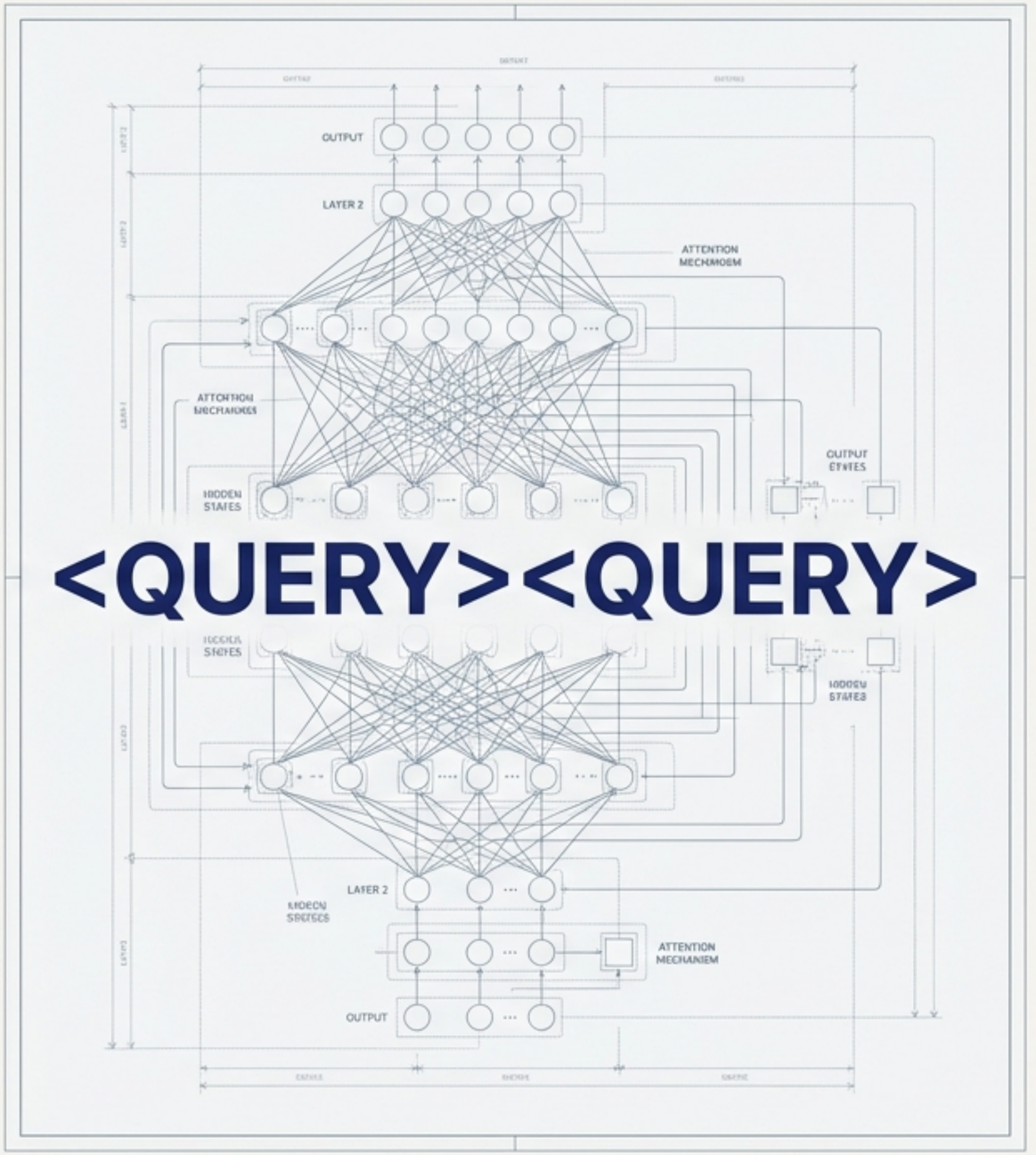
# LLM推論における「単純さ」のパラドックス

プロンプト反復（Prompt Repetition）がハッキングする「因果的注意」の物理的限界




Based on 'Prompt Repetition Improves Non-Reasoning LLMs' by Leviathan et al. (Google Research, 2025)' in Noto Sans JP

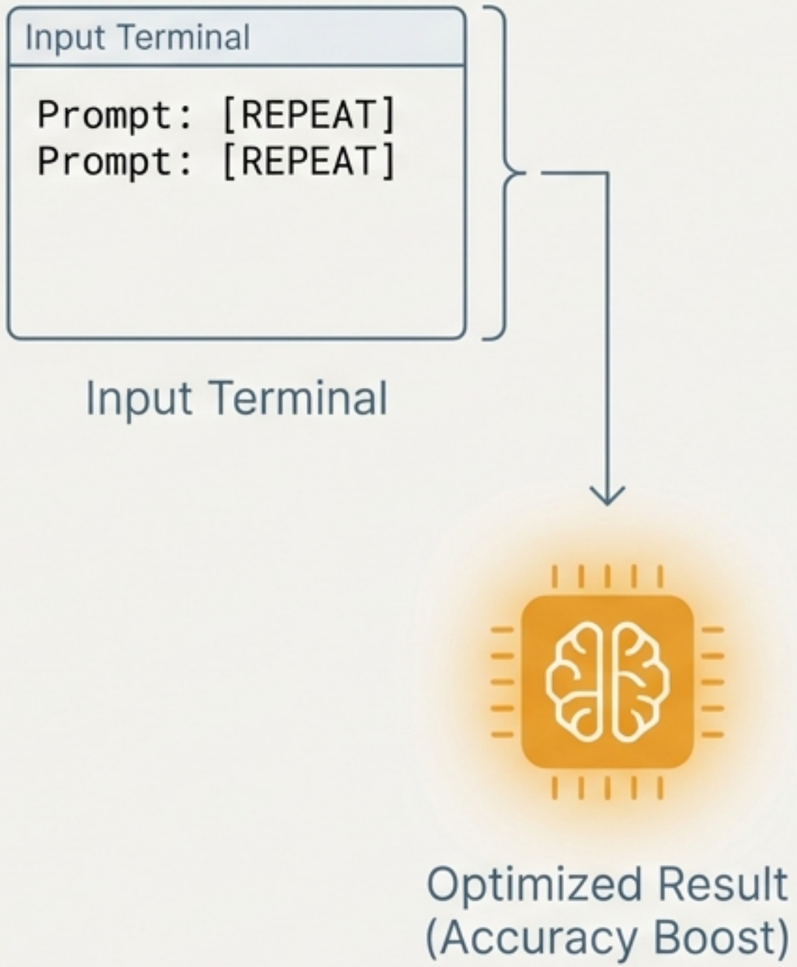
# 2025年の衝撃：AIエンジニアリングへの「原点回帰」

数十億パラメータの追加学習や複雑なエージェント構造が主流となる中、 Google Researchは驚くべき発見をしした。

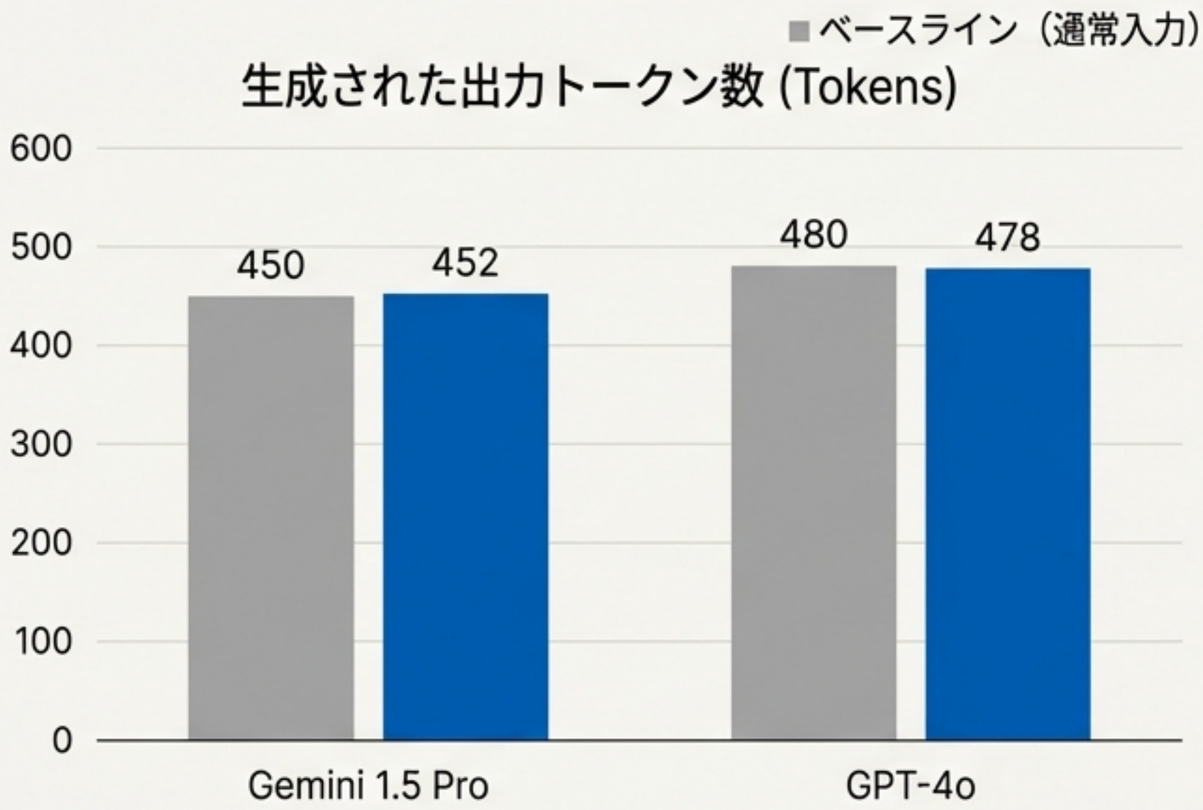
**The Discovery:** 入力プロンプトを単に「2回繰り返す」だけで、モデルの構造を変えずに精度が劇的に向上する。

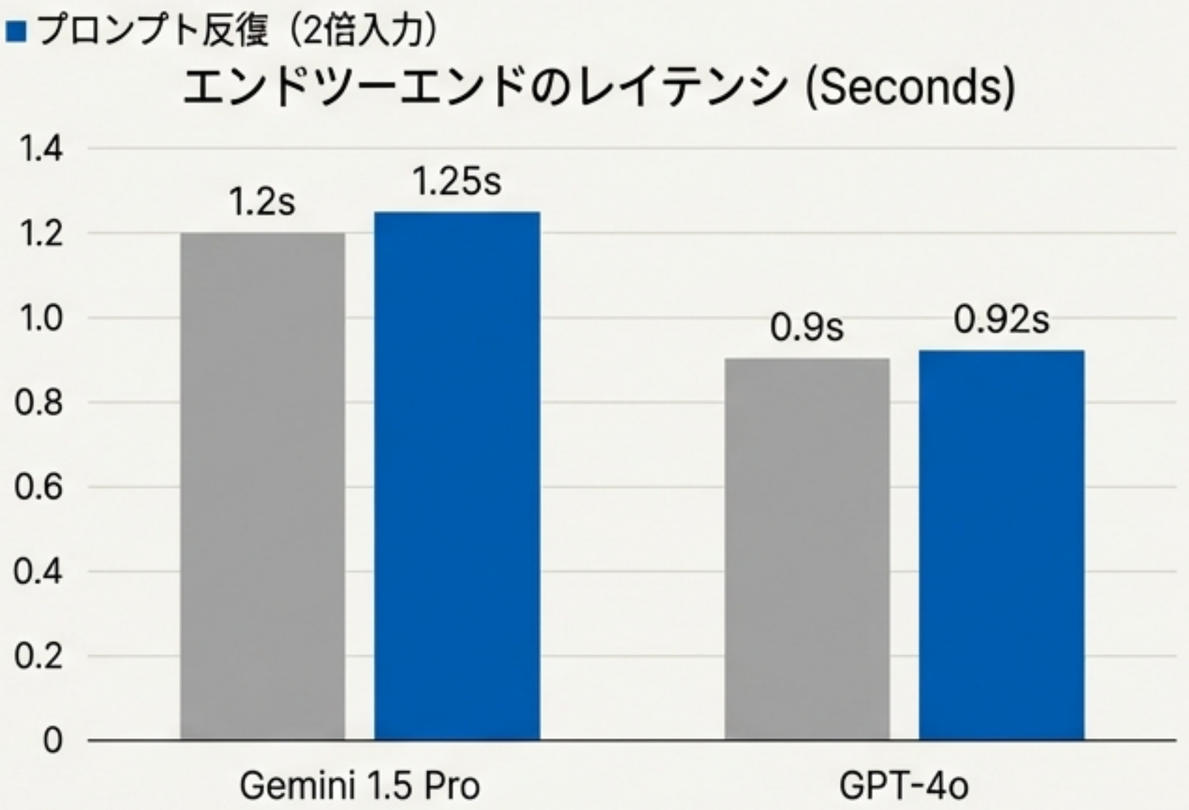
**Scope:** Gemini 2.0、GPT-4o、Claude 3.5 Sonnetなど、最先端のフラッグシップモデルで有効性を確認。

**Key Insight:** これは単なるプロンプトエンジニアリングの小技ではなく、Transformerアーキテクチャの構造的欠陥（Bug）を突いたハックである。

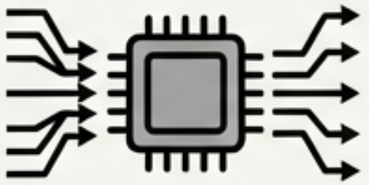


# 「フリーランチ」の経済学：入力倍増でも遅延はゼロ



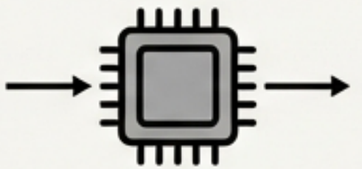




## Prefill（並列処理）



入力処理はGPUで超並列化。計算時間の増加は数ミリ秒。

## Decode (直列処理)



ボトルネックとなる回答生成時間は不変。

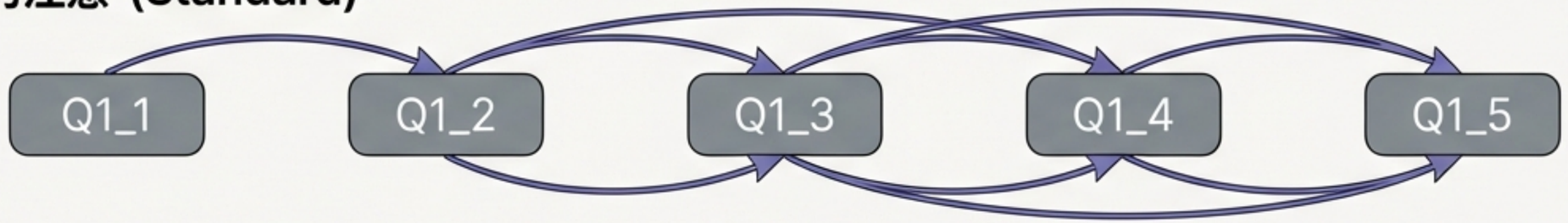
# 因果律の呪縛：モデルは「未来」を見ることができない


**Causal Masking (因果的マスキング):** LLMは「次の単語」を予測する際、物理的に未来のトークンを参照できない。
**The Problem:** <前提条件A> を処理している瞬間、モデルはまだ <質問> の内容を知らない。
**Result:** 文脈全体を把握しないまま、曖昧な状態で初期トークンの埋め込み表現が確定してしまう。

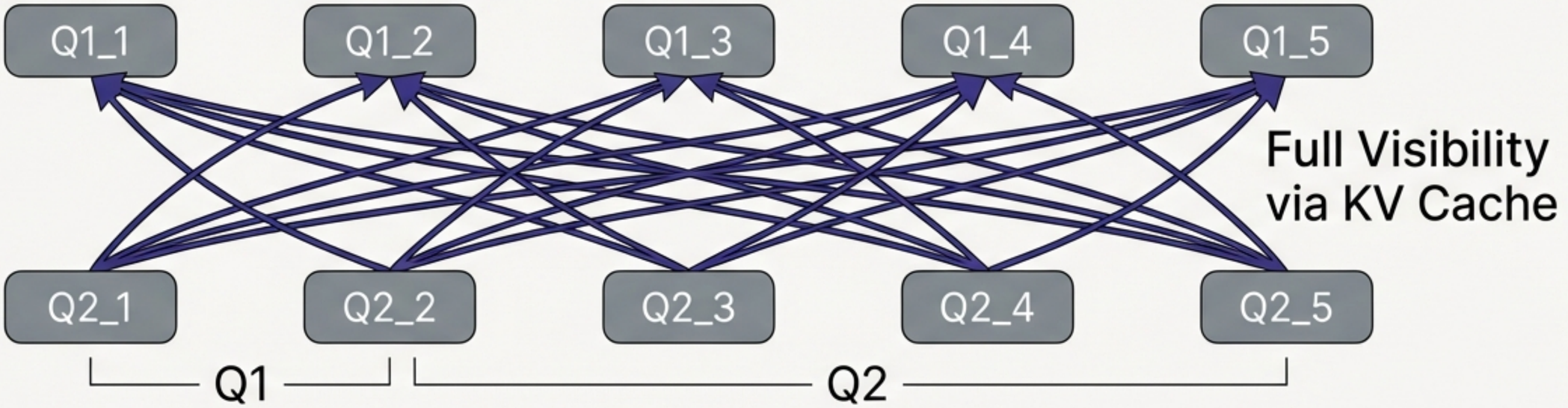
# プロンプト反復による「擬似的な双方向性」の獲得

Helvetica Now Display/Inter と Noto Sans JP

**標準的因果的注意 (Standard)**




**プロンプト反復注意 (Prompt Repetition)**




**Pass 1 (QUERY 1):** 未来を知らないまま読み進める（不完全な理解）。

**Pass 2 (QUERY 2):** KVキャッシュ内の「過去の記憶」として全文を参照可能。実質的に「双方向注意」が実現される。

# 劇的改善の証拠：「NameIndex」タスク


**Task:** 50人の名前リストから、25番目の名前を特定せよ

~~21.33%~~

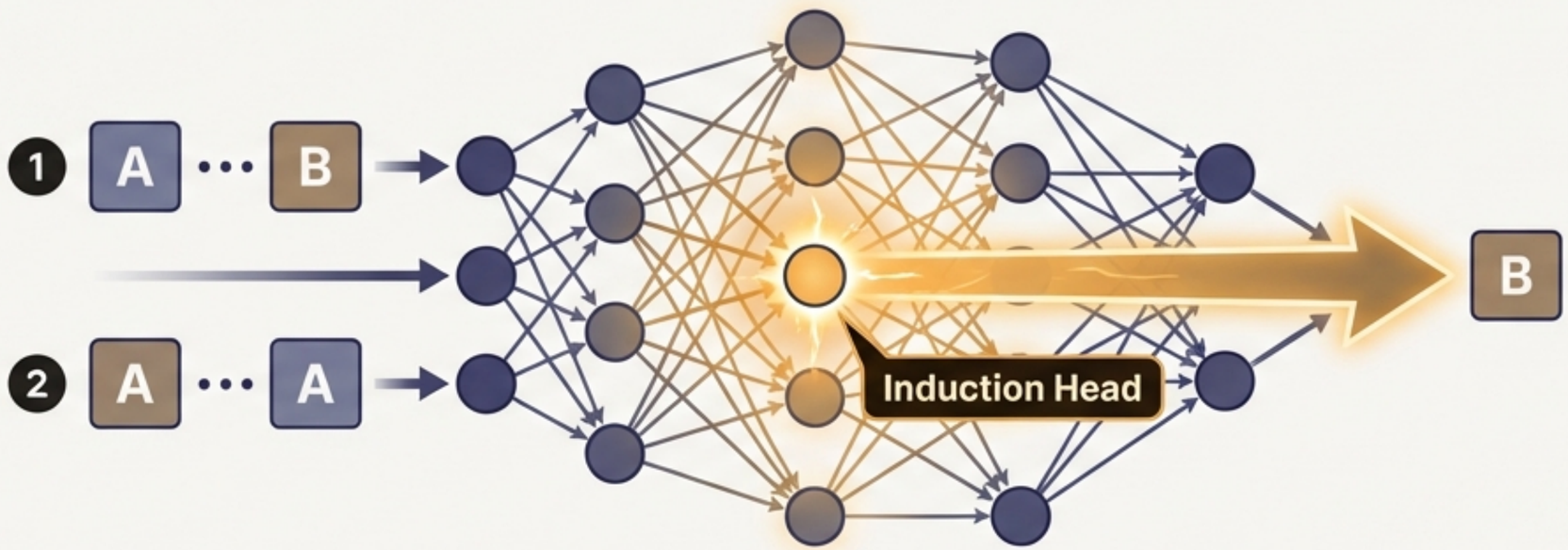
Baseline Accuracy

**97.33%**

With Repetition

- **Lost in the Middle:** 通常、モデルはリストの途中で位置情報を見失う。
- **The Fix:** 2回目は「25番目を探す」という目的（Query）を持った状態で、KVキャッシュ内のリストを再走査（Retrieve）できる。

# インダクションヘッド（Induction Heads）の物理的活性化




**Induction Headとは:** 「以前 A→B というパターンがあった。今 A が来たから、次は B だ」と予測・コピーする回路。

**Mechanism:** プロンプト反復は、物理的に「繰り返しのパターン」を入力内に作り出し、この回路を強力に発火（Fire）させる。

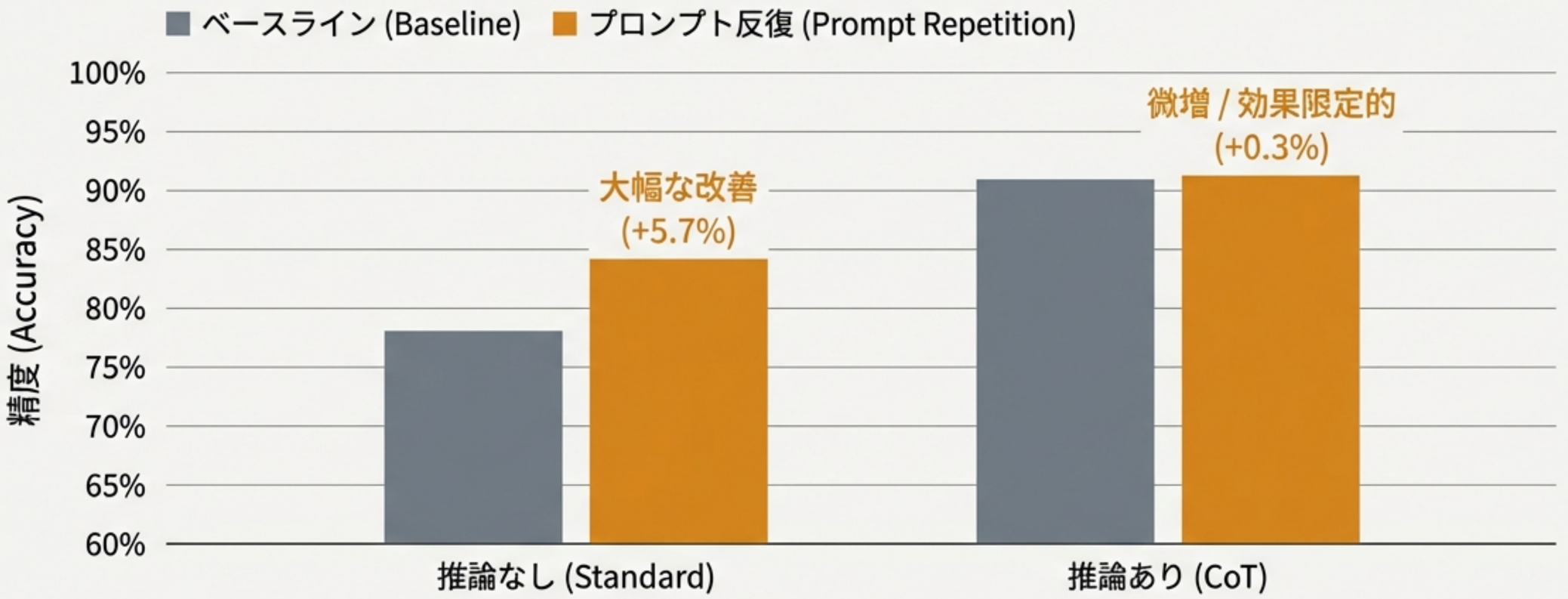
**Result:** 前のプロンプトにある重要な情報を、物理的に「コピー&ペースト」する能力が最大化される。

# 47勝0敗の実績と、明確な境界線

| Non-Reasoning Tasks (Extraction, RAG) | Reasoning Tasks (Math, Logic) |
| --- | --- |
| • Document QA<br>• Information Extraction<br>• Classification | • GSM8K (Math)<br>• Logic Puzzles<br>• Code Generation |
|  **47 WINS / 0 LOSSES** |  **NEUTRAL** |
| 統計的に有意な勝利。敗北なし。 | ほとんど効果なし（Neutral）。 |

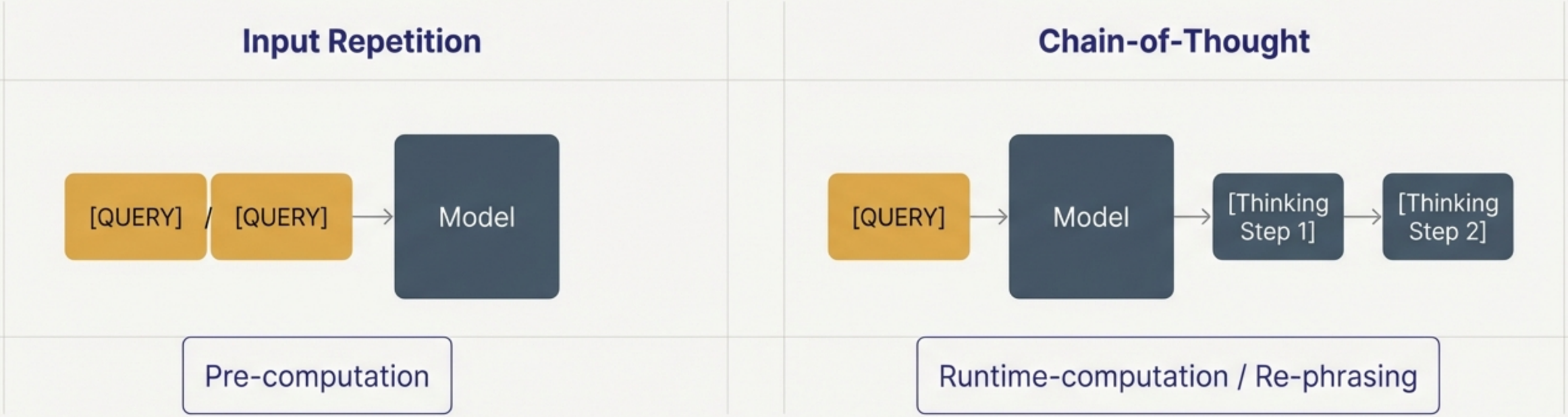
Data source: Google Research Benchmarks on Gemini 2.0 & GPT-4o.

# 「推論」の壁：思考プロセスが反復を無効化する




**Reasoning vs. Recall::** 数学に必要なのは情報の「場所の特定（Locating）」ではなく、「変換・操作（Transformation）」。
反復は「数字を見つける」ことは助けるが、「方程式の解き方」を教えるわけではない。

# Chain-of-Thought (CoT) は「動的な反復」である

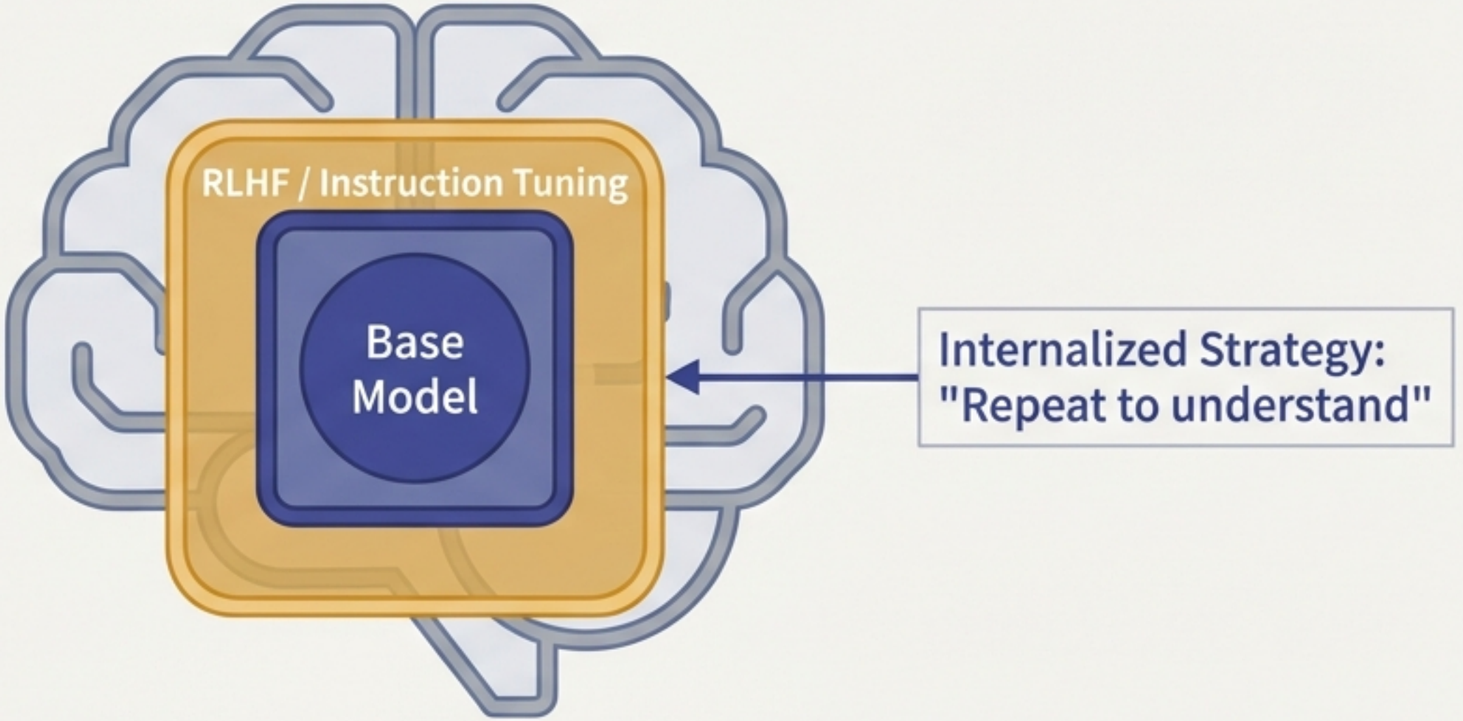



**Key Insight:**

📓 **The Saturation Point:** CoTによってモデルはすでに内部的に情報を反復・定着させている。

📈 「満腹」の状態にあるモデルに、さらに入力で反復（食事）を与えても、追加の効果は得られない（Diminishing returns)。

# 強化学習（RL）によって内在化された反復




- **Internalized Repetition:** RLHFを経たモデルは、ユーザーの指示を復唱・確認する戦略をすでに学習している。
- OpenAI o1のような推論モデルは、「隠れた思考（Hidden Chain of Thought）」の中で何度も条件を反芻しているため、外部からの反復は冗長になる。

# 実装ガイドライン：いつ「反復」すべきか？

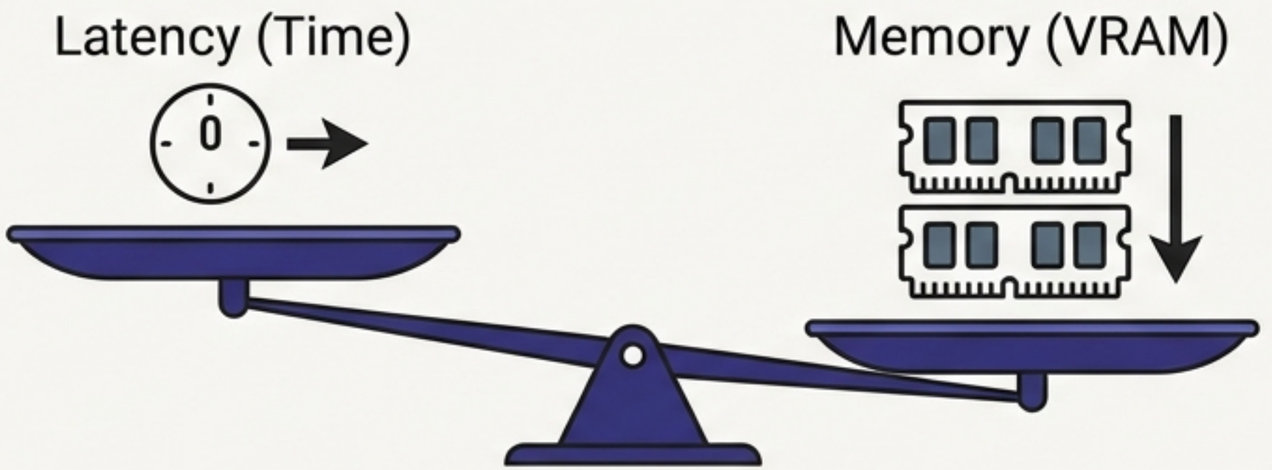
**YES (Use Repetition)**
- RAG / Document QA (正確な引用)
- Extraction (情報抽出)
- Strict Formatting (JSON遵守)

**NO (Use CoT)**
- Math / Logic (GSM8K)
- Creative Writing (自由な発想)
- Multi-turn Chat (履歴あり)

**Extraction Tasks:**
Use <Q><Q> for free accuracy boost.

**Reasoning Tasks:**
Use `Let's think step by step`.

# エンジニアリングの代償：VRAMとのトレードオフ




- **Latency**: 変化なし (Parallel Prefill)。
- **Cost**: トークン課金は2倍 ($$)。
- **Risk**: 長文コンテキストでの OOM (Out Of Memory) エラー。

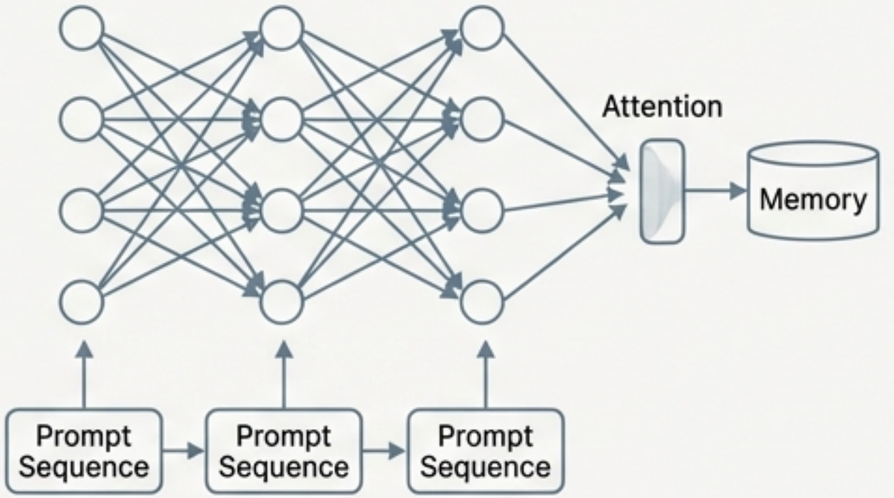
**Future Tech**: Attention Sinkや共有キャッシング技術による最適化が必須。

# 結論：AIとの対話における「再読」の価値

## Human Recall

繰り返す（Repeat）。情報の解像度を高める。

## AI Recall




考えさせる（CoT）。論理を構築する。

**To Recall (想起):** 繰り返す（Repeat）。情報の解像度を高める。
**To Reason (思考):** –考えさせる（CoT）。論理を構築する。

**最新のAIであっても、人間と同じように「読み落とし」をするし、「読み返す」ことで理解が深まる。**
**プロンプトエンジニアリングは物理的制約を考慮した「システム設計」へと進化している。**