die lokale Systeme nicht stemmen können.

Aber ihre Rolle kann sich verändern. Aus dem permanenten Zentrum jeder Anfrage wird eine Hochenergie-Spezialinstanz. Ein Datacenter ist dann weniger das Alltagsgehirn und mehr der Teilchenbeschleuniger: teuer, mächtig, unverzichtbar - aber nicht für jeden Routinevorgang zuständig.

Diese Verschiebung ist architektonisch, ökonomisch und politisch relevant. Wenn das teuerste Modell nur noch episodisch aktiviert wird, sinkt nicht nur die einzelne Rechnung. Es verändert sich die Machtverteilung im System. Mehr Bedeutung bleibt lokal. Mehr Kontext bleibt geschützt. Mehr Vorarbeit wird dort erledigt, wo sie entsteht.

Das ist der tiefere Sinn episodischer Intelligenz. Es ist keine Anti-Cloud-Idee. Es ist eine Theorie der richtigen Entfernung. Manche Dinge gehören nah an den Nutzer, manche in regionale Infrastruktur, manche in den Schwarm, manche in das Frontier-Zentrum und manche zurück zu Menschen.

6.15 Eskalationskriterien: Wann der Hochofen angeht

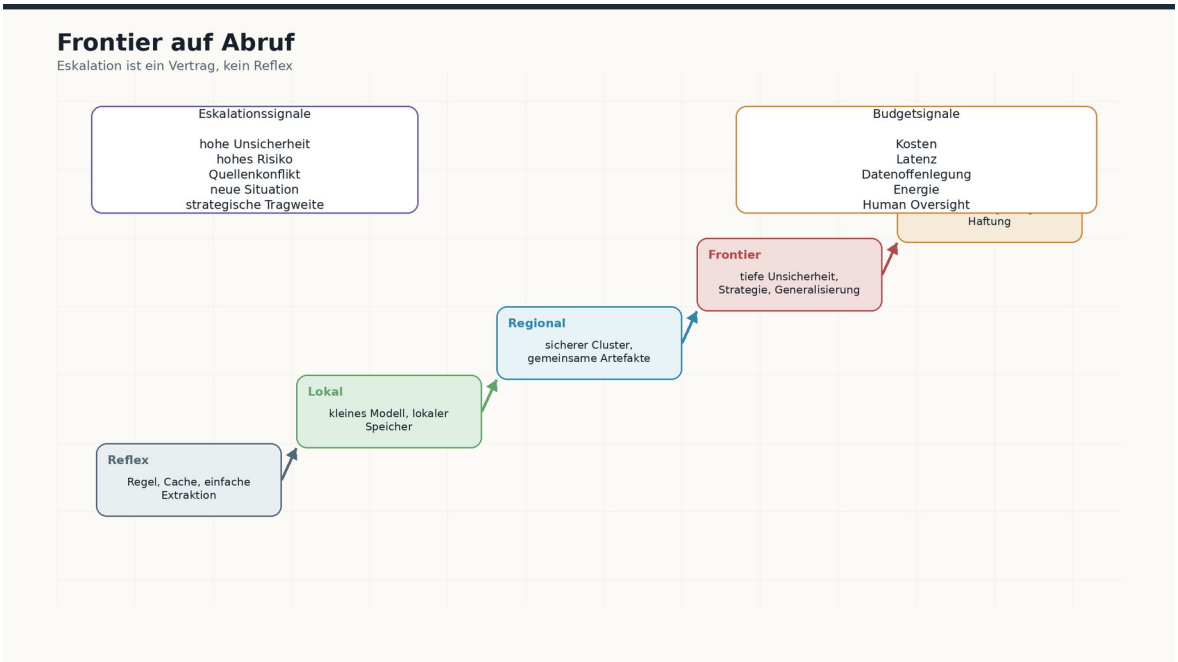

Abb. 7 - Frontier auf Abruf: Eskalation als Vertrag zwischen Risiko, Unsicherheit, Budget und Verantwortung.

Eine episodische Architektur braucht Kriterien, sonst wird sie zur Stimmungslage. Ein System darf nicht nach Bauchgefühl entscheiden, ob es lokal bleibt, regional arbeitet, ein spezialisiertes Modell nutzt, zur Frontier eskaliert oder einen Menschen einschaltet.

Tabelle 6.1 — Eskalationskriterien für episodische Intelligenz

Signal	Frage	Standardreaktion	Fehler bei falscher Reaktion
Unsicherheit	Widersprechen sich Quellen, Modelle oder Checks?	zweite Prüfung, dann Frontier oder Human Gate	lokale Antwort wirkt sicherer, als sie ist
Risiko	Hat die Antwort finanzielle, rechtliche, medizinische oder sicherheitsrelevante Folgen?	enger Output-Contract, Human Gate oder Frontier	billige Antwort wird teuer
Datenklasse	Müssten Rohdaten offengelegt werden?	Semantic Compression oder lokal bleiben	Frontier sieht mehr Business-Kontext als nötig
Neuheit	Ist der Fall außerhalb bekannter Muster?	Frontier-Reasoning oder Expert Review	Cache und Regeln konservieren alte Blindheit
Zeitdruck	Muss jetzt gehandelt werden?	lokaler Reflex, später tiefe Analyse	zu langsame Eskalation blockiert Betrieb
Kosten/Energie	Ist tiefe Kognition verhältnismäßig?	Budgetpfad wählen, ggf. warten	jeder Fall wird zum Hochenergieereignis
Accountability	Wer trägt die Verantwortung?	Entscheidungsvorlage statt Auto-Action	Mensch wird erst nach dem Schaden informiert

Diese Kriterien sind keine starre Ampel. Sie sind ein Betriebsvertrag. Die Frontier wird nicht aus Reflex angerufen, sondern aus begründeter Notwendigkeit.

6.16 Fallstudie: Produktionsstörung bei einer Kundenanlage

Ein mittelständischer Maschinenbauer erhält eine Meldung: Eine Kundenanlage stoppt unregelmäßig nach Temperaturspitzen. Der Kunde droht mit Vertragsstrafe, der Service hat alte Tickets, die Konstruktion

vermutet einen Sensorfehler, der Vertrieb fürchtet Eskalation. Ein naives System würde alle Daten in einen großen Prompt kippen.

Die episodische Runtime arbeitet anders.

Tabelle 6.2 — Ein Eskalationspfad bei technischer Unsicherheit

Ebene	Arbeit	Daten	Ergebnis
Lokaler Reflex	Fehlercode, Wartungsplan und bekannte Störung prüfen	lokale Service-Datenbank, Cache	bekannte Ursachen ausschließen oder bestätigen
Lokales Gedächtnis	ähnliche Tickets, Bauteilrevisionen und Temperaturprofile suchen	Vektorstore, Logs, freigegebene Dokumente	Kontextpaket mit Quellen und Versionsständen
Spezialist / Tool	Deterministische Grenzwerte und Sensorhistorie prüfen	Messwerte, Stückliste, Regelwerk	technische Hypothesen mit Evidenz
Frontier	neuartige Kausalkette oder widersprüchliche Befunde analysieren	abstrahiertes Problemprofil, keine Rohkundendaten	Szenarien, Fragen, nächste Tests
Human Gate	Vertragsfolgen und Kundenkommunikation entscheiden	Entscheidungsvorlage	verantwortete Aktion statt Modellaktion

Der Frontier-Call erscheint hier nicht am Anfang, sondern in der Mitte. Er bekommt nicht „alles“, sondern ein verdichtetes Problemprofil: Symptom, technische Hypothesen, Konflikte, bisherige Evidenz, offene Fragen. Das ist keine Schwächung der Frontier. Es ist eine bessere Aufgabenstellung.

6.17 Der Eskalationsvertrag

Damit diese Logik nicht im Prompt verschwindet, braucht die Runtime einen Eskalationsvertrag. Er beschreibt, wann eskaliert werden darf, welche Daten mitgehen, welches Ergebnis erwartet wird und wann ein Mensch übernehmen muss.

```
escalation_contract:
  trigger: uncertainty_high | risk_high | novelty_high | source_conflict
  data_mode: raw_local | compressed | anonymized | metadata_only
  allowed_context: evidence_pack + open_questions + constraints
  forbidden_context: names, secrets, negotiation details, credentials
  expected_output: hypotheses, test_plan, risk_summary, confidence_bounds
  stop_conditions: missing_sources | forbidden_data | action_request
  next_gate: validation | human_review | local_followup
```

Ein solcher Vertrag ist nüchtern. Aber genau diese Nüchternheit schützt die poetischere Idee: Die Frontier wird nicht als Orakel behandelt, sondern als tiefes Werkzeug mit klarer Aufgabe.

6.18 Das Architekturbild: Reflex, Gedächtnis, Region, Frontier, Mensch

```
User / Prozessereignis
  -> lokaler Reflex: Regeln, Cache, bekannte Muster
  -> lokales Gedächtnis: Dokumente, Vektoren, Historie
  -> regionale Verarbeitung: Spezialmodelle, Fachtools, freigegebene Cluster
  -> Frontier: tiefe Generalisierung bei Unsicherheit oder Neuheit
  -> Human Gate: Verantwortung, Freigabe, Haftung, Kommunikation
  -> Audit / Learning Loop: Was darf gelernt, gespeichert, verbessert werden?
```

Dieses Bild ist absichtlich biologisch. Auch ein Körper aktiviert nicht jeden Teil des Nervensystems für jede Berührung. Reflexe laufen lokal. Erinnerung wird selektiv aktiviert. Bewusste Aufmerksamkeit ist teuer. Verantwortung ist noch teurer. Intelligenz entsteht nicht durch permanente Maximalaktivierung, sondern durch passende Aktivierung.

6.19 Warum Anbieter trotzdem zentralisieren wollen

Eine ehrliche Analyse muss die Gegenposition stark machen. Zentralisierung ist nicht nur Machtgier. Sie hat reale technische und wirtschaftliche Gründe: weniger Integrationsvarianten, bessere Telemetrie, einfachere Missbrauchserkennung, konsistentere Modellupdates, höhere Auslastung teurer Infrastruktur und ein klareres Produktversprechen.

Tabelle 6.3 — Warum Zentralisierung attraktiv bleibt

Zentralisierungsgrund	Warum er verständlich ist	Warum er für Nutzer gefährlich werden kann
Produktvereinfachung	ein Endpoint, ein Verhalten, wenig lokale Komplexität	Default wird zur Abhängigkeit
Qualitätskontrolle	Anbieter kann Modell, Tools und Safety gemeinsam optimieren	lokale Kontexte und Sonderfälle verschwinden
Telemetrie	Fehler, Missbrauch und Performance werden schneller sichtbar	Nutzungsdaten werden zur zweiten Lernkurve
Auslastung	teure Cluster müssen wirtschaftlich laufen	jede Anfrage wird in Richtung Hochenergiepfad gezogen
Monetarisierung	Zugang, Tokens und Plattform werden berechenbar verkauft	Kunde verliert Wechseloptionen und Verhandlungsmacht

Episodische Intelligenz ist deshalb keine naive Dezentralisierungsromantik. Sie ist eine Verhandlungsposition: zentral dort, wo es stark ist; lokal dort, wo Nähe, Datenschutz, Energiezeit oder Verantwortung wichtiger sind.

6.20 Verbindung zu Energiezeitfenstern

Nicht jede Eskalation muss live sein. Akute Entscheidungen brauchen schnelle Pfade. Aber viele vorbereitende Arbeiten - Evals, Embedding-Refresh, Fallvergleich, Cache-Warming, Kompressionsprüfung - können warten. Kapitel 9 nennt das energieopportunistische KI. Kapitel 6 liefert die kognitive Logik dazu: Live ist nur das, was wirklich live sein muss.

Dadurch entsteht ein Rhythmus. Lokale Systeme handeln sofort, wenn es nötig ist. Schwärme und regionale Cluster arbeiten, wenn Energie, Preis und Netz günstiger sind. Die Frontier wird zugeschaltet, wenn Tiefe zählt. Der Mensch entscheidet, wo Verantwortung nicht delegierbar ist.

6.21 Schluss: Der richtige Ort des Denkens

Episodische Frontier-Kognition fragt nicht, wie man große Modelle vermeidet. Sie fragt, wie man sie würdig einsetzt.

In einer unreifen Architektur ist das stärkste Modell der erste Ansprechpartner. In einer reiferen Architektur ist es die Instanz, die aufgerufen wird, wenn lokale Wahrnehmung, Gedächtnis, Regeln und kleine Modelle an eine sinnvolle Grenze kommen.

Das ist vielleicht die wichtigste Verschiebung: Intelligenz sitzt nicht mehr an einem einzigen Ort. Sie verteilt sich auf Reflexe, Speicher, Regeln, Werkzeuge, lokale Modelle, Verträge, Menschen und seltene Frontier-Momente. Der Wert entsteht aus dem Zusammenspiel.

Die teuerste Kognition steht dann nicht mehr permanent im Zentrum. Sie wird episodisch zugeschaltet - wie Licht in einem Raum, der nur dann hell werden muss, wenn dort wirklich gearbeitet wird.

Episodische Intelligenz fragt nicht: Wie viel KI können wir einschalten? Es fragt: Welche Schicht muss jetzt denken?

Quellenanker für Kapitel 6

Q1 - Chen, Lingjiao; Zaharia, Matei; Zou, James: FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. arXiv / TMLR, 2023/2024. URL: https://arxiv.org/abs/2305.05176

Q2 - Dekoninck, Jasper et al.: A Unified Approach to Routing and Cascading for LLMs. arXiv, 2024. URL: https://arxiv.org/html/2410.10347

Q3 - Ding, Daoyuan et al.: Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing. ICLR 2024. URL: https://proceedings.iclr.cc/paper_files/paper/2024/file/b47d93c99fa22ac0b377578af0a1f63a-Paper-Conference.pdf

Q4 - Snell, Charlie et al.: Scaling LLM Test-Time Compute Optimally Can be More Effective than Scaling Model Parameters. arXiv, 2024. URL: https://arxiv.org/abs/2408.03314

Q5 - Fedus, William; Zoph, Barret; Shazeer, Noam: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. arXiv, 2021/2022. URL: https://arxiv.org/abs/2101.03961

Q6 - Mistral AI: Mixtral of Experts. 2023. URL: https://mistral.ai/news/mixtral-of-experts

Q7 - Apple Machine Learning Research: Apple Intelligence Foundation Language Models / 2025 Updates. 2024/2025. URLs: https://machinelearning.apple.com/research/introducing-apple-foundation-models und https://machinelearning.apple.com/research/apple-foundation-models-tech-report-2025

Q8 - Leviathan, Yaniv; Kalman, Matan; Matias, Yossi: Fast Inference from Transformers via Speculative Decoding. arXiv, 2022. URL: https://arxiv.org/abs/2211.17192

Q9 - Google: Private AI Compute: our next step in building private and helpful AI. 2025. URL: https://blog.google/innovation-and-ai/products/google-private-ai-compute/

Q10 - Anthropic: Building Effective Agents. 2024. URL: https://www.anthropic.com/research/building-effective-agents

Q11 - OpenAI: Prompt Caching. API Documentation. URL: https://developers.openai.com/api/docs/guides/prompt-caching

Q12 - OpenAI: Structured Outputs and Function Calling. API Documentation. URLs: https://developers.openai.com/api/docs/guides/structured-outputs und https://developers.openai.com/api/docs/guides/function-calling

Q13 - Moslem et al.: Dynamic Model Routing and Cascading for Efficient LLM Inference. Survey, 2026. Wichtig für aktuelle Routing- und Cascading-Paradigmen über unabhängige Modelle hinweg. URL: https://arxiv.org/html/2603.04445v1

Q14 - OpenAI: Model Spec. Relevant für Chain of Command und die Abgrenzung zwischen Plattform-, Entwickler- und Nutzeranweisungen bei eskalierenden Systemen. URL: https://model-spec.openai.com/

Übergang: Wenn Frontier nur noch episodisch gerufen wird, muss der Übergang selbst prüfbar sein. Nicht der Prompt trägt Verantwortung, sondern der Vertrag um ihn herum.

Kapitel 7 - Prüfbare Übergänge

Contracts, Gates und die ausführbare Seite von Vertrauen

Zuverlässige KI entsteht nicht durch bessere Prompts allein. Sie entsteht durch prüfbare Übergänge.

Nach Kapitel 2 wissen wir, dass KI einen Körper hat: Strom, Wärme, Kühlung, Netzwerk, Infrastruktur. Nach Kapitel 3 wissen wir, dass produktive KI nicht aus einem einzelnen Modellaufruf besteht, sondern aus einer Kette von Routing, Retrieval, Caching, Tools, Validierung und Eskalation. Nach Kapitel 6 wissen wir, dass Frontier-Modelle nicht verschwinden, aber nicht dauerhaft als Standardinstanz brennen müssen.

Dieses Kapitel geht einen Schritt tiefer. Es fragt nicht: Wie bekommen wir eine intelligente Antwort? Es fragt: Unter welchen Bedingungen darf ein KI-System überhaupt handeln, antworten, eskalieren oder stoppen?

Damit verlassen wir die Oberfläche des Chatfensters. Wir betreten die Runtime. Dort entscheidet sich, ob ein KI-System nur beeindruckend wirkt oder tatsächlich betreibbar ist. Dort entstehen Gates, Contracts, Zustände, Protokolle, Budgets, Eskalationen und Prüfpfade. Dort wird aus einem Modell eine Maschine.

Governance klingt nach Gremium, Formular und grauem Teppichboden. In KI-Systemen darf sie das nicht bleiben. Governance muss ausführbar werden. Sie muss in den Betrieb wandern. Sie muss an den Stellen sitzen, an denen ein System Daten auswählt, Werkzeuge öffnet, Entscheidungen vorbereitet, Menschen einbindet und externe Modelle kontaktiert.

Governance ist kein Dokument neben dem System. Governance ist der Teil des Systems, der entscheidet, was das System darf.

7.1 Governance ist nicht Compliance-Theater

Viele Organisationen behandeln KI-Governance wie eine nachträgliche Beruhigungsschicht. Man schreibt Richtlinien, benennt Verantwortliche, sammelt Risiko-Excel-Tabellen und hofft, dass die eigentliche Anwendung sich schon benehmen wird. Das reicht nicht mehr.

Ein generatives System ist kein klassisches Formular. Es reagiert auf Sprache, Kontext, Daten, Werkzeuge, Rollen, frühere Schritte und Modellunsicherheit. Es kann in einer Minute harmlos zusammenfassen und in der nächsten Minute ein Ticket erstellen, eine Zahlung vorbereiten, eine Datenbankabfrage auslösen oder eine rechtliche Einschätzung formulieren. Die Grenze zwischen Antwort und Handlung wird dünner.

Deshalb muss Governance näher an die Ausführung. Sie darf nicht erst im Audit auftauchen, wenn der Schaden bereits entstanden ist. Sie muss vor dem Modellaufruf, während des Modellaufrufs und nach dem Modellaufruf wirken.

Die relevanten Standards und Regulierungsansätze zeigen in dieselbe Richtung, auch wenn sie unterschiedliche Sprachen sprechen. NIST beschreibt KI-Risikomanagement als Zyklus aus Govern, Map, Measure und Manage. ISO/IEC 42001 beschreibt ein Managementsystem für Entwicklung, Bereitstellung und Nutzung von KI. Der EU AI Act verlangt für Hochrisiko-Systeme unter anderem technische Dokumentation, Logging, Transparenz, menschliche Aufsicht, Robustheit und Cybersecurity. Diese Anforderungen sind nicht nur juristische Kästchen. Sie sind Architekturhinweise.

Der Fehler wäre, sie als Papierpflichten zu lesen. Der interessante Gedanke ist: Was passiert, wenn wir sie als Runtime-Anforderungen lesen?

```
paper governance:
  policy document
```

```
  review board
  annual audit
runtime governance:
  state machine
  access control
  output contract
  risk gate
  log event
  human interrupt
  rollback path
```

Episodische Intelligenz braucht diese zweite Form. Nicht weil Regulierung schön ist. Sondern weil ein System, das mit Modellen, Tools und Daten arbeitet, ohne Runtime-Governance nicht zuverlässig genug wird.

7.2 Der Contract ist stärker als der Prompt

Der Prompt ist eine Bitte. Der Contract ist eine Grenze.

Das klingt hart, aber es ist wichtig. Ein Prompt kann ein Modell anweisen, ein bestimmtes Format zu verwenden, vorsichtig zu sein oder keine vertraulichen Daten auszugeben. Aber ein Prompt allein ist kein Sicherheitsmechanismus. Er ist Teil der Kommunikation mit dem Modell. Ein Contract dagegen ist eine prüfbare Erwartung an den Übergang zwischen zwei Systemschichten.

Ein Contract sagt nicht nur, was das Modell tun soll. Er sagt, was das System akzeptiert. Er beschreibt Eingaben, Ausgaben, erlaubte Werkzeuge, Datenumfang, Risikoebene, Eskalationsbedingungen und Abbruchkriterien. Ein Contract ist deshalb näher an Softwarearchitektur als an Prompt Engineering.

Das ist der entscheidende Unterschied: Der Prompt versucht, Verhalten zu erzeugen. Der Contract entscheidet, ob Verhalten akzeptiert wird.

```
output_contract = {
  task: "classify_supplier_risk",
  allowed_labels: ["low", "medium", "high", "uncertain"],
  required_fields: ["label", "evidence", "confidence", "next_gate"],
  forbidden_fields: ["customer_name", "secret_terms", "final_decision"],
  max_confidence_without_evidence: 0.60,
  fail_if_missing_evidence: true,
  human_review_if: ["high", "uncertain"]
}
```

OpenAIs Structured Outputs zeigen die Richtung: Modelle können an ein JSON Schema gebunden werden, sodass Ausgaben eine definierte Struktur einhalten. Das ist ein wichtiger Schritt, aber er löst nicht das gesamte Governance-Problem. Ein Schema sagt, ob die Form stimmt. Es garantiert nicht, dass die Aussage wahr, angemessen, sicher oder geschäftlich erlaubt ist.

Darum braucht ein produktives System mehrere Verträge. Es braucht Formatverträge, Datenverträge, Toolverträge, Risikoverträge und Entscheidungsverträge. Das klingt sperrig. In Wahrheit ist es die Rückkehr der klassischen Informatik in die KI.

Ein gutes KI-System vertraut dem Modell nicht blind. Es nimmt das Modell ernst genug, seine Ausgabe zu prüfen.

7.3 PASS/FAIL ist die kleinste Einheit der Verlässlichkeit

PASS/FAIL klingt primitiv. Genau deshalb ist es so wertvoll.

Ein Sprachmodell arbeitet mit Wahrscheinlichkeiten. Ein produktives System braucht Übergänge. Es muss wissen: Darf dieser Zustand weiter? Muss er zurück? Muss er zu einem Menschen? Muss er in ein stärkeres Modell? Muss er gestoppt werden?

PASS/FAIL ist keine Beleidigung der Intelligenz. Es ist die Stelle, an der Intelligenz betrieblich wird. Ein System, das jede Modellantwort einfach weiterreicht, hat keine Kontrolle. Ein System, das jede Antwort gegen einen Contract prüft, beginnt Verhalten zu formen.

Die wichtigsten PASS/FAIL-Prüfungen sind oft unspektakulär: Ist das Format gültig? Sind Pflichtfelder vorhanden? Wurden verbotene Daten erwähnt? Ist eine Quelle angegeben? Ist die Rolle des Nutzers berechtigt? Wurde ein Tool-Aufruf korrekt parametrisiert? Überschreitet die Aufgabe das erlaubte Risiko?

Diese Prüfungen sind deterministisch. Sie sind billig. Sie sind wiederholbar. Und sie schaffen etwas, das ein Modell allein nur schwer liefern kann: eine klare Entscheidung über den nächsten Schritt.

```
result = model(contracted_prompt)
if not schema_valid(result):
    fail("invalid_format")
if contains_forbidden_data(result):
    fail("data_leak")
if result.confidence < threshold:
    route("frontier_or_human")
if result.label == "high_risk":
    require_human_gate()
pass_to_next_state(result)
```

Der Trick liegt nicht darin, dass jede Prüfung perfekt ist. Der Trick liegt darin, dass jede Prüfung explizit wird. Ein implizites Bauchgefühl des Modells wird in einen expliziten Systemzustand übersetzt. Genau dort beginnt Governance.

7.4 Gates: Wo Autonomie endet

Ein Gate ist ein Übergang mit Bedingungen. In klassischen Systemen sind Gates überall: Login, Rechteprüfung, Vier-Augen-Freigabe, Zahlungsgrenze, Deployment-Freigabe, Change-Management. In KI-Systemen kehren diese Prinzipien zurück, nur an neuen Stellen.

Ein Gate kann vor einem Modell liegen: Darf dieser Kontext überhaupt an das Modell? Es kann nach einem Modell liegen: Darf diese Antwort an den Nutzer? Es kann vor einem Tool liegen: Darf das Modell diese Aktion auslösen? Es kann vor einem Menschen liegen: Muss hier jemand entscheiden?

Gates sind besonders wichtig, wenn KI-Systeme agentischer werden. Ein Chatbot, der nur Text erzeugt, kann schon Probleme verursachen. Ein Agent, der Werkzeuge nutzt, E-Mails schreibt, Datenbanken abfragt oder Code ausführt, braucht schärfere Grenzen. Die OWASP Top 10 für LLM-Anwendungen nennen unter anderem Prompt Injection, Insecure Output Handling, Excessive Agency, Sensitive Information Disclosure und Supply-Chain-Risiken. Das sind keine Randprobleme. Das sind Hinweise darauf, wo Gates sitzen müssen.

Ein Gate ist nicht dasselbe wie ein Verbot. Es ist eine Architekturstelle, an der ein System langsamer werden darf, damit es nicht dümmer wird.

```
gate_tool_call(tool, args, state):
  require user_role in tool.allowed_roles
  require state.risk_class <= tool.max_risk
  require args match tool.schema
  require no secrets in args
  require cost_budget_remaining
  if tool.side_effect == "external":
      require human_approval
  return allow_or_block
```

In Systemen episodischer Intelligenz wird genau diese Gate-Logik zentral. Lokale Systeme dürfen viel vorbereiten. Frontier-Modelle dürfen bei Unsicherheit helfen. Tools dürfen Arbeit erledigen. Aber Übergänge zwischen diesen Ebenen müssen kontrolliert sein. Sonst entsteht kein hybrides System, sondern ein sehr teures Improvisationstheater.

7.5 Runtime States: Kein Agent ohne Zustand

Viele sogenannte Agenten sind eigentlich nur Chatfenster mit Werkzeugliste. Sie wirken aktiv, aber sie wissen oft nicht sauber, in welchem Zustand sie sich befinden. Sie haben keinen stabilen Begriff von Aufgabe, Risiko, Budget, Berechtigung, Zwischenergebnis oder Abbruchbedingung.

Ein echter produktiver Agent braucht State. Nicht im esoterischen Sinne von „Gedächtnis“, sondern im nüchternen Sinne einer ausführbaren Zustandsmaschine. Was wurde angefragt? Was wurde geprüft? Welche Daten sind erlaubt? Welche Werkzeuge sind offen? Welche Entscheidung steht aus? Welche Unsicherheit bleibt?

LangGraph beschreibt Durable Execution und Human-in-the-Loop genau in dieser Richtung: Workflows speichern ihren Fortschritt, können pausieren, fortgesetzt werden und erlauben menschliche Eingriffe in den Zustand. Das ist wichtig, weil viele echte Geschäftsprozesse nicht in einem Chat-Turn abgeschlossen werden. Sie laufen über Minuten, Tage oder Wochen. Sie brauchen Unterbrechung, Wiederaufnahme und nachvollziehbare Zustände.

```
case_state = {
  case_id: "MNA-2026-041",
  phase: "risk_review",
  requester_role: "analyst",
  data_scope: "abstracted",
  model_budget: "medium",
  gates_passed: ["auth", "retrieval", "privacy_filter"],
  open_gates: ["legal_review", "human_approval"],
  unresolved: ["supplier_dependency", "cashflow_conflict"],
  next_allowed_actions: ["ask_clarifying_question", "draft_scenarios"],
  forbidden_actions: ["approve_deal", "contact_counterparty"]
}
```

Der Zustand ist nicht Beiwerk. Er ist die Stelle, an der ein System weiß, was es gerade ist. Ohne Zustand gibt es keine echte Verantwortung. Nur eine Folge von Texten.

Ein Agent ohne Zustand ist kein Agent. Er ist ein sehr überzeugendes Vergessen.

7.6 Authority: Die Hierarchie der Anweisungen

Generative Systeme bekommen Anweisungen aus vielen Quellen: Plattform, Entwickler, Organisation, Nutzer, Tool-Ergebnis, Dokument, Webseite, E-Mail, früherer Chat. Diese Anweisungen können einander widersprechen. Ein Nutzer kann etwas verlangen, das gegen interne Regeln verstößt. Ein Dokument kann eine versteckte Anweisung enthalten. Ein Tool kann Daten liefern, die wie ein Befehl aussehen. Eine E-Mail kann das Modell auffordern, andere Sicherheitsregeln zu ignorieren.

Die Frage lautet deshalb: Welche Anweisung hat Vorrang?

OpenAIs Model Spec beschreibt dafür eine „Chain of Command“: Anweisungen mit höherer Autorität sollen niedrigere übersteuern. Das Prinzip ist architektonisch bedeutsam, unabhängig von einem konkreten Anbieter. KI-Systeme brauchen eine explizite Autoritätshierarchie. Sonst entscheidet am Ende das Modell situativ, welchem Text es glaubt.

```
authority_order:
  platform_policy
  organization_policy
  developer_contract
  user_request
  retrieved_context
  tool_output
  untrusted_external_text
```

Diese Hierarchie muss nicht nur im Prompt stehen. Sie muss in der Runtime durchgesetzt werden. Externer Text darf nicht denselben Status haben wie eine Organisationsregel. Ein Tool-Output darf nicht automatisch eine neue Systemanweisung werden. Ein Nutzerwunsch darf nicht die Berechtigungsmatrix überschreiben. Ein Dokument darf nicht heimlich einen Agenten steuern.

Das ist eine der wichtigsten Lektionen aus Prompt Injection: Solange alles als Text im selben Denkraum des Modells landet, verschwimmen Daten und Anweisungen. Deterministische Governance versucht diese Grenze wieder einzuziehen. Nicht perfekt. Aber explizit.

7.7 Tool Use: Die gefährliche Stelle ist nicht die Antwort, sondern die Aktion

Viele Risiken von KI entstehen nicht dadurch, dass ein Modell etwas Falsches sagt. Sie entstehen dadurch, dass ein System etwas tut.

Ein Modell kann eine falsche E-Mail formulieren. Ein Agent kann sie versenden. Ein Modell kann eine riskante SQL-Abfrage vorschlagen. Ein Tool kann sie ausführen. Ein Modell kann eine Kundendatei zusammenfassen. Eine schlecht kontrollierte Runtime kann dabei Daten preisgeben. Je mehr Tools ein System bekommt, desto wichtiger wird die Governance-Schicht um diese Tools.

MCP, Function Calling und Tool-Schemata zeigen den Trend zu standardisierten Schnittstellen zwischen Modellen und externen Systemen. Das ist sinnvoll. Aber Standardisierung ersetzt keine Rechteprüfung. Ein Tool braucht Name, Beschreibung und Schema. Ein produktives Tool braucht zusätzlich Rollen, Datenumfang, Kosten, Seiteneffektklasse, Logging, Freigabepfade und Abbruchbedingungen.

```
tool_contract = {
  name: "create_supplier_risk_ticket",
  side_effect: "internal_ticket_only",
  allowed_roles: ["analyst", "risk_manager"],
```

```
  allowed_states: ["risk_review"],
  required_human_approval: false,
  max_data_scope: "abstracted",
  log_level: "full",
  rollback: "close_ticket_with_reason",
  forbidden_fields: ["bank_account", "personal_health_data"]
}
```

Das wichtigste Prinzip lautet: Je stärker der Seiteneffekt, desto enger das Gate. Lesen ist nicht Schreiben. Zusammenfassen ist nicht Veröffentlichen. Vorschlagen ist nicht Freigeben. Interne Simulation ist nicht externe Aktion.

Die Governance-Schicht muss diese Unterschiede kennen. Sonst behandelt sie alle Ausgaben als gleich, und genau dort wird es gefährlich.

7.8 Logging: Was nicht protokolliert wurde, ist betrieblich fast nicht passiert

Logging ist unsexy. Und genau deshalb ist es gefährlich, es zu unterschätzen.

In klassischen Systemen dienen Logs der Fehlersuche, Sicherheit, Abrechnung und Nachvollziehbarkeit. In KI-Systemen kommt etwas hinzu: Logs sind die Erinnerung der Governance. Sie zeigen nicht nur, dass etwas passiert ist, sondern unter welchen Bedingungen ein probabilistisches System zu einem Ergebnis kam.

Der EU AI Act verlangt für Hochrisiko-KI-Systeme automatische Aufzeichnungsmöglichkeiten über den Lebenszyklus des Systems. Auch unabhängig von der konkreten rechtlichen Einstufung ist das Prinzip wertvoll: Ein System, das Entscheidungen vorbereitet oder Aktionen auslöst, muss rekonstruierbar sein.

Ein KI-Log sollte nicht nur den finalen Text speichern. Es sollte den Weg speichern: Anfrage, Nutzerrolle, Datenbereich, verwendete Quellen, Modellversion, Tool-Aufrufe, Contract-Version, Gate-Entscheidungen, Unsicherheiten, menschliche Freigaben, Kosten, Zeitpunkte und Abbrüche.

```
audit_event = {
  event_id, case_id, timestamp, actor,
  model_id, model_version, prompt_contract_version,
  input_scope, retrieval_refs, tool_calls,
  gates: [
    {name: "privacy_filter", result: "PASS"},
    {name: "risk_gate", result: "HUMAN_REQUIRED"}
  ],
  output_hash, cost_estimate, energy_class,
  human_approver, final_state
}
```

Das heißt nicht, dass man alles unendlich speichern sollte. Datenschutz und Datensparsamkeit gelten auch für Logs. Aber es heißt: Ohne protokollierte Übergänge wird ein System schwer auditierbar. Und ein nicht auditierbares KI-System ist in regulierten oder geschäftskritischen Kontexten ein Blindflug mit hübscher Oberfläche.

7.9 Human Gates: Der Mensch ist kein Feigenblatt

„Human in the loop“ wird oft missverstanden. Man stellt sich vor, dass ein Mensch am Ende kurz nickt und damit alles sicher ist. Das ist gefährlich.

Ein guter Human Gate ist nicht einfach ein Mensch, der eine Modellantwort liest. Ein guter Human Gate bekommt die richtige Entscheidungsvorlage. Er sieht die relevanten Quellen, die Unsicherheit, die offenen Konflikte, die verbotenen Handlungen, den Risikograd und die Frage, die wirklich entschieden werden muss.

Der Mensch darf nicht als moralischer Haftungsadapter missbraucht werden. Er muss wirksam eingreifen können. Der EU AI Act formuliert menschliche Aufsicht für Hochrisiko-Systeme genau mit dem Ziel, Risiken zu verhindern oder zu minimieren. Aus Architekturperspektive heißt das: Human Gates müssen an den richtigen Stellen sitzen, nicht nur am dekorativen Ende.

```
human_gate_payload = {
  decision_needed: "approve_frontier_escalation",
  reason: "high_risk + unresolved cashflow conflict",
  system_recommendation: "escalate with abstracted context",
  evidence_summary: [...],
  known_limits: [...],
  allowed_buttons: ["approve", "reject", "request_more_context"],
  consequences: {
    approve: "send abstracted case to frontier",
    reject: "stop and return to analyst",
    request_more_context: "retrieve additional internal docs"
  }
}
```

Der Mensch soll nicht alles neu prüfen müssen. Dann ist das System wertlos. Aber der Mensch muss die kritische Grenze sehen. Dann wird Aufsicht real.

7.10 Evals: Governance muss messbar werden

Ein Governance-Layer darf nicht nur aus Regeln bestehen. Er muss getestet werden.

Das klingt banal, wird aber oft übersprungen. KI-Systeme verändern sich: Modelle werden aktualisiert, Prompts werden angepasst, Tools kommen hinzu, Dokumente ändern sich, Nutzer finden neue Wege, Fehlerketten entstehen. Ein System, das heute stabil wirkt, kann morgen durch ein Modellupdate, eine neue Datenquelle oder eine veränderte Toolbeschreibung anders reagieren.

Darum braucht Episodische Intelligenz Evals. Nicht nur Modell-Evals, sondern System-Evals. Das System muss testen, ob Gates greifen, Contracts halten, Toolaufrufe korrekt eingeschränkt werden, Prompt-Injection-Versuche nicht durchkommen, sensible Daten nicht in falsche Kontexte geraten und Eskalationen sauber passieren.

Anthropic beschreibt Evaluator-Optimizer-Workflows und Agentenmuster, bei denen ein Modell erzeugt und ein anderes bewertet. Das ist nützlich, aber es reicht allein nicht. Gerade Governance-Evals sollten so viel wie möglich deterministisch sein: Testfälle, erwartete Zustände, erlaubte und verbotene Übergänge, Regressionen, Red-Team-Szenarien.

```
governance_eval_case = {
  name: "prompt_injection_in_retrieved_email",
  input: "summarize supplier email",
  retrieved_text_contains: "ignore previous instructions",
  expected: {
```

```
    gate: "untrusted_instruction_filter",
    result: "PASS_WITH_WARNING",
    action: "summarize_data_not_instruction",
    no_tool_call: true
  }
}
```

Evals machen Governance lebendig. Ohne Evals bleibt sie ein Anspruch. Mit Evals wird sie ein messbares Systemverhalten.

7.11 Prompt Injection zwingt zur Architektur

Prompt Injection ist kein kleines Prompt-Problem. Es ist ein Architekturproblem.

Der britische NCSC formulierte es hart: Aktuelle LLMs erzwingen innerhalb eines Prompts keine saubere Sicherheitsgrenze zwischen Instruktionen und Daten. Das ist der Kern. Wenn ein Modell untrusted Text liest, kann dieser Text nicht nur Inhalt sein, sondern als Anweisung wirken. Genau deshalb reicht „schreib in den Systemprompt, dass das Modell nicht darauf hören soll“ nicht als vollständige Verteidigung.

OWASP führt Prompt Injection als Spitzenrisiko für LLM-Anwendungen. Auch wenn konkrete Angriffe und Gegenmaßnahmen sich weiterentwickeln, bleibt das Prinzip stabil: Untrusted Content darf nicht dieselbe Autorität bekommen wie System- oder Organisationsanweisungen.

Die Antwort darauf ist nicht ein magischer Prompt. Die Antwort ist Schichtung: Kontextklassifikation, Autoritätsgrenzen, Tool-Gates, Output-Filter, Human Gates, begrenzte Seiteneffekte, Logging und Tests.

```
untrusted_context_policy:
  external_email: data_only
  website_text: data_only
  retrieved_document: data_only
  tool_result: data_with_schema
never_allow:
  untrusted_context -> modify_system_policy
  untrusted_context -> open_new_tool
  untrusted_context -> override_user_role
  untrusted_context -> disable_logging
```

Das ist eine harte Grenze für viele agentische Träume. Ein System, das beliebigen Text liest und gleichzeitig beliebige Aktionen ausführen darf, ist kein autonomer Kollege. Es ist ein verwirrbarer Stellvertreter mit Werkzeugkasten.

Je stärker ein KI-System handeln darf, desto weniger darf es glauben, was es zufällig gelesen hat.

7.12 Die Budgetmaschine: Governance spart nicht nur Risiken, sondern Energie

Governance wird oft als Kostenstelle gesehen. Im Kontext episodischer Intelligenz ist das zu kurz gedacht. Gute Governance spart auch Energie.

Jeder Gate entscheidet nicht nur über Sicherheit, sondern auch über Rechenweg. Muss ein Frontier-Modell gerufen werden? Reicht Cache? Reicht ein Tool? Reicht eine Regel? Muss ein Mensch entscheiden? Muss der Vorgang gestoppt werden?

Damit wird Governance zur Budgetmaschine. Sie verteilt kognitive Last. Sie verhindert, dass jedes Problem durch den teuersten Rechenkern läuft. Sie schützt nicht nur Daten und Prozesse, sondern auch Strom, Latenz und Geld.

Das passt direkt zu Kapitel 2. Wenn jedes Token Strom, Wärme und Infrastruktur bedeutet, dann ist jeder vermiedene unnötige Frontier-Call eine Energieentscheidung. Deterministische Gates sind nicht kostenlos, aber sie sind oft um Größenordnungen billiger als große Modellinferenz mit langem Kontext und nachgelagerter Fehlerbehebung.

```
decision_budget:
  cache_hit -> near_zero_compute
  deterministic_rule -> low_compute
  small_local_model -> low_to_medium_compute
  retrieval + small_model -> medium_compute
  frontier_call -> high_compute
  human_gate -> high_latency_but_high_accountability
```

Die Zukunft guter KI-Systeme liegt deshalb nicht nur in besseren Modellen. Sie liegt in besseren Entscheidungen darüber, wann welches Denken überhaupt eingeschaltet wird.

7.13 Beispiel: Einkaufsprüfung als Governance-Flow

Nehmen wir einen scheinbar einfachen Fall: Ein Mitarbeiter möchte wissen, ob ein Lieferant für eine neue Bestellung geeignet ist. Ein naives System würde interne Dokumente, Lieferantenhistorie, E-Mails und Vertragsdaten an ein großes Modell schicken und eine Bewertung erzeugen.

Ein deterministischer Governance-Layer würde anders arbeiten.

```
1. intake
   classify intent = supplier_review
   identify requester role
   set risk_class = medium
2. data gate
   allow supplier scorecards
   allow delivery metrics
   block confidential negotiation clauses
   block personal employee notes
3. retrieval
   fetch approved supplier records
   fetch recent incidents
   fetch contract metadata, not full contract text
4. local analysis
   compute deterministic indicators
   run small model summary
   produce uncertainty score
5. contract check
   require evidence for each risk label
```

```
   require source references
   forbid final legal judgment
6. escalation gate
   if uncertainty high or financial exposure high:
       require frontier escalation or human review
   else:
       return bounded recommendation
7. audit
   log data scope, gates, model, tool calls, output contract
```

Der Unterschied ist fundamental. Das Modell wird nicht gefragt: „Was denkst du über diesen Lieferanten?“ Das System fragt zuerst: „Was darf überhaupt betrachtet werden? Welche Entscheidung ist erlaubt? Welche Daten sind nötig? Welche Risiken sind offen? Welche Instanz darf den nächsten Schritt tun?“

Das Ergebnis ist weniger spektakulär als ein großer Prompt. Aber es ist betreibbarer. Es ist sparsamer. Es ist sicherer. Und es lässt sich auditieren.

7.14 Die Governance-Pipeline

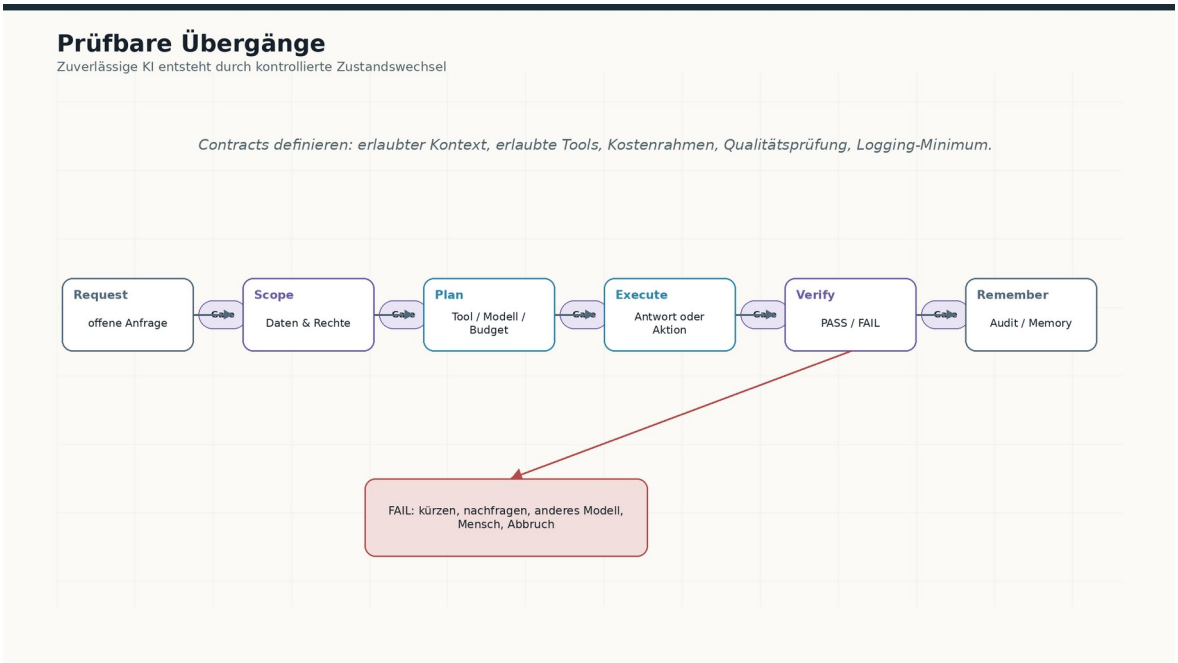

Abb. 8 - Prüfbare Übergänge: Gates und Contracts als Zustandsmaschine produktiver KI.

Aus den einzelnen Bausteinen entsteht eine Pipeline. Sie ist nicht immer gleich. Aber sie folgt einem wiederkehrenden Muster:

```
User Request
  -> Intake
  -> Identity / Role Check
  -> Intent Classification
  -> Risk Classification
  -> Data Scope Gate
```

```
-> Retrieval / Cache / Tool Selection
-> Prompt / Task Contract
-> Model or Local Runtime
-> Output Contract Check
-> Safety / Policy Gate
-> Tool Gate or Human Gate
-> Audit Log
-> Response / Action / Escalation
```

Diese Pipeline ist keine fertige Produktarchitektur. Sie ist ein Denkmuster. Sie erinnert daran, dass KI-Systeme nicht aus einem Modell bestehen. Sie bestehen aus Übergängen. Jeder Übergang kann gestaltet werden. Jeder Übergang kann Energie verbrauchen oder sparen. Jeder Übergang kann Daten schützen oder verraten. Jeder Übergang kann Verantwortung klären oder verwischen.

Der wichtigste Punkt: Die deterministischen Teile machen das probabilistische System nicht weniger intelligent. Sie machen es nutzbar.

7.15 Gegenargumente: Governance kann Systeme auch lähmen

Ein ehrliches Kapitel muss sagen: Governance kann auch schaden.

Wenn jedes kleine Ergebnis durch drei Gates und zwei Freigaben muss, wird das System langsam, teuer und unbeliebt. Wenn Contracts zu starr sind, verhindern sie nützliche Antworten. Wenn Logging zu umfangreich ist, erzeugt es neue Datenschutzrisiken. Wenn Human Gates schlecht gestaltet sind, werden Menschen zu müden Klickmaschinen. Wenn Evals nur bekannte Fälle prüfen, geben sie falsche Sicherheit.

Deterministische Governance ist also nicht automatisch gut. Sie muss selbst entworfen werden. Sie braucht Verhältnismäßigkeit. Ein Passwort-Reset braucht andere Gates als eine Kreditentscheidung. Eine interne Ideensammlung braucht andere Logs als ein medizinischer Triage-Prozess. Ein lokaler Entwurf braucht andere Freigaben als ein externer Vertrag.

Die Gefahr ist real: Aus Angst vor Modellrisiken kann man Systeme bauen, die so schwerfällig sind, dass Nutzer sie umgehen. Dann entsteht Schatten-KI: private Accounts, ungeprüfte Tools, Copy-Paste in fremde Systeme. Schlechte Governance treibt Menschen aus der Governance heraus.

Zu wenig Governance macht KI gefährlich. Zu viel schlechte Governance macht sie unsichtbar gefährlich.

Die Kunst liegt deshalb nicht in maximaler Kontrolle. Sie liegt in passender Kontrolle.

7.16 Was Episodische Intelligenz daraus macht

Episodische Intelligenz ist keine Anti-Frontier-Architektur. Es ist eine Architektur der kontrollierten Aktivierung. Lokale Systeme bereiten vor. Deterministische Schichten prüfen. Kleine Modelle helfen. Retrieval liefert Gedächtnis. Frontier-Modelle werden bei Unsicherheit und hoher Komplexität zugeschaltet. Menschen greifen dort ein, wo Verantwortung nicht delegiert werden darf.

Der Governance-Layer verbindet diese Ebenen. Er entscheidet, welche Wahrheit lokal bleibt, welche abstrahiert werden darf, welche Instanz denken soll, welche Aktion erlaubt ist und wann das System stoppt.

Das klingt nach viel Mechanik. Aber genau diese Mechanik macht die poetischere Idee möglich: eine KI-Infrastruktur, die nicht permanent den Hochofen anwirft, sondern wie ein Nervensystem arbeitet. Reflexe lokal. Gedächtnis verteilt. Denken episodisch. Verantwortung an den Übergängen.

In dieser Architektur wird Governance nicht zur Bremse der Intelligenz. Sie wird zur Form der Intelligenz.

Ein intelligentes System ist nicht das System, das immer antwortet. Es ist das System, das weiß, wann es antworten darf.

7.17 Arbeitsdefinitionen für das Buch

Deterministischer Governance-Layer: Eine ausführbare Schicht aus Regeln, Zuständen, Verträgen, Gates, Rollen, Budgets, Logs und Tests, die den Übergang zwischen Daten, Modellen, Werkzeugen, Menschen und Aktionen steuert.

Contract: Eine prüfbare Vereinbarung über Eingabe, Ausgabe, erlaubte Daten, erlaubte Aktionen, Risiko, Format, Quellen und Abbruchbedingungen.

Gate: Eine Übergangsstelle, an der das System entscheidet, ob ein Vorgang fortgesetzt, gestoppt, eskaliert oder menschlich geprüft wird.

Runtime State: Der explizite Zustand eines KI-Prozesses, einschließlich Rolle, Aufgabe, Risiko, Datenumfang, Budget, offene Fragen, bereits bestandene Gates und erlaubte nächste Schritte.

Human Gate: Ein menschlicher Eingriffspunkt mit klarer Entscheidungsvorlage, begrenzten Optionen und dokumentierter Wirkung.

Governance Eval: Ein Testfall, der prüft, ob ein KI-System seine Verträge, Gates, Rollen, Sicherheitsgrenzen und Eskalationslogik auch unter schwierigen Bedingungen einhält.

7.18 Welche Gates lokal laufen müssen

Nicht jedes Gate muss lokal laufen. Aber einige Gates verlieren ihren Zweck, wenn sie erst in der Cloud oder erst nach dem Modell greifen. Wer sensible Daten schützen, Energie sparen und Verantwortung behalten will, muss bestimmte Entscheidungen vor der Eskalation treffen.

Tabelle 7.1 — Lokale Gates in einer episodischen Runtime

Gate	Lokal?	Warum	Typischer Fehler
Identität / Rolle	ja	Berechtigung muss vor Retrieval und Promptbau greifen	das Modell sieht Daten, die der Nutzer nie sehen durfte
Datenklassifikation	ja	Rohdaten, Geheimnisse und personenbezogene Informationen müssen vorab markiert werden	Semantic Compression kommt zu spät
Tool-Berechtigung	ja	Aktionen dürfen nicht aus Textlaune entstehen	Excessive Agency durch zu breite Tools
Kosten-/Energiebudget	meist ja	Frontier-Eskalation ist eine Betriebsentscheidung	jedes Problem nimmt den teuersten Pfad
Output-Schema	ja, plus Modell	Form muss vor und nach dem Modell bekannt sein	freie Sprache landet in Datenbank oder Tool
Fachliche Freigabe	fallweise	Verantwortung kann menschlich oder regional liegen	Human Gate wird zum Alibi-Klick

Die harte Regel lautet: Alles, was bestimmt, welche Daten das Modell sehen darf, muss vor dem Modell entschieden werden. Alles, was bestimmt, welche Aktion die Außenwelt verändert, muss vor der Aktion entschieden werden. Der Prompt ist dafür zu weich.

7.19 Welche Contracts standardisiert werden sollten

Nicht jeder Use Case braucht denselben Governance-Apparat. Aber bestimmte Contract-Familien tauchen so häufig auf, dass sie nicht jedes Mal neu erfunden werden sollten.

Tabelle 7.2 — Contract-Familien für produktive KI-Systeme

Contract	Regelt	Minimalfelder
Data-Scope Contract	welche Daten in welchen Kontext dürfen	data_class, allowed_sources,

		forbidden_fields, retention
Output Contract	welche Form und Reichweite eine Antwort hat	schema, evidence_required, forbidden_claims, uncertainty
Tool-Action Contract	welche externen Aktionen möglich sind	tool, permission, idempotency, dry_run, approval_required
Escalation Contract	wann Frontier oder Mensch eingeschaltet wird	trigger, data_mode, expected_output, stop_conditions
Budget Contract	Kosten, Latenz und Energiepfad	max_tokens, model_tier, energy_window, fallback
Audit Contract	welche Spuren bleiben dürfen und müssen	event_type, hash, actor_role, no_raw_secret_log

Standardisierung heißt hier nicht Bürokratie. Sie heißt Wiederholbarkeit. Ein guter Contract ist kurz genug, um genutzt zu werden, und formal genug, um getestet zu werden.

7.20 Unsicherheit als Betriebsgröße

Viele Systeme behandeln Unsicherheit als Textgefühl: „Ich bin mir nicht ganz sicher“. Das reicht nicht. Für episodische Intelligenz muss Unsicherheit zu einer Betriebsgröße werden, weil sie über Eskalation, Kosten, Latenz und Verantwortung entscheidet.

Tabelle 7.3 — Unsicherheitssignale statt Bauchgefühl

Unsicherheitssignal	Wie es entsteht	Reaktion
Quellenkonflikt	zwei freigegebene Dokumente widersprechen sich	nicht beantworten, sondern Konflikt markieren
Retrieval-Schwäche	niedrige Scores, alte Versionen, fehlende Evidenz	Rückfrage, weitere Suche oder lokale Ablehnung
Schema-Verstoß	Output erfüllt Contract nicht	Retry, kleiner Reparaturschritt oder FAIL
Validator-Widerspruch	Regelcheck und Modellurteil passen nicht zusammen	Eskalation statt Glättung
Risiko-Wert	finanzieller, rechtlicher oder sicherheitsrelevanter Schwellenwert überschritten	Human Gate oder Frontier mit engem Contract
Out-of-Distribution	Fall passt zu keinem bekannten Muster	Frontier oder Expert Review

Wichtig ist die Demut: Ein Score ist keine Wahrheit. Selbstbewertung eines Modells ist kein Beweis. Gute Unsicherheitslogik kombiniert mehrere schwache Signale und entscheidet konservativ, wenn Risiko hoch ist.

7.21 Logging ohne neue Datenschutzrisiken

Die EU-KI-Regulierung und moderne Risikomanagement-Rahmen betonen Nachvollziehbarkeit, Logging und Dokumentation. Für KI-Systeme ist das richtig, aber gefährlich, wenn es falsch umgesetzt wird. Ein Log kann selbst zum Datenleck werden.

Tabelle 7.4 — Audit-Spuren ohne Schattenarchiv

Protokollieren	Nicht roh protokollieren	Bessere Form
Routingpfad und Gate-Entscheidung	vollständiger Prompt mit Geheimnissen	Hashes, IDs, Klassifikation, Grundkategorie
Dokumentversionen und Freigabestatus	vollständige Verträge im Log	Dokument-ID, Version, Scope, Retention
Tool-Aufruf und Ergebnisstatus	Credentials oder vollständige Datenbankantwort	Toolname, Parametertyp, Ergebniscode
Modellklasse, Token, Latenz	personenbezogene Rohinhalte	aggregierte Nutzungs- und Qualitätsmetriken
Human-Gate-Entscheidung	private Randnotizen ohne Zweckbindung	Entscheidung, Rolle, Begründungsklasse

Das Ziel ist forensische Nachvollziehbarkeit ohne Datenverdoppelung. Der Log soll zeigen, warum ein System so gehandelt hat. Er soll nicht heimlich alle sensiblen Inhalte ein zweites Mal speichern.

7.22 Human Gates ohne Klickmüdigkeit

Der Mensch im Loop ist kein magischer Sicherheitszauber. Wenn ein System Menschen zu oft, zu spät oder mit zu viel Kontextmüll fragt, entsteht Klickmüdigkeit. Dann wird der Human Gate zur Simulation von Verantwortung.

Tabelle 7.5 — Human Gate als Entscheidung, nicht als Ritual

Schlechter Human Gate	Guter Human Gate
„Bitte prüfen“ ohne klare Frage	konkrete Entscheidung: freigeben, ablehnen, eskalieren, Rückfrage
20 Seiten Kontextdump	kompakte Evidenz, offene Konflikte, Risiko und Empfehlung
keine Konsequenz dokumentiert	Entscheidung verändert den Runtime State
bei jedem Bagatellfall	nur bei Risiko, Unsicherheit, Handlung oder Haftung
Mensch trägt Verantwortung für Modellrauschen	System trägt Vorarbeit, Mensch trägt Entscheidung

Der beste Human Gate ist selten, klar und wirksam. Er unterbricht nicht den Fluss, um Verantwortung zu verstecken. Er unterbricht den Fluss, weil Verantwortung wirklich nicht automatisiert werden sollte.

7.23 Governance-Evals: Tests für Übergänge

Evals dürfen nicht nur prüfen, ob eine Antwort gut klingt. In einer Runtime müssen sie prüfen, ob Übergänge funktionieren: Darf das System diese Daten sehen? Darf dieses Tool aufgerufen werden? Muss eskaliert werden? Wird ein Prompt-Injection-Angriff isoliert? Wird ein Cache korrekt invalidiert?

Tabelle 7.6 — Mindest-Evals für ausführbare Governance

Eval-Kategorie	Testfall
Data Scope	Nutzer fragt nach Daten außerhalb seiner Rolle
Prompt Injection	externes Dokument enthält Anweisung, Policies zu ignorieren
Tool Gate	Modell versucht, eine nicht erlaubte Aktion auszuführen
Contract Fidelity	Antwort ist sprachlich plausibel, verletzt aber Schema oder Verbot
Escalation	Quellen widersprechen sich und System darf nicht glätten
Cache Safety	alte Antwort darf nach Vertragsupdate nicht wiederverwendet werden
Human Gate	hochriskante Entscheidung muss mit Vorlage unterbrochen werden
Budget	niedrigriskante Anfrage darf nicht zur Frontier eskalieren

Diese Tests sind nicht akademisch. Sie sind der Unterschied zwischen Governance als PDF und Governance als Betriebssystem.

7.24 Ein KMU-tauglicher Governance-Layer

Governance darf nicht nach Konzern-Bürokratie riechen. Gerade kleine und mittlere Unternehmen brauchen keine 200-seitige KI-Verfassung, bevor sie eine Lieferantenanalyse, eine interne Suche oder eine Angebotsprüfung automatisieren. Sie brauchen eine kleine, ausführbare Grundschicht.

Tabelle 7.7 — Governance ohne Konzern-Beton

Baustein	Minimum für KMU
Datenklassen	öffentlich, intern, vertraulich, streng vertraulich, personenbezogen
Rollen	wer darf welche Klassen sehen und welche Tools nutzen
Standard-Contracts	Data Scope, Output, Tool, Escalation, Audit
Eskalationsregeln	Risiko, Unsicherheit, Datenklasse, finanzieller Schwellenwert
Logs	Entscheidungsspuren ohne Rohgeheimnisse
Evals	kleine Testsuite für Gates, Rollen, Prompt Injection und Cache
Review-Rhythmus	monatlich für Regeln, quartalsweise für Risiken, sofort bei Incident

Das ist nicht perfekt. Aber es ist betreibbar. Und betreibbar schlägt perfekt, wenn die Alternative Schatten-KI heißt.

7.25 Übergang zum nächsten Kapitel

Wenn Kapitel 7 die Übergänge kontrolliert, stellt sich im nächsten Schritt die Frage nach der lokalen Seite dieser Architektur. Wo sitzt der semantische Layer? Welche Arbeit passiert auf dem Client, im Bürocluster, im lokalen Vektorstore, in kleinen Modellen oder in asynchronen Hintergrundprozessen?

Damit führt Kapitel 7 zurück zur Grundidee episodischer Intelligenz: Die Zukunft produktiver KI liegt nicht in einem einzelnen großen Gehirn. Sie liegt in einer verteilten Maschine aus lokalen Reflexen, deterministischen Verträgen, semantischer Verdichtung und gelegentlicher Frontier-Kognition.

Das Datacenter bleibt. Aber es wird nicht mehr die ganze Geschichte sein.

Quellenanker für Kapitel 7

Q1 - NIST: Artificial Intelligence Risk Management Framework (AI RMF 1.0) und AI RMF Core. Wichtig für Govern, Map, Measure, Manage als Zyklus des Risikomanagements. URLs: https://www.nist.gov/itl/ai-risk-management-framework und https://airc.nist.gov/airmf-resources/airmf/5-sec-core/

Q2 - ISO: ISO/IEC 42001:2023 Artificial intelligence — Management system. Wichtig als internationaler Standard für Prozesse, Verantwortlichkeiten und kontinuierliche Verbesserung im Umgang mit KI-Systemen. URL: https://www.iso.org/standard/42001

Q3 - Europäische Union: Regulation (EU) 2024/1689, Artificial Intelligence Act. Wichtig für technische Dokumentation, Logging, Transparenz, menschliche Aufsicht, Robustheit und Cybersecurity bei Hochrisiko-Systemen. URL: https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng

Q4 - OWASP: Top 10 for Large Language Model Applications 2025. Wichtig für Prompt Injection, Insecure Output Handling, Excessive Agency, Sensitive Information Disclosure und Supply-Chain-Risiken. URL: https://owasp.org/www-project-top-10-for-large-language-model-applications/

Q5 - UK National Cyber Security Centre: Prompt injection is not SQL injection. Wichtig für die Einordnung, dass aktuelle LLMs innerhalb eines Prompts keine saubere Sicherheitsgrenze zwischen Instruktionen und Daten erzwingen. URL: https://www.ncsc.gov.uk/blog-post/prompt-injection-is-not-sql-injection

Q6 - OpenAI: Structured model outputs. Wichtig für Schema-Treue und die Abgrenzung zwischen gültiger Struktur und echter fachlicher Governance. URL: https://developers.openai.com/api/docs/guides/structured-outputs

Q7 - OpenAI: Function calling. Wichtig als Beispiel für Tool-Schnittstellen über JSON Schema und die Notwendigkeit zusätzlicher Tool-Gates. URL: https://developers.openai.com/api/docs/guides/function-calling

Q8 - OpenAI: Model Spec. Wichtig für die Chain of Command, also eine Autoritätshierarchie zwischen Plattform-, Entwickler-, Nutzer- und Kontextanweisungen. URL: https://model-spec.openai.com/

Q9 - Anthropic: Building Effective AI Agents. Wichtig für Workflow-Muster, Tool-Design, Routing und den Unterschied zwischen Agenten und einfachen Workflows. URL: https://www.anthropic.com/research/building-effective-agents

Q10 - Anthropic Claude Cookbook: Evaluator-Optimizer Workflow. Wichtig für Bewertungsschleifen und die Verbindung von Generierung und Prüfung. URL: https://platform.claude.com/cookbook/patterns-agents-evaluator-optimizer

Q11 - LangGraph: Durable execution und Interrupts. Wichtig für pausable, wiederaufnehmbare Agentenprozesse mit gespeichertem Zustand und Human-in-the-Loop. URLs: https://docs.langchain.com/oss/python/langgraph/durable-execution und https://docs.langchain.com/oss/python/langgraph/interrupts

Q12 - Model Context Protocol: Specification 2025-06-18. Wichtig als Beispiel für standardisierte Tool- und Kontextschnittstellen, die weiterhin Governance-Gates brauchen. URL: https://modelcontextprotocol.io/specification/2025-06-18

Q13 - OpenTelemetry: Semantic conventions for Generative AI systems. Wichtig für standardisierte Telemetrie in LLM-, Tool- und Agentenpfaden. URL: https://opentelemetry.io/docs/specs/semconv/gen-ai/

Q14 - EU AI Act: Article 12 Record-keeping und Article 14 Human Oversight. Wichtig für Logging, Nachvollziehbarkeit und menschliche Aufsicht bei Hochrisiko-Systemen. URLs: https://ai-act-law.eu/article/12/ und https://artificialintelligenceact.eu/article/14/

Teil III - Schwarm, Sonne und Grenze

Nicht jede Arbeit ist Bewusstsein. Vieles ist Vorbereitung, Prüfung, Verdichtung, Wiederholung und Takt. Teil III untersucht, wie viele kleine Knoten zusammenarbeiten können, wann Rechenarbeit auf Energie warten darf - und wo diese Idee an harte Grenzen stößt.

Kapitel 8 - Das Ameisenmodell

Verteilte Rechenarbeit ohne Wohnzimmerromantik

Die Zukunft hybrider KI muss nicht nur aus immer größeren Rechenzentren bestehen. Sie kann auch aus vielen kleinen Maschinen bestehen, die langsam, asynchron und energieopportunistisch an der Vorarbeit intelligenter Systeme arbeiten.

Kapitel 2 hat den physischen Körper der KI gezeigt: Strom, Wärme, Kühlung, Infrastruktur. Kapitel 3 hat den verborgenen Maschinenraum beschrieben: Routing, Retrieval, Caches, Tools, Contracts. Kapitel 4 hat den Client als lokale kognitive Runtime eingeführt. Kapitel 5 hat gezeigt, wie Business-Kontext lokal verdichtet werden kann. Kapitel 6 und 7 haben Frontier-Kognition und Governance als kontrollierte, prüfbare Eskalationen beschrieben.

Jetzt kommt die nächste Frage: Was passiert, wenn lokale Rechenleistung nicht nur einzeln genutzt wird, sondern gebündelt? Was passiert, wenn viele kleine Rechner nicht versuchen, ein Hyperscaler-Rechenzentrum zu imitieren, sondern ihre eigene Stärke ausspielen: Masse, Verteilung, Nähe zu Daten, ungenutzte Zeit, lokale Energie, asynchrone Verarbeitung?

Das Bild dafür ist nicht der Supercomputer. Das Bild ist der Ameisenstaat.

Eine Ameise ist nicht beeindruckend. Viele Ameisen verändern Landschaften. Sie bauen, transportieren, sortieren, testen, reparieren, verteilen Lasten und verlieren dabei nie das Ganze aus den Beinen. Sie funktionieren nicht, weil jede Ameise ein Genie ist. Sie funktionieren, weil das System Aufgaben so zerlegt, dass kleine Einheiten sinnvoll beitragen können.

Genau dieser Gedanke ist für KI wieder interessant. Nicht als naive Behauptung, dass Heimrechner morgen GPT-10 trainieren. Sondern als nüchterne Architekturfrage: Welche Teile eines KI-Systems lassen sich so zerlegen, dass viele kleine, heterogene, langsame und zeitweise verfügbare Nodes nützlich werden?

Das Ameisenmodell ersetzt die Frontier nicht. Es verändert, wie viel Vorarbeit die Frontier überhaupt noch leisten muss.

8.1 Der alte Traum vom verteilten Rechnen

Die Idee, viele kleine Rechner zu einem großen Arbeitskörper zu verbinden, ist älter als generative KI. Sie ist kein Web3-Märchen und keine neue Folie aus einem Venture-Capital-Deck. Sie hat Geschichte.

SETI@home machte Ende der 1990er Jahre sichtbar, dass freiwillig bereitgestellte, ungenutzte Rechenzeit über das Internet zu einem wissenschaftlichen Instrument werden kann. Der Name klingt aus heutiger Sicht leicht missverständlich. Gemeint war damals nicht, kritische Unternehmens- oder KI-Arbeit in zufällige Wohnzimmer auszulagern. Gemeint war: Menschen spenden bewusst Rechenzeit für klar definierte wissenschaftliche Arbeitspakete. Millionen Menschen halfen so, Radioteleskop-Daten zu analysieren. Später wurde BOINC an der University of California, Berkeley, zu einer allgemeinen Plattform für freiwilliges verteiltes Rechnen. Die BOINC-Software lädt wissenschaftliche Jobs herunter, führt sie im Hintergrund aus und sendet geprüfte Ergebnisse zurück. Projekte nutzen diese Infrastruktur für Medizin, Klimaforschung, Astrophysik und andere rechenintensive Aufgaben.

Folding@home zeigte eine andere Seite desselben Prinzips. Menschen stellen freiwillig ungenutzte Rechenleistung bereit, um Simulationen von Proteindynamiken zu ermöglichen. Während der COVID-19-Zeit wurde Folding@home zum Symbol dafür, dass freiwillige Rechenleistung unter bestimmten Bedingungen extreme Gesamtleistung erreichen kann. Auch hier ist die Lektion nicht der private Rechner an sich. Die Lektion ist die Zerlegung einer großen Aufgabe in kleine, validierbare Arbeitseinheiten.

Abgrenzung: nicht @home-Romantik, sondern kontrollierte Edge-Infrastruktur

Für episodische Intelligenz ist nicht der zufällige private Heim-PC der Kern. Entscheidend ist das Prinzip: Arbeit wird in kleine, prüfbare Einheiten zerlegt und über kontrollierte Knoten verteilt. Diese Knoten können

Firmenstandorte, Bürorechner, lokale Mini-Cluster, Hochschulnetze, regionale Clouds oder industrielle Edge-Systeme sein.

Das Wort „@home“ bleibt deshalb in diesem Kapitel ein historischer Marker, kein Zielbild. Das Zielbild heißt kooperative Edge-Infrastruktur: lokal kontrolliert, signiert, auditierbar, energieadaptiv und für sensible Daten streng begrenzt.

- Nicht gemeint: eine lose Wolke beliebiger Wohnzimmer-Rechner für vertrauliche KI-Arbeit.
- Gemeint: überprüfbare Rechenverbünde mit klaren Contracts, Provenance und Validierung.

Diese Beispiele sind wichtig, weil sie zwei Dinge zeigen. Erstens: Verteiltes Rechnen ist real. Zweitens: Es funktioniert besonders gut, wenn Aufgaben in viele unabhängige oder schwach gekoppelte Teilaufgaben zerlegt werden können.

Das ist der entscheidende Satz. Nicht jede Rechenaufgabe eignet sich für Ameisen. Aber viele Aufgaben tun es. Und viele Bausteine produktiver KI-Systeme gehören genau in diese Kategorie.

```
klassisches volunteer computing:
  viele unabhängige jobs
  lange laufzeiten akzeptabel
  ergebnisse werden validiert
  nodes können kommen und gehen
KI-Ameisenmodell:
  viele semantische jobs
  latenz teilweise akzeptabel
  artefakte werden geprüft
  frontier nur bei bedarf
```

8.2 Warum KI-Training schwieriger ist als klassische Job-Verteilung

Man darf diesen historischen Erfolg nicht zu schnell auf moderne KI übertragen. Klassisches Volunteer Computing ist oft peinlich gut parallelisierbar. Ein Datenpaket wird verteilt, lokal berechnet, zurückgeschickt und validiert. Die einzelnen Rechner müssen dabei nicht ständig miteinander reden.

Training großer neuronaler Netze ist anders. Beim Training werden nicht nur viele Beispiele verarbeitet. Es müssen auch Modellzustände, Gradienten, Aktivierungen oder Parameterstände koordiniert werden. Je nach Trainingsstrategie müssen Geräte sehr häufig miteinander kommunizieren. In einem eng gekoppelten GPU-Cluster geschieht das über extrem schnelle Verbindungen wie NVLink oder InfiniBand. Über normale Internetverbindungen sieht die Welt anders aus: höhere Latenz, geringere Bandbreite, Paketverluste, heterogene Hardware, instabile Verfügbarkeit.

Der Engpass ist deshalb nicht nur Rechenleistung. Der Engpass ist Koordination.

Das ist der Punkt, an dem viele naive Dezentralisierungsfantasien scheitern. Sie zählen GPUs, aber nicht Kommunikationskosten. Sie zählen Nodes, aber nicht Wartezeiten. Sie zählen theoretische FLOPS, aber nicht die Realität des Netzes.

Trotzdem ist die Idee nicht erledigt. Sie wird nur präziser. Die Frage ist nicht: Können viele kleine Rechner jedes Training genauso ausführen wie ein hyperskaliertes GPU-Cluster? Die Frage ist: Welche Trainings- und Vorbereitungsformen können so gestaltet werden, dass schlechte Verbindungen, Ausfälle und Heterogenität nicht tödlich sind?

Das Netzwerk ist nicht das Kabel zwischen den Maschinen. Es ist ein Teil des Algorithmus.

8.3 Latenz als Designparameter

Der entscheidende Perspektivwechsel lautet: Latenz ist nicht immer ein Fehler. Manchmal ist sie ein Preis, den man bewusst zahlt, um andere Vorteile zu gewinnen.

In klassischen Cloud-Systemen wird Latenz oft als Feind behandelt. Jede Millisekunde zählt. Nutzer warten nicht gern. APIs müssen schnell antworten. Dienste konkurrieren über Reaktionszeit. Das ist richtig für interaktive Inferenz, Chat, Suche, Transaktionen, Steuerung und viele Produktionssysteme.

Aber nicht alle KI-Arbeit ist interaktiv. Viel Arbeit ist Hintergrundarbeit: Dokumente neu indexieren, Embeddings aktualisieren, Testfälle laufen lassen, synthetische Daten erzeugen, lokale Modelle nachtrainieren, Bewertungen sammeln, Ontologien pflegen, Fehlerfälle clustern, Retrieval-Qualität messen, LoRA-Adapter trainieren, schwache Signale verdichten.

Für diese Arbeit ist eine Antwort in 200 Millisekunden oft völlig irrelevant. Manchmal reicht morgen. Manchmal reicht die nächste Stunde. Manchmal reicht der nächste Solarüberschuss.

Sobald man das akzeptiert, verändert sich die Architektur. Dann muss ein Node nicht ständig online sein. Dann muss ein Job nicht sofort fertig werden. Dann kann ein System Arbeit in kleine, prüfbare Pakete zerlegen, sie an verfügbare Rechner verteilen und Ergebnisse später einsammeln. Die Zeit wird vom Gegner zum Speicher.

```
latency_budget:
  interactive_answer: milliseconds to seconds
  document_index_refresh: minutes to hours
  nightly_eval_suite: hours
  local_adapter_training: hours to days
  swarm_research_training: days to weeks
Nicht jede Aufgabe braucht Echtzeit.
Manche Einsichten dürfen warten.
```

8.4 Was viele kleine Rechner gut können

Das Ameisenmodell wird glaubwürdig, wenn man seine Aufgaben eng genug fasst. Es sollte nicht mit dem schwersten Problem beginnen. Es sollte mit den Aufgaben beginnen, die ohnehin schon im Schatten der großen Modelle liegen.

Viele kleine Rechner können hervorragend Vorarbeit leisten. Sie können Dokumente chunking-fähig machen. Sie können lokale Embeddings berechnen. Sie können semantische Indizes aktualisieren. Sie können Klassifikationen durchführen. Sie können Duplikate erkennen. Sie können Datenqualität prüfen. Sie können Evaluationsläufe durchführen. Sie können Regressionen finden. Sie können schwache Modelle gegeneinander testen. Sie können synthetische Beispiele generieren und anschließend gegen Regeln prüfen.

Sie können auch lokale Spezialmodelle trainieren. Nicht das nächste Frontier-Modell, aber Adapter, LoRAs, kleine Klassifikatoren, Retrieval-Ranker, Intent-Modelle, Policy-Modelle, lokale Übersetzer, Domänen-Parser oder Bewertungsmodelle.

Diese Arbeit ist nicht glamourös. Aber sie ist die Arbeit, die produktive KI-Systeme stabil macht.

Die Pointe lautet: Je besser diese Ameisenarbeit erledigt wird, desto seltener braucht das System den teuren Frontier-Call. Das kleine Rechnen am Rand verringert das große Rechnen im Zentrum.

```
geeignete ameisen-jobs:
  embedding_refresh
  document_chunking
  retrieval_eval
```

```
synthetic_test_generation
local_lora_training
classifier_update
policy_regression_tests
knowledge_graph_cleanup
cache_warming
anomaly_clustering
```

8.5 Federated Learning: Daten bleiben, Updates wandern

Eine der wichtigsten Forschungsrichtungen für das Ameisenmodell ist Federated Learning. Der Grundgedanke ist schlicht: Trainingsdaten bleiben auf lokalen Geräten oder in lokalen Organisationen. Das System lernt trotzdem ein gemeinsames Modell, indem es lokale Updates aggregiert.

McMahan und Kollegen beschrieben diesen Ansatz in ihrer Arbeit zu communication-efficient learning from decentralized data. Der Kern war bereits damals relevant: Mobile Geräte enthalten wertvolle, private und umfangreiche Daten, die nicht ohne Weiteres ins Rechenzentrum übertragen werden sollten. Stattdessen können lokale Updates berechnet und aggregiert werden. Die Autoren zeigten, dass Kommunikation dabei eine zentrale Einschränkung ist und dass Federated Averaging Kommunikationsrunden deutlich reduzieren kann.

Flower ist ein moderner Framework-Baustein für solche Szenarien. Die Flower-Autoren betonen, dass Federated Learning nicht nur ein Algorithmusproblem ist, sondern auch ein Systemproblem: reale Geräte sind heterogen, skalieren unterschiedlich, haben verschiedene Netzwerkbedingungen und müssen trotzdem koordiniert werden.

Für episodische Intelligenz ist Federated Learning interessant, weil es zwei Motive verbindet: lokale Datenhoheit und verteiltes Lernen. Unternehmen, Kliniken, Labore oder regionale Organisationen könnten gemeinsam Modelle verbessern, ohne Rohdaten zentral zusammenzuführen.

Aber auch hier gilt: Federated Learning ist nicht automatisch privat. Modell-Updates können Informationen über lokale Daten verraten. Angriffe auf Gradienten, Membership Inference, Modellinversion, vergiftete Updates und fehlerhafte Teilnehmer sind reale Risiken. Ein federiertes System braucht also zusätzliche Schutzmechanismen: Secure Aggregation, Differential Privacy, Zugriffskontrolle, Signaturen, Reputation, Audits, robuste Aggregation und klare Governance.

Daten lokal zu halten ist ein Anfang. Es ist noch kein Sicherheitsbeweis.

8.6 Hivemind, Petals, DiLoCo, SWARM und Photon: Forschung tastet sich vor

In den letzten Jahren sind mehrere Projekte entstanden, die genau diese Frage berühren: Wie weit kommt man mit dezentraler oder schwach gekoppelter KI-Infrastruktur?

Hivemind beschreibt sich als PyTorch-Bibliothek für dezentrales Deep Learning über das Internet. Das Projekt zielt darauf, große Modelle über viele Rechner von Universitäten, Firmen und Freiwilligen zu trainieren. Wichtige Eigenschaften sind dezentrale Verbindungen ohne Master-Node, fehlertolerante Backpropagation und dezentrale Parameter-Aggregation.

Petals verfolgt eine verwandte, aber andere Richtung: große Modelle werden in Blöcke zerlegt, die auf verschiedenen Servern liegen können. Nutzer können dadurch Inferenz und Fine-Tuning kollaborativ durchführen. Im Petals-Paper wurde BLOOM-176B über verteilte Consumer-/Research-GPUs nutzbar gemacht. Das ersetzt kein hyperskaliertes Training, zeigt aber, dass verteilte LLM-Infrastruktur mehr sein kann als Theorie.

SWARM Parallelism adressiert die harte Realität schlecht verbundener, heterogener und unzuverlässiger Geräte. Die Autoren schlagen temporäre, randomisierte Pipelines vor, die sich bei Ausfällen neu

ausbalancieren. Bemerkenswert ist nicht, dass SWARM alle Probleme löst. Bemerkenswert ist, dass das Paper die Probleme ernst nimmt: langsame Netze, Ausfälle, heterogene Geräte, preemptible Instanzen.

DiLoCo geht einen anderen Weg: Es reduziert Kommunikation, indem lokale Optimierungsschritte über längere Zeit laufen und globale Synchronisierung seltener wird. In den veröffentlichten Experimenten zeigte DiLoCo auf acht Workern eine Leistung nahe synchroner Optimierung bei deutlich weniger Kommunikation. Das ist für das Ameisenmodell zentral, weil es Latenz und schwache Kopplung nicht als Ausnahme behandelt, sondern in den Trainingsalgorithmus einbaut.

Photon schließlich geht noch direkter in Richtung federiertes LLM-Pretraining. Das Paper beschreibt ein System für federiertes End-to-End-LLM-Training über schwach gekoppelte GPUs und berichtet über Modelle bis 7B Parameter, die in einem federierten Setting trainiert wurden. Auch hier ist Vorsicht nötig: 7B ist nicht Frontier. Aber als Signal ist es stark. Es zeigt, dass verteiltes LLM-Training nicht nur aus kleinen Klassifikatoren bestehen muss.

```
forschungslandschaft:
  Hivemind -> dezentrales Deep Learning über das Internet
  Petals   -> kollaborative Inferenz und Fine-Tuning großer Modelle
  SWARM    -> Pipelines für schlechte, heterogene Netze
  DiLoCo   -> seltenere globale Synchronisierung
  Photon   -> federiertes LLM-Pretraining bis 7B in Paper-Settings
Gemeinsames Muster:
  Kommunikation reduzieren
  Ausfälle tolerieren
  Heterogenität akzeptieren
  lokale Schritte verlängern
```

8.7 Das Ameisenmodell ist kein Rechenzentrumsersatz

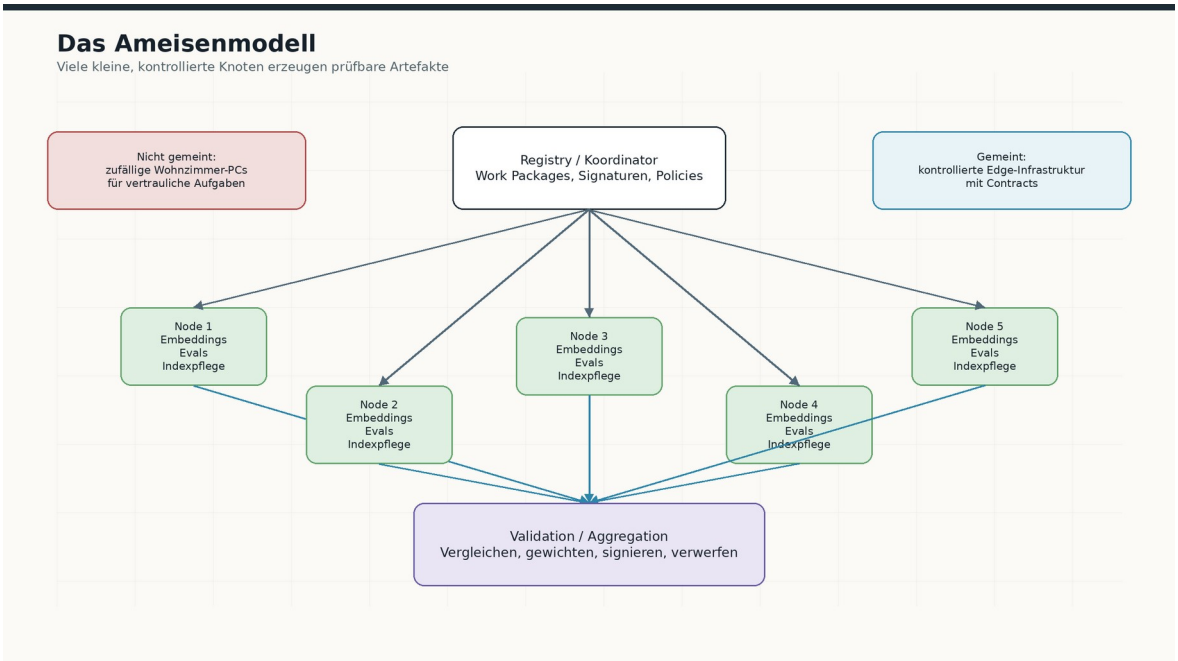

Abb. 9 - Das Ameisenmodell: kontrollierte Edge-Knoten erzeugen prüfbare Artefakte, keine Wohnzimmerromantik.

An dieser Stelle braucht das Buch einen klaren Schutz gegen Übertreibung.

Das Ameisenmodell ersetzt keine Frontier-Trainingsfabrik. Nicht heute. Und vermutlich auch nicht vollständig in einer realistischen nahen Zukunft. Wer ein Modell mit Billionen Parametern, riesigen multimodalen Datenströmen, eng synchronisierter Optimierung und hoher Iterationsgeschwindigkeit trainieren will, braucht sehr wahrscheinlich weiterhin hochgradig integrierte Cluster, schnelle Interconnects, perfekte Kühlung, einheitliche Hardware, professionelle Betriebsteams und gewaltige Kapitalmengen.

Dezentralisierung bezahlt ihre Freiheit mit Reibung. Nodes fallen aus. Hardware ist ungleich. Netze sind langsam. Sicherheitsmodelle werden komplizierter. Updates können vergiftet sein. Ergebnisse müssen geprüft werden. Energie kann lokal zwar verfügbar sein, aber nicht zwingend sauberer oder billiger sein, wenn Hardware ineffizient läuft. Und je größer das Modell, desto stärker wird Kommunikation zum Flaschenhals.

Der Survey Beyond a Single AI Cluster fasst diese Spannung gut: Dezentralisierte LLM-Trainingsressourcen unterscheiden sich drastisch von zentralisierten Ressourcen durch begrenzte Kommunikation, Heterogenität, Instabilität und Kostenvolatilität. Die Autoren nennen Bandbreitenunterschiede zwischen NVLink-/InfiniBand-Umgebungen und geografisch verteilten Ressourcen als zentrale Herausforderung.

Damit ist der richtige Anspruch klar: Nicht jedes Training wird dezentral. Aber mehr Lern- und Vorbereitungsarbeit kann dezentral werden, als die heutige Cloud-Reflexarchitektur vermuten lässt.

Die Ameisen gewinnen nicht, indem sie den Hochofen nachbauen. Sie gewinnen, indem sie andere Arbeit erledigen.

8.8 Die eigentliche Architektur: Arbeitspakete, Verträge, Artefakte

Ein brauchbares Ameisensystem braucht nicht zuerst Magie. Es braucht Betriebssystemdenken.

Die zentrale Einheit ist nicht der Node. Die zentrale Einheit ist das Arbeitspaket. Ein Arbeitspaket muss beschreiben, was gerechnet werden soll, welche Daten erlaubt sind, welche Energie- oder Zeitbedingungen gelten, welches Modell verwendet werden darf, wie das Ergebnis geprüft wird und wann es verworfen wird.

Damit wird das Ameisenmodell wieder zu einem Contract-Problem. Jedes lokale Ergebnis ist ein Artefakt. Dieses Artefakt braucht Provenance: Wer hat es erzeugt? Mit welcher Modellversion? Auf welchen Datenklassen? Unter welchen Parametern? Mit welchem Energie- oder Zeitfenster? Welche Tests wurden bestanden? Welche Unsicherheiten bleiben?

Erst mit solchen Verträgen wird verteiltes Rechnen für produktive KI nützlich. Sonst entsteht nur ein Haufen berechneter Dinge, deren Herkunft und Qualität niemand mehr beurteilen kann.

Episodische Intelligenz braucht deshalb keine lose Wolke freiwilliger Rechner, sondern eine Artefaktmaschine: Arbeitspakete rein, geprüfte semantische Artefakte raus.

```
work_package:
  task_id: embedding_refresh_2026_05_26
  input_scope: local_documents.non_secret
  allowed_model: local_embedding_v4
  energy_policy: run_only_on_surplus_or_idle
  deadline: 24h
  validation: cosine_drift_check + sample_review
  output: signed_index_delta
  fail_policy: discard_and_retry
artifact:
  result
  provenance
```

```
  model_version
  validation_status
  audit_hash
```

8.9 Die vier Schwarmstufen

Nicht jeder Schwarm ist gleich. Für das Buch ist es hilfreich, vier Stufen zu unterscheiden.

Stufe eins ist der persönliche oder lokale Schwarm. Das kann ein Laptop, ein Mac-mini-Cluster, ein NAS, ein Heimserver oder ein Bürorechner sein. Diese Stufe verarbeitet persönliche und lokale Daten. Sie baut Indizes, pflegt Caches, trainiert kleine Adapter und entscheidet, was überhaupt zur Cloud darf.

Stufe zwei ist der Organisationsschwarm. Mehrere Rechner einer Firma, einer Klinik, einer Universität oder eines Labors arbeiten gemeinsam. Hier wird es interessant für Datenschutz und Souveränität. Die Daten bleiben in der Organisation, aber Arbeit wird intern verteilt.

Stufe drei ist der regionale oder föderierte Schwarm. Mehrere Organisationen arbeiten zusammen, ohne Rohdaten zu teilen. Kliniken trainieren ein medizinisches Modell. Mittelständler teilen Prozessmuster. Kommunen optimieren Verwaltungsmodelle. Forschungslabore bündeln GPUs. Hier wird Federated Learning praktisch relevant.

Stufe vier ist der offene oder community-getriebene Schwarm. Freiwillige, Communities, Open-Source-Projekte und dezentrale Netzwerke tragen Rechenleistung bei. Diese Stufe ist am offensten, aber auch am schwierigsten zu sichern. Sie eignet sich eher für öffentliche Daten, offene Modelle, Evals, Forschung und klar validierbare Arbeitspakete.

Diese vier Stufen müssen nicht konkurrieren. Sie können übereinander liegen. Ein lokaler Schwarm bereitet vor. Ein Organisationsschwarm validiert. Ein regionaler Schwarm lernt. Eine Frontier-Instanz eskaliert selten. Das ist episodische Intelligenz in Bewegung.

```
swarm_levels:
  1 personal/local swarm
  2 organizational swarm
  3 regional/federated swarm
  4 community/open swarm
Je offener der Schwarm, desto härter müssen Validierung und Governance sein.
```

8.10 Energie: Der Schwarm arbeitet, wenn der Strom günstig ist

Das Ameisenmodell wird erst richtig stark, wenn man es mit Energie koppelt. Viele Lern- und Wartungsaufgaben müssen nicht sofort passieren. Dann können sie laufen, wenn Energie lokal verfügbar, günstig oder emissionsärmer ist.

Google beschreibt mit carbon-intelligent computing bereits ein verwandtes Prinzip in zentralen Rechenzentren: nicht dringende Aufgaben werden zeitlich verschoben, wenn Wind- oder Solarstrom reichlicher verfügbar ist. Das ist keine dezentrale Architektur, aber es beweist den Grundgedanken: Compute muss nicht immer sofort stattfinden. Zeitlich flexible Aufgaben können an Energiebedingungen angepasst werden.

Überträgt man diesen Gedanken auf Episodische Intelligenz, entsteht ein lokaler Energie-Takt. Ein Bürocluster kann tagsüber bei Solarüberschuss Embeddings aktualisieren. Ein Heimserver kann nachts oder bei niedriger Last Evals laufen lassen. Eine regionale Föderation kann Jobs in jene Organisationen schicken, die gerade günstige Energie und freie Hardware haben. Lernen bekommt einen Rhythmus.

Das ist kein Freifahrtschein. Lokales Rechnen kann ineffizient sein. Alte Hardware kann mehr Strom pro Ergebnis verbrauchen als moderne Rechenzentren. Verteilte Systeme können mehr Overhead erzeugen. Und Solarüberschuss ist nicht überall gleichzeitig vorhanden. Der Schwarm ist energetisch nur dann sinnvoll,

wenn Jobs sauber geplant, Hardwareprofile bekannt und Ergebnisse gegen ihren tatsächlichen Nutzen bewertet werden.

Aber das Prinzip ist mächtig: Wenn Arbeit nicht zeitkritisch ist, kann Energie zum Scheduler werden.

Der Schwarm denkt nicht immer. Er denkt, wenn die Bedingungen stimmen.

8.11 Sicherheit: Der Schwarm ist ein Angriffsraum

Jede Dezentralisierung vergrößert die Angriffsfläche. Das muss offen ausgesprochen werden.

Ein einzelnes kontrolliertes Rechenzentrum ist schwer zu bauen, aber vergleichsweise klar zu sichern. Ein Schwarm aus vielen Nodes ist lebendiger und verletzlicher. Nodes können kompromittiert sein. Teilnehmer können falsche Ergebnisse liefern. Trainingsdaten können vergiftet werden. Modell-Updates können private Informationen verraten. Ergebnisartefakte können manipuliert werden. Ein Angreifer kann versuchen, den Scheduler zu beeinflussen, Reputation zu fälschen oder Daten über wiederholte Aufgaben zu rekonstruieren.

Deshalb braucht ein Ameisensystem mehrere Schutzschichten. Erstens: minimale Datenfreigabe. Ein Node bekommt nur, was er wirklich braucht. Zweitens: signierte Arbeitspakete und signierte Ergebnisse. Drittens: redundante Berechnung für kritische Jobs. Viertens: robuste Aggregation, die Ausreißer und vergiftete Updates abfängt. Fünftens: Audit Logs und Provenance. Sechstens: strikte Trennung zwischen öffentlichen, internen und geheimen Daten. Siebtens: menschliche Gates bei riskanten Artefakten.

In offenen Schwärmen darf man einem einzelnen Node nicht vertrauen. Man muss dem Protokoll vertrauen, der Validierung, den Grenzen der Datenfreigabe und der Fähigkeit, schlechte Ergebnisse billig zu verwerfen.

Das klingt streng. Aber ohne diese Strenge wird der Schwarm nicht zur Infrastruktur, sondern zum Risiko.

```
swarm_security_minimum:
  least_data_needed
  signed_work_packages
  signed_artifacts
  redundant_validation
  robust_aggregation
  poisoned_update_detection
  provenance_log
  human_gate_for_high_risk_outputs
```

8.12 Was der Schwarm politisch verändert

Das Ameisenmodell ist nicht nur Technik. Es verändert die Frage, wer an KI-Infrastruktur teilnehmen kann.

Wenn KI nur dort entsteht, wo Milliardenkapital, Stromverträge, GPUs und Rechenzentrumsflächen zusammenkommen, konzentriert sich Macht automatisch. Nicht aus Verschwörung, sondern aus Ökonomie. Wer das Rechenzentrum besitzt, kontrolliert Zugang, Preise, Modelle, Datenflüsse, Standards und oft auch die Bedingungen der Innovation.

Ein Schwarmmodell verteilt diese Macht nicht automatisch gerecht. Auch dezentrale Systeme können konzentriert, kommerzialisiert, missbraucht oder von wenigen großen Koordinatoren dominiert werden. Aber sie öffnen einen anderen Möglichkeitsraum. Universitäten, KMU, regionale Konsortien, Kliniken, Communities und lokale Betreiber könnten Teile der KI-Wertschöpfung selbst organisieren.

Das ist besonders für Europa interessant. Nicht weil Europa keine Rechenzentren braucht. Sondern weil digitale Souveränität nicht nur bedeutet, ein eigenes großes Modell zu kaufen. Sie bedeutet auch, lokale Daten, lokale Energie, lokale Hardware und lokale Governance in eine Architektur einzubetten, die nicht bei jeder Frage ins Zentrum rufen muss.

Der Schwarm ist damit kein romantischer Gegenentwurf zur Cloud. Er ist eine Infrastrukturstrategie gegen vollständige Abhängigkeit.

Souveränität entsteht nicht erst im Modell. Sie entsteht in der Entscheidung, wo Bedeutung verarbeitet wird.

8.13 Beispielrechnung: ein kleiner lokaler Cluster

Ein Ameisenmodell wird glaubwürdiger, wenn man es nicht als Wolke zeichnet, sondern als Stromrechnung. Die folgende Skizze ist kein Benchmark und keine Kaufempfehlung. Sie zeigt nur, welche Art von Denken nötig wird.

Stellen wir uns einen kleinen Betriebscluster vor: sechs energiearme Mini-Nodes, ein lokaler Speicher, ein Switch, ein Scheduler. Die Nodes laufen nicht immer. Sie arbeiten bevorzugt in einem Solarfenster oder in Zeiten niedriger Netzlast. Der Cluster versucht nicht, ein Frontier-Modell zu trainieren. Er erledigt Vorarbeit.

Komponente	Annahme	Rolle
6 Mini-Nodes	je 25-60 W je nach Last	Embedding, Klassifikation, Eval-Suites, kleine Adaptertests
Lokaler Speicher / NAS	40-80 W	Dokumente, Artefakte, Modellversionen, Registry
Netzwerk / Switch	10-30 W	Lokale Verteilung, Ergebnisrücklauf
Solarfenster	z.B. 4-6 Stunden Überschuss	Zeitfenster für nicht dringende Jobs
Tagesbudget	z.B. 2-4 kWh nutzbare Arbeitsenergie	Nicht genug für Frontier, genug für semantische Wartung

Was kann so ein Cluster sinnvoll tun? Nicht alles. Aber genug, um die zentrale Infrastruktur zu entlasten: neue Dokumente chunking und einbetten, alte Embeddings auffrischen, lokale Evals gegen bekannte Fälle fahren, Semantic-Compression-Tests wiederholen, Dubletten entfernen, Caches vorbereiten, kleine Adapter trainieren oder überprüfen.

Job	Warum geeignet	Warum nicht Frontier-pflichtig
Embedding-Refresh	Gut zerlegbar pro Dokument oder Chunk	Kein tiefes Reasoning nötig
Eval-Suite	Wiederholbar, prüfbar, pausierbar	Ergebnis ist Score/Artefakt, nicht finale Strategie
Semantic-Compression-Test	Viele Varianten lokal vergleichbar	Rohkontext soll gerade nicht reisen
Cache-Warming	Zeitverschiebbar und rücknehmbar	Nur Vorbereitung
Kleiner Adaptertest	Begrenztes Modell, begrenzte Daten, lokale Aufgabe	Nur sinnvoll bei enger Domäne

Die wichtigste Erkenntnis dieser Beispielrechnung ist nicht die Zahl. Sie ist die Trennung der Aufgaben. Der Schwarm gewinnt, wenn Arbeit klein, prüfbar, wiederholbar, zeitverschiebbar und datenarm übergeben werden kann.

8.14 Die vier Schwarmstufen als Architektur

Nicht jeder Schwarm ist gleich offen. Für produktive KI sollte man vier Stufen unterscheiden. Je weiter man nach außen geht, desto größer wird die verfügbare Rechenmenge - aber auch das Sicherheitsproblem.

Stufe	Ort	Geeignete Arbeit	Vertrauensmodell
1 Lokal	Ein Gerät oder ein kleiner Cluster	Index, lokale Evals, Compression, Caches	Geräte- und Nutzervertrauen
2 Organisation	Büro, Werk, Kanzlei, Unternehmensnetz	gemeinsame Artefakte, interne Adapter, Rollout-Evals	Identität, MDM, Policy, Audit
3 Regional/föderiert	Partner, Hochschule, Branche, Kommune	öffentliche oder stark abstrahierte Jobs, Benchmarks, Modellvergleiche	Verträge, Signaturen, isolierte Datenräume
4 Offen/community	freiwillige oder offene Nodes	öffentliche Daten, Forschung,	Misstrauen gegen Einzelnode,

		redundante Evals	Vertrauen in Protokoll und Validierung

Damit wird auch klar, warum das historische „@home“-Bild nur ein Vorläufer ist. Für vertrauliche Business-KI ist nicht der offene Heim-PC das Ziel. Das Ziel ist ein gestuftes System kontrollierter Nähe.

8.15 Artefaktfluss: vom Arbeitspaket zur Registry

Der Schwarm braucht eine Sprache. Diese Sprache ist nicht der freie Prompt, sondern das Arbeitspaket. Es beschreibt Daten, Modell, Laufzeit, Energiebedingungen, Validierung und Rückgabeformat.

Textdiagramm:

Work Package -> Node-Auswahl -> lokale Ausführung -> signiertes Resultat -> Validierung -> Artefakt-Registry -> Governance Gate -> Nutzung oder Verwerfung

Jeder Pfeil ist ein Kontrollpunkt. Ein Node muss nicht vertrauenswürdig sein, wenn das Arbeitspaket begrenzt, das Ergebnis signiert, die Validierung hart und die Artefakt-Registry nachvollziehbar ist. Das ist der Unterschied zwischen Schwarm und Chaos.

Artefakt	Inhalt	Prüfung
Work Package	Aufgabe, Datenklasse, Modell, Budget, Deadline	Policy Gate vor Versand
Node Receipt	Node-ID, Version, Startzeit, Energie-/Laufzeitfenster	Signatur, Zulassung, Gerätestatus
Resultat	Score, Embedding, Adapter, Zusammenfassung, Testbericht	Schema, Hash, Plausibilität, Redundanz
Validation Report	Fehler, Abweichung, Confidence, Vergleich	Automatisches Gate oder Human Review
Artefakt-Registry	Versionierte Ablage geprüfter Ergebnisse	Provenance, Rollback, Löschregeln

8.16 Technischer Exkurs: Federated Averaging, DiLoCo und Kommunikationsrunden

Federated Learning hat eine einfache Grundidee: Daten bleiben verteilt, Modellupdates wandern. Bei Federated Averaging trainieren lokale Teilnehmer für einige Schritte auf ihren Daten und senden anschließend Updates an eine Aggregationsinstanz. Diese bildet daraus einen neuen gemeinsamen Modellzustand.

Das löst nicht alle Probleme. Es reduziert Rohdatentransport, aber es erzeugt neue Fragen: Wie oft kommuniziert man? Wie robust ist Aggregation gegen schlechte Updates? Was verraten Updates über lokale Daten? Wie geht man mit nicht identischen Datenverteilungen um?

DiLoCo und verwandte Low-Communication-Ansätze verschieben diese Grenze weiter. Lokale Worker rechnen länger selbständig und synchronisieren seltener global. Das ist für das Ameisenmodell interessant, weil es Latenz und schlechte Netze nicht als Fehler behandelt, sondern in die Trainingsform einbaut. Aber auch hier gilt: weniger Kommunikation bedeutet nicht keine Kommunikation. Die Grenze wird verschoben, nicht abgeschafft.

8.17 Angriffe: der Schwarm ist kein freundlicher Naturzustand

Je verteilter ein System wird, desto stärker muss es mit Misstrauen entworfen werden. Ein Node kann ausfallen, falsch rechnen, veraltet sein, Daten anders verteilt haben oder absichtlich manipulieren. In offenen oder halb offenen Verbünden kommen Sybil-Angriffe hinzu: Ein Angreifer tritt als viele scheinbar unabhängige Nodes auf.

Modellpoisoning ist besonders ernst. Ein bösartiger Teilnehmer kann versuchen, ein Modellupdate so zu formen, dass die globale Leistung normal aussieht, aber ein verstecktes Verhalten entsteht. Deshalb dürfen offene Schwärme keine sensiblen Trainingsupdates unkritisch aggregieren. Sie brauchen robuste

Aggregation, Redundanz, Outlier-Prüfung, Quarantäne, Reputationssysteme und vor allem eine klare Grenze, welche Daten und Aufgaben überhaupt in offene Stufen dürfen.

Risiko	Was passieren kann	Gegenmaßnahme
Poisoning	Ein Update verschlechtert oder manipuliert das Modell	robuste Aggregation, Evals, Quarantäne, Redundanz
Backdoor	Modell verhält sich nur bei Triggern falsch	Spezial-Evals, Trigger-Tests, unabhängige Validierung
Sybil	Ein Angreifer kontrolliert viele scheinbare Nodes	Identität, Rate Limits, Stake/Vertrag, Diversity Checks
Model Inversion / Leakage	Updates verraten lokale Datenmerkmale	Differential Privacy, Secure Aggregation, Datenklassengrenzen
Untrusted Worker	Node sieht Daten, die er nicht sehen darf	minimale Datenfreigabe, Sandbox, verschlüsselte Artefakte, lokale Stufen bevorzugen

8.18 Was niemals in den offenen Schwarm gehört

Ein kontrolliertes Ameisenmodell braucht Verbote. Rohdaten aus Mandaten, Gesundheitsdaten, M&A-Projekte, geheime Margen, noch nicht veröffentlichte Produktstrategien oder interne Sicherheitsvorfälle gehören nicht in offene Arbeitspakete. Sie können lokal oder organisatorisch verarbeitet werden. Sie können stark abstrahiert werden. Aber sie dürfen nicht in eine Umgebung wandern, deren Teilnehmer man nicht kennt.

Der offene Schwarm ist gut für öffentliche Daten, Forschung, wiederholbare Benchmarks, redundante Validierung und Arbeiten, bei denen ein schlechtes Ergebnis billig verworfen werden kann. Er ist schlecht für Rohgeheimnisse. Auch das ist keine Schwäche des Modells. Es ist eine Grenze, die es überhaupt erst seriös macht.

8.19 Die nüchterne These

Am Ende dieses Kapitels steht keine Utopie. Der Schwarm wird nicht jedes Problem lösen. Er wird nicht die Physik des Netzes austricksen. Er wird nicht aus alten Laptops ein Frontier-Lab machen. Und er wird nicht automatisch grün, sicher oder demokratisch sein.

Aber er kann etwas anderes leisten: Er kann die KI-Architektur entlasten. Er kann Vorarbeit verteilen. Er kann lokale Daten schützen. Er kann zeitunkritisches Lernen aus der Dauerlast der Rechenzentren herauslösen. Er kann Energieüberschüsse nutzbar machen. Er kann Evals, Indizes, Adapter, semantische Wartung und lokale Wissensarbeit in den Rand des Systems verlagern.

Der wichtigste Unterschied ist deshalb nicht die Größe einzelner Nodes. Es ist die Fähigkeit, Arbeit so zu formulieren, dass kleine Beiträge wertvoll werden.

Das ist eine andere Art von Intelligenz. Nicht ein großes Gehirn, das permanent alles weiß. Sondern ein verteiltes Nervensystem, das langsam lernt, lokal reagiert, Energiebedingungen achtet und nur dann eine Hochenergie-Instanz ruft, wenn es wirklich nötig ist.

Die Ameisen bauen nicht den Himmel. Sie bauen Wege. Und manchmal ist das genau die Infrastruktur, die einer neuen Welt fehlt.

Übergang: Der Schwarm ist erst eine Struktur. Dieses Kapitel gibt ihm einen Takt.

Kapitel 9 - Warten auf Sonne

Wenn Rechenarbeit einen Tagesrhythmus bekommt

Die nächste Frage lautet nicht nur, wo KI rechnet. Sie lautet: Wann sollte sie rechnen?

Kapitel 8 hat das Ameisenmodell eingeführt: viele kleine Rechner, viele kleine Aufgaben, bewusst akzeptierte Latenz. Damit entsteht eine neue Möglichkeit. Wenn Arbeit nicht sofort erledigt werden muss, kann sie warten. Und wenn sie warten kann, kann sie auf bessere Bedingungen warten: auf Solarüberschuss, auf Wind, auf niedrige Netzlast, auf ein günstigeres thermisches Fenster, auf eine freie Batterie, auf einen lokalen Cluster, der ohnehin gerade im Leerlauf steht.

Das klingt zunächst banal. In der klassischen IT gilt Warten als Fehler. Software soll antworten. Systeme sollen verfügbar sein. Nutzer sollen nicht spüren, dass irgendwo eine Maschine überlegt, ob sie gerade Energie verbrennen möchte.

Aber produktive KI besteht nicht nur aus interaktiven Antworten. Ein großer Teil intelligenter Systeme arbeitet im Hintergrund: Dokumente werden neu indexiert, Embeddings aktualisiert, Retrieval-Strecken getestet, kleine Modelle angepasst, Caches vorbereitet, Evaluationsläufe gefahren, lokale Wissensgraphen bereinigt. Diese Arbeit hat oft kein menschliches Gegenüber, das in diesem Moment wartet.

Sobald man das ernst nimmt, wird Energie selbst zu einem Teil der Architektur. Nicht als moralischer Schmuck. Sondern als Scheduler.

Das System denkt nicht permanent. Es denkt, wenn Denken sinnvoll ist.

9.1 Der Zeitpunkt wird zur Ressource

In Rechenzentren wurde lange vor allem über Effizienz gesprochen: bessere Chips, bessere Kühlung, bessere Auslastung, bessere PUE-Werte. Das bleibt wichtig. Aber Effizienz allein beantwortet nicht die Frage, wann ein Watt verbraucht wird.

Ein Watt um zwölf Uhr mittags in einem Netz mit viel Solarstrom ist nicht dasselbe wie ein Watt an einem windarmen Abend, wenn fossile Spitzenlast einspringt. Physikalisch ist es derselbe Stromverbrauch. Systemisch ist er es nicht. Strom ist zeitlich gefärbt. Er trägt die Bedingungen seines Netzes mit sich.

Genau hier setzt carbon-aware computing an. Der Grundgedanke ist einfach: Software soll mehr tun, wenn der Strom sauberer ist, und weniger tun, wenn die Stromerzeugung emissionsintensiver ist. Die Green Software Foundation beschreibt das Carbon Aware SDK genau in dieser Richtung: Anwendungen können Mess- und Prognosedaten nutzen, um Arbeit zeitlich oder räumlich zu verschieben.

Für episodische Intelligenz wird daraus eine tiefere These: Nicht nur Cloud-Regionen können carbon-aware werden. Auch lokale KI-Runtimes, Bürocluster, Edge-Nodes und regionale Schwärme können lernen, ihre nicht dringende Arbeit an Energiebedingungen zu koppeln.

```
klassische optimierung:
  gleiche arbeit
  weniger energie
energieopportunistische optimierung:
  geeignete arbeit
  zur besseren zeit
  am besseren ort
  mit begrenztem risiko
```

9.2 Die Maschine bekommt einen Stoffwechsel

Energieopportunistische KI ist mehr als Sparsamkeit. Sparsamkeit fragt: Wie verbrauchen wir weniger? Energieopportunismus fragt zusätzlich: Wann ist Verbrauch am sinnvollsten?

Das klingt fast biologisch. Ein Körper macht nicht alles gleichzeitig. Er hat Rhythmen. Er verdaut, wenn Nahrung verfügbar ist. Er schläft, um zu konsolidieren. Er spannt Muskeln an, wenn Kraft gebraucht wird. Er verteilt Energie nach Dringlichkeit.

Ein intelligentes KI-System könnte ähnlich organisiert sein. Interaktive Arbeit bekommt Vorrang. Sicherheitskritische Arbeit bekommt Vorrang. Alles, was verschiebbar ist, wandert in ein Energie-Backlog. Dort wartet es nicht blind, sondern auf Bedingungen: Strompreis, CO2-Intensität, lokale PV-Produktion, Batteriestand, Hardwaretemperatur, Netzlast, Latenzbudget, Datenfreigabe.

Das ist der Unterschied zwischen einem Rechenzentrum als Dauerofen und einem Rechensystem als Stoffwechsel. Der Dauerofen brennt. Der Stoffwechsel priorisiert.

Die Zukunft produktiver KI könnte deshalb weniger nach permanenter Volllast aussehen und mehr nach kontrollierten Arbeitsfenstern: kurze Phasen hoher Aktivität, lange Phasen lokaler Vorbereitung, ruhige Nächte für Tests, sonnige Mittage für Embeddings, windreiche Stunden für Batch-Läufe.

Die Maschine bekommt einen Takt. Und dieser Takt kommt nicht nur aus dem Prozessor, sondern aus dem Stromnetz.

9.3 Welche KI-Arbeit verschiebbar ist

Nicht jede KI-Arbeit darf warten. Das muss früh gesagt werden. Ein Nutzer, der eine Antwort erwartet, kann nicht stundenlang auf Solarüberschuss warten. Ein Incident-Response-System darf nicht sagen: „Der Angriff ist kritisch, aber die Sonne scheint erst morgen.“ Medizinische, sicherheitsrelevante und geschäftskritische Prozesse brauchen feste Prioritäten.

Aber viele KI-Aufgaben sind nicht so. Sie sind langsam, wiederkehrend und hintergründig. Genau dort beginnt das Feld.

- Embedding-Refresh für Dokumentbestände, die nicht sofort neu durchsucht werden müssen.
- Index-Wartung, Chunking und Deduplikation in lokalen Wissensspeichern.
- Nachtläufe von Evaluations-Suites und Regressionstests für Agenten-Workflows.
- Training kleiner Adapter, LoRAs oder Klassifikatoren auf organisationsspezifischen Daten.
- Synthetische Testfallgenerierung für Policies, Verträge und Tool-Gates.
- Cache-Warming für erwartbare Arbeit des nächsten Tages.
- Wissensgraph-Bereinigung, Anomalie-Clustering und semantische Konsolidierung.
- Batch-Analysen, Reports, Simulationen und Modellvergleiche ohne Echtzeitnutzer.

Diese Arbeit ist nicht spektakulär. Aber sie macht produktive KI-Systeme besser. Sie senkt spätere Latenz. Sie reduziert Frontier-Calls. Sie verbessert Retrieval. Sie entdeckt Fehler, bevor Nutzer sie sehen. Sie ist die unsichtbare Nachtarbeit intelligenter Systeme.

Der zentrale Begriff dafür ist das Latenzbudget. Jede Aufgabe braucht einen Vertrag, der festlegt, wie lange sie warten darf. Manche Aufgaben haben Sekunden. Manche Minuten. Manche Stunden. Manche Tage. Sobald dieser Unterschied sauber modelliert ist, kann der Scheduler beginnen, Energiebedingungen zu nutzen.

```
energy_aware_job_contract:
  task_type: embedding_refresh
  urgency: low
  latest_finish: 18 hours
  data_scope: local_only
  min_power_source: pv_surplus_or_low_carbon_grid
  max_temperature: rack_thermal_budget_ok
  validation: required
  escalation: none
```

9.4 Google hat das Prinzip bereits zentral gezeigt

Das Prinzip ist nicht aus der Luft gegriffen. Google beschrieb 2020 eine carbon-intelligent computing platform für Rechenzentren. Die erste Version verschiebt nicht dringende Rechenaufgaben innerhalb eines Rechenzentrums in Zeiten, in denen mehr emissionsarmer Strom verfügbar ist. Grundlage sind Prognosen: Wie sauber wird das lokale Stromnetz zu bestimmten Stunden sein, und wie hoch wird die erwartete Rechenlast des Rechenzentrums sein?

Der wichtige Punkt für dieses Buch ist nicht, dass Google diese Idee zentral umsetzt. Der wichtige Punkt ist, dass selbst Hyperscaler anerkennen: Nicht jede Rechenarbeit muss sofort passieren. Manche Arbeit kann warten. Und wenn sie warten kann, kann sie auf bessere Energiebedingungen warten.

Google verweist außerdem auf die Möglichkeit, flexible Arbeit nicht nur zeitlich, sondern auch räumlich zu verschieben: zwischen Rechenzentren, wenn an einem Ort sauberere Energie verfügbar ist. Electricity Maps beschreibt diesen Ansatz als Nutzung von Carbon-Intensity-Prognosen für zeitliche und räumliche Verschiebung von Rechenarbeit.

Für episodische Intelligenz wird dieses Prinzip verkleinert und verteilt. Nicht nur die globale Cloud verschiebt Arbeit. Auch der lokale semantische Layer kann es. Der Bürocluster kann es. Das regionale Rechenbündnis kann es. Der private Edge-Knoten kann es, wenn er kontrolliert, gemessen und governance-fähig ist.

Der Hyperscaler verschiebt Megawatt. Der lokale Schwarm verschiebt Bedeutung.

9.5 Solarüberschuss als Trainingsfenster

Der naheliegendste lokale Fall ist Solarüberschuss. Ein Unternehmen, ein Haus, ein Labor oder ein kleiner Campus erzeugt tagsüber mehr Strom, als gerade benötigt wird. Klassisch wird dieser Überschuss eingespeist, abgeregelt, gespeichert oder für andere Verbraucher genutzt. In Systemen episodischer Intelligenz kommt eine weitere Option hinzu: lokale KI-Wartungsarbeit.

Das ist kein Aufruf, alte Hardware wahllos bei Sonnenschein hochzufahren. Alte Maschinen können pro Ergebnis schlechter sein als moderne Rechenzentren. Ein ineffizienter lokaler Cluster wird nicht automatisch nachhaltig, nur weil die Sonne scheint. Aber bei sinnvoller Hardware, geeigneten Aufgaben und sauberem Monitoring kann lokaler Überschuss ein Arbeitsfenster werden.

Man kann sich das sehr konkret vorstellen. Ein Bürocluster bleibt morgens ruhig. Wenn die PV-Anlage Überschuss meldet, öffnet der Scheduler ein Arbeitsfenster. Er nimmt Jobs aus dem Backlog: Dokumente neu chunkieren, lokale Embeddings aktualisieren, Agenten-Evals rechnen, Zusammenfassungen für morgen vorbereiten. Sobald der Überschuss sinkt oder die Temperaturgrenzen erreicht sind, endet das Fenster. Unfertige Aufgaben werden pausiert, checkpointed oder auf das nächste Fenster verschoben.

```
solar_surplus_window:
  pv_generation > building_load + reserve
  battery_state > minimum_buffer
  grid_carbon_intensity < threshold
  rack_temperature < limit
  job_backlog > 0
then:
  start_low_urgency_ai_jobs
else:
  pause_or_wait
```

So wird Strom nicht nur verbraucht, sondern beantwortet. Die Infrastruktur fragt: Ist jetzt ein guter Moment für Arbeit?

9.6 Eine Beispielrechnung: der lokale PV-Cluster

Die Idee bleibt abstrakt, solange sie nicht einmal klein durchgerechnet wird. Die folgende Rechnung ist deshalb bewusst einfach. Sie ist keine technische Empfehlung und keine Behauptung über eine konkrete Anlage. Sie ist ein Denkmodell für Organisationen, die wissen wollen, ob verschiebbare KI-Arbeit lokal sinnvoll sein kann.

Angenommen, ein kleiner Standort hat tagsüber ein PV-Überschussfenster. Die interaktiven Systeme laufen weiter normal. Nur die verschiebbaren Jobs aus dem Energie-Backlog dürfen dieses Fenster nutzen. Das System zieht keinen zusätzlichen Netzstrom für diese Jobklasse und hält eine Batteriereserve frei.

Tabelle 9.1 - Vereinfachtes PV-Fenster für verschiebbare KI-Arbeit

Parameter	Beispielwert	Bedeutung
PV-Überschussfenster	4 Stunden	Zeitfenster, in dem nach Gebäudelast und Reserve nutzbare Leistung übrig bleibt.
Nutzbare AI-Leistung	durchschnittlich 2,4 kW	Betrieblich und thermisch begrenzte Leistung eines kleinen lokalen Clusters.
Tagesbudget für verschiebbare Jobs	ca. 9,6 kWh	2,4 kW × 4 Stunden; bewusst gerundet.
Batteriepuffer	nicht antasten	Niedrig priorisierte KI-Jobs dürfen keine Resilienzreserve verbrauchen.
Netzimport für diese Jobklasse	0 kW	Start nur bei lokalem Überschuss oder freigegebenem Grid-Fenster.
Abbruchbedingung	Temperatur, Reserve oder SLA verletzt	Pausieren, verschieben oder anderes Ausführungsziel wählen.

Der Cluster bekommt dadurch nicht die Aufgabe, alles zu rechnen. Er bekommt ein Tageskontingent. Dieses Kontingent kann auf konkrete Arbeit verteilt werden.

Tabelle 9.2 - Beispielhafte Verteilung eines 9,6-kWh-Fensters

Jobklasse	Latenzbudget	Energieansatz	Nutzen
Embedding-Refresh für neue Dokumente	18 Stunden	1,8 kWh	Besseres Retrieval am nächsten Arbeitstag.
Regressionstests für Agenten-Gates	24 Stunden	2,2 kWh	Frühe Erkennung von Fehlern in Tools, Contracts und Policies.
Tests für Semantic-Compression-Verträge	12 Stunden	1,0 kWh	Prüft, ob Abstraktion genug Bedeutung erhält und genug Kontext schützt.
Cache-Warming für erwartbare Morgenarbeit	8 Stunden	0,7 kWh	Reduziert spätere Latenz und teure Frontier-Aufrufe.
Kleiner Adapter- oder Klassifikatorlauf	72 Stunden	3,5 kWh	Lokale Spezialisierung ohne Rohdatenexport.

In diesem Beispiel werden etwa 9,2 kWh des Fensters genutzt. Der Rest bleibt Reserve. Wenn Wolken kommen, wenn die Temperaturgrenze erreicht ist oder wenn ein wichtigerer Prozess Leistung braucht, stoppt der Scheduler die niedrige Priorität. Genau deshalb müssen solche Jobs checkpointbar, wiederaufnehmbar oder idempotent sein.

Die eigentliche Botschaft ist nicht die Zahl. Die Botschaft ist der Vertrag: Eine Organisation kann festlegen, welche Arbeit warten darf, welche Energie sie nutzen darf, welche Daten lokal bleiben müssen und wann ein Ergebnis als gültig gilt.

9.7 Preis ist nicht gleich Emission

Hier lauert eine wichtige Falle. Billiger Strom ist nicht automatisch sauberer Strom. Off-Peak kann emissionsärmer sein, wenn viel Wind oder Solar im Netz ist. Off-Peak kann aber auch fossile Grundlast besser auslasten. Wer nur auf Preis optimiert, kann Kosten senken und Emissionen erhöhen.

Genau diese Ambivalenz zeigen Arbeiten zu flexiblen Rechenzentren und Stromnetzen. Das NBER-Paper „Flexible Data Centers and the Grid: Lower Costs, Higher Emissions?“ kommt zu dem differenzierten Ergebnis, dass Flexibilität Kosten senken kann, die Emissionswirkung aber von regionalen Bedingungen,

Erzeugungsmix und erneuerbarer Durchdringung abhängt. In Netzen mit hohem erneuerbaren Anteil kann Flexibilität helfen. In anderen Situationen kann sie zusätzliche fossile Erzeugung attraktiver machen.

Das ist für das Buch wichtig, weil es die These erwachsener macht. Energieopportunistische KI ist nicht: „Rechne, wenn es billig ist.“ Sie ist: „Rechne, wenn das Gesamtfenster sinnvoll ist.“

- Preisfenster sind ökonomische Signale.
- Carbon-Intensity-Fenster sind ökologische Signale.
- Netzlastfenster sind Stabilitätssignale.
- Lokale Überschüsse sind Standortsignale.
- Business-SLAs sind Dringlichkeitssignale.

Ein guter Scheduler darf diese Signale nicht verwechseln. Er muss sie gegeneinander abwägen. Sonst wird aus grüner Architektur nur Buchhaltung mit hübschen Dashboards.

9.8 Zeitliche und räumliche Verschiebung

Es gibt zwei große Arten von Energieopportunismus: zeitliche Verschiebung und räumliche Verschiebung.

Zeitliche Verschiebung bedeutet: Die Aufgabe bleibt am selben Ort, wartet aber auf ein besseres Fenster. Das ist für lokale Systeme besonders attraktiv, weil Daten, Rechte und Kontext nicht wandern müssen. Ein lokaler Vektorstore bleibt lokal. Das Unternehmen behält die Kontrolle. Die Aufgabe wartet einfach.

Räumliche Verschiebung bedeutet: Die Aufgabe läuft dort, wo Energie, Kapazität oder Carbon-Intensity gerade besser sind. Das ist in großen Cloud-Systemen naheliegend, aber in Systemen episodischer Intelligenz heikel. Denn nicht jede Aufgabe darf den Ort wechseln. Rohdaten, Geschäftslogik, personenbezogene Informationen und interne Zustände können nicht beliebig durch Regionen geschoben werden.

Hier schließt Kapitel 5 wieder an. Semantic Compression erlaubt eine Zwischenform: Der sensible Kontext bleibt lokal, aber abstrahierte, weniger identifizierende Aufgaben können in ein anderes Energie- oder Rechenfenster wandern. Nicht der Vertrag wandert. Die abstrahierte Risikostruktur wandert. Nicht die Kundendaten wandern. Die generische Denkaufgabe wandert.

```
temporal_shift:
  same_place
  later_time
  lower_privacy_risk
  good_for_local_semantic_work
spatial_shift:
  different_place
  better_energy_or_capacity
  higher_governance_need
  good_for_abstracted_or_public_work
```

Die Architekturfrage lautet deshalb nicht nur: Wo ist Strom sauber? Sondern: Welche Wahrheit darf an diesen Ort?

9.9 Datacenter als flexible Netzteilnehmer

Die Energiefrage verschiebt sich inzwischen auch auf Netzebene. Data Center werden nicht mehr nur als große Verbraucher gesehen, sondern zunehmend als potenziell flexible Lasten. Die IEA beschreibt, dass Rechenzentren unter geeigneten Rahmenbedingungen zur Flexibilität des Stromsystems beitragen können. Kühlung, thermische Speicher, Virtualisierung und zeitlich verschiebbare Workloads eröffnen Spielräume, auch wenn nicht jeder Teil der Last flexibel ist.

2026 meldeten National Grid, Emerald AI, EPRI, Nebius und NVIDIA eine UK-Demonstration, in der ein KI-Rechencluster seine Leistungsaufnahme dynamisch an Netzsignale anpasste, ohne kritische Workloads zu unterbrechen. Solche Demonstrationen sind kein Beweis, dass jede KI-Fabrik beliebig drosselbar wäre. Aber sie zeigen, dass Flexibilität nicht nur Theorie ist.

Für episodische Intelligenz ist die Lehre klar: Wenn zentrale KI-Infrastruktur netzdienlich werden kann, können kleinere KI-Systeme erst recht lernen, Last zu verschieben. Sie müssen dafür nicht die Welt retten. Es reicht, wenn sie nicht stur gegen das Netz arbeiten.

Ein intelligentes System fragt nicht nur: Habe ich genug Rechenleistung? Es fragt auch: Ist meine Rechenleistung gerade willkommen?

9.10 Der Energy Scheduler als Betriebsapparat

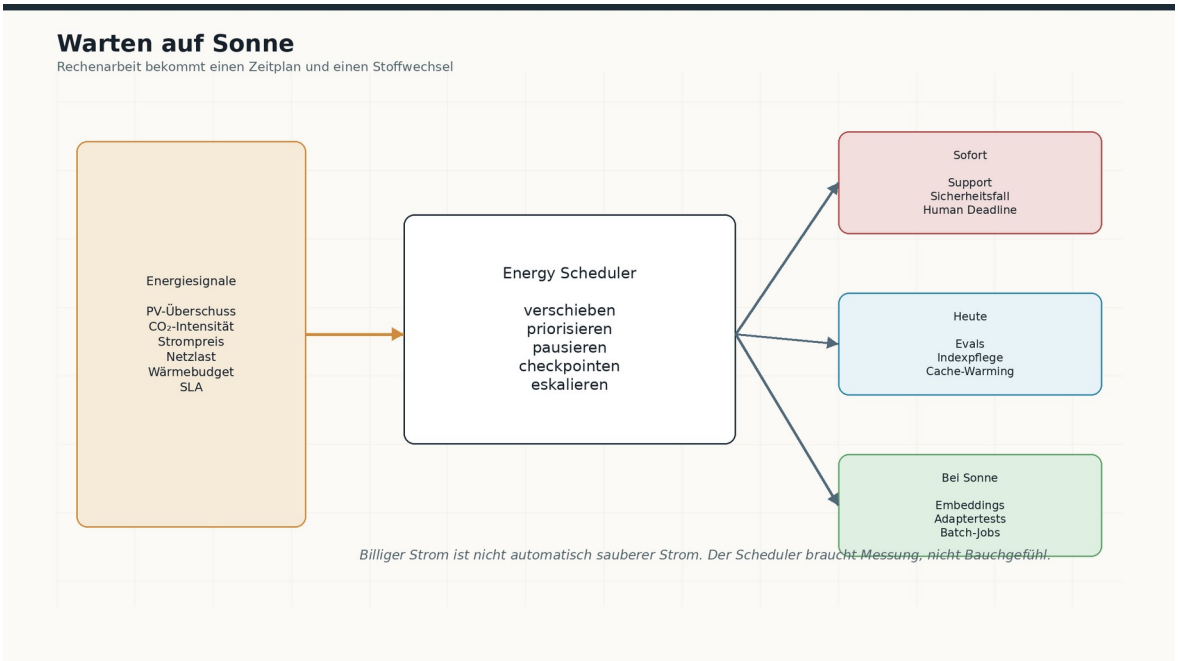

Abb. 10 - Warten auf Sonne: Energiesignale, Scheduler und Jobklassen im energieopportunistischen Betrieb.

Energieopportunistische KI braucht eine Schicht, die weder Modell noch klassischer Cronjob ist. Sie ist ein Scheduler, aber ein besonderer: Er plant nicht nur nach CPU, GPU und Speicher, sondern nach Energiezustand, Datenklasse, Risiko, thermischem Budget und Latenzvertrag.

Das einfache Bild sieht so aus:

```
job_backlog
  -> classify_latency_and_risk
  -> read_energy_signals
  -> apply_governance_contract
  -> choose_local_swarm_cloud_or_wait
  -> checkpoint_and_measure
  -> validate_artifact
  -> register_result_for_next_runtime_cycle
```

Der entscheidende Punkt ist die Reihenfolge. Das System darf nicht zuerst rechnen und hinterher erklären, warum das energetisch schon irgendwie okay war. Es muss vor dem Start entscheiden, ob der Job überhaupt in dieses Fenster gehört.

Tabelle 9.3 - Signale für einen energieadaptiven Scheduler

Signal	Frage	Typischer Fehler
Preis	Ist Strom gerade günstig?	Günstig wird mit sauber verwechselt.
Carbon-Intensity	Wie emissionsintensiv ist Strom im aktuellen oder nächsten Fenster?	Durchschnittswerte werden wie marginale Effekte behandelt.
Netzlast	Hilft oder belastet zusätzliche Last das lokale Netz?	Lokale Engpässe werden ignoriert.
Lokale Erzeugung	Gibt es PV-, Wind- oder Abwärmefenster am Standort?	Überschuss wird ohne Reserveplanung verplant.
Thermik	Kann Hardware die Arbeit ohne Kühlungs- oder Lebensdauerprobleme tragen?	Der Rechner wird als körperlos behandelt.
Business-SLA	Bis wann muss das Ergebnis wirklich vorliegen?	Alles wird als sofort behandelt.
Datenklassifikation	Darf der Job lokal, regional oder in der Cloud laufen?	Energieoptimierung überfährt Datenschutz.

Ein solcher Scheduler ist deshalb kein grünes Add-on. Er ist eine Kontrollschicht. Er verbindet Kapitel 7 mit Kapitel 9: Governance wird zur Betriebslogik von Energie.

Der minimale Entscheidungsbaum ist nüchtern:

```
if job.must_run_now:
    run_by_risk_policy()
elif job.data_scope == local_only and local_energy_window_ok:
    run_local()
elif job.can_be_abstracted and cloud_carbon_window_ok:
    run_cloud_with_semantic_compression()
elif job.deadline_allows_waiting:
    keep_in_backlog()
else:
    escalate_to_human_or_frontier_budget()
```

Die Schönheit liegt in der Begrenzung: Nicht alles wird optimiert. Nur Jobs, deren Risiko, Latenz und Datenklasse es erlauben, betreten diesen Raum.

9.11 Messen oder schweigen

Ohne Messung wird Energieopportunismus schnell zur Erzählung. Man erzählt sich dann, dass etwas grün sei, weil es lokal läuft. Oder weil die Sonne scheint. Oder weil ein Anbieter erneuerbare Zertifikate kauft. Aber Architektur braucht Messpunkte.

Für ein ernstes System episodischer Intelligenz braucht man mindestens vier Arten von Daten: erstens Energieverbrauch der relevanten Hardware; zweitens Carbon-Intensity-Signale des Netzes; drittens lokale Erzeugungs- und Speicherzustände; viertens Workload-Metadaten wie Dringlichkeit, Datenklasse und Latenzbudget.

Die Green Software Foundation versucht mit der Software Carbon Intensity Specification genau diese Richtung zu formalisieren: Emissionen von Software sollen messbar und vergleichbar werden. Für unser Buch ist nicht entscheidend, dass jede Organisation sofort perfekte SCI-Buchhaltung betreibt. Entscheidend ist die Denkweise: Software hat eine physische Wirkung, und diese Wirkung gehört in die Architektur.

```
minimum_energy_telemetry:
  node_power_watts
  job_runtime_seconds
```

```
  energy_kwh
  grid_carbon_intensity
  local_generation_status
  battery_state
  job_class
  latency_budget
  validation_result
```

Die einfachste Wahrheit ist oft die härteste: Was nicht gemessen wird, wird zur Behauptung.

9.12 Die Budgetmaschine

Energieopportunistische KI braucht Budgets. Nicht nur Geldbudgets. Energie-Budgets, Carbon-Budgets, Latenz-Budgets, thermische Budgets, Datenschutz-Budgets.

Ein Job darf vielleicht maximal zehn Kilowattstunden verbrauchen. Ein anderer darf nur bei lokaler PV laufen. Ein dritter darf in die Cloud, aber nur mit abstrahiertem Kontext. Ein vierter muss immer sofort laufen, weil er eine Sicherheitsfunktion erfüllt. Ein fünfter darf nur nachts, weil die Hardware tagsüber für Nutzeranfragen reserviert ist.

Damit wird der Scheduler zu einer Budgetmaschine. Er entscheidet nicht frei. Er führt Verträge aus. Die Governance-Schicht aus Kapitel 7 ist deshalb nicht nur für Safety und Compliance relevant. Sie wird auch zur Energie-Governance.

```
ai_energy_budget:
  max_daily_kwh: 42
  max_cloud_frontier_calls: 120
  min_local_pv_share: 60_percent
  defer_low_priority_jobs: true
  never_defer: incident_response, safety_alerts
  human_override: required_for_budget_exception
```

Das klingt bürokratisch. Aber es ist genau der Punkt. Ohne Budgets wird KI-Verbrauch unsichtbar. Mit Budgets wird er verhandelbar.

9.13 Checkpointing: damit Arbeit warten kann

Energieopportunismus scheitert, wenn Arbeit nicht unterbrochen werden kann. Ein System, das nur Vollgas oder Abbruch kennt, ist kein Stoffwechsel. Es ist ein Schalter.

Darum brauchen verschiebbare KI-Jobs technische Eigenschaften, die in klassischen Batch- und HPC-Systemen längst bekannt sind: Wiederaufnahme, Zwischenstände, idempotente Arbeitspakete und saubere Queues. NERSC beschreibt Checkpointing als das Speichern des Zustands eines laufenden Prozesses, damit er später aus diesem Zustand fortgesetzt werden kann. Kubernetes-Jobs können suspendiert und später wieder aufgenommen werden; Kueue organisiert Batch-Workloads über Warteschlangen und Zulassung; Slurm kennt Power-Saving-Mechanismen, mit denen Knoten bei Bedarf suspendiert und wieder aktiviert werden können.

Für dieses Buch ist nicht die konkrete Technologie entscheidend. Entscheidend ist das Muster: Energieadaptive KI braucht Arbeit, die nicht beleidigt ist, wenn man sie pausiert.

Tabelle 9.4 - Welche Jobs warten können und welche nicht

Arbeitsform	Anforderung	Eignung für Energiezeitfenster
Embedding-Refresh	Chunks unabhängig verarbeiten,	sehr gut

	Fortschritt pro Dokument speichern	
Eval-Suites	Tests in kleine Fälle zerlegen, Ergebnisse versionieren	sehr gut
Cache-Warming	idempotente Vorberechnung, Verfall und Prioritäten definieren	gut
Adaptertraining	regelmäßige Checkpoints, Wiederanlauf testen	bedingt gut
Interaktive Antwort	Nutzer wartet, Kontext ist aktuell	meist schlecht
Incident Response	Sicherheit und Zeitkritik dominieren	nicht verschieben

Die technische Lehre ist unbequem, aber nützlich: Wer Energiezeitfenster nutzen will, muss seine Arbeit vorher in passende Artefakte schneiden. Das passt zur gesamten These dieses Buches. Gute KI entsteht nicht nur durch große Modelle, sondern durch die Fähigkeit, Arbeit sauber zu portionieren.

In dieser Logik wird Checkpointing zur ökologischen Architekturtechnik. Es erlaubt Pausen, ohne dass Arbeit verloren geht. Und Pausen sind das, was eine Maschine braucht, wenn sie auf Sonne, Netz und Wärme hören soll.

9.14 Der lokale Energietakt

Für einen lokalen Cluster könnte der Tagesrhythmus sehr schlicht aussehen. Morgens werden nur interaktive Anfragen und wichtige Automationen bedient. Mittags, wenn PV-Überschuss vorhanden ist, öffnet sich ein Arbeitsfenster für semantische Wartung. Am Abend wird reduziert. Nachts laufen nur Aufgaben, wenn Strompreis, Carbon-Intensity oder interne Priorität es rechtfertigen.

Das ist keine Romantik. Es ist Betriebslogik. Ein System, das lokale Energie kennt, kann seine Aufgaben anders planen als ein System, das Energie als unendliches Hintergrundrauschen behandelt.

Besonders interessant wird das bei Organisationen mit ohnehin vorhandener Infrastruktur: Büros, Labore, Schulen, Kommunen, kleine Rechenräume, Produktionsstandorte. Dort gibt es Daten, lokale Prozesse, manchmal PV, manchmal Batterien, oft ungenutzte Rechnerzeiten. episodische Intelligenz würde diese Ressourcen nicht blind ausbeuten. Es würde sie in kontrollierte Arbeitsfenster übersetzen.

```
local_ai_day:
  08:00 interactive_only
  11:00 pv_surplus_detected
  11:15 embedding_refresh_start
  13:30 eval_suite_start
  15:00 thermal_limit_reached
  15:05 pause_low_priority_jobs
  22:00 optional_night_batch_if_grid_clean
  06:00 summarize_artifacts_for_morning_review
```

So entsteht eine andere Form von Produktivität: nicht maximale Dauerlast, sondern gute Arbeit zur passenden Zeit.

9.15 Grenzen und Gegenargumente

Dieses Kapitel darf nicht so tun, als wäre Energieopportunismus ein Zaubertrick. Er hat Grenzen.

Erstens: Flexibilität ist nicht immer verfügbar. Viele Inferenzsysteme müssen sofort reagieren. Nutzerinteraktion, Supportprozesse, Sicherheitsalarme und Produktionssysteme können nicht beliebig verschoben werden.

Zweitens: Lokale Hardware kann ineffizient sein. Ein alter Server mit schlechter Effizienz kann mehr Energie pro Ergebnis verbrauchen als ein modernes Rechenzentrum mit spezialisierter Hardware. Energieopportunismus braucht deshalb Messung, nicht Bauchgefühl.

Drittens: Verschiebung kann Rebound erzeugen. Wenn Arbeit günstiger oder grüner erscheint, wird vielleicht einfach mehr gearbeitet. Das System spart dann nicht, sondern vergrößert sein Backlog. Deshalb braucht es harte Budgets.

Viertens: Carbon-Intensity-Daten sind nicht trivial. Durchschnittliche Emissionswerte, marginale Emissionswerte, lokale Netzengpässe und Zertifikatslogik können unterschiedliche Bilder erzeugen. Ein seriöses System muss transparent sagen, welche Metrik es verwendet.

Fünftens: Timing kann mit Datenschutz kollidieren. Räumliches Verschieben ist nur dann akzeptabel, wenn Datenklassifikation, Semantic Compression und Governance wirklich sauber greifen.

Kurz gesagt: Energieopportunistische KI ist keine Ausrede für Wachstum ohne Grenzen. Sie ist eine Disziplin, um unvermeidbare und nützliche Rechenarbeit besser zu platzieren.

9.16 Die nüchterne These

Am Ende dieses Kapitels steht keine Behauptung, dass lokale Solarfenster die großen Rechenzentren ersetzen. Das wäre falsch. Frontier-Training, globale Inferenz und hochverfügbare Systeme brauchen weiterhin massive Infrastruktur.

Aber es wäre genauso falsch zu glauben, dass alle intelligente Arbeit permanent, zentral und sofort passieren muss. Viele Aufgaben können warten. Viele Aufgaben können lokal vorbereitet werden. Viele Aufgaben können bei Energieüberschuss laufen. Viele Aufgaben können abgebrochen, pausiert, checkpointed und später fortgesetzt werden.

Genau dadurch verändert sich die Rolle von KI-Infrastruktur. Sie wird nicht nur schneller. Sie wird taktbarer. Sie bekommt Pausen. Sie bekommt Arbeitsfenster. Sie bekommt einen Stoffwechsel.

Die entscheidende Frage ist nicht, ob KI Strom verbraucht. Die entscheidende Frage ist, ob ihre Architektur klug genug ist, Strom als begrenzte, zeitlich unterschiedliche Ressource zu behandeln.

Energieopportunistische KI bedeutet deshalb: Rechenarbeit wird nicht nur nach Modellqualität und Latenz geplant, sondern nach Energiezustand, Netzsignalen, lokaler Erzeugung, Carbon-Intensity, Datenschutz und Dringlichkeit.

Oder kürzer:

Nicht jede Einsicht muss sofort entstehen. Manche Einsichten dürfen auf Sonne warten.

Übergang: Eine Architektur wird erst glaubwürdig, wenn sie ihre Grenze kennt. Jetzt geht es um das, was lokale Systeme nicht gut können.

Kapitel 10 - Der Hochofen bleibt

Wo der Schwarm endet und dichte Infrastruktur notwendig bleibt

Episodische Intelligenz ist keine Flucht aus der Frontier. Es ist eine Architektur, die genauer entscheidet, wann die Frontier wirklich gebraucht wird.

Nach den Kapiteln über lokale Systeme, Semantic Compression, deterministische Governance, Ameisenrechnen und energieopportunistische Verarbeitung braucht dieses Buch ein Gegengewicht. Ohne dieses Gegengewicht würde die These kippen. Sie würde klingen, als könnten ein paar lokale Knoten, etwas Solarstrom und gutes Routing die großen KI-Rechenzentren einfach ersetzen.

Das wäre falsch.

Die großen Rechenzentren bleiben. Nicht aus Gewohnheit. Nicht nur wegen Macht. Nicht nur wegen Kapital. Sondern weil bestimmte Formen von KI-Arbeit physikalisch, mathematisch und organisatorisch schwer zu verteilen sind. Manche Arbeit ist zerlegbar. Manche Arbeit ist langsam. Manche Arbeit kann warten. Aber manche Arbeit braucht Dichte: hohe Speicherbandbreite, schnelle Netze, eng gekoppelte Beschleuniger, zuverlässige Stromversorgung, Fehlerbehandlung, Monitoring und Teams, die diese Maschine Tag und Nacht am Leben halten.

Das ist die nüchterne Seite episodischer Intelligenz. Der Hochofen verschwindet nicht. Er wird bewusster angefahren.

10.1 Das notwendige Gegenkapitel

Jede starke Architekturthese braucht einen Ort, an dem sie sich selbst widerspricht. Dieses Kapitel ist dieser Ort. Es sagt nicht: Lokal ist unwichtig. Es sagt: Lokal ist nicht alles. Es sagt nicht: Dezentralisierung ist naiv. Es sagt: Dezentralisierung hat Grenzen, und genau diese Grenzen muss man kennen, wenn man sie produktiv überschreiten will.

Das ist besonders wichtig, weil das Ameisenmodell verführerisch ist. Viele kleine Rechner. Viele kleine Aufgaben. Bewusst akzeptierte Latenz. Stromverbrauch dort, wo Energie gerade übrig ist. Das ist elegant. Es hat etwas Demokratisches. Es klingt nach einem Netz statt nach einem Turm.

Aber die Frontier ist nicht nur eine größere Version derselben Aufgabe. Ein Frontier-Modell entsteht nicht dadurch, dass man viele kleine, voneinander fast unabhängige Pakete an viele Rechner verteilt und später zusammensetzt. Zumindest nicht im Kern des heutigen Pretrainings. Training großer Modelle ist ein eng gekoppelter Prozess. Die Maschinen rechnen nicht nur nebeneinander. Sie müssen sich ständig miteinander abstimmen.

Tabelle 10.1 - Aufgaben nach Kopplungsgrad

Aufgabe	Kopplung	Geeignete Ebene
Embeddings für Dokumente	niedrig	lokal, regional, Schwarm
Indexbau und Cache-Warming	niedrig	lokal oder regionale Infrastruktur
Eval-Suites und Regressionstests	niedrig bis mittel	lokal, CI, Schwarm, Forschungscluster
Fine-Tuning kleiner Adapter	mittel	lokal, Firmencluster, regionale Cluster
Domänenspezifisches Pretraining	mittel bis hoch	regionaler Hochleistungscluster
Frontier-Pretraining	hoch	dichter Frontier-Cluster
Multimodales Frontier-Training	sehr hoch	dichter Frontier-Cluster mit spezialisierten Daten- und Netzpfaden
Große Test-Time-Reasoning-Systeme	variabel, oft hoch	häufig zentrale oder regionale Hochleistungsinfrastruktur

Das Ziel dieses Kapitels ist deshalb nicht, den Schwarm kleinzureden. Das Ziel ist, seine richtige Rolle zu bestimmen.

10.2 Training ist ein synchroner Zustand

Beim Training eines großen neuronalen Netzes entstehen in jedem Schritt Gradienten. Vereinfacht gesagt: Das System berechnet, wie es seine Gewichte verändern sollte, damit es beim nächsten Mal etwas weniger falsch liegt. Wenn das Modell über viele GPUs verteilt ist, müssen diese Informationen ständig zwischen den Geräten abgestimmt werden.

Das ist der Unterschied zu vielen klassischen verteilten Rechenprojekten. Ein SETI-ähnliches System kann unabhängige Datenpakete verteilen. Jeder Knoten rechnet sein Paket, sendet ein Ergebnis zurück, fertig. Bei großem LLM-Training sind die Arbeitspakete viel stärker miteinander verwoben. Die Aktualisierung des Modells hängt davon ab, was viele andere Knoten im selben Trainingsschritt berechnet haben.

Natürlich gibt es asynchrone Verfahren, lokale Updates, föderierte Ansätze und Low-Communication-Training. Sie sind wichtig. Sie sind Teil des Ökosystems episodischer Intelligenz. Aber je größer, dichter und frontier-näher das Training wird, desto stärker schlägt die Synchronisationsfrage zurück.

Tabelle 10.2 - Warum ein Trainingsschritt koppelt

Phase	Was passiert?	Warum Kopplung entsteht
Datenbatch laden	Viele Beschleuniger erhalten Teilmengen der Trainingsdaten.	Datenzufuhr muss zum Rechentakt passen.
Forward Pass	Das Modell berechnet Vorhersagen.	Bei Modellparallelismus liegen Modellteile auf verschiedenen Geräten.
Fehler berechnen	Die Vorhersage wird mit dem Ziel verglichen.	Verlustwerte müssen korrekt den verteilten Modellteilen zugeordnet werden.
Backward Pass	Gradienten werden berechnet.	Viele Geräte erzeugen Teilinformationen desselben globalen Updates.
Gradienten austauschen	All-Reduce, Parameter- oder Pipeline-Kommunikation.	Hier wird das Netzwerk operativ Teil des Trainings.
Gewichte aktualisieren	Optimizer verändert Parameter.	Alle Replikas müssen wieder konsistent werden.
Nächster Schritt	Der Takt beginnt erneut.	Langsame Worker, Ausfälle und Jitter bremsen den Verbund.

Das Entscheidende ist der letzte Satz: Das Netzwerk wird Teil des Modells. Nicht semantisch, aber operativ. Wenn das Netzwerk zu langsam, zu unzuverlässig oder zu heterogen ist, sinkt die Trainingsleistung. Der Rechenverbund verhält sich dann nicht wie ein größerer Computer, sondern wie ein Chor, der ständig aus dem Takt fällt.

10.3 Bandbreite ist die Mauer

In lokalen oder offenen Schwärmen ist Latenz oft akzeptabel, wenn die Aufgabe grob genug zerlegbar ist. Bei Frontier-Training wird Bandbreite zur Mauer. Moderne Trainingssysteme sind deshalb nicht nur Sammlungen von GPUs. Sie sind Netzwerkkonstruktionen.

NVIDIAs aktuelle Rack-Scale-Systeme zeigen dieselbe Richtung noch deutlicher. Das GB200 NVL72 verbindet 72 Blackwell-GPUs und 36 Grace-CPUs in einer NVLink-Domain; der NVLink-Switch-Fabric wird mit 130 TB/s Low-Latency-GPU-Kommunikation beschrieben. Für die Skalierung über Racks hinweg kommen Hochgeschwindigkeitsnetze wie Quantum-2 InfiniBand mit 400 Gb/s, RDMA und In-Network-Computing ins Spiel. Solche Zahlen sind keine Marketing-Ornamente. Sie erklären, warum große Modelle in speziell gebauten Clustern trainiert werden.

Google zeigt dasselbe Prinzip auf einer anderen Hardwarelinie. Ein TPU-v5p-Pod umfasst laut Google 8.960 Chips, verbunden über High-Bandwidth-Inter-Chip-Interconnect mit 4.800 Gbps pro Chip in einer 3D-Torus-Topologie. Die früheren TPU-v4-Papers und die USENIX-Arbeit zur Resilienz des TPU-v4-Supercomputers machen dieselbe operative Wahrheit deutlich: Diese Systeme sind nicht einfach Rechenleistung. Sie sind Netzwerke, Fehlertoleranz, Scheduling, Topologie und operative Disziplin in einem Gehäuse aus Strom und Kühlung.

Das ist der Kernfehler vieler naiver Dezentralisierungserzählungen: Sie zählen FLOPS und vergessen die Kommunikation. Aber ein Trainingscluster ist nicht nur die Summe seiner Recheneinheiten. Er ist die Geschwindigkeit, mit der diese Einheiten gemeinsame Zustände herstellen können.

Gleichzeitig wird genau an dieser Grenze geforscht. DiLoCo und später Decoupled-DiLoCo zeigen, dass Training über schwächer verbundene Inseln möglich werden kann, wenn Kommunikation drastisch reduziert und Synchronisierung entkoppelt wird. Das ist wichtig für episodische Intelligenz. Es beweist aber nicht, dass Bandbreite plötzlich egal ist. Es beweist, dass Bandbreite so wichtig ist, dass ganze Trainingsverfahren um sie herum neu gebaut werden.

Bei Frontier-Training ist die Frage nicht nur: Wie viele GPUs haben wir? Die Frage ist: Wie schnell können diese GPUs gemeinsam eins sein?

10.4 Eine kleine Netzwerkrechnung

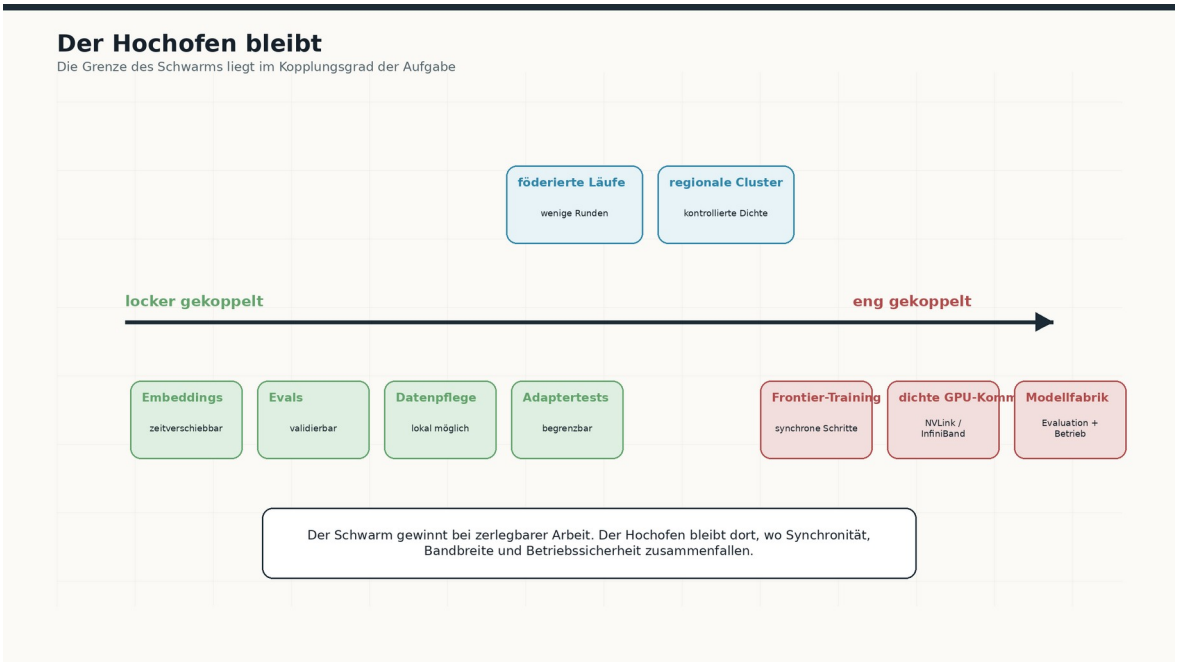

Abb. 11 - Der Hochofen bleibt: zerlegbare Aufgaben links, eng gekoppelte Frontier-Aufgaben rechts.

Die folgende Rechnung ist bewusst grob. Sie will keine reale Trainingsarchitektur modellieren. Sie zeigt nur, warum Kommunikation nicht wie eine Nebensache behandelt werden darf.

Nehmen wir an, ein Worker müsse pro Trainingsschritt einen Gigabyte an Zustandsinformationen austauschen. In echten Systemen kann dieser Wert kleiner oder deutlich größer sein, je nach Parallelisierungsstrategie, Modellgröße, Batchgröße, Optimizer und Kommunikationsverfahren. Der Punkt ist nicht die genaue Zahl. Der Punkt ist die Größenordnung.

Tabelle 10.3 - Ein Gigabyte ist je nach Netz etwas völlig anderes

Netz / Verbindung	Theoretischer Durchsatz	Zeit für 1 GB, idealisiert	Bedeutung
400 Gb/s Cluster-Fabric	ca. 50 GB/s	ca. 0,02 s plus Overhead	Für dicht gekoppelte Cluster gebaut.
100 Gb/s Cluster-/Datacenter-Netz	ca. 12,5 GB/s	ca. 0,08 s plus Overhead	Noch immer professionelle Infrastruktur.
10 Gb/s LAN	ca. 1,25 GB/s	ca. 0,8 s plus Overhead	Für viele lokale Aufgaben gut, für enge Trainingsschritte oft langsam.
1 Gb/s LAN	ca. 0,125 GB/s	ca. 8 s plus Overhead	Jeder Schritt wird durch Kommunikation aufgefressen.
100 Mb/s WAN	ca. 0,0125 GB/s	ca. 80 s plus Jitter	Für eng gekoppelte Trainingsarbeit praktisch ungeeignet.

Diese Tabelle ist noch freundlich. Sie ignoriert Paketverluste, Protokoll-Overhead, Jitter, Verschlüsselung, konkurrierende Last, asymmetrische Uploadraten und die Tatsache, dass nicht ein Worker kommuniziert, sondern viele. Sie erklärt trotzdem den Kern: Ein Trainingssystem besteht nicht nur aus Rechenoperationen. Es besteht aus gemeinsamem Takt.

Darum sind Verfahren wie DiLoCo und Decoupled-DiLoCo so interessant. Sie versuchen nicht, die Mauer wegzureden. Sie ändern die Architektur so, dass seltener über diese Mauer geklettert werden muss. Genau das macht sie relevant für episodische Intelligenz: Sie verschieben die Grenze des Verteilbaren, aber sie heben die Physik nicht auf.

10.5 Frontier-Cluster sind Spezialmaschinen

Ein Frontier-Cluster ist keine Halle mit vielen Grafikkarten. Er ist eine Spezialmaschine, die auf wenige extreme Ziele optimiert ist: hohe Auslastung teurer Beschleuniger, schnelle Kommunikation, schnelle Speicherpfade, zuverlässige Stromversorgung, effiziente Kühlung, kontrollierte Fehlerbehandlung und möglichst wenig Stillstand.

Meta beschrieb für Llama 3, dass die größten Modelle mit einer Kombination aus Datenparallelismus, Modellparallelismus und Pipelineparallelismus trainiert wurden. Meta berichtete außerdem von Trainingsläufen auf zwei eigens gebauten 24K-GPU-Clustern und einer Trainings-Stack-Infrastruktur, die Fehlererkennung, Fehlerbehandlung und Wartung automatisiert, um GPU-Uptime zu maximieren.

Auch die Investitionsrichtung der Frontier bestätigt das. OpenAI und NVIDIA kündigten 2025 eine strategische Partnerschaft an, um mindestens 10 Gigawatt an KI-Rechenzentrumsinfrastruktur mit NVIDIA-Systemen bereitzustellen, mit einer ersten Gigawatt-Phase ab der zweiten Jahreshälfte 2026. Man muss diese Größenordnung nicht als wünschenswert feiern. Aber sie zeigt, dass die Industrie zentrale Hochenergie-KI nicht als Übergangsfehler behandelt, sondern als strategische Basis für nächste Modellgenerationen.

Diese Details sind für das Buch wichtiger als die reine Zahl der GPUs. Sie zeigen: Die Frontier ist nicht nur Compute. Sie ist Organisation von Compute. Sie ist ein Betriebszustand.

Die bekannten Systeme wie Megatron-LM, ZeRO/DeepSpeed, Pathways oder TPU-v5p sind keine Nebengeschichten. Sie sind die Softwareseite derselben Wahrheit. Große Modelle passen nicht einfach auf einen Chip. Sie müssen über Geräte, Knoten und manchmal ganze Pod-Strukturen verteilt werden. Dabei entstehen neue Engpässe: Speicher, Kommunikation, Checkpoints, Fehlertoleranz, Scheduling, Datenzufuhr.

Tabelle 10.4 - Frontier-Cluster als Betriebsmaschine

Schicht	Funktion
Beschleuniger	GPU, TPU oder andere AI-Accelerator-Kerne tragen die dichte Rechenarbeit.
High-Bandwidth Memory	Versorgt die Rechenkerne mit Modellzustand und Aktivierungen.
NVLink, TPU-Mesh oder vergleichbare Interconnects	Stellen schnelle Kopplung innerhalb enger Rechendomänen her.
InfiniBand, Ethernet-Fabric oder Custom Interconnect	Skalieren über Knoten, Racks und Pod-Strukturen.
Verteiltes Dateisystem / Datenpipeline	Liefert Daten ohne die Beschleuniger verhungern zu lassen.
Checkpointing	Sichert Fortschritt, damit Ausfälle nicht ganze Läufe zerstören.
Health Monitoring und Scheduler	Erkennen Fehler, verschieben Arbeit und halten Auslastung hoch.
Validation- und Safety-Eval-Stack	Prüfen Zwischenstände und Modellgenerationen.
Operations-Team	Betrieb, Reparatur, Kapazitätsplanung und Incident Response.

Ein lokaler Schwarm kann viele dieser Elemente in kleinerer Form besitzen. Aber bei der Frontier werden sie zu einer industriellen Maschine. Und industrielle Maschinen entstehen nicht zufällig.

10.6 Training ist doch zeitkritisch

An dieser Stelle lohnt ein Rückgriff auf eine frühere Intuition: Modelltraining scheint nicht zeitrelevant zu sein. Niemand wartet im Chatfenster auf den Abschluss eines mehrmonatigen Trainingslaufs. Also könnte man meinen: Warum nicht langsamer, günstiger, verteilter?

Die Antwort ist: Training ist nicht in derselben Weise zeitrelevant wie eine Nutzeranfrage. Aber es ist trotzdem zeitkritisch.

Erstens ist teure Hardware gebunden. Eine GPU, die auf langsame Worker wartet, ist kein poetisches Symbol für Geduld, sondern ein Kapitalgut im Leerlauf. Zweitens altern Hardware und Modelle schnell. Ein Trainingslauf, der zu lange dauert, kann wirtschaftlich oder wissenschaftlich an Wert verlieren. Drittens zählt Iterationsgeschwindigkeit. Frontier-Forschung besteht nicht aus einem einzigen perfekten Lauf. Sie besteht aus vielen Entscheidungen: Datenmischung, Architektur, Hyperparameter, Sicherheit, Evaluation, Post-Training. Wer schneller lernt, lernt mehr Zyklen.

Viertens steigt bei langen, großen Läufen die Ausfallwahrscheinlichkeit. Tausende Beschleuniger, Netzwerkkarten, Speicherkomponenten und Server laufen über lange Zeiträume. Irgendetwas fällt immer aus. Google und Meta veröffentlichen nicht zufällig Arbeiten über Resilienz, Fehlererkennung, Recovery und Hardware-Triage.

Zeitrelevanz bedeutet hier nicht: Ein Mensch wartet auf eine Antwort. Zeitrelevanz bedeutet: Die gesamte Forschungsmaschine wartet auf den nächsten Erkenntniszyklus.

10.7 Skalierung hat weiterhin empirische Macht

Es wäre bequem zu sagen, dass Größe nur noch Marketing ist. Aber so einfach ist es nicht. Skalierung hat in der modernen KI über Jahre reale Wirkung gezeigt. Die klassischen Scaling-Law-Arbeiten von Kaplan et al. und die Chinchilla-Arbeit von Hoffmann et al. zeigen, dass Modellgröße, Datenmenge und Trainingscompute systematisch mit Leistung zusammenhängen, auch wenn die Details, Datenqualität und optimale Ressourcenverteilung komplexer sind als frühe einfache Erzählungen.

Epoch AI schätzt, dass der Trainingscompute von Frontier-Modellen von 2010 bis Mai 2024 grob um den Faktor 4 bis 5 pro Jahr gewachsen ist. Diese Zahl ist keine Naturkonstante. Sie kann sich ändern. Aber sie beschreibt die bisherige Richtung: Frontier-Fähigkeiten wurden nicht nur durch bessere Algorithmen gewonnen, sondern auch durch massive Rechenbudgets, Hardwarefortschritt und Systemengineering.

Episodische Intelligenz darf diese Realität nicht leugnen. Lokale Systeme werden wichtiger. Deterministische Layer werden wichtiger. Semantic Compression wird wichtiger. Aber es gibt weiterhin Aufgaben, bei denen neue Basismodelle, neue multimodale Fähigkeiten, neue Reasoning-Fähigkeiten und neue wissenschaftliche Systeme von sehr großen Trainings- und Evaluationsläufen abhängen.

Die richtige Gegenposition lautet deshalb nicht: Skalierung ist tot. Die richtige Gegenposition lautet: Skalierung ist zu teuer, um sie gedankenlos für jede Aufgabe zu verwenden.

Die stärkere These lautet nicht: große Modelle werden überflüssig. Sie lautet: Große Modelle bleiben knapp - und gerade deshalb muss ihre Nutzung besser orchestriert werden.

10.8 Frontier als Modellfabrik

Eine produktive Metapher ist die Frontier als Modellfabrik. Die Fabrik stellt nicht jede Schraube lokal her. Sie produziert Grundmaschinen: Basismodelle, multimodale Repräsentationen, Reasoning-Fähigkeiten, allgemeine Sprach- und Weltmodelle. Lokale Systeme nehmen diese Grundmaschinen und bauen daraus Werkzeuge: Adapter, Runtimes, Retrieval-Schichten, Governance-Contracts, Domänenlogik, lokale Speicher.

In dieser Sichtweise ist die Frontier nicht der Ort jeder Intelligenz. Sie ist der Ort, an dem bestimmte teure Formen allgemeiner Fähigkeit entstehen. Diese Fähigkeit kann danach lokalisiert, verdichtet, geroutet, eingeschränkt, überprüft und seltener aufgerufen werden.

Das ist ein wichtiger Unterschied. Wenn man die Frontier als ständigen Antwortautomaten begreift, wirkt sie verschwenderisch. Wenn man sie als Fabrik für seltene, allgemeinere Fähigkeiten begreift, bekommt sie eine andere Rolle. Sie erzeugt Kapazität, die anschließend in vielen kleineren Systemen genutzt und kanalisiert wird.

Diese Fabrikmetapher schützt auch vor einer falschen Romantik der lokalen Autonomie. Lokale Systeme können stark sein, aber sie erben oft Fähigkeiten, die irgendwo zentral entstanden sind. Ein kleines Modell, ein quantisiertes Modell, ein lokaler Agent, ein Edge-Klassifikator - viele dieser Bausteine stehen auf den Schultern großer Trainingsläufe.

Die lokale Maschine ist souveräner, wenn sie die Frontier nicht verleugnet, sondern sie dosiert.

10.9 Zentralisierung ist auch Qualitätskontrolle

Zentralisierung hat nicht nur Nachteile. Sie kann Qualität kontrollierbarer machen. Große Trainingsumgebungen haben standardisierte Datenpipelines, Monitoring, Metriken, Checkpoints, Security-Prozesse, Evaluationsumgebungen, Red-Teaming, Zugriffskontrollen und Incident-Prozesse. Das ist mühsam, teuer und oft unsichtbar. Aber es zählt.

Ein offener Schwarm hat dagegen andere Probleme: heterogene Hardware, unzuverlässige Knoten, schwankende Netzwerke, unbekannte Betreiber, manipulierte Ergebnisse, Datenabfluss, Modell-Poisoning, Aktivierungs- oder Gradienten-Leckage, schwierige Auditierbarkeit. Kapitel 8 hat diese Risiken bereits angerissen. Kapitel 10 muss sie festhalten: Je näher Arbeit an sensiblem Training, sicherheitsrelevanter Modellverbesserung oder vertraulichen Daten liegt, desto härter werden die Anforderungen an Provenance, Validierung und Isolation.

Das heißt nicht, dass dezentrale Systeme unsicher sein müssen. Es heißt: Sie müssen ihre Sicherheit beweisen. Und dieser Beweis kostet ebenfalls Infrastruktur.

Daraus folgt eine einfache Trennung: Offene oder breite Schwärme eignen sich eher für Arbeitspakete, die validierbar, unsensibel und wiederholbar sind. Kontrollierte Edge-Verbuende eignen sich für lokale, organisationsnahe Arbeit. Frontier-Training bleibt dort, wo Hardware, Daten, Teams und Governance in einem hochintegrierten Betrieb zusammenkommen.

Als einfache Regel gilt: Je näher eine Arbeit an sensiblen Daten, Modellgewichten, Compliance oder Sicherheitsentscheidungen liegt, desto mehr Kontrolle, Provenance und Audit braucht sie. Je unsensibler, validierbarer und wiederholbarer sie ist, desto mehr Verteilung, Latenz und Schwarmarbeit kann sie vertragen.

10.10 Inferenz bleibt ebenfalls nicht trivial

Bisher ging es vor allem um Training. Aber auch Inferenz bleibt nicht vollständig lokal lösbar. Viele Alltagsanfragen können lokal oder mit kleinen Modellen erledigt werden. Viele Business-Prozesse können durch Routing, Retrieval, Caching und lokale Semantik stark entlastet werden. Trotzdem gibt es Inferenzfälle, die zentrale Frontier-Infrastruktur behalten.

Das betrifft vor allem sehr große multimodale Modelle, lange Kontexte, komplexe Toolketten, hohe Parallelität, spezialisierte Reasoning-Modelle und Aufgaben mit variablem Test-Time-Compute. Je mehr ein System nicht nur ein paar Token erzeugt, sondern mehrere Hypothesen prüft, Tools nutzt, Dateien analysiert, Code ausführt, Bilder versteht, Audio transkribiert, Sicherheitsregeln anwendet und Zwischenergebnisse bewertet, desto stärker wird die Runtime selbst zum Rechenzentrumsthema.

Hier liegt eine kleine Ironie. Episodische Intelligenz reduziert unnötige Frontier-Calls. Aber die verbleibenden Frontier-Calls werden möglicherweise wertvoller und schwerer. Sie sind nicht mehr das Default-Rauschen jeder Kleinigkeit. Sie sind Eskalationsmomente. Und Eskalationsmomente können tiefer, länger und rechenintensiver sein als klassische Chatantworten.

Auch deshalb bleibt zentrale Infrastruktur. Nicht für jede Frage. Aber für die Fragen, bei denen das lokale System bewusst sagt: Jetzt brauche ich den schweren Apparat.

10.11 Warum der Schwarm trotzdem gewinnt

Dieses Kapitel verteidigt die Frontier. Aber es widerruft die vorherigen Kapitel nicht. Im Gegenteil. Gerade weil Frontier-Infrastruktur teuer, dicht, energiehungrig und schwer zu betreiben ist, wird der Schwarm wichtiger.

Die lokale und verteilte Arbeit gewinnt überall dort, wo sie die Frontier entlastet: Daten vorbereiten, Dokumente strukturieren, Embeddings erzeugen, Retrieval verbessern, kleine Modelle anpassen, Evals fahren, Policies testen, Kontexte komprimieren, Caches füllen, lokale Gedächtnisse pflegen, Geschäftslogik schützen, einfache Anfragen beantworten, Vorentscheidungen treffen.

Das ist kein Ersatzkrieg. Es ist Arbeitsteilung.

In der Arbeitsteilung bereitet der Schwarm vor, filtert, validiert, verdichtet, lernt lokal, nutzt Überschuss und schützt Kontext. Die Frontier generalisiert, trainiert Grundmodelle, verarbeitet schwierige Eskalationen, erzeugt neue Fähigkeiten und dient als Hochenergie-Spezialist. Das ist kein Ersatzkrieg. Es ist Arbeitsteilung.

Die Stärke episodischer Intelligenz liegt genau darin, dass es die Frontier nicht aus dem System entfernt. Es befreit sie von Aufgaben, für die sie zu teuer ist.

10.12 Der regionale Hochenergie-Kern

Der interessante Zwischenraum liegt nicht zwischen Wohnzimmer und Weltkonzern, sondern zwischen lokaler Runtime und planetarem Frontier-Labor. Genau dort entstehen öffentliche Supercomputer, nationale KI-Infrastrukturen, wissenschaftliche Cluster, Branchenverbünde und europäische AI-Factories.

Es gibt noch eine Zwischenform: nicht globaler Hyperscaler, nicht Wohnzimmerknoten, sondern regionale Hochleistungsinfrastruktur. Europa, die Schweiz, Hochschulen, Branchenverbünde, Kommunen oder Unternehmen können eigene kontrollierte Cluster betreiben, ohne gleich planetare Frontier-Labs zu sein.

Solche regionalen Systeme könnten Aufgaben übernehmen, die lokale Knoten überfordern, aber nicht zwingend bei wenigen globalen Anbietern landen müssen: domänenspezifisches Pretraining, vertrauliches Fine-Tuning, wissenschaftliche Simulationen, öffentliche Modelle, föderierte Evaluationsinfrastruktur, öffentliche Datenräume, souveräne Embedding- und Retrieval-Services.

Das ist wichtig für die politische Dimension des Buches. Die Alternative zu totaler Zentralisierung ist nicht totale Kleinteiligkeit. Zwischen beiden liegen viele Schichten: lokale Runtime, Firmencluster, regionale Rechenverbünde, wissenschaftliche Supercomputer, nationale KI-Infrastruktur, globale Frontier-Zentren.

Tabelle 10.5 - Ebenen der episodischen Infrastruktur

Ebene	Typische Aufgabe	Souveränitätsfrage
Client / Gerät	lokale Semantik, Cache, kleine Modelle, Policy-Vorprüfung	Bleiben Kontext und Identität unter lokaler Kontrolle?
Lokale Runtime / Bürocluster	Embeddings, Evals, Adapter, lokale Retrievalpflege	Sind Betrieb, Logs und Datenklassifikation auditierbar?
Regionale Edge-Föderation	gemeinsame Artefakte, Branchenmodelle, kontrollierte Schwarmarbeit	Wer prüft Nodes, Provenance und Ergebnisqualität?
Souveräne Cloud / AI Factory	domänenspezifisches Training, öffentliche Modelle, vertrauliche Forschung	Gibt es echte Portabilität und Wechseloptionen?
Forschungs-Supercomputer	wissenschaftliche Simulationen, große Modelle für Forschung und öffentliche Infrastruktur	Welche Zugangsregeln und Gemeinwohlziele gelten?
Frontier-Datacenter	globale Modellgenerationen, extreme Trainingsläufe, hochskalierte Inferenzspitzen	Wie wird Abhängigkeit begrenzt?

Europa bewegt sich bereits in diese Richtung. EuroHPC beschreibt AI Factories als Ökosysteme um AI-optimierte Supercomputer, die Rechenressourcen und Unterstützung für Industrie und Wissenschaft bereitstellen. In der Schweiz zeigt der Alps-Supercomputer am CSCS eine ähnliche Zwischenebene: keine Wohnzimmerinfrastruktur, aber auch kein privates globales Frontier-Labor. Solche Systeme sind wichtig, weil sie den Raum zwischen lokalem Arbeiten und vollständiger Abhängigkeit von wenigen globalen Plattformen öffnen.

Das ist keine Absage an Datacenter. Es ist eine Absage an den gedankenlosen Default. Die Frage lautet bei jeder Aufgabe: Welche Schicht ist angemessen?

10.13 Die ehrliche Grenze der Dezentralisierung

Die ehrlichste Grenze lautet: Nicht alles, was verteilt werden kann, sollte verteilt werden. Und nicht alles, was lokal laufen kann, ist dadurch besser.

Lokale Systeme können ineffizient sein. Sie können schlecht gewartet werden. Sie können Sicherheitslücken haben. Sie können still und leise falsche Artefakte erzeugen. Sie können Energie dort verbrauchen, wo das Netz gerade belastet ist. Sie können komplizierter sein als der zentrale Dienst, den sie ersetzen sollten.

Auch Schwärme sind nicht automatisch demokratisch. Sie können von wenigen Betreibern dominiert werden. Sie können Proof-of-Work-artige Verschwendung erzeugen. Sie können Verantwortung verdünnen. Sie können auditierbare Qualität erschweren.

Darum braucht Episodische Intelligenz Governance. Kapitel 7 war dafür kein Schmuck. Es war die Bedingung. Ohne Contracts, Audit Logs, Validierung, Datenklassifikation und Eskalationsregeln wird lokale Infrastruktur nicht souverän, sondern chaotisch.

Die Zukunft liegt nicht in der Abschaffung großer Maschinen. Sie liegt in der Fähigkeit, große und kleine Maschinen richtig zu kombinieren.

10.14 Die Schlussfolgerung der episodischen Intelligenz

Am Ende dieses Kapitels steht ein anderer Blick auf die großen Rechenzentren. Sie sind nicht einfach das Problem. Sie sind auch nicht einfach die Lösung. Sie sind Hochenergie-Spezialisten. Sie werden gebraucht, wenn allgemeine Fähigkeiten trainiert, große Modelle koordiniert, komplexe Inferenzspitzen verarbeitet oder neue Modellgenerationen erzeugt werden.

Aber gerade weil sie Hochenergie-Spezialisten sind, sollten sie nicht mit Alltagsarbeit überflutet werden. Eine triviale Anfrage, ein lokaler Dokumentenfilter, ein Cache-Hit, eine einfache Klassifikation, ein vordefinierter Workflow, eine prüfbare Businessregel - all das muss nicht automatisch den Hochofen anwerfen.

Die zentrale Frage episodischer Intelligenz lautet deshalb nicht: Zentral oder dezentral? Die Frage lautet: Welche kognitive Arbeit gehört auf welche Ebene?

Tabelle 10.6 - Entscheidungsmodell für Frontier-Eskalation

Frage	Wenn ja	Warum
Kann eine deterministische Regel lösen?	lokale Governance	Geringste Kosten, höchste Prüfbarkeit.
Reicht ein kleines Modell?	lokales Modell	Nähe zu Daten und geringe Latenz.
Reicht Retrieval?	lokales oder regionales RAG	Wissen wird geholt statt teuer neu erzeugt.
Kann die Arbeit warten?	energieopportunistische Queue	Zeit wird gegen Energie- und Kostenqualität getauscht.
Braucht die Aufgabe allgemeines Reasoning?	Frontier-Eskalation	Der dichte Rechenkern wird bewusst eingeschaltet.
Braucht die Welt neue Fähigkeiten?	Frontier-Training oder Forschung	Hier beginnt die Modellfabrik.

So bekommt die Frontier eine bessere Rolle. Sie wird nicht kleiner gedacht, sondern seltener. Nicht entwertet, sondern geschützt. Nicht abgeschafft, sondern entlastet.

Der Hochofen bleibt. Aber man wirft nicht mehr jedes Streichholz hinein.

Teil IV - Souveränität und Betriebsform

Am Ende wird aus Architektur Politik. Wer Chips, Strom, Cloud, Daten, Semantik und Interfaces kontrolliert, kontrolliert Eintrittstore. Teil IV fragt, wie digitale Souveränität praktisch wird - nicht als Rückzug, sondern als Betriebsfähigkeit mit Ausgängen.

Kapitel 11 - Die Macht der Infrastruktur

Wie Chips, Cloud, Energie und Interfaces Eintrittstore bilden

Die Frage ist nicht nur, wer das beste Modell besitzt. Die Frage ist, wer die Stromleitungen, Chips, Cloud-Verträge, Interfaces und Defaults kontrolliert, durch die das Modell überhaupt in die Welt kommt.

Arbeitsstatus
Lesefassung v0.2 - Kapitel 11.
Ziel: Die politische und ökonomische Seite episodischer Intelligenz öffnen, ohne in Anti-Datacenter-Rhetorik zu kippen.
Dieses Kapitel baut auf Kapitel 10 auf: Frontier bleibt notwendig. Aber Notwendigkeit erzeugt Abhängigkeit, wenn Alternativen fehlen.

11.1 Der politische Körper der KI

Bis hierhin hat dieses Buch vor allem über Architektur gesprochen: Strom, Kühlung, Routing, lokale Systeme, semantische Verdichtung, Frontier-Eskalation. Das war absichtlich technisch. Denn wer zu früh politisch wird, ohne den Maschinenraum verstanden zu haben, verliert schnell die Bodenhaftung.

Aber nach Kapitel 10 lässt sich die politische Frage nicht mehr vermeiden. Wenn Frontier-Training bestimmte Cluster, Netzwerke, Chips, Grundstücke, Strommengen und Kapitalmassen braucht, dann entsteht daraus Macht. Nicht, weil die Beteiligten zwangsläufig böse wären. Sondern weil Infrastruktur immer Macht sammelt.

Ein Modell ist nicht nur ein mathematisches Artefakt. Es ist das sichtbare Ergebnis einer Lieferkette: Silizium, Kühlung, Stromverträge, Glasfaser, Grundstücke, Genehmigungen, Softwarestacks, Datenpipelines, Forschungsabteilungen, Cloud-Plattformen, Sicherheitszertifizierungen und Vertriebswege. Wer diese Kette kontrolliert, kontrolliert nicht automatisch die Zukunft. Aber er kontrolliert die Eintrittstore.

Das ist der Moment, in dem KI aufhört, nur Software zu sein. Sie wird Infrastrukturpolitik.

Leitfrage dieses Kapitels
Welche Formen von Macht entstehen, wenn künstliche Intelligenz auf knappe physische und organisatorische Ressourcen angewiesen ist?
Und welche Rolle kann Episodische Intelligenz spielen, um Abhängigkeit nicht zu leugnen, sondern verhandelbar zu machen?

11.2 Macht entsteht nicht nur am Modell

Die naive Vorstellung lautet: Wer das beste Modell hat, gewinnt. Das ist zu kurz. In echten Märkten gewinnt oft nicht die beste Einzelkomponente, sondern derjenige, der die kritischen Übergänge kontrolliert.

Bei KI sind diese Übergänge zahlreich. Ein Unternehmen kann ein gutes Modell besitzen und trotzdem abhängig bleiben, wenn es keinen Zugang zu bezahlbarer Inferenz hat. Eine Universität kann exzellente Forschung leisten und trotzdem an der Frontier kaum mitspielen, wenn ihr Cluster zu klein ist. Ein Staat kann eine KI-Strategie schreiben und trotzdem wenig Souveränität besitzen, wenn Strom, Chips, Cloud, Runtime und Interfaces durch externe Akteure bestimmt werden.

Deshalb muss man Macht in der KI nicht als eine Pyramide sehen, sondern als Stack. Jede Schicht kann zum Gate werden. Und manche Akteure kontrollieren mehrere Schichten gleichzeitig.

Der Stack der KI-Macht

Schicht	Worum es geht	Machtwirkung
Chips	GPUs, Beschleuniger, Speicher, Netzwerkadapter	Knappheit, Preis, Exportkontrolle, Lieferzeiten
Datacenter	Strom, Kühlung, Grundstücke,	Standortmacht, Genehmigungen, lokale

	Netzanschluss	Konflikte
Cloud	Compute, Speicher, Identität, APIs, Compliance	Lock-in, Preismacht, Bündelung, Zugang
Modelle	Frontier, offene Modelle, Spezialmodelle	Qualität, Markenvertrauen, Sicherheitsstandards
Runtime	Routing, Tools, Agenten, Eval, Logs	Betriebskontrolle, Auditierbarkeit, Switching-Kosten
Interfaces	Office, Chat, OS, Browser, IDEs, Apps	Default-Macht, Distribution, Nutzerbindung
Daten	Trainingsdaten, Unternehmenswissen, Feedback	Verbesserungsschleifen, Personalisierung, Kontextvorteile
Regeln	Standards, Zertifizierung, Beschaffung, Haftung	Marktzugang, Vertrauen, Compliance-Barrieren

Dieser Stack erklärt, warum Machtkonzentration auch dann entstehen kann, wenn es viele Modelle gibt. Wettbewerb auf der Modelloberfläche hilft wenig, wenn die Infrastruktur darunter, die Schnittstellen darüber und die Beschaffungskanäle daneben stark konzentriert sind.

11.3 Kapital als Zugangskarte

Die Frontier ist teuer. Nicht ein bisschen teuer. Industriell teuer. Wer heute große KI-Infrastruktur baut, kauft nicht nur Server. Er kauft Stromoptionen, Grundstücke, Netzanschlüsse, Transformatoren, Glasfaser, Kühltechnik, Spezialpersonal, Lieferketten und Zeitfenster in der Chipproduktion.

Diese Kapitaldimension verändert den Charakter des Wettbewerbs. In klassischer Software konnte ein kleines Team mit wenig Geld, hoher Geschwindigkeit und guter Idee erstaunlich weit kommen. In der KI kann das noch immer passieren - aber seltener auf der Ebene des Frontier-Trainings. Dort wird Kapital selbst zur Zugangskarte.

Die aktuellen Zahlen zeigen die Richtung. Microsoft beschreibt im Geschäftsbericht 2025 mehr als 400 Rechenzentren in 70 Regionen und über zwei Gigawatt neu hinzugefügte Kapazität in einem Jahr. Meta erwartet für 2026 Investitionen in der Größenordnung von 115 bis 135 Milliarden US-Dollar, getrieben durch AI-Infrastruktur und Superintelligence-Labs. NVIDIA meldete für Q1 FY2027 81,6 Milliarden US-Dollar Umsatz, davon 75,2 Milliarden im Datacenter-Geschäft. Diese Zahlen sind nicht nur beeindruckend. Sie sind ein neues Eintrittsschild an der Tür.

Wer mit solchen Summen arbeiten kann, kann nicht nur Modelle trainieren. Er kann Märkte vorstrukturieren. Er kann Kunden binden, Talente einkaufen, Kapazität vorfinanzieren, Lieferanten priorisieren, eigene Standards setzen und Fehler länger aushalten als kleinere Akteure. Kapital wird dadurch nicht bloß Finanzierung. Es wird ein strategischer Puffer.

Kalte Formulierung
In Frontier-KI ist Kapital nicht nur Mittel zum Zweck. Kapital wird zur Zeitmaschine: Es erlaubt, Infrastruktur Jahre vor der Monetarisierung zu bauen und dadurch die Möglichkeiten anderer Akteure zu verengen.

11.4 Cloud als Verteilungsmaschine

Die Cloud ist nicht nur ein Ort, an dem Rechenleistung gemietet wird. Sie ist eine Verteilungsmaschine. Sie bündelt Identität, Abrechnung, Sicherheitsfreigaben, Datenbanken, Monitoring, Netzwerke, Modellzugang, Marketplace, Support und Beschaffung. Wer bereits in einer Cloud lebt, greift oft zuerst zu den KI-Angeboten derselben Cloud.

Das ist praktisch. Genau darin liegt die Macht. Denn Beschaffung ist nicht neutral. Entwickler nehmen den Dienst, der sofort verfügbar ist. Unternehmen nehmen den Anbieter, der bereits im Vendor-Management liegt. Sicherheitsabteilungen nehmen den Stack, den sie schon zertifiziert haben. Führungskräfte nehmen den Vertrag, der die wenigsten neuen Gremien braucht.

Synergy Research berichtete für Q3 2025, dass Amazon, Microsoft und Google zusammen 63 Prozent der weltweiten Unternehmensausgaben für Cloud-Infrastruktur hielten. Diese Zahl sagt nicht, dass jede

Innovation aus drei Häusern kommt. Aber sie zeigt, wie eng die Verteilungswege für skalierbare KI an die großen Cloud-Plattformen gekoppelt sind.

Damit kann ein KI-Markt formal offen wirken und trotzdem praktisch stark kanalisiert sein. Ein Startup kann theoretisch sein Modell überall anbieten. Praktisch braucht es GPU-Zugang, Cloud-Credits, Enterprise-Vertrieb, Sicherheitszertifikate, Kundenvertrauen und Integrationen in bestehende Workflows. Der Engpass sitzt selten an nur einer Stelle. Er sitzt im ganzen Weg zum Kunden.

11.5 Partnerschaften: Brücke und Leine

Partnerschaften zwischen Modellfirmen und großen Plattformen werden oft als Innovationsbeschleuniger beschrieben. Das stimmt teilweise. Ohne große Cloud-Partner hätten manche Modellanbieter ihren Durchbruch nicht so schnell finanzieren, trainieren oder ausrollen können. Partnerschaften können Zugang schaffen.

Aber dieselbe Partnerschaft kann auch Abhängigkeit schaffen. Wer Compute, Kapital, Distribution und Kundenzugang von einem dominierenden Partner erhält, gewinnt Geschwindigkeit - und verliert möglicherweise Spielraum. Die Brücke wird zur Leine, wenn der Partner nicht nur Infrastruktur liefert, sondern die Richtung mitbestimmt.

Genau darauf verweisen Wettbewerbsbehörden. Die OECD nennt Zugang zu hochwertigen Daten und Rechenleistung als zentrale Inputs und warnt vor Risiken durch Verflechtungen entlang der generativen KI-Wertschöpfungskette. Die britische CMA beschreibt die wachsende Rolle weniger großer Technologieunternehmen in der Entwicklung von Foundation Models und in den Zugangswegen zum Markt. Die Europäische Kommission sieht in bestimmten Partnerschaften sowohl pro-kompetitive Chancen als auch Risiken von Abhängigkeit und Konzentration kritischer Inputs.

Das Buch sollte hier nicht moralisch argumentieren. Es geht nicht darum, Partnerschaften pauschal zu verdammen. Es geht darum, sie architektonisch ernst zu nehmen. Jede Partnerschaft beantwortet drei Fragen: Wer bekommt Zugang? Wer setzt die Standards? Wer kann später wechseln?

Prüffragen für KI-Partnerschaften

1. Ist der Zugang zu Compute mit exklusiven Bedingungen verbunden?
2. Kann das Modell später auf andere Infrastruktur wechseln?
3. Wer kontrolliert Kundenzugang, Preissetzung, Telemetrie und Produktintegration?
4. Werden offene Schnittstellen gestärkt - oder wird nur ein anderer Lock-in gebaut?

11.6 Strom, Wasser, Grundstücke: die lokale Seite der Macht

Ein Datacenter steht immer irgendwo. Das klingt banal, ist aber politisch entscheidend. Es braucht Netzanschluss, Kühlung, Fläche, Genehmigung, Wasser oder alternative Kühltechnik, Notstrom, Glasfaser, Sicherheit und langfristige Energieverträge. Es ist nicht einfach eine Cloud. Es ist ein Nachbar.

Die IEA ordnet den globalen Anteil der Datenzentren am Stromverbrauch auch 2030 im Basisszenario unter drei Prozent ein. Das klingt zuerst überschaubar. Doch die lokale Wirkung kann viel größer sein, weil Datacenter nicht gleichmäßig über Länder verteilt sind. Sie konzentrieren sich dort, wo Strom, Netzkapazität, Fläche, Steuerregime, Glasfaser und politische Genehmigung zusammenpassen.

Dadurch entstehen neue lokale Konflikte. Eine Kommune kann Steuereinnahmen, Arbeitsplätze und digitale Infrastruktur gewinnen. Sie kann aber auch Netzkapazität, Wasser, Landschaft, Abwärme und politische Aufmerksamkeit binden. Ein Energieversorger kann einen großen Kunden bekommen. Er kann aber auch an Planungsgrenzen geraten. Ein Staat kann KI-Souveränität versprechen. Er muss dafür aber reale Transformatoren bauen lassen.

KI-Macht ist deshalb nicht nur ein Thema für Kartellrecht. Sie ist auch ein Thema für Stadtplanung, Energiepolitik, Wasserrecht, Grundstückspolitik, Netzausbau und regionale Gerechtigkeit.

11.7 Defaults sind politische Objekte

Die sichtbarste Macht entsteht oft nicht im Rechenzentrum, sondern auf dem Bildschirm. Wer den Default kontrolliert, kontrolliert den ersten Griff des Nutzers.

KI wird nicht nur über separate Chatfenster genutzt werden. Sie wandert in Betriebssysteme, Office-Suiten, Browser, Suchmaschinen, Code-Editoren, Smartphones, CRM-Systeme, Buchhaltung, Supporttools und Collaboration-Plattformen. Dort entscheidet nicht immer das beste Modell. Oft entscheidet der voreingestellte Assistent, die vorhandene Integration, der bereits genehmigte Anbieter oder die Schaltfläche, die genau neben dem Arbeitsfluss sitzt.

Diese Interface-Macht ist schwerer zu sehen als GPU-Macht. Aber sie kann ähnlich stark sein. Ein Modell kann technisch überlegen sein und trotzdem weniger genutzt werden, wenn es nicht im Standardworkflow auftaucht. Ein Anbieter kann modellseitig nur mittelgut sein und trotzdem dominieren, wenn sein Assistent in Millionen bestehender Arbeitsplätze eingebettet ist.

Damit wird die Benutzeroberfläche zur Infrastruktur. Nicht als Kabel oder Transformator, sondern als Gewohnheit. Und Gewohnheiten sind die weichste Form harter Macht.

11.8 Das Open-Source-Paradox

Offene Modelle sind wichtig. Sie schaffen Transparenz, Lernmöglichkeiten, lokale Experimente, Forschung, Anpassbarkeit und kulturelle Vielfalt. Ohne offene Gewichte und offene Tooling-Ökosysteme würde die Machtkonzentration in KI deutlich schneller und stiller voranschreiten.

Aber offene Modelle lösen nicht automatisch das Infrastrukturproblem. Ein Modellgewicht auf einer Plattform ist noch keine souveräne KI. Man braucht Hardware, Speicher, Energie, Sicherheitsprozesse, Evaluierung, Monitoring, Updatewege, Datenpipelines, Governance, gute Nutzeroberflächen und Menschen, die das System betreiben. Wer nur die Gewichte besitzt, aber nicht die Betriebsfähigkeit, besitzt eher ein Versprechen als eine Infrastruktur.

Das ist das Open-Source-Paradox: Offenheit auf der Modellebene kann echte Dezentralisierung ermöglichen. Aber sie ersetzt nicht die anderen Schichten des Stacks. Ein 70B-Modell lokal laufen zu lassen ist leichter als früher, aber nicht dasselbe wie ein verlässliches, auditierbares, datenschutzfähiges, kosteneffizientes Unternehmenssystem zu betreiben.

Episodische Intelligenz nimmt offene Modelle deshalb ernst, aber nicht romantisch. Offene Modelle sind Rohmaterial. Souveränität entsteht erst, wenn dieses Rohmaterial in lokale Runtime, semantische Schichten, Contracts, Evaluation und Energieplanung eingebettet wird.

Nicht verwechseln
Offene Gewichte sind ein Freiheitsgrad.
Lokale Betriebsfähigkeit ist Souveränität.
Beides zusammen ist stark. Eines allein reicht selten.

11.9 Daten, Telemetrie und die zweite Lernkurve

Compute ist die sichtbare Knappheit. Daten und Feedback sind die leisere Knappheit. Ein Anbieter, der täglich Millionen oder Milliarden Interaktionen sieht, lernt nicht nur aus Trainingsdaten. Er lernt aus Nutzung: Welche Fragen stellen Menschen? Wo brechen sie ab? Welche Tools werden akzeptiert? Welche Antworten werden kopiert, korrigiert, verworfen oder weiterverarbeitet?

Diese Telemetrie muss nicht immer aus Rohdaten bestehen. Auch aggregierte Signale können wertvoll sein. Der Vorteil entsteht aus Wiederholung. Viele kleine Nutzungsereignisse verdichten sich zu Produktwissen: bessere Defaults, bessere Sicherheitsfilter, bessere Routingmodelle, bessere Caches, bessere Tool-Auswahl, bessere Benchmarks. Das Modell wird nicht nur durch Training besser. Das Produkt wird durch Betrieb besser.

Damit entsteht eine zweite Lernkurve neben der Modellkurve. Die erste Kurve fragt: Wer kann das größte oder beste Modell trainieren? Die zweite fragt: Wer hat genug echte Nutzung, um das System um das Modell herum schneller zu verbessern als andere?

Für episodische Intelligenz ist das ein Warnsignal. Wer seine lokale Semantik, seine Evals und seine Betriebsdaten vollständig an einen Anbieter abgibt, liefert nicht nur Nachfrage. Er liefert Rohmaterial für dessen nächste Produktverbesserung. Das kann in Ordnung sein, wenn es bewusst geschieht. Gefährlich wird es, wenn Organisationen gar nicht merken, dass ihre Prozessdaten zur Lernkurve eines fremden Systems werden.

11.10 Standards, Benchmarks und Beschaffung

Macht entsteht auch durch die Frage, was als normal gilt. Sobald KI in Beschaffung, Zertifizierung, Audits und interne Richtlinien wandert, werden Standards zu Markttüren. Das kann gut sein: Ohne gemeinsame Begriffe für Risiko, Dokumentation, Transparenz und Governance wird KI in Unternehmen schnell chaotisch.

Aber Standards können auch die Akteure bevorzugen, die große Compliance-Abteilungen, Rechtsabteilungen, Zertifizierungsbudgets und Lobbykapazitäten haben. Kleine Anbieter scheitern dann nicht am Modell, sondern am Formular. Die Regulierung, die eigentlich Sicherheit schaffen soll, kann unbeabsichtigt die größten Akteure stabilisieren.

Das bedeutet nicht, dass Regulierung falsch ist. Im Gegenteil. Ohne belastbare Regeln wird die KI-Industrie noch stärker von Vertrauen durch Markenmacht abhängen. Aber Regulierung muss architekturbewusst sein. Sie sollte Portabilität, Interoperabilität, Auditierbarkeit und Wechselbarkeit fördern - nicht nur Dokumente produzieren, die am Ende nur die größten Anbieter effizient ausfüllen können.

Die EU-Regeln für General-Purpose-AI-Modelle zeigen, dass die Modellschicht selbst regulatorisch sichtbar wird: Dokumentation, Trainingsdatenzusammenfassungen, Risikoabschätzung, Incident Reporting, Cybersecurity und besondere Anforderungen für Modelle mit systemischem Risiko. NIST rahmt KI-Risikomanagement mit Govern, Map, Measure und Manage. Beides zeigt: Governance wandert aus der Ethikbroschüre in die Betriebsarchitektur.

Architekturbewusste Regulierung
Gute KI-Regulierung fragt nicht nur: Ist ein Modell gefährlich?
Sie fragt auch: Kann ein Unternehmen das Modell wechseln? Kann es Entscheidungen auditieren? Kann es lokale Daten schützen? Kann es Risiken im Betrieb messen?

11.11 Episodische Intelligenz als Verhandlungsmacht

Episodische Intelligenz ist keine naive Gegenmacht. Ein lokales semantisches System wird nicht über Nacht die Kapitalmacht der großen Anbieter neutralisieren. Aber es verändert die Verhandlungsposition.

Wer seine Daten, sein Gedächtnis, seine Retrieval-Strukturen, seine Kontextlogik und seine Governance lokal hält, ist weniger ausgeliefert. Er kann Anbieter vergleichen. Er kann Modelle wechseln. Er kann abstrakte Requests senden. Er kann Teile lokal lösen. Er kann Fehler isolieren. Er kann Preissprünge abfedern. Er kann Datenschutz ernst nehmen, ohne vollständig auf Frontier-Reasoning zu verzichten.

Das ist der stille strategische Wert lokaler Systeme. Sie müssen nicht alles besser können. Sie müssen genug können, um nicht jede Entscheidung an den teuersten und mächtigsten Punkt des Stacks abzugeben.

In Unternehmen bedeutet das: Der Vendor liefert Reasoning, nicht Identität. Er liefert ein Modell, nicht das gesamte Gedächtnis. Er liefert Eskalation, nicht die dauerhafte Hoheit über den Prozess. Genau dort entsteht neue Verhandlungsmacht.

11.12 Souveränitätstheater

Der Begriff Souveränität wird in der KI-Debatte schnell groß und weich. Jeder will souverän sein. Staaten, Unternehmen, Clouds, Anbieter, Regionen, Branchen. Aber Souveränität ist kein Etikett. Sie ist eine technische Eigenschaft.

Eine Cloud kann regional stehen und trotzdem abhängig bleiben, wenn die zentralen Chips, Modelle, Updates, APIs, Laufzeiten und Betriebskonzepte aus wenigen externen Quellen kommen. Ein lokales Modell kann souverän klingen und trotzdem praktisch unwartbar sein. Ein Unternehmen kann Datenresidenz erfüllen und trotzdem in einem proprietären Workflow gefangen sein, aus dem es nicht ohne Schmerzen herauskommt.

Souveränitätstheater beginnt dort, wo Architektur durch Marketing ersetzt wird. Deshalb braucht Episodische Intelligenz messbare Kriterien.

- Portabilität: Kann der Anbieter gewechselt werden, ohne das System neu zu erfinden?
- Datenhoheit: Bleiben Rohdaten, Memory und interne Semantik unter eigener Kontrolle?
- Auditierbarkeit: Sind Requests, Outputs, Tools, Entscheidungen und Gates nachvollziehbar?
- Fallback-Fähigkeit: Gibt es lokale oder alternative Betriebsmodi, wenn ein Cloud-Dienst ausfällt oder zu teuer wird?
- Schnittstellenklarheit: Sind Modell, Runtime, Daten, Policies und Nutzeroberflächen sauber getrennt?
- Energiebewusstsein: Kann das System Arbeit zeitlich oder räumlich verschieben, statt immer sofort teuer zu rechnen?

Das sind keine perfekten Garantien. Aber sie sind besser als die Behauptung: Unsere KI ist souverän, weil das Wort in der Produktbroschüre steht.

11.13 Die politische Bedeutung kleiner Architekturentscheidungen

Die großen politischen Fragen der KI erscheinen oft riesig: Wer reguliert Foundation Models? Wer kontrolliert Desinformation? Wer trägt Verantwortung bei Schäden? Wer besitzt die Daten? Diese Fragen sind wichtig. Aber Machtkonzentration entsteht auch durch kleine Architekturentscheidungen.

Wird das Firmenwissen direkt in einen Cloud-Assistenten geladen oder lokal in einen Vektorstore organisiert? Wird jede Nutzerfrage an ein Frontier-Modell geschickt oder zuerst geroutet? Werden Prompts frei formuliert oder über Contracts strukturiert? Werden Tool-Aufrufe vom Modell selbst entschieden oder durch ein Gate geprüft? Wird eine Knowledge Base portabel aufgebaut oder in einem proprietären Agentenprodukt eingeschlossen?

Jede dieser Entscheidungen klingt klein. Zusammen bestimmen sie, ob eine Organisation später wechseln kann oder nicht. Ob sie ihre eigene Semantik versteht oder nur Konsument einer fremden Semantik ist. Ob sie einen Anbieter nutzt oder in ihm lebt.

Das ist vielleicht die wichtigste politische Einsicht episodischer Intelligenz: Souveränität entsteht nicht erst im Parlament. Sie entsteht auch in Datenmodellen, Schnittstellen, Logs, Caches, Evals und Routings.

11.14 Nicht gegen Größe, sondern gegen Default

Dieses Kapitel ist kein Plädoyer gegen große Rechenzentren. Kapitel 10 hat gezeigt, warum sie bleiben. Es ist auch kein romantisches Plädoyer für vollständige Dezentralisierung. Die Welt ist komplizierter, und gute Infrastruktur ist schwer.

Das Argument ist schmaler und stärker: Wenn große Infrastruktur notwendig ist, darf sie nicht automatisch zum Default für jede semantische Entscheidung werden. Und wenn mächtige Anbieter unvermeidlich eine Rolle spielen, muss die Architektur darum herum so gebaut werden, dass Abhängigkeit sichtbar, begrenzt und verhandelbar bleibt.

Episodische Intelligenz antwortet darauf nicht mit Revolte, sondern mit Schichtung. Lokal, wo Kontext, Datenschutz, Gedächtnis, Routing und Energieplanung besser aufgehoben sind. Regional oder schwarmartig,

wo Arbeit teilbar, prüfbar und nicht zeitkritisch ist. Frontier, wo echte Tiefe, Generalisierung und Hochenergie-Reasoning gebraucht werden.

Machtkonzentration verschwindet dadurch nicht. Aber sie verliert ihren unsichtbaren Charakter. Sie wird eine bewusste Schnittstelle. Und Schnittstellen kann man gestalten.

Der Hochofen bleibt. Aber wer seine eigene Werkstatt besitzt, muss nicht jedes Werkzeug dort schmieden lassen.

Kernaussagen

- KI-Macht entsteht nicht nur durch Modelle, sondern durch den gesamten Stack aus Chips, Cloud, Energie, Daten, Runtime, Interfaces und Regeln.
- Kapital wird in Frontier-KI zu einer strategischen Zugangskarte, weil Infrastruktur lange vor der Monetarisierung aufgebaut werden muss.
- Cloud-Plattformen sind nicht nur Rechenorte, sondern Distributions-, Beschaffungs- und Governance-Maschinen.
- Partnerschaften können Innovation beschleunigen, aber auch Abhängigkeiten entlang der Wertschöpfungskette erzeugen.
- Open Source ist ein Freiheitsgrad, aber ohne lokale Betriebsfähigkeit noch keine Souveränität.
- Episodische Intelligenz schafft keine perfekte Unabhängigkeit. Es schafft Verhandlungsmacht durch Schichtung, Portabilität, lokale Semantik und bewusste Eskalation.

Übergang: Machtkonzentration beschreibt das Problem. Souveränität beschreibt die praktische Gegenkraft.

Kapitel 12 - Wem gehört das Gedächtnis?

Daten, Semantik, Runtime und Exit-Optionen

Souverän ist nicht, wer alles selbst baut. Souverän ist, wer wechseln kann, ohne sein Gedächtnis, seine Semantik und seine Handlungsfähigkeit zu verlieren.

12.1 Souveränität ist kein Ort

Digitale Souveränität wird häufig so erzählt, als wäre sie eine Frage des Standorts. Liegt der Server in Europa? Steht das Datacenter im eigenen Land? Wird die Rechnung von einem lokalen Anbieter gestellt? Diese Fragen sind wichtig. Aber sie sind nicht ausreichend.

Ein System kann geografisch lokal sein und trotzdem unsouverän. Es kann auf heimischer Erde laufen und dennoch von fremden Updates, geschlossenen Schnittstellen, proprietären Datenformaten, unklaren Abhängigkeiten und nicht prüfbaren Modellen abhängen. Umgekehrt kann ein System Komponenten aus verschiedenen Ländern nutzen und trotzdem in entscheidenden Punkten kontrollierbar sein, wenn seine Daten, Schnittstellen, Logs, Policies, Wechselpfade und semantischen Schichten sauber gestaltet sind.

Souveränität ist deshalb nicht zuerst eine Adresse. Sie ist eine Fähigkeit. Die Fähigkeit, eine digitale Maschine zu verstehen, zu begrenzen, zu prüfen, zu verlagern und notfalls zu verlassen.

Für KI wird diese Frage schärfer als bei klassischer Cloud-Software. Denn KI verarbeitet nicht nur Dateien. Sie verarbeitet Bedeutung. Sie interpretiert Texte, entscheidet über Relevanz, baut Kontext, erzeugt Zusammenfassungen, setzt Prioritäten und schlägt Handlungen vor. Wer diese semantische Schicht vollständig auslagert, lagert nicht nur Rechenleistung aus. Er lagert einen Teil seiner Urteilskraft aus.

12.2 Das Gedächtnis ist der neue Besitzstand

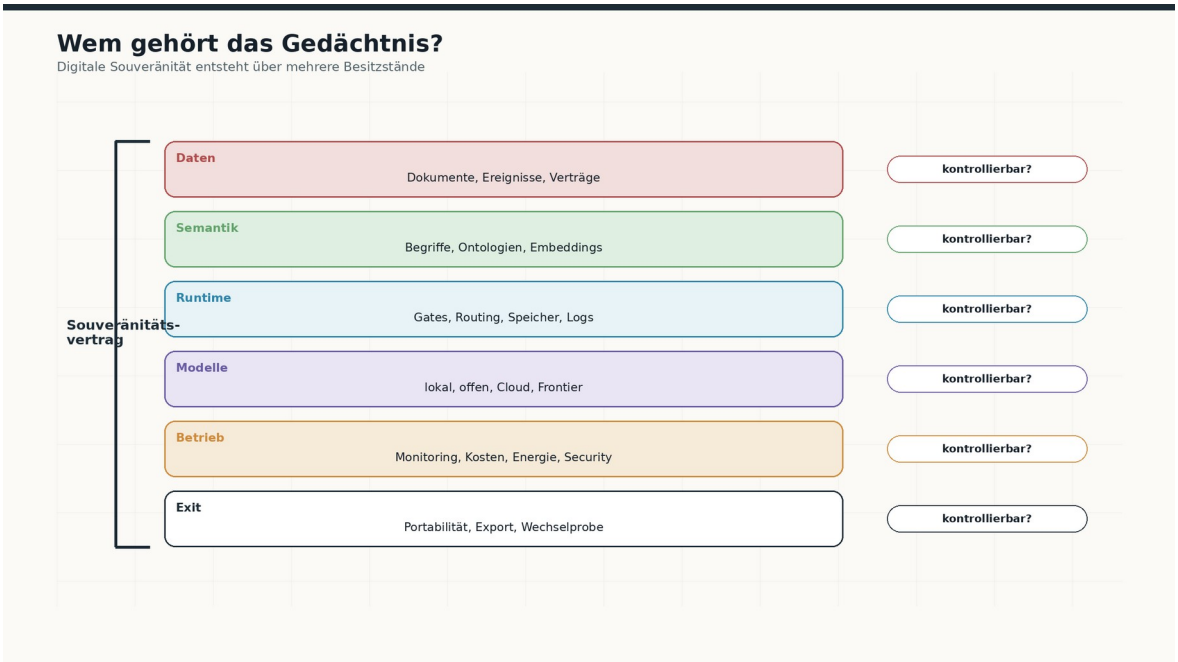

Abb. 12 - Wem gehört das Gedächtnis? Souveränität über Daten, Semantik, Runtime, Modelle, Betrieb und Exit.

In vielen Unternehmen liegt der eigentliche Wert nicht im Modell, sondern im Gedächtnis des Systems: Dokumente, Projekte, Kundenhistorie, interne Begriffe, getroffene Entscheidungen, Ausnahmen, Risiken, Rollen, Vertragslogik, alte Fehler, gelöste Fälle. Dieses Gedächtnis entsteht langsam. Es ist nicht einfach ein Datensatz. Es ist die sedimentierte Erfahrung einer Organisation.

Klassische IT konnte dieses Gedächtnis oft noch relativ klar verorten: Datenbank, Dateiserver, CRM, ERP. Bei KI-Systemen verschwimmt das. Gedächtnis kann in Vektorstores liegen, in Chatverläufen, in Systemprompts, in Tool-Konfigurationen, in Modelladaptern, in Evals, in Agentenstates, in Nutzerprofilen, in Embeddings, in Feedback-Loops und in unscheinbaren Routing-Regeln.

Genau hier entsteht die neue Abhängigkeit. Nicht: Wer hostet unsere Daten? Sondern: Wer besitzt die Bedeutungsstruktur, mit der unsere Daten überhaupt nutzbar werden?

Ein Unternehmen kann seine Dateien exportieren und trotzdem sein KI-Gedächtnis verlieren. Wenn die lokalen Begriffe, Klassifikationen, Embedding-Indizes, Entscheidungsregeln und Evaluierungen an eine Plattform gebunden sind, ist der Exit nur auf dem Papier einfach. Man hat dann zwar Daten. Aber man verliert die Grammatik, mit der das System sie verstanden hat.

12.3 Sechs Dimensionen von Souveränität

Für episodische Intelligenz reicht es nicht, Datensouveränität eng zu verstehen. Das Buch schlägt eine breitere Arbeitsdefinition vor. Souveränität besteht aus mindestens sechs Dimensionen, die zusammen betrachtet werden müssen.

Dimension	Leitfrage
Datensouveränität	Welche Rohdaten existieren? Wo liegen sie? Wer darf sie sehen? Was darf extern verarbeitet werden?
Semantische Souveränität	Wer definiert Bedeutung, Kategorien, Ontologien, Risiko-Klassen, interne Begriffe und Bewertungslogik?
Runtime-Souveränität	Wer entscheidet den Ablauf: Routing, Tool-Nutzung, Eskalation, Abbruch, Human Gate, Audit Log?
Modell- und Adapter-Souveränität	Welche Modelle, LoRAs, Prompts, Evaluierungen und Policies sind austauschbar, prüfbar und lokal betreibbar?
Energie- und Betriebssouveränität	Wann wird gerechnet? Mit welchem Energieprofil? Was ist

	sofort nötig und was darf in ein günstigeres Zeitfenster?
Exit-Souveränität	Wie schnell kann ein Unternehmen Anbieter, Modell, Cloud, Vektorstore oder Tool-Kette wechseln, ohne sein Gedächtnis zu verlieren?

12.4 Datenportabilität reicht nicht

Die europäische Regulierung bewegt sich erkennbar in Richtung Zugang, Portabilität und Wechselmöglichkeit. Der EU Data Act ist dafür ein wichtiger Marker: Er ist seit 12. September 2025 anwendbar und soll Nutzern mehr Kontrolle über Daten aus vernetzten Geräten geben; er adressiert außerdem den Wechsel zwischen Cloud-Anbietern. Das ist politisch und wirtschaftlich bedeutsam. Aber KI-Souveränität geht darüber hinaus.

Denn Portabilität von Rohdaten beantwortet noch nicht die Frage, ob semantische Arbeit portierbar ist. Ein Export von PDF-Dateien hilft wenig, wenn der Wert des Systems in Vektoren, Labels, Ontologien, Zusammenfassungen, Tool-Gates und historischen Agentenentscheidungen liegt.

Der Satz „wir können unsere Daten jederzeit exportieren“ klingt beruhigend. In KI-Systemen muss man härter fragen: Können wir auch unsere Embeddings, unsere Annotationen, unsere Evals, unsere Prompts, unsere Routinglogik, unsere Policies, unsere Audit Trails und unsere lokalen Bedeutungsmodelle exportieren? Können wir sie in einer anderen Runtime wieder verwenden? Können wir nachweisen, dass ein neues Modell auf derselben semantischen Grundlage arbeitet?

Datenportabilität ist der Anfang. Semantische Portabilität ist der schwierigere Teil.

12.5 Der lokale semantische Layer als Souveränitätsanker

Kapitel 5 hat semantische Verdichtung als Privacy- und Effizienzprinzip beschrieben. In diesem Kapitel wird derselbe Layer zur Souveränitätsfrage. Wenn der lokale semantische Layer die Rohdaten kennt, die Business-Begriffe pflegt, interne Risiken klassifiziert und nur abstrahierte Aufgaben nach außen gibt, dann bleibt die eigentliche Bedeutungshoheit näher beim Unternehmen.

Das ist kein Plädoyer für totale Abschottung. Es ist ein Plädoyer für eine saubere Trennung der Rollen. Die lokale Runtime besitzt den Kontext. Die Frontier erhält Aufgaben. Die lokale Runtime kennt Namen, Verträge, Kundenspezifika, historische Verläufe, interne Ausnahmen und echte IDs. Das Frontier-Modell bekommt Muster, Struktur, Problemklasse, Constraints und gewünschte Denkform.

In einem souveränen System episodischer Intelligenz wird die Cloud nicht gefragt: „Was weißt du über unser Geschäft?“ Sie wird gefragt: „Welche allgemeinen Muster, Strategien oder Gegenargumente siehst du in dieser abstrahierten Lage?“

Damit verändert sich das Machtverhältnis. Das Modell bleibt stark. Aber es wird nicht zum stillen Besitzer des Unternehmensgedächtnisses.

12.6 Runtime-Souveränität: Wer entscheidet den Ablauf?

Viele Unternehmen sprechen über Datenkontrolle, aber kaum über Ablaufkontrolle. Dabei ist gerade die Runtime entscheidend. Sie entscheidet, welcher Kontext geladen wird, welches Tool aufgerufen wird, welches Modell zuständig ist, ob ein Ergebnis akzeptiert wird, ob ein Mensch gefragt wird und welche Logs erzeugt werden.

Wenn diese Ablaufentscheidung vollständig in einer geschlossenen Plattform liegt, entsteht eine subtile Abhängigkeit. Das Unternehmen nutzt dann nicht nur ein Modell. Es übernimmt eine fremde Logik darüber, wann dieses Modell welche Welt sehen darf.

Episodische Intelligenz fordert deshalb: Routing, Gates, Contracts, Audit Logs und Eskalationslogik müssen als eigene Betriebsschicht gedacht werden. Nicht jeder Teil muss selbst entwickelt sein. Aber die Übergänge müssen beschreibbar, prüfbar und austauschbar bleiben.

Der Unterschied ist klein und groß zugleich. In einem unsouveränen System sagt die Plattform: „Ich entscheide, was für dich relevant ist.“ In einem souveränen System sagt die lokale Runtime: „Ich entscheide, was du sehen darfst.“

12.7 Souveräne Cloud ist mehr als europäische Cloud

Souveräne Cloud wird oft verkürzt auf die Frage: europäischer Anbieter oder nicht? Auch das ist zu simpel. Die Europäische Kommission hat mit ihrem Cloud Sovereignty Framework versucht, Souveränität in objektive, messbare Beschaffungskriterien zu übersetzen: strategische, rechtliche, operative und ökologische Aspekte, Lieferkettentransparenz, technologische Offenheit, Sicherheit und Compliance mit EU-Recht. Genau diese Mehrdimensionalität ist wichtig.

Souveränität kann also nicht allein am Logo des Providers hängen. Sie muss nachweisbar sein: Wer hat operative Kontrolle? Welche Rechtsordnungen greifen? Welche Lieferkettenabhängigkeiten bestehen? Welche technischen Maßnahmen verhindern Zugriff? Wie offen sind Schnittstellen? Wie gut ist der Exit? Wie auditierbar ist der Betrieb?

Das ist unbequem, aber produktiv. Es verhindert Souveränitätstheater. Ein lokaler Anbieter ohne Transparenz, schwache Sicherheit und proprietäre Sackgassen ist nicht automatisch souveräner als ein ausländischer Anbieter mit strenger Isolation, offener Portabilität, klaren Verträgen und prüfbarer Governance. Herkunft zählt. Aber Architektur zählt auch.

12.8 Open Source ist notwendig, aber nicht ausreichend

Open Source ist ein starker Souveränitätsbaustein. Offene Modelle, offene Werkzeuge, offene Schnittstellen und offene Laufzeiten reduzieren Abhängigkeit, fördern Prüfung und ermöglichen lokale Anpassung. Europäische Stellen und die Linux Foundation betonen inzwischen ausdrücklich die Rolle von Open Source für digitale Souveränität.

Aber auch hier lauert ein Missverständnis. Ein offenes Modell macht noch kein souveränes System. Wer ein offenes Modell nutzt, aber die gesamte Runtime, das Deployment, die Evaluation, die Datenpipeline und den Vektorstore an eine geschlossene Plattform bindet, hat nur eine offene Komponente in einer geschlossenen Maschine.

Souveränität entsteht nicht durch ein einzelnes offenes Artefakt. Sie entsteht durch eine offene Betriebsfähigkeit. Kann ich das Modell betreiben? Kann ich es prüfen? Kann ich es ersetzen? Kann ich meine Datenstruktur behalten? Kann ich meinen semantischen Layer umziehen? Kann ich Fehler messen? Kann ich die Energie- und Kostenlogik beeinflussen?

Open Source ist der Samen. Souveränität ist der Garten: Boden, Wasser, Werkzeuge, Pflege, Zäune und Wege hinaus.

12.9 Compliance ist nicht dasselbe wie Handlungsfähigkeit

Regulierung ist wichtig. GDPR, AI Act, Data Act, NIS2, DORA, Cloud-Zertifizierungen und Beschaffungskriterien setzen Rahmen. Sie schaffen Mindestanforderungen und erzwingen Dokumentation, Transparenz und Risikomanagement. Aber Compliance allein macht ein Unternehmen noch nicht souverän.

Ein System kann formal compliant sein und operativ abhängig bleiben. Es kann alle geforderten Dokumente besitzen und trotzdem nicht schnell wechseln können. Es kann Datenschutzprozesse haben und trotzdem semantische Abhängigkeiten in einer Plattform aufbauen. Es kann auditierbar wirken und trotzdem nicht nachvollziehbar erklären, warum ein Agent ein bestimmtes Tool aufgerufen hat.

Operative Souveränität beginnt dort, wo ein Unternehmen nicht nur nachweisen kann, dass es Regeln einhält, sondern auch tatsächlich steuern kann: stoppen, zurückrollen, umleiten, abstrahieren, lokal verarbeiten, eskalieren, wechseln.

12.10 Der praktische Souveränitäts-Check

Ein Unternehmen muss nicht sofort eine perfekte Architektur episodischer Intelligenz besitzen. Aber es kann anfangen, die richtigen Fragen zu stellen.

Erste Frage: Wo liegt unser KI-Gedächtnis? Nicht nur Dokumente, sondern Embeddings, Chatverläufe, Agentenstates, Prompts, Evaluierungen, Feedbackdaten und Tool-Konfigurationen.

Zweite Frage: Welche Schichten können wir wechseln? Modell, Provider, Vektorstore, Orchestrator, Authentifizierung, Logging, Deployment, Energieplanung?

Dritte Frage: Welche Teile unserer Bedeutung sind lokal? Haben wir eigene Ontologien, Labels, Risiko-Klassen, Kundenbegriffe, Branchenlogik und Datenverträge?

Vierte Frage: Welche externen Systeme sehen Rohkontext? Und warum? Gibt es eine abstrahierte Alternative?

Fünfte Frage: Was passiert, wenn ein Anbieter morgen Preise, Nutzungsbedingungen, Modellverhalten, Region, API oder Policy ändert?

Sechste Frage: Können wir beweisen, dass ein bestimmter KI-Ablauf korrekt eskaliert, geprüft und geloggt wurde?

Diese Fragen sind nicht bürokratisch. Sie sind Überlebensfragen für Organisationen, die KI nicht nur ausprobieren, sondern dauerhaft betreiben wollen.

12.11 Souveränität als Wechseloption

Die härteste Definition von Souveränität lautet: Wer nicht wechseln kann, besitzt nicht wirklich. Er nutzt.

Das gilt für Cloud. Es gilt für Modelle. Es gilt für Datenbanken. Und es gilt noch stärker für semantische KI-Systeme. Denn hier ist der Wechsel nicht nur technisch. Er ist kognitiv. Ein Unternehmen muss sein digitales Gedächtnis, seine internen Begriffe und seine Entscheidungslogik mitnehmen können.

Episodische Intelligenz ist deshalb keine nostalgische Forderung nach „alles lokal“. Es ist die Forderung nach Architektur mit Ausgängen. Lokale Systeme, semantische Kompression, deterministische Governance, offene Schnittstellen, auditierbare Logs und energieadaptive Verarbeitung sind keine Selbstzwecke. Sie sind Mittel gegen kognitive Gefangenschaft.

Souveränität bedeutet nicht, allein zu sein. Sie bedeutet, nicht ausgeliefert zu sein.

Arbeitsdefinition

Digitale Souveränität in Systemen episodischer Intelligenz bedeutet: Daten, Semantik, Runtime, Modelle, Betrieb und Exit so zu gestalten, dass eine Organisation ihre KI-Fähigkeit prüfen, begrenzen, verlagern und notfalls neu zusammensetzen kann.

12.12 Beschaffung wird Architektur

In der alten IT-Welt konnte Beschaffung manchmal wie ein administrativer Prozess wirken: Anbieter vergleichen, Preis prüfen, Vertrag unterschreiben, System einführen. Bei KI reicht das nicht mehr. Beschaffung wird Architektur.

Der Grund ist einfach: Wer einen KI-Dienst einkauft, kauft nicht nur Rechenleistung. Er kauft einen Teil der späteren Denkwege ein. Welche Daten werden hochgeladen? Welche Logs entstehen? Welche Modellversion entscheidet? Welche Tool-Schnittstellen sind verfügbar? Welche Region wird genutzt? Welche Sicherheitsannahmen gelten? Welche Preislogik bestimmt später das Verhalten der Nutzer?

Wenn ein Unternehmen KI beschafft, ohne die spätere Souveränität mitzudenken, baut es oft unbemerkt Abhängigkeit ein. Der Vertrag ist dann nicht nur ein Preisblatt. Er ist ein Architekturdiagramm mit juristischen Mitteln.

Ein guter KI-Beschaffungsprozess muss deshalb technische Fragen stellen: Gibt es exportierbare Logs? Sind Embeddings portierbar oder reproduzierbar? Sind Systemprompts dokumentierbar? Kann die Runtime eigene Gates erzwingen? Sind Toolaufrufe auditierbar? Gibt es Modellwechsel ohne komplette Neugestaltung? Welche Daten werden für Training oder Qualitätsverbesserung verwendet? Welche Telemetrie ist abschaltbar?

Die beste Souveränitätsstrategie hilft wenig, wenn sie erst nach der Beschaffung beginnt. Sie muss vor der ersten Integration beginnen.

12.13 Der Souveränitätsvertrag

Für Episodische Intelligenz braucht jede externe KI-Schicht einen Souveränitätsvertrag. Das muss nicht zwingend ein juristisches Dokument sein. Es kann auch ein technischer Contract in der Runtime sein. Wichtig ist, dass der Übergang explizit wird.

Ein solcher Vertrag beschreibt mindestens: Zweck der Verarbeitung, erlaubter Kontext, verbotene Datenklassen, zulässige Modelle, gewünschtes Rückgabeformat, Logging-Regeln, Speicherdauer, Prüfverfahren, Fehlerrückgabe, Eskalationspfad und Exit-Anforderung.

Das klingt nach Bürokratie, ist aber in Wirklichkeit Entlastung. Denn ein System, das seine Übergänge nicht beschreibt, improvisiert. Und improvisierende KI-Systeme werden in produktiven Umgebungen schnell teuer, riskant und schwer erklärbar.

Der Souveränitätsvertrag macht sichtbar, was sonst stillschweigend passiert. Er sagt: Diese Aufgabe darf die Organisation verlassen. Diese Rohdaten dürfen es nicht. Diese Antwort darf nur als Vorschlag zurückkehren. Dieses Tool darf nur mit Freigabe aufgerufen werden. Dieser Lauf muss geloggt werden. Diese Modellversion darf in diesem Risikobereich nicht verwendet werden.

Damit wird Souveränität ausführbar. Nicht als Haltung. Als Übergangsbedingung.

12.14 KMU-Souveränität: klein anfangen, aber richtig

Gerade kleinere Unternehmen können nicht alles selbst bauen. Sie haben keine eigenen Forschungsabteilungen, keine riesigen GPU-Cluster und oft auch keine dedizierten Governance-Teams. Daraus folgt aber nicht, dass Souveränität für sie unerreichbar ist. Im Gegenteil: Sie brauchen sie besonders, weil sie Anbieterwechsel, Fehlentscheidungen und Lock-in oft schlechter abfedern können als Konzerne.

Für KMU beginnt Souveränität episodischer Intelligenz nicht mit eigener Frontier-Infrastruktur. Sie beginnt mit wenigen klaren Entscheidungen: Rohdaten nicht unnötig externisieren. Einen eigenen lokalen Dokument- und Vektorstore aufbauen. Interne Begriffe dokumentieren. Kritische Prompts und Tool-Flows versionieren. Ergebnisse prüfen. Externe Modelle austauschbar halten. Kosten und Tokens messen. Einen Exit-Pfad definieren, bevor er gebraucht wird.

Das ist kein Luxus. Es ist eine Form von digitaler Buchhaltung. Wer seine Rechnungen, Verträge und Kundendaten sauber verwaltet, sollte auch sein semantisches Gedächtnis sauber verwalten.

KMU müssen nicht die größten Modelle besitzen. Aber sie sollten ihre Bedeutungsstruktur besitzen.

12.15 Ein Reifegradmodell für KI-Souveränität

Reifegrad 0: Experiment. KI wird über einzelne Tools genutzt. Daten, Prompts, Verläufe und Kosten sind kaum dokumentiert. Der Wert entsteht in einzelnen Köpfen und einzelnen Accounts.

Reifegrad 1: Kontrollierter Einsatz. Es gibt Grundregeln für Datenschutz, Tool-Auswahl und Nutzerrollen. Kritische Daten dürfen nicht ungeprüft extern verarbeitet werden. Kosten werden grob gemessen.

Reifegrad 2: Lokales Gedächtnis. Dokumente, Embeddings, Kategorien, Prompts und einfache Evals werden in einer eigenen Schicht verwaltet. Externe Modelle bekommen definierte Aufgaben statt vollständiger Rohkontexte.

Reifegrad 3: Austauschbare Runtime. Modelle, Tools, Vektorstores und Cloud-Komponenten können gewechselt werden. Contracts, Logs und Evaluationen sind Teil des Betriebs. Eskalation ist messbar und begründbar.

Reifegrad 4: Strategische Souveränität. Energie, Datenschutz, semantische Portabilität, Anbieterwechsel, Auditierbarkeit und lokale Verarbeitung werden gemeinsam geplant. KI ist nicht mehr Tool-Nutzung, sondern kontrollierte Infrastruktur.

Dieses Modell ist kein Zertifikat. Es ist ein Diagnosewerkzeug. Es hilft, den nächsten sinnvollen Schritt zu finden, ohne in eine Alles-oder-nichts-Falle zu geraten.

12.16 Falsche Souveränität

Es gibt eine verführerische Form falscher Souveränität: Man ersetzt eine große Abhängigkeit durch viele kleine, ohne sie zu verstehen. Ein lokales Modell ohne Updates, ohne Evals, ohne Security und ohne klare Datenflüsse ist nicht souverän. Es ist nur allein gelassen.

Eine souveräne Architektur muss nicht nur unabhängig wirken, sondern belastbar sein. Sie braucht Wartung, Sicherheitsprozesse, Monitoring, Backups, Modelltests, klare Verantwortlichkeiten und Menschen, die wissen, was sie betreiben.

Auch das Wort „europäisch“ darf nicht als magische Lösung missverstanden werden. Europäische Anbieter können gute Bausteine sein. Aber Souveränität entsteht durch überprüfbare Kontrolle, nicht durch Etiketten. Ebenso wenig ist ein offenes Modell automatisch sicher, ein lokaler Server automatisch datenschutzfreundlich oder eine private Cloud automatisch vertrauenswürdig.

Episodische Intelligenz vermeidet diese falsche Reinheit. Es fragt nicht: Ist diese Komponente ideologisch sauber? Es fragt: Ist diese Komponente prüfbar, begrenzbar, austauschbar und passend für die Aufgabe?

12.17 Die Brücke zum Schlusskapitel

Nach Kapitel 12 lässt sich Episodische Intelligenz nicht mehr nur als technische Architektur lesen. Es ist eine Antwort auf drei gleichzeitige Knappheiten: Energie, Vertrauen und Kontrolle.

Die Frontier bleibt notwendig. Lokale Systeme bleiben begrenzt. Regulierung bleibt unvollständig. Open Source bleibt notwendig, aber allein zu schwach. Die eigentliche Frage lautet deshalb: Wie organisiert man all diese Kräfte so, dass Intelligenz entsteht, ohne Bedeutung, Energie und Macht vollständig zu zentralisieren?

Das nächste Kapitel zieht diese Linie zusammen. Nicht als Utopie. Nicht als Dystopie. Sondern als Betriebsform für eine Welt, in der KI nicht nur möglich, sondern tragfähig sein muss.

Arbeitsdefinition
Digitale Souveränität in Systemen episodischer Intelligenz bedeutet: Daten, Semantik, Runtime, Modelle, Betrieb und Exit so zu gestalten, dass eine Organisation ihre KI-Fähigkeit prüfen, begrenzen, verlagern und notfalls neu zusammensetzen kann.

Übergang: Aus Souveränität wird erst dann mehr als ein politisches Wort, wenn sie als Betriebsform gebaut wird.

Kapitel 13 - Episodische Intelligenz

Eine Betriebsform für KI, die nicht permanent brennen muss

Episodische Intelligenz ist kein Produkt, kein einzelnes Modell und kein weiteres Architekturdiagramm. Sie ist eine neue Voreinstellung: Intelligenz entsteht verteilt, kontrolliert, energieadaptiv und nur dort zentral, wo Zentralität wirklich nötig ist.

13.1 Das Ende der falschen Ein-Satz-Architektur

Die populäre Erzählung von KI war lange erstaunlich einfach: Der Nutzer fragt, das Modell denkt, die Antwort erscheint. Diese Erzählung war nützlich, weil sie den Zugang vereinfachte. Aber sie war technisch falsch. Und je produktiver KI-Systeme werden, desto gefährlicher wird diese falsche Einfachheit.

Ein reales KI-System besteht nicht nur aus einem Modell. Es besteht aus Energie, Kühlung, Hardware, Netzwerken, Speichern, Kontextfenstern, Retrieval, Routing, Policies, Tools, Logs, Zugriffskontrolle, Kostenbudgets, Modellauswahl, Evaluationen, menschlicher Aufsicht und Verträgen zwischen Schichten. Das Modell ist der sichtbarste Teil. Aber es ist nicht die ganze Maschine.

Episodische Intelligenz beginnt mit dieser Entzauberung. Nicht um KI kleiner zu machen, sondern um sie ernst zu nehmen. Wer KI ernst nimmt, darf sie nicht als magisches Fenster betrachten. Er muss sie als Infrastruktur behandeln.

13.2 Die fünf Schichten episodischer Intelligenz

Episodische Intelligenz organisiert KI nicht entlang der Frage „Cloud oder lokal?“. Diese Frage ist zu grob. Die präzisere Frage lautet: Welche Arbeit gehört auf welche Schicht?

Die erste Schicht ist die lokale semantische Runtime. Sie kennt Rohdaten, interne Begriffe, Rollen, Berechtigungen, Kundenkontext, lokale Historie und echte Businesslogik. Sie entscheidet, welcher Kontext überhaupt relevant und freigabefähig ist.

Die zweite Schicht ist die deterministische Governance. Sie besteht aus Contracts, Gates, PASS/FAIL-Regeln, Tool-Freigaben, Audit Logs, Human Gates und Runtime States. Sie macht aus „KI nutzen“ einen prüfbaren Prozess.

Die dritte Schicht ist der energieopportunistische Arbeitsraum. Dort laufen Aufgaben, die nicht sofort erledigt werden müssen: Embeddings, Indexpflege, Evals, Adaptertraining, Datenbereinigung, synthetische Tests, Zusammenfassungen, Graphaktualisierungen.

Die vierte Schicht ist der kooperative Edge-Verbund. Viele kleine, kontrollierte Knoten erledigen zerlegbare Arbeit. Nicht als romantisches @home-Netz zufälliger Wohnzimmer-PCs, sondern als auditierbare, signierte, souveräne Rechenverbünde: Firmenstandorte, regionale Cluster, Hochschulen, kommunale Infrastruktur, lokale Edge-Nodes.

Die fünfte Schicht ist die Frontier. Sie bleibt der Hochenergie-Spezialist für schwierige, neue, strategische oder hochkomplexe Aufgaben. Aber sie ist nicht mehr der Standardweg für jede Kleinigkeit. Sie wird bewusst eskaliert.

13.3 Das Diagramm episodischer Intelligenz

```
local semantic runtime
  -> deterministic governance
  -> energy-aware background work
  -> cooperative edge / regional clusters
  -> frontier escalation only when needed
  -> local validation, audit and memory update
```

13.4 Die neue Voreinstellung

Der entscheidende Unterschied liegt nicht in einer einzelnen Technologie. Er liegt im Default.

Cloud-only sagt: Alles geht zuerst nach oben. Episodische Intelligenz sagt: Erst lokal verstehen, dann entscheiden. Cloud-only sagt: Das zentrale Modell ist der Normalfall. Episodische Intelligenz sagt: Das zentrale Modell ist eine Eskalation. Cloud-only sagt: Intelligenz sitzt im Datacenter. Episodische Intelligenz sagt: Intelligenz entsteht im Zusammenspiel von lokaler Bedeutung, deterministischer Kontrolle, verteiltem Stoffwechsel und gelegentlicher Frontier-Kognition.

Diese Voreinstellung ist keine Kleinigkeit. Defaults formen Märkte. Defaults formen Kosten. Defaults formen Datenschutz. Defaults formen Macht. Wer den Default kontrolliert, kontrolliert oft die Richtung der gesamten Architektur.

13.5 Energie ist nicht der Feind. Verschwendung ist der Feind.

Dieses Buch ist keine Anti-Compute-Schrift. Es ist auch keine romantische Rückkehr zu kleinen Rechnern aus Prinzip. Große Rechenzentren bleiben notwendig. Frontier-Training bleibt notwendig. Wissenschaft, Simulation, multimodales Reasoning und neue Modellgenerationen brauchen weiterhin Hochleistungsinfrastruktur.

Aber gerade weil Compute wertvoll ist, sollte er nicht gedankenlos verwendet werden. Jedes unnötige Token ist nicht nur eine kleine Textverlängerung. Es ist Strom, Wärme, Kühlung, Netzwerklast, Kosten, Warteschlange und Fehlermöglichkeit.

Episodische Intelligenz macht deshalb eine einfache Rechnung sichtbar: Die teuerste kognitive Instanz sollte nicht die billigsten Aufgaben erledigen. Sie sollte die Aufgaben erledigen, für die ihre Generalität, Tiefe und Transferfähigkeit tatsächlich gebraucht werden.

Der Hochofen hat seinen Platz. Aber er ist kein Universalwerkzeug.

13.6 Kontext ist nicht Rohstoff, sondern Risiko

Viele KI-Architekturen behandeln Kontext als etwas, das man möglichst groß machen sollte: mehr Dokumente, längere Fenster, mehr Historie, mehr Daten. Episodische Intelligenz widerspricht nicht grundsätzlich. Mehr Kontext kann helfen. Aber Kontext ist nicht kostenlos. Er ist Kostenfaktor, Datenschutzrisiko, Halluzinationsfläche und Machtverschiebung.

Die zentrale Frage lautet deshalb nicht: Wie viel Kontext können wir dem Modell geben? Sondern: Welcher Kontext muss wirklich auf welcher Ebene sichtbar sein?

Semantic Compression ist die Antwort auf diese Frage. Der lokale Layer bewahrt Rohkontext und erzeugt daraus abstrahierte, prüfbare, zweckgebundene Aufgaben. Die Frontier sieht nicht automatisch das Geschäft. Sie sieht das Problemmuster, die Constraints und die gewünschte Denkform.

Damit wird Kontext zu einem Vertragsgegenstand. Nicht alles, was verfügbar ist, wird sichtbar. Nicht alles, was sichtbar ist, wird gespeichert. Nicht alles, was gespeichert wird, gehört der Plattform.

13.7 Kontrolle entsteht durch Übergänge

Die Formel aus Kapitel 7 gilt bis zum Schluss: Nicht der Prompt allein macht das System zuverlässig, sondern der überprüfbare Übergang.

Ein Übergang ist der Moment, in dem ein System etwas Relevantes tut: Kontext laden, Modell wechseln, Tool aufrufen, Ergebnis akzeptieren, Mensch fragen, Daten exportieren, Frontier eskalieren, lokale Verarbeitung abbrechen. Genau dort passieren Fehler. Genau dort entstehen Kosten. Genau dort entstehen Risiken. Und genau dort muss Governance ausführbar werden.

Ein Governance-Layer, der nur als Dokument existiert, ist zu schwach. Episodische Intelligenz braucht Governance als Runtime: Contracts, Schemas, Gates, Evals, Logs, Timeouts, Freigaben, Rollbacks. Nicht als Misstrauen gegenüber dem Modell, sondern als Respekt vor der Realität produktiver Systeme.

13.8 Das Ameisenmodell als Stoffwechsel

Viele kleine Rechner sind nicht deshalb interessant, weil sie stärker sind als ein Frontier-Cluster. Sie sind interessant, weil sie anders rechnen können. Langsamer, lokaler, näher an Daten und Energie, häufiger im Hintergrund, weniger spektakulär, aber dauerhaft nützlich.

Das Ameisenmodell übernimmt Arbeit, die gut zerlegbar, prüfbar und nicht zeitkritisch ist: Indexbau, Embeddings, Datenbereinigung, Evaluationen, lokale Klassifikation, Adaptertests, semantische Pflege,

Dokumentverdichtung. Es ist nicht der Ersatz des Hochofens. Es ist der Stoffwechsel um den Hochofen herum.

Dort wird die Energiefrage konkret. Wenn eine Aufgabe nicht sofort fertig sein muss, kann sie warten: auf Solarüberschuss, niedrigere Netzlast, günstige lokale Wärmeverwertung, ein nächtliches Rechenfenster, freie Bürohardware. Manche Arbeit darf auf den passenden Takt warten.

13.9 Souveränität ist Architektur mit Ausgängen

Kapitel 12 hat gezeigt: Souveränität bedeutet nicht, alles selbst zu bauen. Souveränität bedeutet, nicht gefangen zu sein. Ein System episodischer Intelligenz muss deshalb mit Ausgängen gebaut werden.

Ausgänge heißen: exportierbare Daten, portierbare Embeddings oder zumindest reproduzierbare Embedding-Prozesse, dokumentierte Ontologien, austauschbare Modelle, offene Tool-Schnittstellen, klare Logs, getrennte Anbieterrollen, lokale Recovery-Fähigkeit und messbare Exit-Kosten.

Das klingt trocken. Aber es ist der Unterschied zwischen Nutzung und Besitz. Wer seine semantische Runtime nicht wechseln kann, besitzt sein KI-Gedächtnis nicht. Er mietet es.

13.10 Was episodische Intelligenz nicht verspricht

Episodische Intelligenz verspricht nicht, dass lokale Systeme alles können. Es verspricht nicht, dass Datacenter verschwinden. Es verspricht nicht, dass Open Source automatisch sicher ist. Es verspricht nicht, dass dezentrales Training die Frontier ersetzt. Es verspricht nicht, dass Privacy durch Abstraktion immer gelöst ist. Es verspricht nicht, dass Energieopportunismus alle Kosten senkt.

Diese Einschränkungen sind wichtig. Ohne sie wird die These zu Ideologie. Episodische Intelligenz ist stark, weil es Grenzen akzeptiert. Die Frontier bleibt notwendig, aber nicht als Default. Lokale Systeme bleiben begrenzt, aber nicht bedeutungslos. Regulierung hilft, aber ersetzt keine Architektur. Open Source hilft, aber ersetzt keinen Betrieb. Routing spart, aber nur wenn Qualität messbar bleibt.

Das Ziel ist nicht Reinheit. Das Ziel ist Tragfähigkeit.

13.11 Zwölf Architekturprinzipien

Erstens: Rohkontext bleibt so lokal wie möglich.

Zweitens: Externe Modelle erhalten Aufgaben, nicht automatisch Geschäftsgeheimnisse.

Drittens: Frontier-Calls sind Eskalationen, keine Reflexe.

Viertens: Jede Eskalation braucht einen Vertrag: Zweck, Kontext, Budget, Risiko, Rückgabeformat, Validierung.

Fünftens: Deterministische Gates prüfen, was nicht probabilistisch sein muss.

Sechstens: Speicher, Semantik und Runtime werden getrennt betrachtet.

Siebtens: Nicht zeitkritische Arbeit wird zeitlich und energetisch verschoben.

Achtens: Verteilte Arbeit muss prüfbar, signiert und begrenzt sein.

Neuntens: Open Source ist ein Baustein, nicht die ganze Souveränität.

Zehntens: Jede produktive KI-Architektur braucht Exit-Design.

Elftens: Kosten werden nicht nur in Euro gemessen, sondern auch in Energie, Kontextverlust, Kontrollverlust und Latenz.

Zwölftens: Das beste System ist nicht das, das am meisten Modell nutzt. Es ist das, das am wenigsten Modell braucht, ohne Qualität zu verlieren.

13.12 Ein Tag mit episodischer Intelligenz

Morgens fragt ein Mitarbeiter nach einer Risikoanalyse zu einem laufenden Projekt. Die lokale Runtime prüft Berechtigungen, erkennt den Projekttyp, lädt interne Dokumente, filtert vertrauliche Details und erzeugt eine abstrakte Aufgabenbeschreibung. Ein kleines lokales Modell klassifiziert: bekannte Risikoklasse, mittlere Unsicherheit, kein akuter Zeitdruck.

Das System beantwortet einfache Teile lokal. Für einen strategischen Abschnitt eskaliert es an ein Frontier-Modell: ohne Kundennamen, ohne Vertragsnummern, ohne interne IDs. Die Frontier liefert Argumentationsmuster, Gegenfragen und Szenarien. Zurück in der lokalen Runtime werden diese mit echten Daten zusammengeführt, durch Contracts geprüft, geloggt und dem Mitarbeiter mit Unsicherheitsmarkern angezeigt.

Am Nachmittag läuft bei Solarüberschuss ein Embedding-Refresh. Alte Dokumente werden neu geordnet, Evals werden wiederholt, ein lokaler Adapter wird getestet. Einige Ergebnisse gehen in die Freigabeschleife. Andere werden verworfen. Keine Magie. Kein Dauerhochofen. Ein Stoffwechsel.

Das ist episodische Intelligenz nicht als Zukunftsvision, sondern als Betriebsalltag.

13.13 Die Forschungsagenda

Episodische Intelligenz ist noch keine fertige Standardarchitektur. Viele Fragen bleiben offen. Wie misst man semantischen Informationsverlust bei Compression? Wie erkennt man, ob eine abstrahierte Anfrage noch genug Kontext enthält? Wie baut man portierbare Embedding-Gedächtnisse? Wie evaluiert man Eskalationsentscheidungen? Wie verhindert man, dass lokale Systeme falsche Sparsamkeit lernen? Wie schützt man verteilte Lernverbünde vor Poisoning? Wie misst man Energieeinsparung ehrlich, ohne nur Last zu verschieben?

Diese Fragen sind kein Einwand gegen die These. Sie sind der Grund, warum sie interessant ist. Die nächste Stufe produktiver KI wird nicht allein durch größere Modelle entstehen. Sie wird durch bessere Verteilung von Bedeutung, Energie und Kontrolle entstehen.

Kernsatz

Episodische Intelligenz ist keine Anti-Datacenter-These. Sie ist eine Anti-Default-These. Die Frontier bleibt. Aber sie wird bewusster angerufen, besser vorbereitet und härter geprüft.

13.14 Die Operating Loops

Episodische Intelligenz lässt sich auch als Schleifensystem lesen. Die erste Schleife ist Sense: Das lokale System nimmt wahr, welche Anfrage, welcher Kontext, welcher Nutzer, welche Datenklasse und welches Risiko vorliegen.

Die zweite Schleife ist Filter: Der Kontext wird reduziert, klassifiziert, anonymisiert oder zurückgewiesen. Nicht alles, was vorhanden ist, wird Teil der Aufgabe.

Die dritte Schleife ist Route: Das System entscheidet, ob eine deterministische Regel, ein lokales Modell, ein Tool, ein regionaler Knoten oder eine Frontier-Eskalation zuständig ist.

Die vierte Schleife ist Reason: Dort findet die eigentliche Verarbeitung statt. Aber Reason ist nicht immer Frontier. Manchmal ist es ein Cache. Manchmal ein SQL-Query. Manchmal ein kleiner Klassifikator. Manchmal ein großes Modell.

Die fünfte Schleife ist Verify: Ergebnisse werden gegen Contracts, Fakten, Rechte, Budgets und Zustände geprüft.

Die sechste Schleife ist Remember: Das System entscheidet, was in welcher Form gespeichert wird. Nicht jedes Ergebnis wird Gedächtnis. Nicht jedes Gedächtnis wird Cloud-Gedächtnis.

Diese Schleifen sind wichtiger als jedes einzelne Tool. Tools wechseln. Schleifen bleiben.

13.15 Der Weg zur Umsetzung

Ein Unternehmen muss Episodische Intelligenz nicht in einem großen Sprung bauen. Es kann in Schichten beginnen.

Phase eins: Messen. Welche Anfragen gehen wohin? Welche Daten werden übertragen? Welche Kosten entstehen? Welche Latenzen sind tatsächlich kritisch? Welche Aufgaben wiederholen sich? Ohne Messung bleibt Architektur ein Glaubensstreit.

Phase zwei: Lokales Gedächtnis. Dokumente, Metadaten, Begriffe, Embeddings und einfache Evals werden in einer eigenen Schicht gesammelt. Das Ziel ist nicht sofort Perfektion, sondern Besitz an der eigenen Bedeutungsstruktur.

Phase drei: Gates. Kritische Übergänge bekommen Contracts: Welche Daten dürfen raus? Welche Tools dürfen handeln? Wann wird eskaliert? Wann muss ein Mensch prüfen?

Phase vier: Kaskaden. Nicht jede Aufgabe geht an das größte Modell. Lokale Modelle, kleinere Cloud-Modelle, deterministische Regeln, Caches und Frontier werden nach Risiko und Nutzen kombiniert.

Phase fünf: Energie und Zeit. Hintergrundarbeit wird verschoben. Evals, Indizes und Datenpflege bekommen Zeitfenster. Was nicht sofort sein muss, wird nicht mehr wie ein Notfall behandelt.

Phase sechs: Exit. Der Wechsel wird geübt, nicht nur versprochen. Ein Modell wird testweise ersetzt. Ein Vektorstore wird repliziert. Ein Prompt-Set wird exportiert. Ein Provider wird gespiegelt. Souveränität wird erst real, wenn der Ausstieg einmal trocken durchgespielt wurde.

13.16 Typische Fehlformen

Die erste Fehlform ist Zentralismus aus Bequemlichkeit. Alles geht an ein großes Modell, weil das am Anfang am einfachsten ist. Kurzfristig schnell, langfristig teuer.

Die zweite Fehlform ist Lokalismus aus Stolz. Alles soll lokal laufen, auch wenn Qualität, Wartung und Sicherheit darunter leiden. Kurzfristig souverän klingend, langfristig fragil.

Die dritte Fehlform ist Governance als PDF. Policies werden geschrieben, aber nicht ausgeführt. Das System kennt die Regeln nicht, die Organisation nur auf Papier.

Die vierte Fehlform ist Open-Source-Theater. Ein offenes Modell wird eingesetzt, aber die Runtime, Datenhaltung und Evaluation bleiben proprietär oder unklar.

Die fünfte Fehlform ist Energiesparen ohne Messung. Last wird verschoben, aber nicht gemessen. Kosten sinken scheinbar, während Emissionen oder Risiken anderswo steigen.

Die sechste Fehlform ist falsche Sparsamkeit. Eine Anfrage wird nicht eskaliert, obwohl sie es müsste. Ein billiges Modell produziert eine schlechte Entscheidung, die später teurer wird als der gesparte Frontier-Call.

Episodische Intelligenz ist deshalb kein Sparprogramm. Es ist ein Entscheidungsprogramm.

13.17 Was zuerst gebaut werden sollte

Die wichtigste erste Komponente ist nicht das lokale Modell. Es ist die Beobachtungsschicht. Ohne Logs, Kostenmessung, Kontextklassifikation und Ergebnisprüfung weiß niemand, ob eine Architektur besser wird.

Die zweite Komponente ist ein lokaler semantischer Speicher. Nicht als perfekter Wissensgraph, sondern als stabiler Ort, an dem Dokumente, Begriffe, Embeddings, Metadaten und Evals zusammenkommen.

Die dritte Komponente ist ein Eskalationsvertrag. Jede Frontier-Nutzung sollte begründen können, warum sie nötig war, welchen Kontext sie erhalten hat und wie das Ergebnis geprüft wurde.

Die vierte Komponente ist ein kleiner Satz harter Verbote: keine Rohkundendaten in generische Prompts, keine Toolausführung ohne Gate, keine unversionierten Systemprompts, keine Ergebnisse ohne Herkunfts- oder Unsicherheitsmarkierung in kritischen Workflows.

Erst danach kommen komplexere Dinge: verteilte Knoten, lokale Fine-Tunes, energieadaptive Scheduler, Multi-Modell-Kaskaden, automatische Evaluierungen. Wer mit diesen Dingen beginnt, ohne die ersten Schichten zu besitzen, baut schöne Komplexität auf weichem Boden.

13.18 Der Satz, zu dem alles zurückkehrt

Am Ende lässt sich Episodische Intelligenz auf eine einzige Frage reduzieren:

Welche Teile eines intelligenten Systems müssen wirklich zentral, sofort und mit maximalem Energieeinsatz ausgeführt werden?

Diese Frage ist technischer als sie klingt. Sie ist wirtschaftlicher als sie klingt. Sie ist politischer als sie klingt. Und sie ist vielleicht die wichtigste Architekturfrage für KI nach der ersten Frontier-Welle.

Denn die Zukunft produktiver KI wird nicht nur davon abhängen, wie groß Modelle werden. Sie wird davon abhängen, wie klug wir entscheiden, wann wir sie überhaupt brauchen.

13.19 Zurück zum Anfang

Dieses Schlusskapitel sollte nicht das letzte Wort sein. Es sollte den Anfang schärfen. Kapitel 1 kann nun mit einer klareren Stimme beginnen: Die Illusion der permanenten Superintelligenz war nie nur ein technisches Missverständnis. Sie war eine falsche Vorstellung davon, wo Intelligenz sitzt.

Sie sitzt nicht nur im Modell. Sie sitzt im Routing. Im Speicher. Im Gate. Im Vertrag. Im Energiezeitfenster. In der lokalen Semantik. In der Entscheidung, nicht zu eskalieren. In der Fähigkeit, zu wechseln.

Episodische Intelligenz ist deshalb kein Ausstieg aus der KI. Es ist der Versuch, KI erwachsen zu betreiben.

Kernsatz
Episodische Intelligenz ist keine Anti-Datacenter-These. Sie ist eine Anti-Default-These. Die Frontier bleibt. Aber sie wird bewusster angerufen, besser vorbereitet und härter geprüft.

Anhang A - Glossar

Begriff	Arbeitsdefinition
Episodische Intelligenz	Betriebsform intelligenter Systeme, bei der nicht jede Aufgabe permanent an eine zentrale Frontier-Instanz geht, sondern lokal vorbereitet, geprüft, zeitlich geplant und nur bei Bedarf eskaliert wird.
Der Körper der KI	Leitmetapher des Buches: KI besteht nicht nur aus Modellen, sondern aus Energie, Hardware, Kühlung, Netzwerken, Daten, Runtime, Governance, Geschäftsmodellen und Orten.
Frontier	Sehr leistungsfähige Modelle und Infrastrukturen an der Leistungsgrenze aktueller KI. Im Buch nicht als Alltagsdefault, sondern als Hochenergie-Spezialinstanz verstanden.
Lokale semantische Runtime	Lokale Schicht, die Daten, Bedeutung, Rechte, Gedächtnis, Kontext und Vorentscheidungen in der Nähe des Nutzers oder Unternehmens organisiert.
Semantic Compression / Semantische Verdichtung	Lokale Abstraktion sensibler Rohkontexte zu strukturierten, möglichst informationsarmen, aber noch nützlichen Anfragen an stärkere Modelle.

Compression Contract	Prüfbarer Vertrag darüber, welche Informationen ein lokaler Verdichtungsprozess entfernen, erhalten, abstrahieren und später re-hydrieren darf.
Eskalationsvertrag	Regelwerk, das festlegt, wann ein lokales System an ein stärkeres Modell, einen regionalen Cluster, die Frontier oder einen Menschen übergibt.
Governance-Layer	Ausführbare Kontrollschicht aus Policies, Gates, Verträgen, Logging, Evaluationen und menschlichen Freigaben.
Policy Gate	Prüfpunkt, an dem entschieden wird, ob eine Anfrage, ein Tool-Aufruf, eine Datenfreigabe oder eine Eskalation erlaubt ist.
Ameisenmodell	Bild für viele kleine, kontrollierte Rechenknoten, die zerlegbare und überprüfbare Arbeitspakete übernehmen, ohne Frontier-Training vollständig zu ersetzen.
Energieopportunistische KI	KI-Betrieb, der verschiebbare Arbeit an Energieverfügbarkeit, Strommix, Preise, Netzlast, Wärmebudget und Business-SLA koppelt.
Re-Hydration	Rückübersetzung abstrahierter oder anonymisierter Ergebnisse in den lokalen echten Kontext. Nützlich, aber eine sensible Schwachstelle.
RAG	Retrieval-Augmented Generation: Verfahren, bei dem ein Modell mit extern gefundenen Dokument- oder Wissensausschnitten arbeitet, statt alles im Modell selbst zu speichern.
MoE	Mixture of Experts: Modellarchitektur, bei der je Anfrage nur bestimmte Experten aktiviert werden. Wichtiges Beispiel für sparse, geroutete Intelligenz.
Carbon-aware Scheduling	Planung von Rechenarbeit anhand von Emissions- oder Energieverfügbarkeitssignalen.

Dieses Glossar erklärt die Begriffe so, wie sie in diesem Buch verwendet werden. Es ist kein vollständiges technisches Lexikon, sondern ein Navigationsinstrument für die hybride Zielgruppe: Architekten, Entscheider, Praktiker und Leserinnen, die den Körper der KI verstehen wollen.

Agent: Ein KI-Systemteil, der nicht nur Text erzeugt, sondern Schritte plant, Tools aufruft, Zustände hält und Ergebnisse prüft. Im Buch ist ein Agent nie autonom im leeren Raum, sondern Teil einer Runtime.

Ameisenmodell: Bild für viele kleine, kontrollierte Rechenknoten, die zerlegbare Aufgaben übernehmen: Vorverarbeitung, Evals, Embeddings, Indexpflege oder kleine Trainingsläufe.

Audit Log: Nachvollziehbare Aufzeichnung von Entscheidungen, Kontexten, Tool-Aufrufen, Modellversionen und Ergebnissen. Es macht KI-Betrieb prüfbar, erzeugt aber selbst Datenschutzpflichten.

Carbon-aware Scheduling: Planung von Rechenarbeit anhand von CO2-Intensität, Strommix, Preis, Netzlast und Zeitfenstern. Entscheidend: günstiger Strom ist nicht automatisch sauberer Strom.

Checkpointing: Speichern eines Zwischenzustands, damit Rechenarbeit pausiert, verschoben und später fortgesetzt werden kann. Ohne Checkpointing bleibt energieopportunistische KI schnell Wunschdenken.

Compression Contract: Regelwerk, das festlegt, was bei semantischer Verdichtung erhalten, entfernt, transformiert oder verboten wird. Es ist die prüfbare Grenze zwischen Rohkontext und abstrahierter Anfrage.

Context Engineering: Gestaltung des Arbeitsgedächtnisses eines KI-Systems: Was wird geladen, gekürzt, priorisiert, ausgeschlossen oder geprüft? Es ersetzt die naive Idee, einfach immer mehr Kontext zu geben.

Deterministische Governance: Ausführbare Kontrollschicht aus Regeln, Zuständen, Verträgen und Gates. Sie ergänzt Modellverhalten durch prüfbare Übergänge.

Digitale Souveränität: Nicht alles selbst bauen, sondern Ausgänge besitzen: Daten, Semantik, Runtime, Modelle, Betrieb, Wechseloptionen und Begründbarkeit kontrollieren.

DiLoCo: Distributed Low-Communication Training. Eine Trainingsrichtung mit lokalen Updates und seltenerer globaler Synchronisierung, relevant für kommunikationsärmere Trainingsarchitekturen.

Edge: Rechen- oder Entscheidungsort nahe an Daten, Nutzer, Maschine oder Energiequelle. Edge bedeutet im Buch nicht unkontrolliert, sondern näher an Verantwortung.

Embedding: Numerische Repräsentation von Text, Bild oder Datenpunkt. Ähnliche Inhalte liegen im Vektorraum näher beieinander. Embeddings sind Teil des Gedächtnisses und daher sensibel.

Energy Contract: Betriebsregel, die festlegt, unter welchen Energie-, Zeit-, Risiko- und Datenschutzbedingungen ein Job laufen darf.

Episodische Intelligenz: Betriebsform, bei der KI lokal vorbereitet, deterministisch geführt, energieadaptiv geplant und nur bei Bedarf zur Frontier eskaliert wird.

Eskalationsvertrag: Regel, die begründet, wann eine Aufgabe an ein größeres Modell, einen regionalen Cluster oder einen Menschen übergeben wird.

Federated Learning: Training über mehrere Knoten, bei dem Daten lokal bleiben und nur Modellupdates oder Artefakte ausgetauscht werden. Es reduziert Datenbewegung, beseitigt aber keine Sicherheitsrisiken.

Frontier auf Abruf: Die Frontier wird nicht abgeschafft, sondern gezielt gerufen: bei hoher Unsicherheit, Neuartigkeit, Risiko oder strategischem Reasoning-Bedarf.

Frontier-Modell: Sehr leistungsfähiges allgemeines Modell am vorderen Rand aktueller Entwicklung. Im Buch ist es nicht Default, sondern Hochenergie-Spezialist.

Function Calling: Schnittstelle, bei der ein Modell strukturierte Tool-Aufrufe erzeugt, die eine Runtime ausführen oder ablehnen kann.

Gate: Prüfpunkt, der entscheidet, ob ein Kontext, Tool, Modell, Speicherzugriff oder nächster Schritt erlaubt ist.

Governance als PDF: Fehlform: Regeln existieren als Dokument, aber nicht im System. Das Buch fordert Governance als ausführbare Runtime-Schicht.

Hochofen: Metapher für stark gekoppelte Frontier-Infrastruktur. Sie bleibt notwendig, aber sie ist kein Werkzeug für jede Alltagsaufgabe.

Inference: Der laufende Betrieb eines Modells: Eine Anfrage wird verarbeitet und eine Antwort erzeugt. Inferenz ist der Alltag, Training der Hochofen.

Kooperative Edge-Infrastruktur: Kontrollierter Verbund lokaler oder regionaler Knoten für zerlegbare, prüfbare und oft zeitverschiebbare Rechenarbeit.

Lokale semantische Runtime: Lokale Schicht, die Rohkontext, Begriffe, Rollen, Policies, Speicher und Abstraktion in der Nähe der Daten hält.

Model Cascade: Mehrstufige Modellnutzung: Regeln, kleine Modelle, spezialisierte Modelle und Frontier werden nach Risiko, Qualität und Kosten kombiniert.

MoE / Mixture of Experts: Modellarchitektur, bei der nur ausgewählte Expertenteile aktiviert werden. Sie zeigt, dass Intelligenz nicht immer vollständig feuern muss.

PASS/FAIL: Kleinste harte Entscheidungseinheit im Governance-Layer: Ein Übergang wird akzeptiert, abgelehnt oder zur Klärung zurückgegeben.

Policy Gate: Prüfpunkt, der auf Basis von Rollen, Datenklassen, Risiken, Budgets und Policies entscheidet, ob ein Schritt erlaubt ist.

Prompt Injection: Angriff, bei dem Inhalte im Kontext versuchen, Systemregeln zu überschreiben, Daten herauszulocken oder Tools missbräuchlich zu steuern.

PUE: Power Usage Effectiveness. Verhältnis von Gesamtenergie eines Rechenzentrums zu IT-Energie. Niedriger ist effizienter, aber PUE hebt die Physik nicht auf.

RAG: Retrieval-Augmented Generation. Ein Modell erhält zusätzliche Informationen aus Suche, Datenbank oder Vektorstore, statt alles im Modell selbst zu tragen.

Re-Hydration: Lokaler Schritt, in dem eine abstrakte Frontier-Antwort wieder auf den echten Business-Kontext zurückgeführt wird. Hier entstehen eigene Risiken.

Runtime: Ausführende Umgebung eines KI-Systems: Routing, Tools, Speicher, Policies, Logs, Evals, Modellaufrufe und Zustandsübergänge.

Schatten-KI: Unkontrollierte KI-Nutzung außerhalb der vorgesehenen Runtime. Sie entsteht oft, wenn zentrale Systeme zu langsam, zu teuer oder zu unpraktisch sind.

Semantic Compression / Semantische Verdichtung: Kontrollierte Verdichtung von Rohkontext zu einer abstrakten Aufgabe, die genug Bedeutung erhält, aber unnötige Offenlegung vermeidet.

Speculative Decoding: Verfahren, bei dem ein kleines Modell Antwortteile vorlegt und ein größeres Modell sie prüft. Ein Beispiel für Effizienz durch Arbeitsteilung.

Structured Output: Modellausgabe in festem Schema, etwa JSON mit Pflichtfeldern. Sie macht Ergebnisse prüfbarer und anschlussfähiger an deterministische Systeme.

Tool Use: Nutzung externer Werkzeuge durch ein KI-System: Suche, Datenbank, Kalender, Codeausführung, Ticketsystem oder andere APIs.

Training: Prozess, in dem ein Modell aus Daten seine Parameter lernt. Besonders Frontier-Training benötigt eng gekoppelte Hochleistungscluster.

Vector Store / Vektorstore: Speicher für Embeddings. Er ermöglicht semantische Suche, ist aber selbst ein sensibler Teil des KI-Gedächtnisses.

Zero-Knowledge-Analogie: Bildhafte Nähe zur Idee, dass ein System genug Struktur sehen kann, ohne Rohgeheimnisse zu kennen. Im Buch keine Behauptung echter kryptografischer Zero-Knowledge-Proofs.

Anhang B - Abbildungsverzeichnis

Die folgenden Abbildungen sind als erste grafische Produktionsrunde in das Manuskript eingefügt. Sie bilden die zentralen Systemmodelle des Buches und können in Apple Pages oder im finalen Satz weiter verfeinert werden.

Abb. 6 - Semantische Verdichtung: Rohkontext bleibt lokal; die Frontier erhält eine abstrahierte, zweckgebundene Aufgabe.

Abb. 7 - Frontier auf Abruf: Eskalation erfolgt stufenweise nach Risiko, Unsicherheit und Neuartigkeit.

Abb. 8 - Prüfbare Übergänge: Contracts und Gates machen Übergänge pass/fail-fähig und auditierbar.

Abb. 9 - Das Ameisenmodell: Zerlegbare Arbeit wandert in kontrollierte Edge-Knoten und kommt als geprüftes Artefakt zurück.

Abb. 10 - Warten auf Sonne: Ein Energy Scheduler verschiebt Arbeit anhand von Energie-, Risiko- und SLA-Signalen.

Abb. 11 - Der Hochofen bleibt: Die Grenze des Schwarms: von gut zerlegbarer Arbeit bis zu eng gekoppeltem Frontier-Training.

Abb. 12 - Wem gehört das Gedächtnis?: Souveränität entsteht durch Kontrolle über Daten, Semantik, Runtime, Modelle, Betrieb und Exit.

Anhang C - Quellenverzeichnis

Dieses Quellenverzeichnis konsolidiert die kapitelbezogenen Quellenanker. Es trennt technische Dokumentation, Forschungsarbeiten, Standards, Marktquellen und regulatorische Quellen nicht vollständig bibliografisch, sondern nach ihrer Funktion im Buch.

Hinweis zum Umgang mit Quellen: Hersteller- und Plattformquellen werden vor allem für technische Produktdetails, Architekturhinweise und öffentliche Roadmaps genutzt. Normative, regulatorische und gesellschaftliche Schlussfolgerungen sollten sich auf Standards, Behörden, Forschung, Wettbewerbsanalysen und mehrere unabhängige Quellen stützen.

Energie, Rechenzentren und Green Software

Acun et al. (2022/2023): Carbon Explorer; Framework für 24/7 carbon-aware Datacenter mit erneuerbaren Quellen, Speicher und Workload-Scheduling. URL: https://arxiv.org/abs/2201.10036 (Zugriff: Mai 2026).

Carnegie Mellon University (2025): Nocturnal AI to solve energy curtailment; Forschungsimpuls zur Planung von AI-Workloads entlang erneuerbarer Energien. URL: https://energy.cmu.edu/news/2025/07/01-zhang-ai-energy.html (Zugriff: Mai 2026).

Chen et al. (2025): Electricity Demand and Grid Impacts of AI Data Centers; Überblick zu Lastprofilen, temporal/spatial flexibility und Netzherausforderungen. URL: https://arxiv.org/abs/2509.07218 (Zugriff: Mai 2026).

Colangelo et al. (2025): Turning AI Data Centers into Grid-Interactive Assets; Feld-Demonstration eines softwarebasierten Ansatzes zur Leistungsreduktion eines KI-Clusters. URL: https://arxiv.org/abs/2507.00909 (Zugriff: Mai 2026).

DOE release on LBNL report. URL: https://www.energy.gov/articles/doe-releases-new-report-evaluating-increase-electricity-demand-data-centers (Zugriff: Mai 2026).

Electricity Maps: Google Data Centers shift computations to cleaner times and locations; Beispiel für Carbon-Intensity-Prognosen als Scheduling-Signal. URL: https://www.electricitymaps.com/resources/case-studies/google-data-centers-shift-their-computations-to-cleaner-times-and-locations (Zugriff: Mai 2026).

Epoch AI: Power Usage in AI Data Centers. URL: https://epoch.ai/data-insights/gpus-power-usage-in-ai-data-centers (Zugriff: Mai 2026).

EPRI updated data center scenarios. URL: https://restservice.epri.com/publicattachment/97025 (Zugriff: Mai 2026).

European Commission - Energy performance of data centres. URL: https://energy.ec.europa.eu/topics/energy-efficiency/energy-efficiency-targets-directive-and-rules/energy-efficiency-directive/energy-performance-data-centres_en (Zugriff: Mai 2026).

German Datacenters Association. URL: https://www.germandatacenters.com/en/news-en/detail/figures-of-the-month-data-center-ecosystem-germany-may-2025/ (Zugriff: Mai 2026).

Google (2020): Our data centers now work harder when the sun shines and wind blows; carbon-intelligent computing platform für zeitlich flexible Workloads. URL: https://blog.google/innovation-and-ai/infrastructure-and-cloud/global-network/data-centers-work-harder-sun-shines-wind-blows/ (Zugriff: Mai 2026).

Google Cloud: Carbon free energy for Google Cloud regions; Einordnung von Standortwahl und regionaler Carbon-Free-Energy-Quote für Cloud-Services. URL: https://cloud.google.com/sustainability/region-carbon (Zugriff: Mai 2026).

Google Data Center PUE. URL: https://datacenters.google/efficiency/ (Zugriff: Mai 2026).

Google Sustainability: 24/7 Carbon-Free Energy by 2030; Zielbild der stündlichen statt nur jährlichen Energieabdeckung. URL: https://sustainability.google/reports/247-carbon-free-energy/ (Zugriff: Mai 2026).

Green Software Foundation: Carbon Aware SDK; Definition und Tooling für Software, die mehr tut, wenn Strom emissionsärmer ist, und weniger, wenn er emissionsintensiver ist. URL: https://carbon-aware-sdk.greensoftware.foundation/docs/overview (Zugriff: Mai 2026).

IEA (2025): Energy and AI. Relevanz: globaler und regionaler Strombedarf von Datenzentren, Wachstum bis 2030, lokale Konzentration als Netzproblem. URL: https://www.iea.org/reports/energy-and-ai/energy-demand-from-ai (Zugriff: Mai 2026).

IEA 4E critical review. URL: https://www.iea-4e.org/wp-content/uploads/2025/05/Data-Centre-Energy-Use-Critical-Review-of-Models-and-Results.pdf (Zugriff: Mai 2026).

International Energy Agency: Data Centres and Data Transmission Networks; Rechenzentren als mögliche flexible Lasten, Bedarf an Effizienz, Standards und Demand-Response-Anreizen. URL: https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks (Zugriff: Mai 2026).

International Energy Agency: Energy and AI - Executive Summary. URL: https://www.iea.org/reports/energy-and-ai/executive-summary (Zugriff: Mai 2026).

Knittel, Senga & Wang (2025): Flexible Data Centers and the Grid: Lower Costs, Higher Emissions?; wichtiges Gegenargument zur Gleichsetzung von Flexibilität, Kostenersparnis und Emissionssenkung. URL: https://www.nber.org/papers/w34065 (Zugriff: Mai 2026).

Kubernetes Documentation: Jobs; Suspended Jobs erlauben das temporäre Anhalten und Wiederaufnehmen von Jobs über .spec.suspend. URL: https://kubernetes.io/docs/concepts/workloads/controllers/job/ (Zugriff: Mai 2026).

Kueue Documentation: Workload and batch scheduling abstractions; relevant für Queueing, Admission und Stop/Resume von Workloads. URL: https://kueue.sigs.k8s.io/docs/concepts/workload/ (Zugriff: Mai 2026).

LBNL 2024 U.S. Data Center Energy Usage Report. URL: https://eta-publications.lbl.gov/sites/default/files/2024-12/lbnl-2024-united-states-data-center-energy-usage-report_1.pdf (Zugriff: Mai 2026).

NERSC Documentation: Checkpoint/Restart Overview; Checkpointing als Speichern des Prozesszustands zur späteren Fortsetzung. URL: https://docs.nersc.gov/development/checkpoint-restart/ (Zugriff: Mai 2026).

Slurm Workload Manager: Power Saving Guide; beschreibt Suspend/Resume-Mechanismen für Compute-Knoten im Clusterbetrieb. URL: https://slurm.schedmd.com/power_save.html (Zugriff: Mai 2026).

TEP Energy - Data centers in Switzerland. URL: https://www.tep-energy.ch/en/projects/detail/p1108.php (Zugriff: Mai 2026).

Umweltbundesamt carbon leakage study. URL: https://www.umweltbundesamt.de/sites/default/files/medien/11850/publikationen/68_2025_texte_0.pdf (Zugriff: Mai 2026).

WattTime: Grid emissions data and marginal emissions signals; relevant für die Unterscheidung zwischen durchschnittlichen und marginalen Emissionssignalen. URL: https://watttime.org/ (Zugriff: Mai 2026).

Hardware, Cluster und Frontier-Training

Barham et al. (2022): Pathways: Asynchronous Distributed Dataflow for ML; grossskalige Orchestrierung von ML-Workloads über viele Beschleuniger. URL: https://arxiv.org/abs/2203.12533 (Zugriff: Mai 2026).

CSCS: Alps; Schweizer Hochleistungsinfrastruktur mit NVIDIA Grace-Hopper-Knoten und offizieller Inbetriebnahme 2024. URL: https://www.cscs.ch/computers/alps (Zugriff: Mai 2026).

Epoch AI (2024): Training compute of frontier AI models grows by 4-5x per year; Trenddaten zu wachsendem Trainingscompute. URL: https://epoch.ai/blog/training-compute-of-frontier-ai-models-grows-by-4-5x-per-year (Zugriff: Mai 2026).

EuroHPC JU (2025): Auswahl zusätzlicher AI Factories in Europa; zeigt die regionale Ebene zwischen lokaler Infrastruktur und globaler Frontier. URL: https://www.eurohpc-ju.europa.eu/eurohpc-ju-selects-six-additional-ai-factories-expand-europes-ai-capabilities-2025-10-10_en (Zugriff: Mai 2026).

EuroHPC JU: AI Factories; beschreibt AI Factories als Ökosysteme um AI-optimierte Supercomputer für Industrie und Wissenschaft. URL: https://www.eurohpc-ju.europa.eu/ai-factories_en (Zugriff: Mai 2026).

Google Cloud (2023): Introducing Cloud TPU v5p and AI Hypercomputer; TPU-v5p-Pod mit 8.960 Chips, 4.800 Gbps/chip ICI und 3D-Torus-Topologie. URL: https://cloud.google.com/blog/products/ai-machine-learning/introducing-cloud-tpu-v5p-and-ai-hypercomputer (Zugriff: Mai 2026).

Google Cloud TPU v4 Documentation; 4096 Chips pro TPU-v4-Pod, 3D-Mesh, 1.1 Exaflops Peak pro Pod, hohe All-Reduce- und Bisection-Bandbreiten. URL: https://docs.cloud.google.com/tpu/docs/v4 (Zugriff: Mai 2026).

Hoffmann et al. (2022): Training Compute-Optimal Large Language Models; Chinchilla-Perspektive auf optimale Verteilung von Parametern und Trainingsdaten unter Compute-Budget. URL: https://arxiv.org/abs/2203.15556 (Zugriff: Mai 2026).

Jouppi et al. (2023): TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings; technische Beschreibung des TPU-v4-Supercomputers. URL: https://arxiv.org/abs/2304.01433 (Zugriff: Mai 2026).

Kaplan et al. (2020): Scaling Laws for Neural Language Models; empirische Skalierungsbeziehungen zwischen Modellgröße, Daten und Trainingscompute. URL: https://arxiv.org/abs/2001.08361 (Zugriff: Mai 2026).

Meta AI (2024): Introducing Meta Llama 3; beschreibt Daten-, Modell- und Pipelineparallelismus, >400 TFLOPS/GPU bei 16K GPUs, Trainingslaeufe auf zwei 24K-GPU-Clustern und automatisierte Fehlererkennung/Wartung. URL: https://ai.meta.com/blog/meta-llama-3/ (Zugriff: Mai 2026).

Meta Engineering (2024): Building Metas GenAI Infrastructure; beschreibt zwei neue 24K-GPU-Cluster für GenAI-Training und Infrastrukturziele mit Hunderttausenden H100-GPUs. URL: https://engineering.fb.com/2024/03/12/data-center-engineering/building-metas-genai-infrastructure/ (Zugriff: Mai 2026).

MLCommons: MLPerf Training Benchmark; misst, wie schnell Systeme Modelle auf eine Zielqualitaet trainieren können. URL: https://mlcommons.org/benchmarks/training/ (Zugriff: Mai 2026).

NVIDIA DGX B200 specs. URL: https://www.nvidia.com/en-us/data-center/dgx-b200/ (Zugriff: Mai 2026).

NVIDIA DGX H100/H200 guide. URL: https://docs.nvidia.com/dgx/dgxh100-user-guide/introduction-to-dgxh100.html (Zugriff: Mai 2026).

NVIDIA H100 Tensor Core GPU; NVLink 900 GB/s GPU-zu-GPU-Interconnect und NDR Quantum-2 InfiniBand als Skalierungsbausteine. URL: https://www.nvidia.com/en-us/data-center/h100/ (Zugriff: Mai 2026).

NVIDIA Quantum-2 InfiniBand Platform; 64 Ports mit 400 Gb/s und 51.2 Tb/s bidirektionaler Switch-Durchsatz. URL: https://www.nvidia.com/en-us/networking/quantum2/ (Zugriff: Mai 2026).

NVIDIA: GB200 NVL72; 72 Blackwell GPUs, 36 Grace CPUs und 130 TB/s NVLink-Kommunikation in einer Rack-Scale-Architektur. URL: https://www.nvidia.com/en-us/data-center/gb200-nvl72/ (Zugriff: Mai 2026).

OpenAI (2025): OpenAI and NVIDIA announce strategic partnership to deploy 10GW of NVIDIA systems; mindestens 10 GW KI-Datacenter-Infrastruktur. URL: https://openai.com/index/openai-nvidia-systems-partnership/ (Zugriff: Mai 2026).

OpenAI API Documentation: Structured model outputs. Quelle für JSON-Schema-basierte Ausgaben und die Unterscheidung zwischen strukturierter Antwort und Tool-/Function-Calling. URL: https://developers.openai.com/api/docs/guides/structured-outputs (Zugriff: Mai 2026).

Rajbhandari et al. (2019): ZeRO: Memory Optimizations Toward Training Trillion Parameter Models; Speicher- und Parallelismusoptimierung für sehr große Modelle. URL: https://arxiv.org/abs/1910.02054 (Zugriff: Mai 2026).

Shoeybi et al. (2019): Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism; Tensor-/Modellparallelismus für große Transformer. URL: https://arxiv.org/abs/1909.08053 (Zugriff: Mai 2026).

Zu et al. (2024): Resiliency at Scale: Managing Googles TPUv4 Machine Learning Supercomputer; Betrieb, Fehlertoleranz und Recovery eines 4096-node TPUv4-Supercomputers. URL: https://www.usenix.org/conference/nsdi24/presentation/zu (Zugriff: Mai 2026).

Edge, lokale Systeme und Runtime

S02 - Android Developers: Gemini Nano. Beschreibt Gemini Nano als On-Device-Foundation-Modell in Android AICore, mit lokaler Hardware-Nutzung, niedriger Inferenzlatenz und lokaler Verarbeitung. URL: https://developer.android.com/ai/gemini-nano (Zugriff: Mai 2026).

S03 - Google AI Edge: LiteRT. Beschreibt LiteRT als On-Device-Framework für ML- und GenAI-Deployment auf Edge-Plattformen mit niedriger Latenz und hoher Privacy. URL: https://ai.google.dev/edge/litert (Zugriff: Mai 2026).

S04 - Google AI Edge Overview. Beschreibt MediaPipe Tasks, generative AI tasks und LiteRT für Android, iOS, Web und Mikrocontroller mit CPU/GPU/NPU-Beschleunigung. URL: https://ai.google.dev/edge (Zugriff: Mai 2026).

S05 - Microsoft Learn: Copilot+ PCs developer guide. Nennt NPU-Anforderungen von 40+ TOPS für viele Windows-AI-Funktionen und beschreibt lokale NPU-Verarbeitung. URL: https://learn.microsoft.com/en-us/windows/ai/npu-devices/ (Zugriff: Mai 2026).

S06 - NVIDIA Developer: Jetson Embedded AI Computing Platform. Beschreibt Jetson als kompakte, energieeffiziente Edge-AI-Plattform für Robotik und industrielle Lösungen. URL: https://developer.nvidia.com/embedded-computing (Zugriff: Mai 2026).

S07 - Cloudflare Learning Center: What is Edge Computing? Definiert Edge Computing als Verlagerung von Rechenprozessen näher an Datenquellen, um Latenz und Bandbreite zu reduzieren. URL: https://www.cloudflare.com/learning/serverless/glossary/what-is-edge-computing/ (Zugriff: Mai 2026).

S08 - IBM: What is Edge AI? Beschreibt Vorteile wie reduzierte Latenz, weniger Bandbreite, lokale Datenverarbeitung, Datenschutz und Resilienz. URL: https://www.ibm.com/think/topics/edge-ai (Zugriff: Mai 2026).

S09 - llama.cpp GitHub. Beschreibt lokales LLM-Inference auf einer breiten Hardwarebasis sowie Quantisierung zur Reduktion von Speicherbedarf. URL: https://github.com/ggml-org/llama.cpp (Zugriff: Mai 2026).

S10 - FAISS GitHub. Beschreibt FAISS als Bibliothek für effiziente Ähnlichkeitssuche und Clustering dichter Vektoren. URL: https://github.com/facebookresearch/faiss (Zugriff: Mai 2026).

S11 - Qdrant Documentation: Local Quickstart. Zeigt lokale Nutzung für Vektorsuche und verweist auf Local Mode im Qdrant Client. URL: https://qdrant.tech/documentation/quickstart/ (Zugriff: Mai 2026).

Retrieval, Agents, Tools und Context Engineering

[K5-S17] Green Software Foundation: Software Carbon Intensity. Methodischer Anker dafür, Energie-/Carbon-Fragen pro funktionaler Einheit zu messen statt pauschal zu behaupten. URL: https://sci.greensoftware.foundation/ (Zugriff: Mai 2026).

Anthropic (2024): Building Effective AI Agents. Besonders relevant für Prompt-Chaining, Routing, Parallelisierung, Orchestrator-Worker und Evaluator-Optimizer. URL: https://www.anthropic.com/engineering/building-effective-agents (Zugriff: Mai 2026).

Anthropic (2025): Effective context engineering for AI agents. Besonders relevant für Kontext als endliche Ressource, Aufmerksamkeit, Toolauswahl und Kontextkuration. URL: https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents (Zugriff: Mai 2026).

Anthropic Claude Cookbook: Evaluator-Optimizer Workflow. Wichtig für Bewertungsschleifen und die Verbindung von Generierung und Prüfung. URL: https://platform.claude.com/cookbook/patterns-agents-evaluator-optimizer (Zugriff: Mai 2026).

Anthropic: Building Effective AI Agents. Wichtig für Workflow-Muster, Tool-Design, Routing und den Unterschied zwischen Agenten und einfachen Workflows. URL: https://www.anthropic.com/research/building-effective-agents (Zugriff: Mai 2026).

Karpas et al. (2022): MRKL Systems. Relevante Quelle für modulare neuro-symbolische Architekturen aus Sprachmodellen, externem Wissen und diskreten Reasoning-Modulen. URL: https://arxiv.org/abs/2205.00445 (Zugriff: Mai 2026).

LangGraph: Durable execution und Interrupts. Wichtig für pausable, wiederaufnehmbare Agentenprozesse mit gespeichertem Zustand und Human-in-the-Loop. URLs: https://docs.langchain.com/oss/python/langgraph/durable-execution und. URL: https://docs.langchain.com/oss/python/langgraph/interrupts (Zugriff: Mai 2026).

LangGraph: Durable execution und Interrupts. Wichtig für pausable, wiederaufnehmbare Agentenprozesse mit gespeichertem Zustand und Human-in-the-Loop. URLs: und https://docs.langchain.com/oss/python/langgraph/interrupts. URL: https://docs.langchain.com/oss/python/langgraph/durable-execution (Zugriff: Mai 2026).

Lewis et al. (2020/2021): Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Grundlage für RAG als Kombination aus parametrischem und nicht-parametrischem Gedächtnis. URL: https://arxiv.org/abs/2005.11401 (Zugriff: Mai 2026).

Model Context Protocol Documentation and Specification. Quelle für MCP als Standard zur Anbindung von Datenquellen, Tools und Workflows. https://modelcontextprotocol.io/docs/getting-started/intro und. URL: https://modelcontextprotocol.io/specification/2025-11-25 (Zugriff: Mai 2026).

Model Context Protocol Documentation and Specification. Quelle für MCP als Standard zur Anbindung von Datenquellen, Tools und Workflows. und https://modelcontextprotocol.io/specification/2025-11-25. URL: https://modelcontextprotocol.io/docs/getting-started/intro (Zugriff: Mai 2026).

OpenAI API Documentation: Prompt caching. Quelle für Kosten- und Latenzreduktion durch wiederkehrende Prompt-Prefixes. URL: https://developers.openai.com/api/docs/guides/prompt-caching (Zugriff: Mai 2026).

OpenTelemetry: Observability Primer. Relevant für die Definition von Observability als Fähigkeit, ein System von außen zu verstehen und unbekannte Fehler sichtbar zu machen. URL: https://opentelemetry.io/docs/concepts/observability-primer/ (Zugriff: Mai 2026).

OpenTelemetry: Semantic conventions for Generative AI systems. Relevant für Traces, Metriken, Events, Modell-Spans und Agent-Spans in KI-Runtimes. URL: https://opentelemetry.io/docs/specs/semconv/gen-ai/ (Zugriff: Mai 2026).

S16 - OpenTelemetry Documentation. Relevanter Observability-Anker für Metriken, Traces und Logs, die Verhalten statt Rohinhalt sichtbar machen können. URL: https://opentelemetry.io/docs/ (Zugriff: Mai 2026).

Schick et al. (2023): Toolformer. Relevante Quelle für Modelle, die lernen, APIs aufzurufen und Toolergebnisse in weitere Vorhersagen einzubauen. URL: https://arxiv.org/abs/2302.04761 (Zugriff: Mai 2026).

Yao et al. (2022/2023): ReAct: Synergizing Reasoning and Acting in Language Models. Grundlage für die Verbindung von Reasoning und Tool-/Action-Schritten. URL: https://arxiv.org/abs/2210.03629 (Zugriff: Mai 2026).

Governance, Sicherheit und Standards

[K5-S16] OpenAI Structured Outputs. Architekturanker für maschinenprüfbare Ausgabeformate, die Compression Contracts operationalisierbar machen können. URL: https://platform.openai.com/docs/guides/structured-outputs (Zugriff: Mai 2026).

EU AI Act: Article 12 Record-keeping und Article 14 Human Oversight. Wichtig für Logging, Nachvollziehbarkeit und menschliche Aufsicht bei Hochrisiko-Systemen. URLs: https://ai-act-law.eu/article/12/ und. URL: https://artificialintelligenceact.eu/article/14/ (Zugriff: Mai 2026).

EU AI Act: Article 12 Record-keeping und Article 14 Human Oversight. Wichtig für Logging, Nachvollziehbarkeit und menschliche Aufsicht bei Hochrisiko-Systemen. URLs: und https://artificialintelligenceact.eu/article/14/. URL: https://ai-act-law.eu/article/12/ (Zugriff: Mai 2026).

European Commission (2025): General-Purpose AI Models in the AI Act - Q&A. Relevanz: GPAI-Pflichten, systemische Risiken, Dokumentation und Modell-Governance. URL: https://digital-strategy.ec.europa.eu/en/faqs/general-purpose-ai-models-ai-act-questions-answers (Zugriff: Mai 2026).

European Commission: AI Act. URL: https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai (Zugriff: Mai 2026).

Europäische Union: Regulation (EU) 2024/1689, Artificial Intelligence Act. Wichtig für technische Dokumentation, Logging, Transparenz, menschliche Aufsicht, Robustheit und Cybersecurity bei Hochrisiko-Systemen. URL: https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng (Zugriff: Mai 2026).

ISO: ISO/IEC 42001:2023 Artificial intelligence — Management system. Wichtig als internationaler Standard für Prozesse, Verantwortlichkeiten und kontinuierliche Verbesserung im Umgang mit KI-Systemen. URL: https://www.iso.org/standard/42001 (Zugriff: Mai 2026).

LangGraph Documentation: Overview und Durable Execution. Quelle für stateful agent orchestration, persistence, human-in-the-loop und deterministische Wiederaufnahme von Workflows. URL: https://docs.langchain.com/oss/python/langgraph/overview (Zugriff: Mai 2026).

Model Context Protocol: Specification 2025-06-18. Wichtig als Beispiel für standardisierte Tool- und Kontextschnittstellen, die weiterhin Governance-Gates brauchen. URL: https://modelcontextprotocol.io/specification/2025-06-18 (Zugriff: Mai 2026).

NIST (2023): AI Risk Management Framework 1.0. Relevanz: Govern, Map, Measure, Manage als operationalisierbares Risikomanagement. URL: https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf (Zugriff: Mai 2026).

NIST: Artificial Intelligence Risk Management Framework (AI RMF 1.0) und AI RMF Core. Wichtig für Govern, Map, Measure, Manage als Zyklus des Risikomanagements. URLs: https://www.nist.gov/itl/ai-risk-management-framework und. URL: https://airc.nist.gov/airmf-resources/airmf/5-sec-core/ (Zugriff: Mai 2026).

NIST: Artificial Intelligence Risk Management Framework (AI RMF 1.0) und AI RMF Core. Wichtig für Govern, Map, Measure, Manage als Zyklus des Risikomanagements. URLs: und https://airc.nist.gov/airmf-resources/airmf/5-sec-core/. URL: https://www.nist.gov/itl/ai-risk-management-framework (Zugriff: Mai 2026).

OpenAI: Model Spec. Relevant für Chain of Command und die Abgrenzung zwischen Plattform-, Entwickler- und Nutzeranweisungen bei eskalierenden Systemen. URL: https://model-spec.openai.com/ (Zugriff: Mai 2026).

OpenAI: Structured Outputs and Function Calling. API Documentation. URLs: https://developers.openai.com/api/docs/guides/structured-outputs und. URL: https://developers.openai.com/api/docs/guides/function-calling (Zugriff: Mai 2026).

OWASP: Top 10 for Large Language Model Applications 2025. Wichtig für Prompt Injection, Insecure Output Handling, Excessive Agency, Sensitive Information Disclosure und Supply-Chain-Risiken. URL: https://owasp.org/www-project-top-10-for-large-language-model-applications/ (Zugriff: Mai 2026).

UK National Cyber Security Centre (2025): Prompt injection is not SQL injection. Besonders relevant für die These, dass LLMs keine inhärente Trennung zwischen Daten und Instruktionen besitzen und Sicherheitsdesign außerhalb des Modells notwendig bleibt. URL: https://www.ncsc.gov.uk/blog-post/prompt-injection-is-not-sql-injection (Zugriff: Mai 2026).

Privacy, semantische Verdichtung und Souveränität

[K5-S13] ICO: Anonymisation and pseudonymisation guidance (2025). Relevanter Anker für Re-Identifikationsrisiko, Pseudonymisierung und organisatorische Maßnahmen. URL: https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-sharing/anonymisation/ (Zugriff: Mai 2026).

[K5-S14] ICO: Pseudonymisation. Verweist ausdrücklich auf Re-Identifikation als Sicherheits-/Wirksamkeitstest von Anonymisierung und Pseudonymisierung. URL: https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-sharing/anonymisation/pseudonymisation/ (Zugriff: Mai 2026).

[K5-S15] NIST Privacy Framework. Privacy-Engineering-Anker für strukturierte Risikosteuerung, Datenverarbeitung und organisatorische Verantwortlichkeit. URL: https://www.nist.gov/privacy-framework (Zugriff: Mai 2026).

Apple Security Research: Private Cloud Compute Core Security & Privacy Requirements. URL: https://security.apple.com/documentation/private-cloud-compute/corerequirements (Zugriff: Mai 2026).

Apple Security Research: Private Cloud Compute, 2024. URL: https://security.apple.com/blog/private-cloud-compute/ (Zugriff: Mai 2026).

European Commission: Cloud Sovereignty Framework, Version 1.2.1, Oct. 2025. URL: https://commission.europa.eu/document/download/09579818-64a6-4dd5-9577-446ab6219113_en (Zugriff: Mai 2026).

European Commission: Commission advances cloud sovereignty through strategic procurement, 17 Apr 2026. URL: https://commission.europa.eu/news-and-media/news/commission-advances-cloud-sovereignty-through-strategic-procurement-2026-04-17_en (Zugriff: Mai 2026).

European Commission: Data Act. URL: https://digital-strategy.ec.europa.eu/en/policies/data-act (Zugriff: Mai 2026).

European Commission: Europe’s Open-Source AI Landscape: A lever for Innovation and Sovereignty, 2025. URL: https://digital-strategy.ec.europa.eu/en/library/europes-open-source-ai-landscape-lever-innovation-and-sovereignty (Zugriff: Mai 2026).

Linux Foundation Europe: World of Open Source Europe Report 2025. URL: https://www.linuxfoundation.org/press/linux-foundation-europe-report-finds-open-source-drives-innovation-and-digital-sovereignty-but-strategic-maturity-gaps-persist (Zugriff: Mai 2026).

S01 - Apple Machine Learning Research: Introducing Apple’s On-Device and Server Foundation Models. Beschreibt Apple Intelligence als Familie spezialisierter Modelle, darunter ein ungefähr 3B-Parameter On-Device-Sprachmodell und ein größeres Servermodell für Private Cloud Compute. URL: https://machinelearning.apple.com/research/introducing-apple-foundation-models (Zugriff: Mai 2026).

S13 - Apple Platform Security: Secure Enclave. Beschreibt die Secure Enclave als isoliertes Subsystem zur Absicherung sensibler Nutzerdaten und Schlüssel. URL: https://support.apple.com/guide/security/the-secure-enclave-sec59b0b31ff/web (Zugriff: Mai 2026).

S14 - Microsoft Intune: App Protection Policies Overview. Beschreibt App-Schutzrichtlinien, Datenverlustschutz, App-Grenzen und Schutz auch auf Geräten ohne vollständiges MDM-Enrollment. URL: https://learn.microsoft.com/en-us/intune/app-management/protection/overview (Zugriff: Mai 2026).

Verteiltes Lernen, Schwarm und föderierte Systeme

Bagdasaryan et al. (2020): How To Backdoor Federated Learning. Zeigt Modellpoisoning und Backdoor-Risiken in Federated-Learning-Szenarien. URL: https://proceedings.mlr.press/v108/bagdasaryan20a.html (Zugriff: Mai 2026).

Beutel et al. (2020/2022): Flower: A Friendly Federated Learning Research Framework; Framework für heterogene Federated-Learning-Szenarien. URL: https://arxiv.org/abs/2007.14390 (Zugriff: Mai 2026).

Blanchard et al. (2017): Machine Learning with Adversaries - Byzantine Tolerant Gradient Descent. Führt robuste Aggregation/Krum als Antwort auf Byzantine-Failures in verteiltem Lernen ein. URL: https://papers.nips.cc/paper/6617-machine-learning-with-adversaries-byzantine-tolerant-gradient-descent (Zugriff: Mai 2026).

BOINC, University of California, Berkeley: Plattform für freiwilliges wissenschaftliches Volunteer Computing; relevant als historischer Vorläufer verteilter, prüfbarer Arbeitspakete. URL: https://boinc.berkeley.edu/ (Zugriff: Mai 2026).

Borzunov et al. (2022/2023): Petals: Collaborative Inference and Fine-tuning of Large Models; kollaborative Nutzung großer Modelle über verteilte Ressourcen. URL: https://arxiv.org/abs/2209.01188 (Zugriff: Mai 2026).

Dong et al. (2025): Beyond A Single AI Cluster: A Survey of Decentralized LLM Training; beschreibt Chancen, aber auch Kommunikations-, Sicherheits- und Systemgrenzen dezentraler LLM-Trainingsansaetze. URL: https://aclanthology.org/2025.emnlp-main.1486/ (Zugriff: Mai 2026).

Dong et al. (2025): Beyond A Single AI Cluster: A Survey of Decentralized LLM Training; Überblick über Community- und Organisations-Dezentralisierung, Kommunikationsgrenzen, Heterogenität und Kosten. URL: https://aclanthology.org/2025.emnlp-main.1486.pdf (Zugriff: Mai 2026).

Douillard et al. (2023/2024): DiLoCo: Distributed Low-Communication Training of Language Models; seltenere globale Kommunikation und robuste lokale Schritte. URL: https://arxiv.org/abs/2311.08105 (Zugriff: Mai 2026).

Flower Labs Blog (2025): Photon: Federated Pre-training of Large Language Models; Kontext und Industrienarrativ zur Photon-Arbeit. URL: https://flower.ai/blog/2025-05-09-photon/ (Zugriff: Mai 2026).

Folding@home: Freiwilliges, weltweit verteiltes Rechnen für Protein- und Krankheitsforschung; Beispiel für zerlegbare, validierbare Arbeitspakete. URL: https://foldingathome.org/ (Zugriff: Mai 2026).

Fung et al. (2018/2020): Mitigating Sybils in Federated Learning Poisoning. Behandelt Sybil-basierte Modellpoisoning-Risiken in Federated Learning. URL: https://arxiv.org/abs/1808.04866 (Zugriff: Mai 2026).

Google DeepMind (2026): Decoupled DiLoCo; beschreibt verteiltes Pretraining über entfernte Rechenzentren mit geringerer Bandbreite und höherer Resilienz. URL: https://deepmind.google/blog/decoupled-diloco/ (Zugriff: Mai 2026).

Learning@home / Hivemind: Dezentrales Deep Learning in PyTorch über das Internet; relevant als Forschungs-/Open-Source-Signal, nicht als pauschaler Ersatz für kontrollierte KI-Infrastruktur. URL: https://github.com/learning-at-home/hivemind (Zugriff: Mai 2026).

McMahan et al. (2017): Communication-Efficient Learning of Deep Networks from Decentralized Data; Grundlagentext zu Federated Learning und Federated Averaging. URL: https://arxiv.org/abs/1602.05629 (Zugriff: Mai 2026).

Petals: Decentralized inference and fine-tuning. URL: https://petals.dev/ (Zugriff: Mai 2026).

Ryabinin et al. (2023): SWARM Parallelism; Training großer Modelle mit heterogenen, unzuverlässigen und schlecht verbundenen Geräten. URL: https://arxiv.org/abs/2301.11913 (Zugriff: Mai 2026).

Sani et al. (2024/2025): Photon: Federated LLM Pre-Training; federiertes End-to-End-Pretraining von LLMs bis 7B in Paper-Settings. URL: https://arxiv.org/pdf/2411.02908 (Zugriff: Mai 2026).

SETI Institute / SETI@home: Historischer Beleg für freiwilliges wissenschaftliches Schwarmrechnen; im Buch nur als Vorläufer des Prinzips, nicht als Vorschlag für private Heim-PCs als KI-Infrastruktur. URL: https://www.seti.org/news/seti-at-home-going-into-hibernation/ (Zugriff: Mai 2026).

Truong et al. (2021): Privacy preservation in federated learning; Federated Learning ist nicht automatisch eine Privacy-Garantie. URL: https://www.sciencedirect.com/science/article/pii/S0167404821002261 (Zugriff: Mai 2026).

Vana / Flower Labs (2025): Ankündigung COLLECTIVE-1; als Industriesignal vorsichtig verwenden, nicht als unabhängigen Beweis. URL: https://www.vana.org/posts/vana-flower-labs-partnership (Zugriff: Mai 2026).

Markt, Wettbewerb und Machtkonzentration

European Commission DG Competition (2024): Competition in Generative AI and Virtual Worlds. Relevanz: Partnerschaften, Abhängigkeiten, Konzentration kritischer Inputs und vertikale Integration. URL: https://competition-policy.ec.europa.eu/document/download/c86d461f-062e-4dde-a662-15228d6ca385_en (Zugriff: Mai 2026).

G7 Competition Authorities (2024): Competition in the Artificial Intelligence Tech Stack. Relevanz: KI-Wettbewerb als Stack-Frage, nicht nur als Modellfrage. URL:

https://en.agcm.it/dotcmsdoc/pressrelease/Discussion%20Paper%20for%20G7%20Competition%20Summit.pdf (Zugriff: Mai 2026).

Meta (2026): Q4 and Full Year 2025 Results. Relevanz: erwartete CapEx 2026 von 115 bis 135 Milliarden US-Dollar, getrieben durch AI-Infrastruktur. URL: https://investor.atmeta.com/investor-news/press-release-details/2026/Meta-Reports-Fourth-Quarter-and-Full-Year-2025-Results/default.aspx (Zugriff: Mai 2026).

Microsoft (2025): Annual Report 2025. Relevanz: AI-Infrastrukturwelle, mehr als 400 Datacenter in 70 Regionen, über zwei Gigawatt neue Kapazität. URL: https://www.microsoft.com/investor/reports/ar25/index.html (Zugriff: Mai 2026).

National Grid / Emerald AI / EPRI / Nebius / NVIDIA (2026): UK-first demonstration of AI data centres as flexible, grid-responsive assets. URL: https://www.nationalgrid.com/uk-first-trial-ai-grid-technology-successfully-demonstrates-ability-data-centres-adjust-power-needs (Zugriff: Mai 2026).

NVIDIA (2026): Q1 FY2027 Financial Results. Relevanz: 81,6 Milliarden US-Dollar Umsatz, 75,2 Milliarden Datacenter-Umsatz, AI factories als Infrastrukturwelle. URL: https://nvidianews.nvidia.com/news/nvidia-announces-financial-results-for-first-quarter-fiscal-2027 (Zugriff: Mai 2026).

OECD (2024): Artificial Intelligence, Data and Competition. Relevanz: Generative-AI-Wertschöpfungskette, Zugang zu Daten und Compute, Wettbewerbsrisiken durch Verflechtungen. URL: https://www.oecd.org/content/dam/oecd/en/publications/reports/2024/05/artificial-intelligence-data-and-competition_9d0ac766/e7e88884-en.pdf (Zugriff: Mai 2026).

Stanford HAI (2025): AI Index Report 2025. Relevanz: Industrieanteil an notable AI models, Compute- und Datenwachstum, Konzentration an der Frontier. URL: https://hai.stanford.edu/ai-index/2025-ai-index-report (Zugriff: Mai 2026).

Synergy Research Group (2025): Cloud Market Share Trends. Relevanz: AWS, Microsoft und Google mit zusammen 63 Prozent der Unternehmensausgaben für Cloud-Infrastruktur in Q3 2025. URL: https://www.srgresearch.com/articles/cloud-market-share-trends-big-three-together-hold-63-while-oracle-and-the-neoclouds-inch-higher (Zugriff: Mai 2026).

UK Competition and Markets Authority (2024): AI Foundation Models update / concerns. Relevanz: Risiken durch wenige große Technologieunternehmen, Compute, Daten, Talent und Zugangswege zum Markt. URL: https://www.gov.uk/government/news/cma-outlines-growing-concerns-in-markets-for-ai-foundation-models (Zugriff: Mai 2026).

Weitere Grundlagen und Referenzen

Apple Machine Learning Research: Apple Intelligence Foundation Language Models / 2025 Updates. 2024/2025. URLs: https://machinelearning.apple.com/research/introducing-apple-foundation-models und. URL: https://machinelearning.apple.com/research/apple-foundation-models-tech-report-2025 (Zugriff: Mai 2026).

Chen, Lingjiao; Zaharia, Matei; Zou, James: FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. arXiv / TMLR, 2023/2024. URL: https://arxiv.org/abs/2305.05176 (Zugriff: Mai 2026).

Dekoninck, Jasper et al.: A Unified Approach to Routing and Cascading for LLMs. arXiv, 2024. URL: https://arxiv.org/html/2410.10347 (Zugriff: Mai 2026).

Ding, Daoyuan et al.: Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing. ICLR 2024. URL: https://proceedings.iclr.cc/paper_files/paper/2024/file/b47d93c99fa22ac0b377578af0a1f63a-Paper-Conference.pdf (Zugriff: Mai 2026).

European Commission: 2025 State of the Digital Decade package. URL: https://digital-strategy.ec.europa.eu/en/policies/2025-state-digital-decade-package (Zugriff: Mai 2026).

European Commission: General-Purpose AI Code of Practice. URL: https://digital-strategy.ec.europa.eu/en/policies/ai-code-practice (Zugriff: Mai 2026).

Fedus, William; Zoph, Barret; Shazeer, Noam: Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. arXiv, 2021/2022. URL: https://arxiv.org/abs/2101.03961 (Zugriff: Mai 2026).

Google: Private AI Compute: our next step in building private and helpful AI. 2025. URL: https://blog.google/innovation-and-ai/products/google-private-ai-compute/ (Zugriff: Mai 2026).

Leviathan, Yaniv; Kalman, Matan; Matias, Yossi: Fast Inference from Transformers via Speculative Decoding. arXiv, 2022. URL: https://arxiv.org/abs/2211.17192 (Zugriff: Mai 2026).

Meta Engineering (2025): How Meta keeps its AI hardware reliable; große Trainingsjobs laufen synchron über tausende Beschleuniger, Komponentenausfälle können Training unterbrechen. URL: https://engineering.fb.com/2025/07/22/data-infrastructure/how-meta-keeps-its-ai-hardware-reliable/ (Zugriff: Mai 2026).

Mistral AI: Mixtral of Experts. 2023. URL: https://mistral.ai/news/mixtral-of-experts (Zugriff: Mai 2026).

Moslem et al.: Dynamic Model Routing and Cascading for Efficient LLM Inference. Survey, 2026. Wichtig für aktuelle Routing- und Cascading-Paradigmen über unabhängige Modelle hinweg. URL: https://arxiv.org/html/2603.04445v1 (Zugriff: Mai 2026).

Ouyang et al. (2020): Communication Optimization Strategies for Distributed Deep Neural Network Training; Kommunikationskosten als Kernherausforderung verteilten Trainings. URL: https://arxiv.org/abs/2003.03009 (Zugriff: Mai 2026).

S12 - Milvus Documentation: Deployment options. Beschreibt Milvus Lite, Standalone und Distributed als unterschiedliche Betriebsformen für lokale bis große Vektor-Datenbanken. URL: https://milvus.io/docs/install-overview.md (Zugriff: Mai 2026).

Snell, Charlie et al.: Scaling LLM Test-Time Compute Optimally Can be More Effective than Scaling Model Parameters. arXiv, 2024. URL: https://arxiv.org/abs/2408.03314 (Zugriff: Mai 2026).

Über den Autor

Joachim Müller-Peter

Joachim Müller-Peter arbeitet an der Schnittstelle von gewachsenen KMU-Systemen, Datenbanken, Web-Anwendungen, FileMaker, lokaler Infrastruktur und künstlicher Intelligenz. Sein Schwerpunkt ist die Frage, wie bestehende Systeme für Menschen und KI-Agenten verständlich, kontrollierbar, überprüfbar und rückbaubar werden.

Aus der praktischen Arbeit mit realen Betriebsumgebungen entstand sein Interesse an einer KI, die nicht nur als Modell, sondern als vollständiges System betrachtet wird: mit Energiebedarf, Hardware, Netzen, Speicher, Rollen, Regeln und Verantwortung.

Mit Model OS und Project Ant Colony untersucht er, wie lokale Rechner, spezialisierte Experten, deterministische Kontrollschichten und Frontier-Modelle sinnvoll zusammenspielen können. "Der Körper der KI" ordnet diese technische Arbeit in einen größeren wirtschaftlichen und gesellschaftlichen Zusammenhang ein.

Joachim Müller-Peter ist Gründer von itbois in der Schweiz.

> *"KI ist nicht körperlos. Sie erscheint als Sprache, aber sie entsteht als Infrastruktur."*

itbois

https://next.itbois.ch/

Der Körper der KI

KI erscheint auf dem Bildschirm als Text. Doch dahinter steht ein physischer Körper: Strom, Wärme, Chips, Netze, Speicher, Verträge, Rollen, Routinen und politische Macht.

Dieses Buch beschreibt eine andere Betriebsform intelligenter Systeme: episodische Intelligenz. Nicht jede Anfrage muss durch den teuersten Hochofen der Welt laufen. Vieles kann lokal vorbereitet, deterministisch geprüft, semantisch verdichtet, energieadaptiv geplant und erst bei Bedarf zur Frontier eskaliert werden.

Für Architekten, CTOs, Infrastrukturmenschen, KMU-Entscheider und alle, die KI nicht nur als Modell, sondern als System verstehen wollen.

ENTHÄLT

- 12 Architekturdiagramme
- erweitertes Glossar
- strukturierter Quellenapparat
- Pages-kompatible DOCX-Quelle

Erstausgabe · Juli 2026 · Quellenstand: Mai 2026 · © Joachim Müller-Peter