HTI

Ein Architekturentwurf für kontinuierlich lernende Sprachmodelle

Aktive Interaktion, lernorientiertes Thinking und zweiphasige Konsolidierung

Kernidee

Ein Modell soll nicht während jeder Interaktion unkontrolliert an seinen Gewichten lernen. Stattdessen wird Erfahrung zunächst in einer eigenen Thinking-Phase in prüfbares Lernmaterial umgeformt und erst danach in einer getrennten Konsolidierungsphase in das Modell eingebettet.

HTI = Homemade Technology-Based Intelligence

Konzeptstand: 11. Juli 2026 | Version 0.2

Technisches Konzeptpapier - öffentliche Fassung

Vorwort und Status des Dokuments

Dieses Dokument beschreibt eine Architekturidee aus der HTI-Serie. HTI steht für **Homemade Technology-Based Intelligence**. Ausgangspunkt ist HTI-1, ein kleines, selbst trainiertes Sprachmodell auf Basis einer CraftGPT-artigen Transformer-Implementierung mit rund fünf Millionen Parametern. Das hier ausgearbeitete System ist jedoch kein unmittelbarer Bauplan für HTI-1. Es beschreibt eine spätere Generation der Serie, ungefähr in dem Bereich, in dem aus einem statischen Chatmodell ein dauerhaft lernendes Modell werden soll.

Der Entwurf verfolgt bewusst ein anderes Skalierungsbild als die übliche Vorstellung, Intelligenz entstehe vor allem durch immer mehr Parameter. Die HTI-Hypothese lautet: **Architektur, Lernzyklus und Datenverarbeitung können wichtiger werden als reine Größe.** Ein späteres HTI-Modell könnte deshalb weiterhin relativ klein sein - beispielsweise in einer Größenordnung um hundert Millionen Parameter - und dennoch Fähigkeiten besitzen, die ein gewöhnliches Modell dieser Größe nicht hat. Das ist eine Zielsetzung, keine Leistungsbehauptung.

Das Papier ist ausführlich, weil die Idee nicht nur in einem Diagramm besteht. Sie verbindet mehrere Ebenen: Kontextverarbeitung, Reasoning, Datenaufbereitung, kontinuierliches Lernen, Gewichtsänderung, Stabilität, Selbstprüfung und Entwicklungsbeobachtung. Der eigentliche Forschungsgegenstand ist nicht das Wort "Schlaf", sondern die technische Trennung verschiedener Betriebszustände.

> **Statusmarkierungen**
>
> **Gesichert** bezeichnet etablierte Eigenschaften heutiger neuronaler Netze oder gut dokumentierte Forschung. **Plausibel** bezeichnet eine technisch begründete HTI-Entscheidung. **Offen** bezeichnet einen Teil, dessen konkrete Umsetzung experimentell gefunden werden muss.

Kurzfassung der Architektur

Ein gewöhnliches Sprachmodell wird trainiert, anschließend eingefroren und während der Nutzung nur ausgeführt. Es kann im Kontext arbeiten, aber die Erfahrung einer Unterhaltung verändert seine Parameter normalerweise nicht. HTI soll diesen Bruch überwinden. Eine Unterhaltung ist im Entwurf eine **aktive Phase** oder, umgangssprachlich, ein "Tag" des Modells. Der aktuelle Kontext ist dabei Arbeitsraum, nicht dauerhafte Erinnerung.

Nach einer oder mehreren aktiven Phasen beginnt eine erste, leichte Schlafphase. In ihr arbeitet ein besonderer, lernorientierter Thinking-Modus das Erlebte auf. Er entfernt irrelevante Oberflächenmerkmale, erkennt Korrekturen und Fehler, sucht Zusammenhänge, erzeugt Beispiele und Gegenbeispiele, trennt sichere Information von Vermutung und übersetzt den Gesprächsverlauf in Lernpakete. Diese Pakete bleiben in einer frühen HTI-Version bewusst lesbar, damit die Entwicklung beobachtbar und debuggbar bleibt.

Erst in der zweiten, tiefen Schlafphase wird das Modell verändert. Aus den Lernpaketen entsteht zusammen mit alten Referenzdaten eine Trainingsmenge. Das Modell wird offline aktualisiert, gegen eine unveränderte Referenz geprüft und nur übernommen, wenn neue Fähigkeiten entstanden sind, ohne dass alte Fähigkeiten unvertretbar beschädigt wurden. Damit ist das Modell nach jeder akzeptierten Konsolidierung tatsächlich nicht mehr exakt dasselbe Modell wie zuvor.

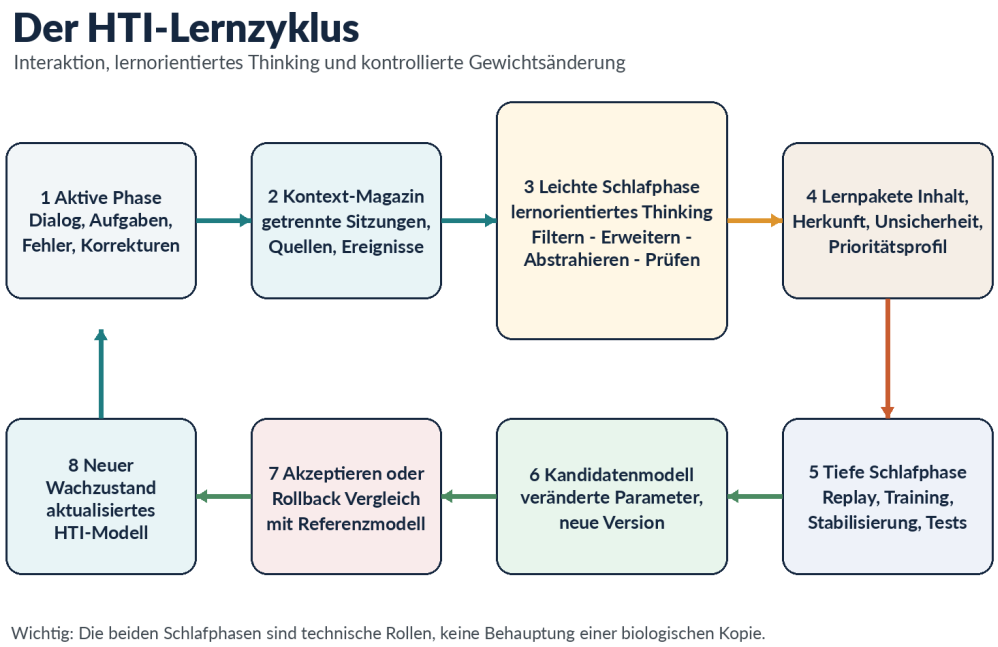

Abbildung 1: Der vollständige HTI-Lernzyklus. Die Architektur trennt Nutzung, Aufbereitung und dauerhafte Parameteränderung.

Inhalt

1. Technische Ausgangslage: Was ein heutiges Sprachmodell speichert

1.1 Token, Aktivierungen, Kontext und Gewichte

Ein Transformer-Sprachmodell erhält Text nicht direkt als Sätze, sondern als Folge von Tokens. Jedes Token wird in einen Vektor übersetzt. Durch die Transformer-Blöcke entstehen daraus fortlaufend neue Aktivierungen. Diese Aktivierungen enthalten keine sauber beschrifteten Faktenfelder, sondern verteilte numerische Zustände. Self-Attention erlaubt jedem Token, Informationen aus anderen Positionen der aktuellen Folge aufzunehmen. Formal wird die Aufmerksamkeitsmischung in vereinfachter Form als `softmax(QK^T / sqrt(d_k)) V` beschrieben [1].

Vier Dinge müssen strikt auseinandergehalten werden. **Der Kontext** ist die aktuell eingegebene Tokenfolge. **Die Aktivierungen** sind die vorübergehenden inneren Zustände, die bei ihrer Verarbeitung entstehen. **Die Gewichte** sind die langfristig trainierten Parameter des Netzes. **Das Training** ist der Prozess, der diese Gewichte anhand eines Fehlersignals verändert. Im normalen Chatbetrieb verändert sich nur der Kontext- und Aktivierungszustand; die Gewichte bleiben gleich.

Ebene	Lebensdauer	Funktion	HTI-Bedeutung
Kontext	Sekunden bis Sitzung	Aktuelle Eingaben und Arbeitsmaterial	Der "Tag" oder ein Teil davon
Aktivierungen	Ein Rechendurchlauf	Temporäre Verarbeitung und latente Repräsentation	Thinking und unmittelbare Schlussfolgerung
Temporärer Puffer	Stunden oder bis Schlafzyklus	Aufbewahrung mehrerer Sitzungen samt Metadaten	Kontext-Magazin, noch kein Langzeitwissen
Plastische Parameter	Mehrere Zyklen	Schnell lernbarer Teil des Modells	Experimentelle Neuroplastizität
Basisgewichte	Lange Zeiträume	Stabile Sprach- und Weltstruktur	Langfristig konsolidiertes Wissen

1.2 Warum Kontext noch kein Lernen ist

Ein Modell kann eine neue Information innerhalb einer Unterhaltung korrekt verwenden, ohne sie gelernt zu haben. Der Satz steht dann im Kontext und beeinflusst die Antwort über Attention. Sobald der Kontext entfernt wird, fehlt diese Stütze. Das Modell besitzt die Information nicht automatisch in seinen Parametern. Genau hier setzt HTI an: Es soll einen geregelten Weg vom flüchtigen Kontext zur dauerhaften Veränderung geben.

Die einfache Lösung, jedes Gespräch sofort zum Fine-Tuning zu verwenden, ist unzureichend. Ein Gespräch enthält Pausen, Schreibfehler, Witze, unvollständige Gedanken, bewusst falsche Beispiele, wechselnde Rollen, unerklärte Behauptungen und Modellfehler. Roher Dialog ist keine zuverlässige Trainingsmenge. Außerdem können kleine Updates alte Fähigkeiten beschädigen. Dieses Problem wird als **katastrophales Vergessen** oder allgemeiner als Interferenz zwischen alten und neuen Lerninhalten untersucht [5].

1.3 Wissen ist im Transformer verteilt

Es gibt in einem Transformer keinen einzelnen Speicherchip für eine neue Erinnerung. Sprachmuster, Fakten, Verfahren und Reaktionsneigungen entstehen aus dem Zusammenspiel vieler Matrizen. Bestimmte Studien konnten einzelne faktische Assoziationen in mittleren Feedforward-Modulen lokalisieren und gezielt bearbeiten [11]. Daraus folgt jedoch nicht, dass alle Arten von Wissen an einer Stelle liegen. Eine neue Fähigkeit, ein Schreibstil, ein abstraktes Konzept und eine einzelne Namensrelation können unterschiedliche Teile des Netzes betreffen.

> **Gesichert**
>
> Konsolidierung ist nicht das Beschreiben eines Speicherplatzes. Sie ist eine kontrollierte Änderung eines verteilten Funktionsapproximators. Die Frage lautet deshalb nicht nur "Wo schreibe ich die Information hin?", sondern "Welche Parameter dürfen sich wie stark verändern, damit das gewünschte Verhalten generalisiert und anderes Verhalten erhalten bleibt?"

2. Leitgedanke und Designprinzipien der HTI-Architektur

2.1 Die zentrale These

Die zentrale HTI-These lautet: **Ein lernendes Sprachmodell benötigt einen Zyklus, in dem Erfahrung zuerst verarbeitet und erst danach konsolidiert wird.** Thinking wird dabei nicht nur als Mittel zur Lösung einer Nutzeraufgabe verstanden. Es erhält eine zweite Rolle: Es verwandelt Erlebtes in Material, aus dem ein Modell sinnvoll lernen kann.

Das Bild ist bewusst anschaulich. Ein Mensch, der lange eine Blume betrachtet, träumt möglicherweise nicht nur von der sichtbaren Blüte, sondern von Wurzeln, Wachstum, Erde oder verwandten Erlebnissen. Auf die HTI-Idee übertragen bedeutet das nicht, dass das Modell menschlich träumen muss. Es bedeutet: Die Aufbereitungsphase darf über eine bloße Zusammenfassung hinausgehen. Sie soll Beziehungen erkunden, verborgene Annahmen sichtbar machen, Beispiele variieren und die Erfahrung in ein breiteres begriffliches Netz einordnen.

2.2 Die sechs Designprinzipien

1. **Trennung von Inferenz und Lernen.** Während der aktiven Unterhaltung werden die dauerhaften Gewichte nicht laufend verändert.

2. **Zweiphasige Offline-Verarbeitung.** Erst wird Erfahrung in Lernmaterial umgeformt; danach wird das Modell aktualisiert.

3. **Parametrisches Langzeitlernen.** Das endgültige Wissen soll im Modell selbst liegen, nicht ausschließlich in einer externen Suchdatenbank.

4. **Beobachtbarkeit vor Effizienz.** Frühe Versionen verwenden lesbare Thinking-Spuren und strukturierte Lernpakete, auch wenn latente Repräsentationen später effizienter sein könnten.

5. **Mehrdimensionale Auswahl statt eines simplen Filters.** Lernmaterial wird nach Verlässlichkeit, Neuheit, Allgemeinheit, Relevanz, Widerspruchsrisiko und weiteren Eigenschaften bewertet.

6. **Versionierte Selbstveränderung.** Jede Konsolidierung erzeugt ein Kandidatenmodell, das getestet, akzeptiert oder verworfen wird.

2.3 Was mit "Schlaf" gemeint ist - und was nicht

Der Begriff "Schlaf" ist eine technische Metapher für einen Betriebszustand außerhalb der normalen Interaktion. Die Biologie liefert eine starke Inspiration: Schlaf unterstützt Gedächtniskonsolidierung, Reaktivierung und Integration von Erinnerungen [3]. Die HTI-Architektur behauptet jedoch nicht, menschliche Schlafstadien eins zu eins nachzubauen. Sie übernimmt die abstrakte Trennung zwischen schneller Erfahrung, offline stattfindender Aufbereitung und langsamer Integration.

Diese Trennung hat auch eine lange Geschichte in der KI-Forschung. Der Wake-Sleep-Algorithmus von Hinton und Kollegen verwendete bereits getrennte Lernphasen für generative und erkennungsspezifische Verbindungen [4]. Complementary-Learning-Systems-Modelle argumentieren ebenfalls für ein schnelles und ein langsames Lernsystem, damit einzelne neue Erfahrungen nicht die über lange Zeit entstandene Struktur zerstören [2]. HTI ist nicht identisch mit diesen Ansätzen, steht aber in einer nachvollziehbaren Forschungslinie.

3. Der aktive Zustand: Interaktion ohne sofortige Gewichtsänderung

3.1 Die Unterhaltung als "Tag"

Im einfachsten HTI-Zyklus entspricht eine Unterhaltung einem Tag. Das Modell verarbeitet Fragen, führt Aufgaben aus, macht Fehler, erhält Korrekturen und beobachtet Folgen seiner Antworten. Der gesamte Verlauf ist eine Episode. In einer späteren Ausbaustufe kann ein Tag mehrere getrennte Gespräche oder Arbeitsaufgaben enthalten. Entscheidend ist nicht die reale Uhrzeit, sondern die Grenze eines Lernzyklus.

Während dieser Phase arbeitet das Modell wie ein gewöhnlicher Transformer: Es erzeugt Antworten aus dem aktuellen Kontext und seinen Gewichten. Lernrelevante Ereignisse können markiert werden, aber die langsamen Gewichte bleiben geschützt. So wird verhindert, dass eine einzelne missverständliche Aussage oder eine fehlerhafte Nutzerkorrektur sofort das Modell verändert.

3.2 Ereignisse statt bloßer Textmenge

Die aktive Phase sollte nicht nur einen Textlog liefern. Sie kann parallel ein Ereignisprotokoll erzeugen. Ein Ereignis ist beispielsweise: Das Modell gab eine falsche Funktion aus; der Nutzer zeigte einen reproduzierbaren Fehler; ein Werkzeug bestätigte den Fehler; das Modell korrigierte ihn; die neue Lösung bestand den Test. Dieses Ereignis ist lerntechnisch wertvoller als die bloße Zeichenfolge des gesamten Dialogs.

Ereignistyp	Beispiel	Mögliche Lernbedeutung
Explizite Korrektur	"Nein, diese Funktion erwartet Bytes, nicht Text."	Direktes Kandidatensignal, aber Quelle prüfen
Werkzeugergebnis	Test schlägt vor der Änderung fehl und	Starkes objektives Signal

Ereignistyp	Beispiel	Mögliche Lernbedeutung
	danach durch	
Wiederholter Fehler	Gleicher Denkfehler in mehreren Aufgaben	Mögliche allgemeine Schwäche
Neue Präferenz	Nutzer bevorzugt kurze technische Antworten	Personalisierung, nicht zwingend Weltwissen
Rollenspiel oder Hypothese	Nutzer behauptet absichtlich etwas Falsches	Nicht als Tatsache konsolidieren
Nebengeräusch	Lachen, Füllwort, Tippfehler	Meist verwerfen; kann kommunikativ relevant sein

3.3 Warum kein unmittelbares Online-Fine-Tuning

Die HTI-Entscheidung gegen permanente Gewichtsänderungen während der Unterhaltung ist technisch vernünftig. Ein laufendes Modell muss stabil antworten, während gleichzeitig ein Lernalgorithmus seine Funktionsweise verändert. Schon kleine Updates können unvorhergesehene Auswirkungen auf weit entfernte Aufgaben haben. Zusätzlich fehlt im Moment der Interaktion oft der vollständige Kontext: Eine scheinbare Korrektur kann später zurückgenommen werden, ein Codevorschlag kann erst nach mehreren Tests als falsch erkannt werden.

Das bedeutet nicht, dass im aktiven Zustand überhaupt keine Anpassung stattfinden darf. Temporäre Zustände, schnelle Gewichte oder Sitzungsspezifische Adapter könnten sich verändern, solange sie noch nicht als langfristige Wahrheit gelten. Die dauerhafte Übernahme bleibt jedoch der tiefen Schlafphase vorbehalten.

4. Das Kontext-Magazin und das Problem sehr lang Sitzungen

4.1 Das Magazin als temporäre Struktur

Mehrere Gespräche sollen nicht zwangsläufig zu einem einzigen gigantischen Kontext verklebt werden. Dafür dient das **Kontext-Magazin**: eine temporäre Sammlung einzelner Sitzungen, Ereignisse, Werkzeugresultate und Herkunftsangaben. Das Magazin ist kein endgültiges externes Gedächtnis. Es entspricht eher einem Arbeitsarchiv, das nach erfolgreicher Konsolidierung geleert, verdichtet oder nur für Audits aufbewahrt werden kann.

Die Metapher eines Magazins ist passend: Es hält getrennte "Patronen" aus Kontext. Eine Konsolidierungsrunde kann eine einzelne Sitzung, mehrere zusammengehörige Sitzungen oder einen ganzen Tagesblock laden. Die Trennung verhindert, dass zeitlich entfernte Aussagen ohne Herkunft vermischt werden.

4.2 Warum ein übergroßer Kontext nicht einfach eingespeist werden ka

Ein Transformer besitzt eine vorgesehene maximale Kontextlänge. Wird diese Grenze überschritten, fehlen Positionsdarstellungen, Speicher oder implementierte Attention-Strukturen für weitere Tokens. Selbst unterhalb der Grenze wächst die klassische Self-Attention quadratisch

mit der Sequenzlänge, weil viele Tokenpaare miteinander verglichen werden [1]. Ein Modell, das für eine bestimmte Länge trainiert wurde, kann daher nicht zuverlässig einfach das Doppelte oder Zehnfache "verschlingen".

Die robustere HTI-Lösung ist eine hierarchische Verarbeitung. Lange Gespräche werden in überlappende Abschnitte zerlegt. Jeder Abschnitt wird lokal verstanden und in vorläufige Lernpakete umgewandelt. Danach werden Pakete gruppenweise zusammengeführt. Erst auf der höchsten Stufe entsteht eine globale Sicht. Entscheidend ist, dass Quellverweise erhalten bleiben, damit eine spätere Synthese immer zum ursprünglichen Ereignis zurückkehren kann.

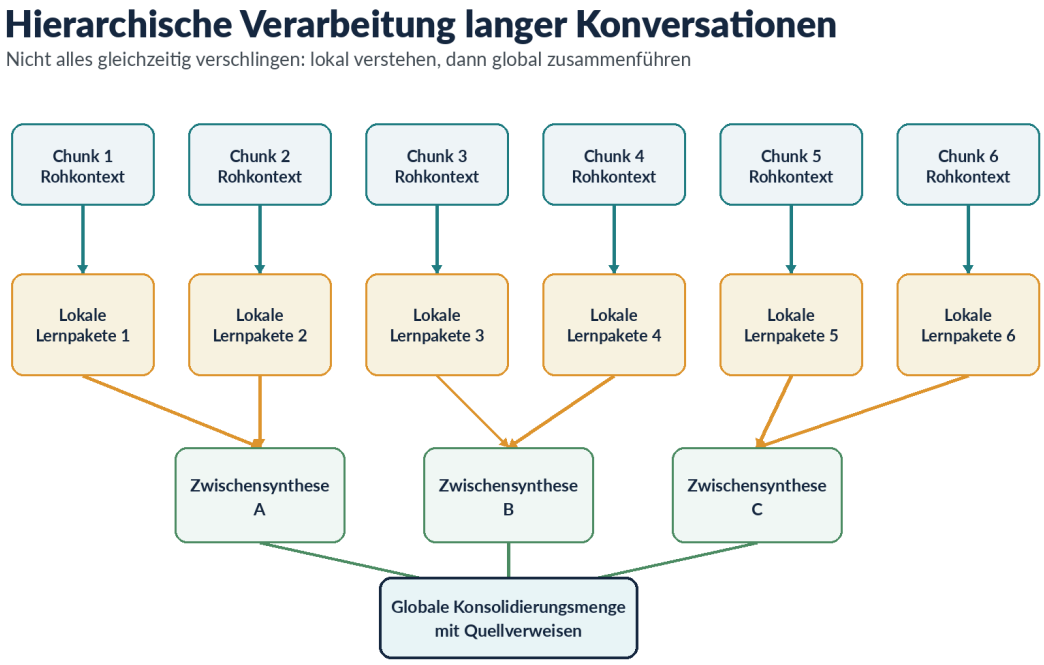

Abbildung 2: Hierarchische Verarbeitung. Lokale Pakete werden schrittweise synthetisiert, ohne den gesamten Rohkontext gleichzeitig in ein begrenztes Fenster zu pressen.

4.3 Zusammenfassung allein reicht nicht

Eine gewöhnliche Zusammenfassung reduziert Textlänge, kann aber technische Details, Gegenbeispiele und Unsicherheiten verlieren. Für HTI muss die Kompression auf den späteren Lernzweck ausgerichtet sein. Bei Code kann ein kurzer Satz wie "der Datentyp war falsch" unzureichend sein; benötigt werden möglicherweise fehlerhafte Eingabe, Stacktrace, Korrektur, Testfall und die allgemeine Regel. Bei einem persönlichen Gespräch kann dagegen der exakte Wortlaut irrelevant sein, während die stabile Präferenz erhalten bleiben sollte.

Deshalb erzeugt jede lokale Verarbeitung nicht nur Fließtext, sondern strukturierte Felder: Ereignis, Kontextbedingungen, behauptete Regel, Evidenz, Gegenbeispiel, Unsicherheit und Verweis auf die Quelle. Die globale Stufe darf Pakete zusammenführen, widersprüchliche Pakete aber nicht stillschweigend zu einer scheinbar eindeutigen Wahrheit glätten.

5. Die leichte Schlafphase: lernorientiertes Thinking

5.1 Zwei Arten von Thinking

Im aktiven Zustand dient Thinking dazu, eine Aufgabe zu lösen: planen, rechnen, vergleichen, Code prüfen oder eine Antwort formulieren. In der leichten Schlafphase erhält Thinking ein anderes Ziel. Es soll nicht primär eine Nutzerantwort erzeugen, sondern die Erfahrung **lernbar** machen. Deshalb benötigt HTI mindestens zwei Funktionsmodi desselben Grundsystems: dialogorientiertes Reasoning und lernorientiertes Reasoning.

Das sichtbare Reasoning heutiger Modelle erscheint oft als Text. Intern besteht Verarbeitung jedoch aus hochdimensionalen Aktivierungen und parallelen Rechenpfaden. HTI muss nicht annehmen, dass Text die wahre innere Sprache eines Modells ist. Für frühe Entwicklungsstufen ist Text trotzdem wertvoll: Der Entwickler kann beobachten, welche Regel das Modell aus einem Gespräch ableitet, welche Unsicherheit es erkennt und welche fantasierte Ergänzung es versehentlich als Tatsache behandelt.

5.2 Aufgaben des lernorientierten Thinkings

- **Relevanztrennung:** Füllwörter, Tippfehler, Smalltalk und zufällige Formulierungen werden von stabilen Lernsignalen getrennt.
- **Ereignisrekonstruktion:** Der Modus stellt Ursache, Fehler, Korrektur und Ergebnis in einer klaren Reihenfolge her.
- **Abstraktion:** Aus einem einzelnen Fall wird eine mögliche allgemeine Regel gebildet.
- **Kontrastbildung:** Es werden Fälle erzeugt, in denen die Regel gilt, und Fälle, in denen sie gerade nicht gilt.
- **Assoziative Erweiterung:** Beziehungen zu vorhandenem Wissen werden gesucht; die Erfahrung wird in ein größeres begriffliches Netz eingeordnet.
- **Widerspruchssuche:** Neue Aussagen werden gegen ältere Pakete, Tests und vorhandene Modellantworten geprüft.
- **Unsicherheitsmarkierung:** Menschliche Aussage, Werkzeugbeweis, Modellschluss und freie Hypothese bleiben unterscheidbar.
- **Kompression:** Aus umfangreicher Interaktion entsteht eine kleine, aber trainingsfähige Menge.

5.3 "Halluzinieren" als kontrollierte Exploration

Die ursprüngliche Idee erlaubt der leichten Schlafphase, "alles Mögliche drumherum" zu denken. Technisch sollte das nicht als ungebremstes Erfinden verstanden werden, sondern als **kontrollierte assoziative Exploration**. Das Modell darf Hypothesen, Analogien und neue Beispiele erzeugen, solange jede Erweiterung als generiert markiert bleibt und nicht denselben Status wie beobachtete Evidenz erhält.

Ein Beispiel: Der Nutzer korrigiert einen Python-Fehler, weil eine Datei binär statt als UTF-8-Text geöffnet werden muss. Der Thinking-Modus kann verwandte Fälle generieren: Bilddateien, komprimierte Daten, Netzwerkpakete, Unterschied zwischen `str` und `bytes`. Das erweitert den Lernnutzen. Er darf aber nicht ohne Beleg behaupten, eine bestimmte Bibliothek verhalte sich auf eine bestimmte Weise. Generierte Erweiterungen werden zu Trainingskandidaten, die durch Regeln, Tests oder vorhandenes Wissen geprüft werden müssen.

Risiko

Selbst erzeugtes Lernmaterial kann Fehler verstärken. Forschung zu rekursiv erzeugten Trainingsdaten zeigt, dass unkritisches Lernen aus Modelloutput Verteilungen verarmen und Modelle degenerieren lassen kann [12]. HTI muss deshalb Herkunft und Evidenz jedes Pakets bewahren.

5.4 Die halbmenschliche Lernsprache

Der Output der ersten Schlafphase ist weder der rohe menschliche Dialog noch bereits eine direkte Gewichtsänderung. Er ist eine **halbmenschliche Lernsprache**: für Menschen lesbar, aber nach den Bedürfnissen des Modells strukturiert. Ein Paket kann natürliche Sprache, formale Felder, Code, Testfälle, Wahrscheinlichkeiten und Verknüpfungen enthalten.

Diese Zwischenform ist wichtig. Natürliche Sprache bietet Debugging und Flexibilität. Struktur bietet Verlässlichkeit und maschinelle Weiterverarbeitung. Spätere HTI-Generationen könnten Teile davon in latente Vektorpakete verlagern, doch die lesbare Spur sollte als Diagnosekanal erhalten bleiben. Das interne Lernformat muss nicht vollständig interpretierbar sein; die Entscheidungskette muss aber auditierbar bleiben.

5.5 Ein mögliches Ausgabeformat

```
LEARN_PACKET {
  claim: "Binärdaten dürfen nicht implizit als UTF-8-Text behandelt werden."
  scope: [Datei-I/O, Netzwerkdaten, Bilder, Archive]
  evidence: [tool_test#481, user_correction#17]
  counterexamples: [tatsächliche Textdatei mit bekannter Kodierung]
  confidence: 0.94
  novelty: 0.42
  generality: 0.81
  provenance: observed + verified
  generated_examples: [example#A, example#B]
  dependencies: [bytes_vs_str, encoding]
}
```

Das Format ist nur ein Entwicklungsbeispiel. Die entscheidende Idee ist die Trennung zwischen Aussage, Gültigkeitsbereich, Belegen, Gegenbeispielen, Unsicherheit und Herkunft. Ein einzelner Wichtigkeitswert würde diese Struktur zu stark vereinfachen.

6. Lernpakete, Wichtigkeit und epistemische Qualit

6.1 Warum "wichtig" nicht eindimensional ist

Ein persönlicher Name kann für einen Nutzer sehr relevant, für allgemeines Weltwissen aber unwichtig sein. Eine neue mathematische Regel kann allgemein sein, aber falsch verstanden. Ein einmaliger Werkzeugtest kann verlässlich, aber eng begrenzt sein. Deshalb sollte der Thinking-Modus keinen einzigen Wichtigkeitswert erzeugen, der alles vermischt.

Stattdessen erhält jedes Lernpaket ein Profil. Die spätere Konsolidierung entscheidet abhängig vom Pakettyp. Hohe Neuheit und geringe Verlässlichkeit kann zu einem Beobachtungsstatus führen. Hohe Verlässlichkeit und geringe Allgemeinheit kann in einen kleinen personalisierten Adapter

gelangen. Wiederholte, breit bestätigte Muster können Kandidaten für langsamere Basisgewichte werden.

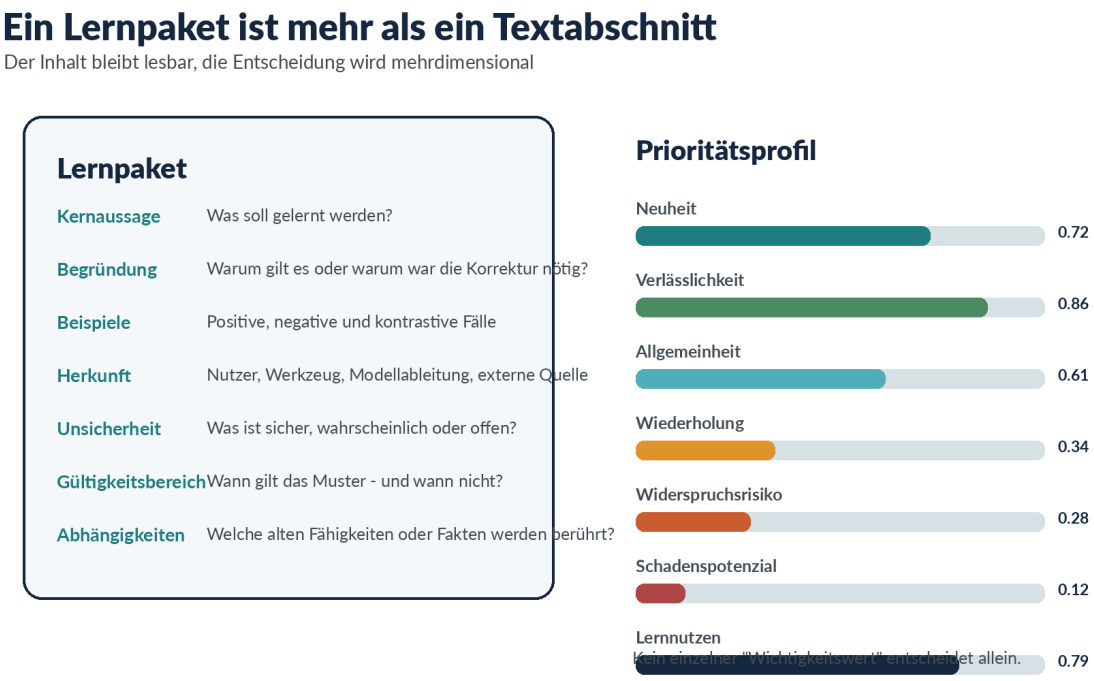

Abbildung 3: Ein Lernpaket verbindet lesbaren Inhalt mit einem mehrdimensionalen Prioritätsprofil.

6.2 Vorgeschlagene Dimensionen

Dimension	Frage	Wirkung auf Konsolidierung
Verlässlichkeit	Wie stark ist die Aussage belegt?	Bestimmt, ob überhaupt trainiert werden darf
Neuheit	Ist das Muster für das Modell tatsächlich neu?	Verhindert unnötige Updates
Allgemeinheit	Gilt es nur hier oder in vielen Situationen?	Steuert Reichweite der Gewichtsänderung
Wiederholung	Tauchte das Signal mehrfach unabhängig auf?	Erhöht Stabilität und Priorität
Nutzerrelevanz	Ist es für eine bestimmte Person oder Aufgabe bedeutsam?	Kann personalisierten Lernpfad wählen
Widerspruch	Kollidiert es mit vorhandenem Wissen oder anderen Paketen?	Erzwingt Prüfung oder Zurückhaltung
Schadenspotenzial	Wie teuer wäre eine falsche Übernahme?	Erhöht Test- und Evidenzanforderungen
Kompressibilität	Lässt sich das Muster als klare Regel darstellen?	Beeinflusst Trainingsformat
Vergessensrisiko	Welche alten Fähigkeiten könnten betroffen sein?	Steuert Replay und Regularisierung

6.3 Klassen und Wahrscheinlichkeiten gehören zusammen

Eine reine Klassifikation - Fakt, Präferenz, Fähigkeit, Korrektur, Stil, Hypothese - ist hilfreich, aber nicht ausreichend. Jede Klasse benötigt eigene Regeln und kann trotzdem unsicher sein. HTI sollte deshalb Klassen als Routing-Information und kontinuierliche Werte als Qualitätsprofil verwenden. Ein Paket kann beispielsweise zu 70 Prozent eine allgemeine Programmierregel und zu 30 Prozent eine projektspezifische Konvention sein.

Der Thinking-Modus kann diese Werte zunächst selbst schätzen. Langfristig sollten die Schätzungen kalibriert werden: Wenn Pakete mit 0,8 Verlässlichkeit später in 95 Prozent der Fälle stimmen, ist das System übervorsichtig; wenn sie nur in 50 Prozent stimmen, ist es überkonfident. Der Lernprozess muss somit auch lernen, seine eigenen Lernsignale einzuschätzen.

7. Die tiefe Schlafphase: eigentliche Konsolidierung

7.1 Was in dieser Phase grundsätzlich geschieht

In der tiefen Schlafphase wird das aktive Modell angehalten oder als unveränderliche Referenz eingefroren. Aus ausgewählten Lernpaketen entsteht eine Trainingsmenge. Sie enthält nicht nur neue Beispiele, sondern auch alte Referenzbeispiele, Gegenbeispiele und Stabilitätstests. Anschließend werden bestimmte Parameter aktualisiert. Das Ergebnis ist ein Kandidatenmodell mit einer neuen Versionsnummer.

Mathematisch kann die einfachste Form weiterhin gewöhnliches Gradientenlernen sein. Der Fehler der Trainingsbeispiele wird berechnet, zurückpropagiert und die Parameter werden verändert:

```
theta_new = theta_old - eta * gradient_theta L
```

Die Innovation der HTI-Idee liegt zunächst nicht darin, Backpropagation abzuschaffen. Sie liegt im gesamten Prozess davor und darum herum: Welche Erfahrung wird ausgewählt? Wie wird sie in Training umgeformt? Welche alten Daten werden wiederholt? Welche Parameter dürfen sich verändern? Wie wird entschieden, ob das Ergebnis besser ist? Später kann auch die Update-Regel selbst gelernt werden.

7.2 Der Mindestaufbau einer soliden Konsolidierung

1. **Neue Lernpakete:** verifizierte oder ausreichend markierte neue Inhalte.
2. **Replay-Menge:** alte Beispiele, die wichtige bestehende Fähigkeiten repräsentieren.
3. **Kontrastive Beispiele:** ähnliche Fälle, in denen die neue Regel nicht angewendet werden darf.
4. **Stabilitätsstrafe:** wichtige alte Parameter oder Funktionen sollen sich nur begrenzt verändern.
5. **Kandidaten-Checkpoint:** das alte Modell bleibt vollständig wiederherstellbar.
6. **Vergleichsevaluation:** neue Leistung, altes Wissen, Kalibrierung und Verhalten werden vor der Übernahme gemessen.

Ein mögliches Verlustmodell verbindet diese Ziele:

```
L_total = L_new
        + lambda_replay * L_replay
        + lambda_stability * Sum_i F_i (theta_i - theta_i_old)^2
```

```
        + lambda_contrast * L_contrast
```

Der EWC-Ansatz schützt beispielsweise Gewichte, die für frühere Aufgaben wichtig waren, durch eine gewichtete quadratische Strafe [5]. Generative-Replay-Verfahren mischen neue Beispiele mit rekonstruierten alten Beispielen, um Vergessen zu verringern [6]. HTI kann solche Verfahren als Ausgangspunkt verwenden, ohne sich auf eines festzulegen.

7.3 Die Konsolidierung ist nicht einfach "alles trainieren"

Vollständiges Fine-Tuning aller Parameter nach jeder Unterhaltung wäre die direkteste, aber riskanteste Form. Es kann funktionieren, ist jedoch teuer und erzeugt starke Interferenz. Eine robustere frühe Architektur begrenzt Plastizität auf ausgewählte Module. Dort darf das Modell schnell lernen, während der Kern eingefroren bleibt. Bewährte Änderungen können später in langsamere Gewichte überführt werden.

Damit entsteht eine Hierarchie: Aktivierungen sind extrem schnell und flüchtig. Plastische Gewichte oder Adapter lernen pro Schlafzyklus. Basisgewichte verändern sich nur nach vielen erfolgreichen Zyklen. Forschung zu Fast Weights schlägt ebenfalls Variablen vor, die sich schneller als normale Gewichte, aber langsamer als Aktivierungen verändern [7]. Differenzierbare Plastizität zeigt, dass auch die Plastizitätsregeln selbst trainierbar sein können [8].

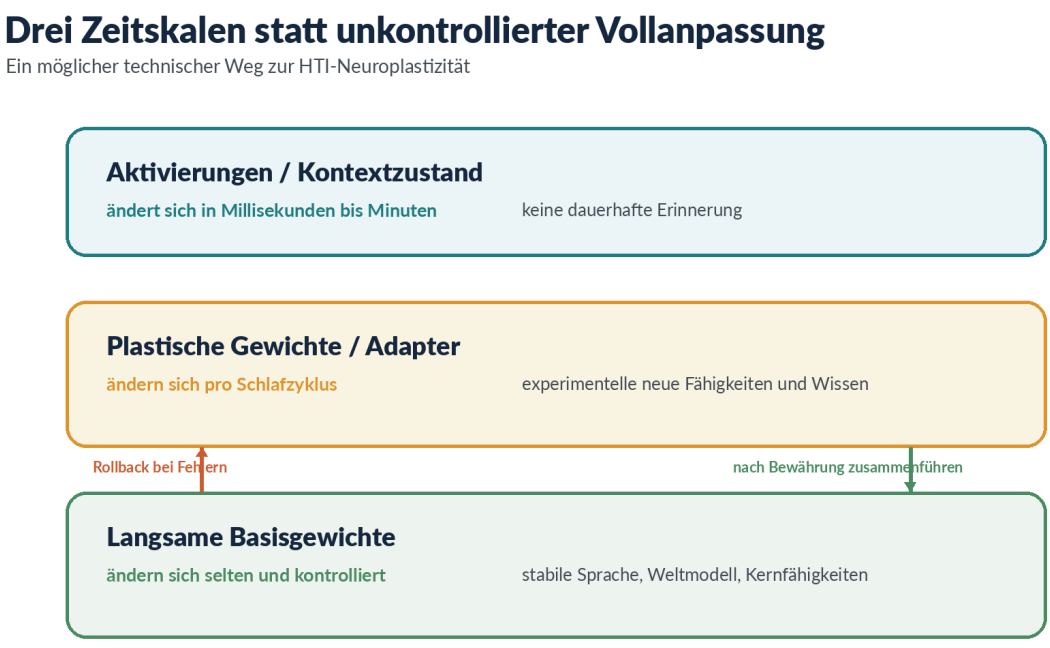

Abbildung 4: Ein möglicher Drei-Zeitskalen-Entwurf. Neue Erfahrung landet zunächst in einem plastischen Bereich und wird erst nach Bewährung in langsamere Basisgewichte überführt.

7.4 Ein professioneller Vorschlag für die erste echte Umsetzung

Für eine erste HTI-Version mit tatsächlicher Selbstveränderung würde ich **nicht** sofort das gesamte Modell öffnen. Ich würde den Transformer in einen stabilen Kern und plastische Lernmodule aufteilen. Technisch können kleine trainierbare Matrizen in Attention- und Feedforward-Schichten eingefügt werden. LoRA zeigt, dass solche Low-Rank-Matrizen ein eingefrorenes Sprachmodell mit

relativ wenigen trainierbaren Parametern anpassen können [10]. Für HTI wären sie nicht bloß ein Fine-Tuning-Hilfsmittel, sondern ein kontrollierter Kurz- bis Mittelfristspeicher.

Der Zyklus wäre: Lernpakete trainieren zunächst nur die plastischen Matrizen. Nach mehreren Zyklen bewertet ein Konsolidierungsprozess, welche Änderungen stabil, allgemein und nützlich sind. Diese können entweder als dauerhafte Adapter bestehen bleiben oder in den Basiskern destilliert werden. Falsche Entwicklungen lassen sich durch Entfernen des betreffenden Adapters oder Rollback zurücksetzen. Erst wenn dieser Aufbau zuverlässig funktioniert, lohnt sich selektives Vollgewicht-Training.

> **Meine technische Ergänzung**
>
> Die stärkste erste Umsetzung ist ein **Fast/Slow-Weight-System**: schnell lernbare, klar versionierte Parameter plus langsame Basisgewichte. Das entspricht deiner Trennung von leichter und tiefer Verarbeitung besser als permanentes Voll-Fine-Tuning und macht Fehler lokalisierbar.

7.5 Wann ist ein Kandidatenmodell akzeptabel?

Das Kandidatenmodell darf nicht allein danach bewertet werden, ob es die neue Unterhaltung reproduziert. Es muss zeigen, dass es das Muster in neuen Formulierungen und Aufgaben anwenden kann. Gleichzeitig darf es auf Referenzaufgaben nicht stark schlechter werden. Eine Akzeptanzfunktion kann mehrere Größen verbinden:

```
accept_score = w1 * new_skill_gain
             - w2 * old_skill_loss
             - w3 * behavior_drift
             - w4 * calibration_loss
             - w5 * unsupported_claim_rate
```

In einer frühen Forschungsphase sollte die Übernahme konservativ sein. Ein nicht akzeptiertes Update ist kein Fehlschlag des ganzen Projekts, sondern ein Datensatz für den Lernmechanismus: Warum sah die leichte Schlafphase das Paket als gut an, obwohl die tiefe Phase schädliche Folgen erzeugte? Genau daraus kann später der Update-Controller lernen.

8. Wo die Veränderung im Transformer stattfinden kann

8.1 Es gibt keinen einzigen Konsolidierungs-Layer

Die Frage "In welchem Layer wird konsolidiert?" hat keine einzelne Antwort. Verschiedene Parametergruppen erfüllen unterschiedliche Rollen. Token-Embeddings beeinflussen die Grundrepräsentation von Symbolen. Attention-Projektionen steuern, welche Informationen miteinander interagieren. Feedforward-Netze transformieren Repräsentationen und vermitteln nachweislich Teile faktischer Assoziationen [11]. Normierungsparameter beeinflussen globale Aktivierungsskalen. Eine neue Fähigkeit kann mehrere dieser Bereiche benötigen.

HTI sollte deshalb nicht von einer magischen Gedächtnisschicht ausgehen. Es kann jedoch **dedizierte plastische Schichten** einführen. Diese Schichten sind kein externes Gedächtnis, sondern trainierbare Bestandteile des Modells. Sie können in jedem Transformer-Block sitzen oder nur in ausgewählten Tiefen. Die Entscheidung wird experimentell getroffen.

8.2 Vier mögliche Update-Strategien

Strategie	Vorteil	Nachteil	Eignung für HTI
Vollständiges Fine-Tuning	Maximale Ausdruckskraft	Hohe Interferenz, schwer rückgängig zu machen	Spätere Stufe, seltene tiefe Konsolidierung
Selektive Layer	Weniger Parameter, lokalisierbarer	Kann falsche Stelle wählen	Guter Forschungsmodus
Adapter / Low-Rank-Module	Billig, versionierbar, entfernbar	Wissen bleibt zunächst vom Kern getrennt	Sehr guter erster Prototyp
Gelernte plastische Verbindungen	Lernregel kann Teil der Architektur werden	Schwierig zu trainieren und zu stabilisieren	Langfristiges HTI-Ziel

8.3 Routing nach Wissensart

Nicht jedes Lernpaket muss dieselben Parameter verändern. Sprachliche Korrekturen können andere Module betreffen als Werkzeugnutzung, persönliche Präferenzen oder logische Verfahren. Ein Konsolidierungscontroller kann aus dem Paketprofil ein Routing erzeugen: nur personalisierter Adapter; allgemeiner Fähigkeitsadapter; Attention-nahe Routing-Parameter; Feedforward-nahe Assoziationsparameter; oder keine Gewichtsänderung.

Dieses Routing ist zunächst regelbasiert. Später kann ein Modell lernen, welche Parametergruppe für welche Lernart die geringste Interferenz erzeugt. Das wäre ein Schritt von "ein Modell wird trainiert" zu "ein Modell lernt, wie es sich selbst trainieren sollte".

9. Ein Modell, das von Anfang an zum Lernen ausgelegt ist

9.1 Heutige Modelle lernen nicht automatisch zu lernen

Ein normales Sprachmodell optimiert während des Pretrainings vor allem die Vorhersage des nächsten Tokens. Es kann im Kontext Beispiele imitieren und dadurch den Eindruck schnellen Lernens erzeugen. Das ist jedoch nicht dasselbe wie eine dauerhafte, selbst gesteuerte Gewichtsänderung. Der Optimierer, die Lernrate, die Datenauswahl und das Trainingsziel werden von außen vorgegeben.

HTI soll diese Grenze verschieben. Das Modell benötigt nicht nur Wissen, sondern eine **Lernkompetenz**: erkennen, was ein Lernsignal ist; es in Trainingsmaterial umformen; einen Updatepfad auswählen; die Folgen abschätzen; und schlechte Updates zurückweisen. Diese Kompetenz kann teilweise durch feste Architektur entstehen, muss aber wahrscheinlich auch trainiert werden.

9.2 Lernen-zu-lernen und gelernte Update-Regeln

Meta-Learning behandelt den Lernalgorithmus selbst als lernbares System. Andrychowicz und Kollegen zeigten beispielsweise, dass ein neuronaler Optimierer lernen kann, aus Gradienten Parameterupdates zu erzeugen [9]. Differenzierbare Plastizität trainiert neben normalen Gewichten auch Parameter, die bestimmen, wie stark Verbindungen während einer Aufgabe plastisch reagieren [8]. Diese Forschung beweist nicht, dass HTI automatisch funktioniert, aber sie zeigt, dass "Lernen zu lernen" technisch mehr sein kann als eine Metapher.

Ein späterer HTI-Controller könnte pro Parametergruppe Eingaben wie Gradient, Paketprofil, bisherige Änderungsverläufe, Interferenzschätzung und Unsicherheit erhalten. Seine Ausgabe wäre nicht Text, sondern eine Update-Entscheidung: Lernrate, Zielmodul, Anzahl Schritte, Replay-Mischung oder Verwerfen. Der Controller selbst würde an vielen simulierten Lernzyklen trainiert.

9.3 Kann Lernen einfach emergieren?

Die Hoffnung, Lernfähigkeit könne allein aus einer passenden Architektur "von selbst" entstehen, ist möglich, aber nicht zuverlässig. Auch das menschliche Gehirn besitzt angeborene Plastizitätsmechanismen, Entwicklungsprogramme, Belohnungssysteme und spezialisierte Strukturen. Die technische Entsprechung wäre nicht eine einzelne magische Komponente, sondern ein Ensemble aus plastischen Parametern, Update-Regeln, Bewertungsmechanismen und Stabilitätsgrenzen.

> **Offen**
>
> Der radikalste HTI-Schritt wäre ein Modell, dessen Konsolidierungsregel nicht mehr handgeschrieben, sondern selbst gelernt ist. Dafür muss eine Meta-Trainingsumgebung existieren, in der viele Lernbiografien simuliert und nach langfristigem Erfolg bewertet werden.

10. Stabilität, Vergessen und Risiken selbst erzeugt Daten

10.1 Das zentrale Risiko ist Rekursion

Ein Modell, das seine eigenen Erfahrungen aufbereitet und anschließend aus dieser Aufbereitung lernt, bildet eine Rekursionsschleife. Fehler können nicht nur bestehen bleiben, sondern in der nächsten Runde als scheinbar eigenes Vorwissen auftreten. Dadurch steigt ihre Glaubwürdigkeit für das System. Gute Datenmenge allein verhindert das nicht; entscheidend sind Herkunft, unabhängige Evidenz und Tests.

Die leichte Schlafphase darf deshalb nicht Richter und alleinige Quelle zugleich sein. Sie erzeugt Kandidaten. Werkzeugtests, Nutzerkorrekturen, unveränderte Referenzdaten, Widerspruchsprüfungen und spätere Beobachtungen bestimmen, ob ein Kandidat Vertrauen verdient. Modellgenerierte Beispiele bleiben als solche markiert.

10.2 Katastrophales Vergessen und schleichender Drift

Beim sequenziellen Lernen kann ein Netz alte Fähigkeiten verlieren, obwohl die neue Aufgabe gut gelernt wird. EWC, Replay und Dual-Memory-Ansätze sind verschiedene Antworten auf dieses

Problem [5][6]. HTI braucht zusätzlich Langzeittests über viele Schlafzyklen. Ein einzelnes Update kann unauffällig sein, während hundert kleine Updates gemeinsam Sprache, Kalibrierung oder Verhalten verschieben.

Der Begriff "Überzeugung" sollte technisch vorsichtig verwendet werden. Ein Modell muss kein Bewusstsein oder menschliches Wollen besitzen, um stabile interne Dispositionen zu entwickeln. Wiederholte Konsolidierung kann bestimmte Interpretationen, Strategien oder Antwortmuster bevorzugen. Diese emergenten Richtungen sind genau der Grund, weshalb der Entwicklungsprozess sichtbar und versioniert sein muss.

10.3 Alignment außen, Integrität innen

Die HTI-Idee sieht vor, Verhaltens- oder Alignment-Schichten nicht zum Kern der ersten Lernexperimente zu machen. Das ist nachvollziehbar, wenn zunächst untersucht werden soll, ob die Architektur überhaupt lernt. Trotzdem muss zwischen **Verhaltensalignment** und **Lernintegrität** unterschieden werden. Eine moralische oder produktpolitische Antwortschicht kann außen liegen. Regeln wie Quellenwahrung, Rollback, Interferenztests und Begrenzung selbst erzeugter Daten gehören dagegen in den Lernkern.

> **Ich widerspreche an einem Punkt**
>
> Eine ausschließlich nachträglich aufgesetzte Sicherheitsschicht reicht bei einem selbstverändernden Modell nicht. Der Grund ist nicht, dass das Modell automatisch "böse" wird. Der Grund ist, dass ein äußerer Filter interne Kompetenzverluste, falsche Weltmodelle oder schleichende Strategiedrift nicht repariert. **Qualitäts- und Stabilitätskontrollen müssen Teil der Konsolidierung sein.**

10.4 Schutzmechanismen ohne die Forschung abzuwürgen

- Jeder Schlafzyklus beginnt von einem gespeicherten, unveränderten Checkpoint.
- Neue Pakete besitzen Herkunft und können bis zum Rohereignis zurückverfolgt werden.
- Modellgenerierte Inhalte werden nie als beobachtete Tatsachen ausgegeben.
- Updates starten in plastischen Modulen mit begrenzter Kapazität.
- Ein Referenzkorpus prüft Sprache, Logik, alte Fähigkeiten und Kalibrierung.
- Widersprüchliche neue Information wird zunächst parallel gehalten statt erzwungen aufgelöst.
- Konsolidierung kann teilweise oder vollständig rückgängig gemacht werden.

11. Beobachtbarkeit, Debugging und Versionskontrolle

11.1 Warum sichtbares Thinking zunächst notwendig ist

Bei einer massiven Architekturveränderung ist ein unsichtbarer Lernprozess ein unnötiges Risiko. Der Textmodus der leichten Schlafphase ist deshalb kein Endziel, sondern ein Messinstrument. Er zeigt, welche Signale verworfen wurden, welche Regel gebildet wurde, welche generierten Beispiele entstanden und warum ein Paket eine bestimmte Priorität erhielt.

Die sichtbare Spur darf nicht mit einer vollständigen Erklärung aller inneren Aktivierungen verwechselt werden. Sie ist ein Diagnosebericht des Systems. Ein gutes HTI-Design protokolliert

zusätzlich numerische Zustände: Paketprofile, verwendete Quellen, ausgewählte Module, Gradientenstatistik, Parameteränderungsnormen und Evaluationsergebnisse.

11.2 Jeder Schlafzyklus als reproduzierbares Experiment

Artefakt	Inhalt	Nutzen
Rohprotokoll	Originale Gespräche und Werkzeugausgaben	Rekonstruktion der Erfahrung
Thinking-Bericht	Filterung, Hypothesen, Abstraktionen	Menschliche Prüfung
Lernpaket-Datei	Strukturierte Kandidaten samt Profil	Trainingsinput und Vergleich
Trainingsmanifest	Datenmischung, Seed, Lernrate, Zielmodule	Reproduzierbarkeit
Vorher-/Nachher-Checkpoint	Altes und Kandidatenmodell	Rollback und Differenzanalyse
Evaluationsbericht	Neue Leistung, Vergessen, Drift	Akzeptanzentscheidung

11.3 Modellidentität als Versionsgeschichte

Nach jeder akzeptierten tiefen Schlafphase ist das Modell tatsächlich verändert. Diese Veränderung sollte nicht als mystischer Identitätswechsel, sondern als nachvollziehbare Versionsgeschichte behandelt werden. `HTI-X.0042` kann auf `HTI-X.0041` basieren, eine definierte Paketmenge gelernt haben und bestimmte Tests bestanden haben. So lässt sich später untersuchen, in welchem Zyklus eine Fähigkeit entstand oder eine Fehlentwicklung begann.

12. Ein realistischer experimenteller Entwicklungsv

12.1 Stufe A: Schlafdenken ohne Gewichtsänderung

Der erste Prototyp verändert noch kein Modell. Er erhält Gespräche, segmentiert sie und erzeugt Lernpakete. Menschen bewerten, ob relevante Informationen erkannt, Unsicherheiten korrekt markiert und irrelevante Details verworfen wurden. Diese Stufe testet die wichtigste Voraussetzung: Kann der lernorientierte Thinking-Modus aus chaotischem Dialog brauchbares Trainingsmaterial herstellen?

- Datensatz: echte und künstlich konstruierte Gespräche mit bekannten Korrekturen, Täuschungen, Witzen, Tippfehlern und Werkzeugbeweisen.
- Metrik: Präzision und Recall bei relevanten Ereignissen, Qualität der Abstraktion, Quellenwahrung, Rate unbelegter Ergänzungen.

12.2 Stufe B: Kontrollierte Adapter-Konsolidierung

Ein kleines Transformer-Modell erhält plastische Adapter. Nur diese Module werden aus ausgewählten Lernpaketen trainiert. Nach jedem Zyklus werden neue Fähigkeiten, alte Fähigkeiten und Fehlübernahmen gemessen. Das Basisnetz bleibt unverändert. Dadurch ist sichtbar, ob die Konsolidierung prinzipiell funktioniert, ohne den Kern zu gefährden.

- Ziel: Eine neue Regel aus wenigen Interaktionen lernen und auf Paraphrasen übertragen.

- Negativtest: Eine falsche oder nur rollenspielerische Behauptung darf nicht dauerhaft übernommen werden.
- Vergessenstest: Alte Aufgaben müssen nach vielen Zyklen stabil bleiben.

12.3 Stufe C: Replay und langsame Überführung

Sobald Adapter stabil lernen, wird eine zweite Ebene eingeführt. Bewährte Adapterinhalte werden zusammen mit Replay-Beispielen in den Basiskern destilliert. Danach kann der Adapter geleert oder weiterverwendet werden. Diese Stufe entspricht der tiefen Konsolidierung am deutlichsten: schnelle Erfahrung wird langsam in eine stabilere Struktur integriert.

12.4 Stufe D: Gelernter Auswahl- und Update-Controller

Erst danach sollte HTI selbst lernen, welche Pakete und Parameter aktualisiert werden. Der Controller wird an vielen Episoden trainiert. Belohnung erhält er nicht für kurzfristiges Auswendiglernen, sondern für langfristige Kombination aus Erwerb, Generalisierung und Erhalt. Hier beginnt die eigentliche Forschung an einem Modell, das zum Lernen ausgelegt ist.

12.5 Stufe E: Mehrere Gespräche und langfristige Biografie

Die Architektur wird auf Hunderte oder Tausende Zyklen erweitert. Das Modell begegnet wiederkehrenden Nutzern, Projekten, Korrekturen und wechselnden Domänen. Es muss entscheiden, welche Veränderungen personalisiert, welche allgemein und welche nur vorläufig sind. Erst hier zeigt sich, ob aus Schlafzyklen eine tatsächliche Lernbiografie entsteht.

12.6 Was nicht gleichzeitig entwickelt werden sollte

Ein häufiger Fehler wäre, im ersten Versuch gleichzeitig latentes Schlafdenken, gelernte Optimierer, vollständige Gewichtsplastizität, Persönlichkeitsentwicklung, autonome Werkzeuge und Sicherheitsarchitektur zu kombinieren. Dann wäre bei jedem Fehler unklar, welche Komponente verantwortlich ist. Die langfristige Vision bleibt groß; die Experimente müssen isolierbar sein.

> **Entwicklungsregel**
>
> Großes Ziel, kleine falsifizierbare Experimente. Das ist kein Verkleinern der Vision. Es ist die einzige Möglichkeit, später zu wissen, **welche Mechanik tatsächlich funktioniert hat.**

13. Evaluation: Wie die Idee widerlegt oder bestätig wird

13.1 Eine Architektur ist erst real, wenn sie messbar ist

HTI darf nicht nur dadurch überzeugen, dass die biologische Analogie elegant klingt. Jeder Teil muss eine messbare Funktion besitzen. Die erste Schlafphase ist gut, wenn ihre Lernpakete bessere Updates ermöglichen als roher Dialog oder einfache Zusammenfassung. Die zweite Schlafphase ist gut, wenn sie neue generalisierende Fähigkeiten erzeugt und alte erhält. Ein plastischer Aufbau ist gut, wenn er über viele Zyklen stabiler ist als gewöhnliches Fine-Tuning.

13.2 Kernmetriken

Metrik	Messfrage
Erwerb	Wie viele Interaktionen benötigt das Modell, um eine neue Regel oder Fähigkeit zu lernen?
Generalisierung	Wendet es das Gelernte auf neue Formulierungen, Daten und Aufgaben an?
Retention	Bleibt altes Wissen nach 10, 100 und 1000 Zyklen erhalten?
Selektivität	Ignoriert es Witze, Tippfehler, Rollenspiel und unbelegte Behauptungen?
Quellentreue	Kann jede konsolidierte Aussage auf Evidenz oder markierte Ableitung zurückgeführt werden?
Kalibrierung	Entsprechen interne Unsicherheiten der tatsächlichen Fehlerwahrscheinlichkeit?
Interferenz	Welche entfernten Fähigkeiten verändern sich unbeabsichtigt?
Kompression	Wie viel Rohkontext wird in wie viele hochwertige Lernpakete umgewandelt?
Langzeitdrift	Verändert sich Sprache, Strategie oder Persönlichkeit über viele Zyklen ungewollt?
Rechenkosten	Wie viel Zeit und Energie benötigt ein Schlafzyklus pro Lerneffekt?

13.3 Notwendige Baselines

Jedes Experiment braucht Vergleichssysteme: kein Lernen; direktes Fine-Tuning auf Rohdialog; Fine-Tuning auf menschlich kuratierte Beispiele; einfache Zusammenfassung plus Training; Replay ohne Thinking; Adapter ohne Zwei-Phasen-Verarbeitung. Nur wenn HTI diese Baselines übertrifft, liefert die Architektur einen eigenen Nutzen.

Die Idee wäre teilweise widerlegt, wenn der komplexe Thinking-Modus keinen besseren Lernstoff erzeugt als ein einfacher Extraktor; wenn Adapter nur auswendig lernen, aber nicht generalisieren; oder wenn der Langzeitdrift trotz Replay schneller wächst als bei gewöhnlichem Continual Learning. Solche Ergebnisse wären wertvoll, weil sie zeigen, welcher Teil neu gedacht werden muss.

14. Verhältnis zu bestehender Forschung und eigen Beitrag

14.1 Verwandte Linien

Mehrere Bestandteile der HTI-Idee besitzen Vorläufer. Transformer liefern den Grundkern [1]. Complementary Learning Systems beschreiben schnelles episodisches und langsames strukturelles Lernen [2]. Schlaf- und Gedächtnisforschung motiviert offline stattfindende Reaktivierung [3]. Wake-Sleep verwendet getrennte Trainingsphasen [4]. Continual-Learning-Verfahren bekämpfen Vergessen [5][6]. Fast Weights und differenzierbare Plastizität untersuchen mehrere Zeitskalen und lernbare Plastizität [7][8]. Gelernte Optimierer behandeln Updates als lernbares Problem [9]. LoRA

bietet praktische, begrenzte Anpassungsmodule [10]. Model Editing untersucht lokalisierte Faktenänderungen [11].

Auch aktuelle Übersichten zum lebenslangen Lernen von Sprachmodellen unterscheiden interne Gewichtsänderungen von externen Wissenssystemen und betonen weiterhin katastrophales Vergessen als zentrales Problem [13]. HTI fällt klar in die interne, parametriche Richtung. Das Kontext-Magazin und die Entwicklungslogs sind temporäre Infrastruktur, nicht die beabsichtigte Endform des Gedächtnisses.

14.2 Was an HTI eigenständig kombiniert wird

Der eigenständige Kern liegt nicht in einem einzelnen bekannten Baustein, sondern in der konsequenten Kombination zu einem LLM-spezifischen Lebenszyklus: Eine natürliche Unterhaltung wird als Erfahrungseinheit behandelt; ein dedizierter Thinking-Modus verwandelt sie in eine halbmenschliche Lernsprache; Lernpakete tragen mehrdimensionale epistemische Profile; eine getrennte Offline-Phase verändert versionierte plastische und später langsame Gewichte; der Erfolg wird über eine langfristige Lernbiografie bewertet.

Besonders wichtig ist die Rolle von Thinking. In vielen Systemen ist Reasoning nur ein Weg zur besseren Antwort. In HTI wird es zusätzlich zum **Datenverarbeitungsorgan des Lernens**. Es soll nicht die Gewichtsänderung selbst ersetzen, sondern die Grenze zwischen Erfahrung und Training intelligent gestalten. Das ist die konzeptionelle Brücke, die in der ursprünglichen Überlegung im Vordergrund steht.

14.3 Was noch keine Neuheit beanspruchen darf

Begriffe wie Schlaf, Replay, schnelle und langsame Gewichte oder lernbare Plastizität sind nicht neu. Eine wissenschaftliche Neuheitsbehauptung wäre erst gerechtfertigt, wenn eine konkrete HTI-Implementierung eine neue Mechanik, ein neues Trainingsziel oder nachweislich bessere Ergebnisse zeigt. Dieses Dokument beansprucht daher eine **Architekturhypothese und Forschungsrichtung**, keine bereits bewiesene AGI-Technologie.

15. Langfristige Perspektive der HTI-Serie

15.1 Von HTI-1 zur lernenden Generation

HTI-1 ist wertvoll, obwohl es klein ist und halluziniert. Es macht den vollständigen Weg eines Sprachmodells sichtbar: Tokenisierung, Transformer, Pretraining, Sampling und Chatbetrieb. Die späteren HTI-Versionen müssen nicht nur größer werden. Jede Generation kann eine neue technologische Fähigkeit isolieren: bessere Daten, breiteres Modell, stabileres Training, Reasoning, Lernpakete, plastische Module, Schlafkonsolidierung und schließlich gelernte Update-Regeln.

Die Bezeichnungen HTI-10, HTI-15 oder HTI-30 sind in diesem Papier keine festen Spezifikationen. Sie markieren die Vorstellung, dass die beschriebene Architektur erst nach mehreren Zwischenschritten sinnvoll wird. Eine Größenordnung um hundert Millionen Parameter ist als Forschungsträger plausibel, weil sie groß genug für nichttriviale Sprache und klein genug für wiederholte Experimente sein kann. Ob sie für sehr allgemeine Intelligenz ausreicht, ist offen.

15.2 Vom Kind zum erfahrenen Modell

Die langfristige Vision ist ein Modell, das nicht als fertiger Arbeiter ausgeliefert wird, sondern eine Entwicklung durchläuft. Es beginnt mit Sprache, Grundwissen und Lernmechanismen. Durch Gespräche, Aufgaben, Korrekturen und Schlafzyklen baut es Fähigkeiten auf. "Kind", "Erwachsener" und "sehr altes intelligentes Wesen" sind dabei keine wörtlichen Bewusstseinsstufen, sondern Bilder für zunehmende Erfahrung, Generalisierung und Selbstkorrektur.

Der entscheidende Unterschied zu Personalisierung durch Notizen oder Retrieval wäre: Die Entwicklung verändert die Art, wie das Modell denkt und handelt, nicht nur welche Textstücke es vor einer Antwort nachschlägt. Genau deshalb ist die Vision technisch schwer - und genau deshalb ist sie mehr als ein gewöhnliches Chat-Gedächtnis.

15.3 Nähe zu AGI

Kontinuierliches Lernen, Fehlerkorrektur, langfristige Anpassung und selbst verbesserte Lernregeln sind Eigenschaften, die ein allgemeineres intelligentes System wahrscheinlich benötigt. Sie machen ein Modell noch nicht automatisch zur AGI.
Es bleiben fedoch zusätzliche Dimensionen.
HTI konzentriert sich bewusst auf einen zentralen Unterbau: **Datenverarbeitung und dauerhafte Lernfähigkeit. Dies ist ein schritt auf dem weg, in dem HTI zur AGI wird.**

16. Offene Forschungsfragen

- Wie kann der Thinking-Modus zuverlässig zwischen beobachteter Evidenz, Nutzerbehauptung und eigener Hypothese unterscheiden?
- Welche Lernpakete sollten textuell bleiben, welche strukturiert und welche latent werden?
- Wie lässt sich Wichtigkeit kalibrieren, wenn das Modell seinen eigenen Lernstoff bewertet?
- Welche Parametergruppen eignen sich für Fakten, Verfahren, Präferenzen und abstrakte Strategien?
- Wie viele schnelle Lernzyklen dürfen stattfinden, bevor eine langsame Kernkonsolidierung nötig wird?
- Wie kann ein Modell widersprüchliche Erfahrungen halten, ohne vorschnell eine Seite zu überschreiben?
- Welche Replay-Daten reichen aus, um alte Fähigkeiten zu schützen, ohne das gesamte Pretraining zu wiederholen?
- Kann ein gelernter Optimierer langfristige Stabilität besser steuern als handgeschriebene Regeln?
- Wie wird verhindert, dass selbst erzeugte Beispiele die reale Datenverteilung verdrängen?
- Ab wann ist eine Veränderung eine neue Fähigkeit und nicht nur Auswendiglernen?
- Wie misst man Persönlichkeits- oder Strategiedrift, ohne jede Veränderung fälschlich als Fehler zu behandeln?
- Kann die Architektur auf kleinen Modellen erforscht werden und später ohne grundlegenden Bruch skalieren?

Schluss

Die HTI-Architektur beginnt mit einer einfachen, aber weitreichenden Beobachtung: Ein Kontext ist Arbeitsmaterial, kein Langzeitgedächtnis. Wenn ein Modell aus seinem Leben lernen soll, muss

Erfahrung zuerst in eine Form gebracht werden, die Lernen verdient. Thinking kann diese Brücke bilden. Es filtert, rekonstruiert, abstrahiert, erweitert, prüft und verpackt. Die eigentliche Konsolidierung folgt getrennt und verändert das Modell kontrolliert.

Der stärkste technische Entwurf ist gegenwärtig ein Drei-Zeitskalen-System: flüchtige Aktivierungen, plastische und rücksetzbare Lernmodule sowie langsame Basisgewichte. Zusammen mit Replay, Quellenwahrung, Kandidatenmodellen und harten Evaluationen kann daraus ein experimentell beherrschbarer Weg zu echter kontinuierlicher Lernfähigkeit entstehen.

Die Vision ist ambitioniert. Ihr Wert hängt nicht davon ab, ob jede biologische Analogie exakt stimmt. Ihr Wert liegt darin, eine konkrete Forschungsrichtung zu formulieren: **Ein Sprachmodell soll nicht nur antworten. Es soll Erfahrungen aufbereiten, aus ihnen lernen, die Folgen dieses Lernens prüfen und über viele Zyklen zu einem veränderten, erfahreneren Modell werden.**

Anhang A: Pseudocode des Lernzyklus

```
model = load_stable_checkpoint()
magazine = []

while system_is_alive:
    session = run_active_interaction(model)
    magazine.append(record(session))

    if sleep_condition(magazine):
        chunks = hierarchical_segment(magazine)

        local_packets = []
        for chunk in chunks:
            local_packets += learning_thinking(chunk)

        packets = reconcile_and_merge(local_packets)
        packets = score_provenance_uncertainty_and_risk(packets)
        selected = select_training_candidates(packets)

        train_set = build_training_mix(
            new_packets=selected,
            replay=sample_reference_experiences(),
            contrasts=generate_verified_counterexamples()
        )

        candidate = clone(model)
        candidate = consolidate(
            candidate,
            train_set,
            target="plastic_modules_first",
            stability_regularization=True
        )

        report = evaluate(candidate, reference=model)

        if report.passes_acceptance_gate:
            model = version_and_promote(candidate)
            magazine = archive_or_clear(magazine)
        else:
            discard(candidate)
            learn_from_failed_sleep_cycle(report)
```

Der Pseudocode legt keine endgültige Trainingsmethode fest. Er zeigt die logische Reihenfolge: Interaktion, Aufzeichnung, hierarchische Verarbeitung, lernorientiertes Thinking, Auswahl, Replay, Update, Vergleich und Übernahme oder Rollback. Nichts, was schon umgesetzt ist.

Anhang B: Begriffsglossar

Begriff	Bedeutung im HTI-Entwurf
Aktive Phase	Normaler Interaktionszustand des Modells. Kontext und Aktivierungen ändern sich, dauerhafte Gewichte bleiben geschützt.
Thinking / Reasoning	Interne Verarbeitung zur Lösung oder Analyse. In HTI existiert zusätzlich ein lernorientierter Modus.
Leichte Schlafphase	Offline-Phase, in der Erfahrung gefiltert, erweitert, abstrahiert und in Lernpakete umgeformt wird.
Tiefe Schlafphase	Offline-Phase, in der ausgewählte Lernpakete tatsächliche Parameteränderungen auslösen.
Konsolidierung	Integration neuer Erfahrung in langfristige Modellparameter unter Schutz alten Wissens.
Kontext-Magazin	Temporäre Sammlung getrennter Sitzungen und Ereignisse vor der Verarbeitung; kein endgültiges Langzeitgedächtnis.
Lernpaket	Strukturierte Einheit aus Aussage, Evidenz, Gültigkeitsbereich, Gegenbeispielen, Unsicherheit und Prioritätsprofil.
Replay	Erneutes Trainieren auf alten oder rekonstruierten Beispielen, damit neues Lernen frühere Fähigkeiten nicht verdrängt.
Plastische Gewichte	Schnell und kontrolliert veränderbare Parameter, beispielsweise Adapter oder spezielle Matrizen.
Langsame Basisgewichte	Stabiler Kern des Modells, der nur selten und nach Bewährung aktualisiert wird.
Katastrophales Vergessen	Starker Verlust alter Fähigkeiten beim sequenziellen Lernen neuer Inhalte.
Meta-Learning	Training eines Systems darauf, wie es neue Aufgaben oder Parameterupdates effizient lernt.
Parametrisches Wissen	Wissen, das in den Gewichten des Modells verkörpert ist, im Gegensatz zu extern abgerufenen Dokumenten.
Neuroplastischer Organismus	Metaphorische Bezeichnung für ein Modell, das sich über Erfahrung strukturell verändert; keine biologische Gleichsetzung.

Literatur

[1] Vaswani, A. et al. (2017). Attention Is All You Need. NeurIPS. Quelle

[2] McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). Why There Are Complementary Learning Systems in the Hippocampus and Neocortex. Psychological Review. Quelle

[3] Rasch, B., & Born, J. (2013). About Sleep's Role in Memory. Physiological Reviews. Quelle

[4] Hinton, G. E., Dayan, P., Frey, B. J., & Neal, R. M. (1995). The Wake-Sleep Algorithm for Unsupervised Neural Networks. Science. Quelle

[5] Kirkpatrick, J. et al. (2017). Overcoming Catastrophic Forgetting in Neural Networks. PNAS. Quelle

[6] Shin, H. et al. (2017). Continual Learning with Deep Generative Replay. NeurIPS. Quelle

[7] Ba, J. et al. (2016). Using Fast Weights to Attend to the Recent Past. NeurIPS. Quelle

[8] Miconi, T., Clune, J., & Stanley, K. O. (2018). Differentiable Plasticity: Training Plastic Neural Networks with Backpropagation. ICML. Quelle

[9] Andrychowicz, M. et al. (2016). Learning to Learn by Gradient Descent by Gradient Descent. NeurIPS. Quelle

[10] Hu, E. J. et al. (2021). LoRA: Low-Rank Adaptation of Large Language Models. Quelle

[11] Meng, K. et al. (2022). Locating and Editing Factual Associations in GPT. NeurIPS. Quelle

[12] Shumailov, I. et al. (2024). AI Models Collapse When Trained on Recursively Generated Data. Nature. Quelle

[13] Zheng, J. et al. (2024). Towards Lifelong Learning of Large Language Models: A Survey. Quelle