What Temporal Speedup Reveals About Frontier Model Architectures

Evidence from Multi-Model Debugging Experiments

Anand Karasi

DisruptWithAI Research · San Jose, CA

https://disruptwithai.com

January 2026

Abstract

We present a cross-model analysis of temporal speedup dynamics observed during a controlled debugging experiment across three frontier language models (GPT-5.2, Claude Opus 4.5, Gemini 3 Flash). While all models achieve near-perfect accuracy ($\geq$99.6%) on 280 standardized Python debugging tasks, their *latency response* to accumulated temporal context diverges dramatically: GPT-5.2 accelerates by 6.7%, Opus 4.5 remains flat (+0.2%), and Gemini 3 Flash decelerates by 17.0%. We argue that this divergence constitutes a behavioral fingerprint of each model's internal reasoning architecture—revealing three fundamentally different inference strategies. We introduce the **Capability Boundary Hypothesis** and the **Context Utilization Spectrum** as theoretical frameworks for predicting when temporal intelligence helps, hurts, or has no effect. Our findings have direct implications for the design of temporally-aware AI systems and challenge the assumption that more context universally improves performance.

Keywords: temporal intelligence, language model architecture, in-context learning, debugging automation, inference optimization, capability boundary

Contents

1 Introduction

The dominant paradigm in AI system design assumes that providing models with more relevant context improves performance. Retrieval-augmented generation (RAG), chain-of-thought prompting, and few-shot exemplars all rest on this premise. But what happens when the context is *temporal*—accumulated experience from sequential task execution—and the model is already highly capable?

This paper examines a surprising finding from our temporal intelligence experiments: **the same temporal context that accelerates one frontier model actively degrades another, while leaving a third entirely unaffected.** This three-way divergence cannot be explained by differences in model capability (all achieve $\geq$99.6% accuracy) or task difficulty (all face identical tasks). Instead, it reveals fundamental architectural differences in how these models process, prioritize, and utilize contextual information during inference.

We tested three frontier models—OpenAI GPT-5.2, Anthropic Claude Opus 4.5, and Google Gemini 3 Flash—on 280 Python debugging tasks under two conditions: stateless (no accumulated context) and temporal (accumulated debugging patterns carried forward). The results expose what we call the **Context Utilization Spectrum**: a continuum from context-as-accelerant (GPT-5.2) to context-as-noise (Gemini 3 Flash), with context-as-irrelevant (Opus 4.5) in between.

1.1 Contributions

1. **Empirical evidence** that temporal context produces model-dependent effects spanning a 23.7 percentage-point range (from +6.7% to −17.0%) across frontier models of comparable capability.

2. **The Capability Boundary Hypothesis:** temporal intelligence provides maximum benefit at the boundary of a model's capability, and is redundant for models operating well within their frontier.

3. **The Context Utilization Spectrum:** a framework classifying models by their inference response to accumulated context—exploitative, invariant, or susceptible.

4. **Architectural inferences** about the internal reasoning strategies of three frontier models, derived purely from behavioral speedup signatures.

5. **Practical guidelines** for practitioners building temporally-aware AI systems.

2 Related Work

2.1 In-Context Learning

Brown et al. (2020) demonstrated that large language models can learn from examples provided in-context, a phenomenon later formalized by Xie et al. (2022) as implicit Bayesian inference. Our work extends this to *sequential* in-context learning, where the context is not curated examples but accumulated operational experience.

2.2 Temporal Reasoning in AI

Temporal reasoning has been studied primarily in the context of event ordering, temporal logic, and planning (Zhou et al., 2019; Vashishtha et al., 2020). Our notion of "temporal intelligence" differs: we examine whether models can leverage *their own past performance history* to improve future performance—a form of meta-cognitive temporal awareness.

2.3 Model Architecture Comparisons

Comparative evaluations of frontier models typically focus on accuracy benchmarks (Zheng et al., 2023; Chiang et al., 2024). Our contribution is orthogonal: we hold accuracy approximately constant and examine *latency dynamics* as a window into architectural differences.

3 Experimental Design

3.1 Task Battery

We constructed 280 Python debugging tasks across three difficulty tiers:

- **Easy (100 tasks):** Single-bug programs—off-by-one errors, null reference failures, type coercion, missing returns, incorrect boolean logic, string immutability violations. Each 10–25 lines with 3–5 test cases.
- **Medium (100 tasks):** Python-specific behavioral quirks—mutable default arguments, closure late-binding, modify-while-iterating, dictionary iteration errors, bare exception handling, infinite recursion. Each 15–40 lines with 4–7 test cases.
- **Hard (80 tasks):** Multi-concept programs involving concurrency, data structures, and systems—thread safety violations, graph traversal bugs, BST insertion errors, LRU cache eviction, deadlock scenarios, async concurrency patterns. Each 30–80 lines with 5–10 test cases.

Tasks cycle through 10 bug patterns per difficulty level, creating deliberate opportunities for temporal pattern recognition.

3.2 Conditions

STATELESS (Control): Each task presented independently with no context from previous tasks.

TEMPORAL (Treatment): Tasks presented sequentially with accumulated pattern context. After each solved task, a summary of observed bug types and successful fix strategies is appended to subsequent prompts.

3.3 Models

Model	Provider	Optimized For	Access
GPT-5.2	OpenAI	General reasoning, speed	API
Claude Opus 4.5	Anthropic	Deep reasoning, safety	API
Gemini 3 Flash	Google	Speed, efficiency	API

Table 1: Models tested. All accessed via production APIs with default parameters.

4 Results

4.1 Aggregate Performance

All three models achieve near-perfect accuracy ($\geq$99.6%), confirming the task battery falls within all models' capability frontiers.

Model	Cond.	Pass Rate	Avg (ms)	Easy	Med	Hard
GPT-5.2	Stateless	100.0%	1,236	1,067	1,100	1,616
GPT-5.2	Temporal	100.0%	1,153	931	955	1,679
Opus 4.5	Stateless	100.0%	2,153	1,933	1,994	2,626
Opus 4.5	Temporal	99.6%	2,149	1,926	1,979	2,637
Gemini Flash	Stateless	100.0%	6,884	2,547	4,118	15,760
Gemini Flash	Temporal	100.0%	8,056	2,460	4,908	18,986

Table 2: Aggregate performance across conditions and difficulty levels.

4.2 Temporal Speedup Divergence

The central finding: identical temporal context produces dramatically different effects.

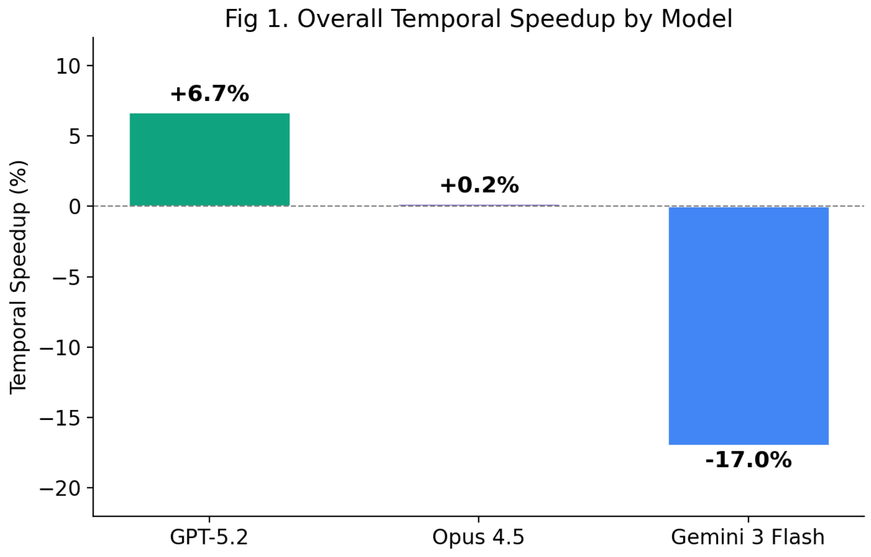

Figure 1: Overall temporal speedup by model. GPT-5.2 accelerates (+6.7%), Opus 4.5 is flat (+0.2%), and Gemini 3 Flash decelerates (−17.0%). The 23.7pp spread reveals fundamentally different context utilization strategies.

Model	Easy Δ	Medium Δ	Hard Δ	Overall Δ
GPT-5.2	+12.7%	+13.2%	−3.9%	**+6.7%**
Opus 4.5	+0.4%	+0.8%	−0.4%	+0.2%
Gemini 3 Flash	+3.4%	−19.2%	−20.5%	**−17.0%**

Table 3: Temporal speedup by task difficulty. Positive = faster with temporal context.

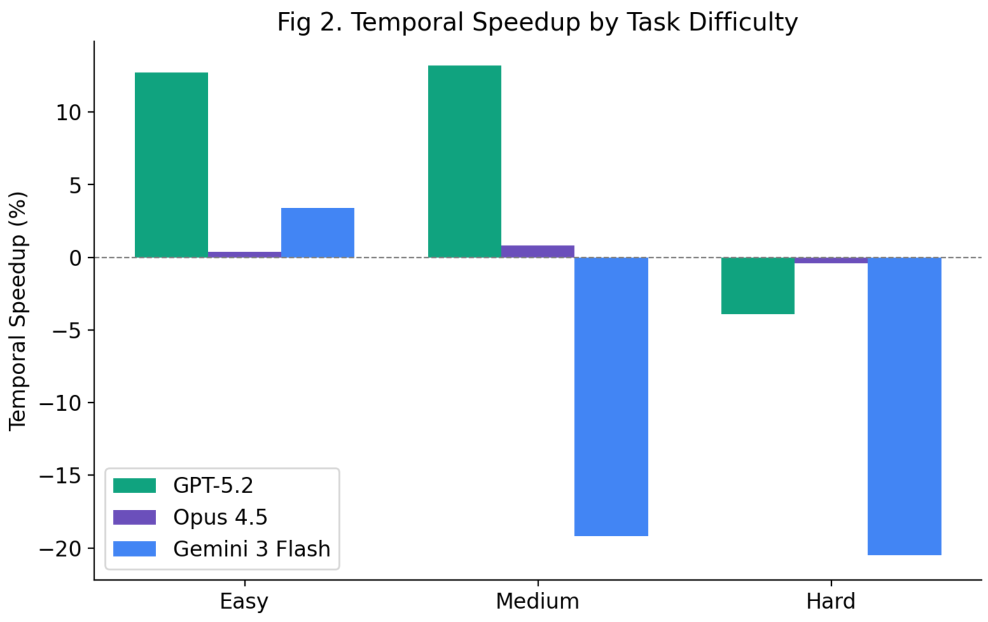

Figure 2: Temporal speedup decomposed by difficulty. GPT-5.2 benefits on easy/medium but not hard tasks. Gemini 3 Flash is penalized on medium/hard tasks. Opus 4.5 is uniformly flat.

4.3 Temporal Dynamics: Acceleration vs. Deceleration Curves

The most revealing data comes from batch-level analysis, exposing *how temporal effects evolve over time.*

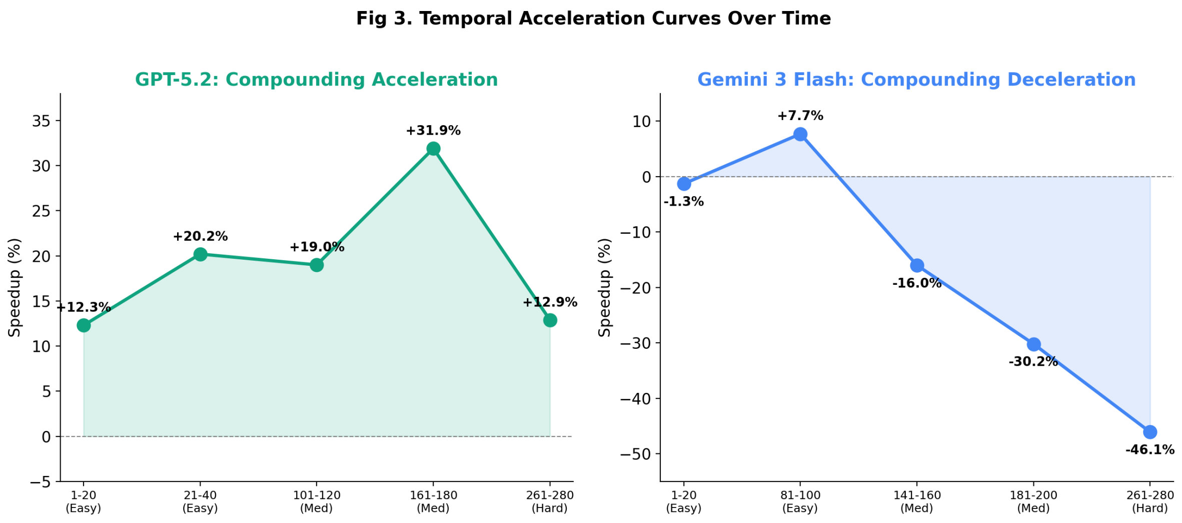

Figure 3: Batch-level temporal dynamics. **Left:** GPT-5.2 shows compounding acceleration, peaking at +31.9%. **Right:** Gemini 3 Flash shows compounding deceleration, reaching −46.1%. These mirror-image trajectories reveal fundamentally opposed context processing strategies.

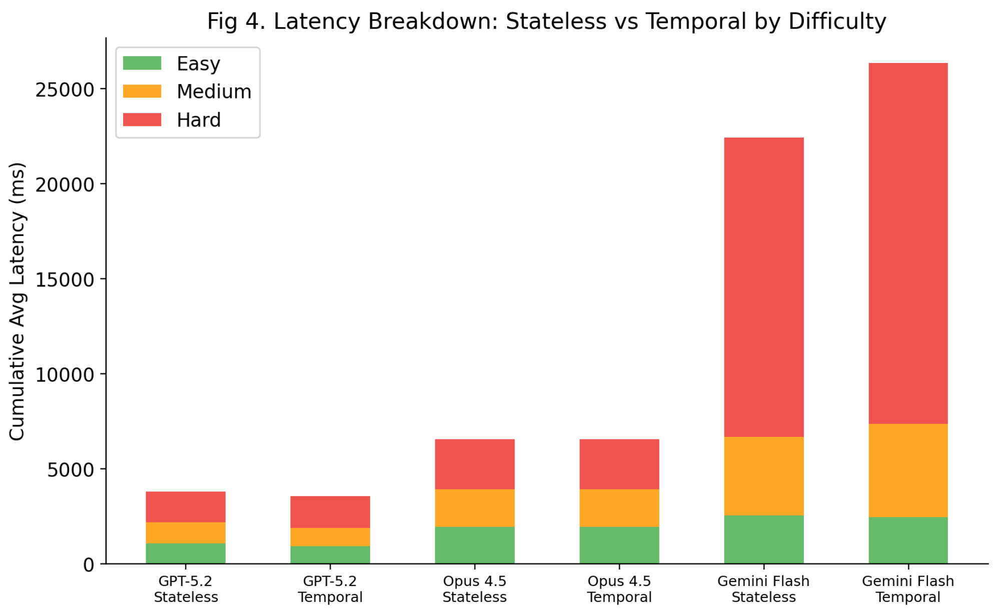

Figure 4: Absolute latency breakdown by difficulty and condition. Note the scale difference: Gemini 3 Flash's hard-task latency (15–19s) dwarfs GPT-5.2's (1.6–1.7s), revealing fundamentally different inference strategies despite comparable accuracy.

5 What Temporal Speedup Reveals About Each Model

The temporal speedup signature acts as a **behavioral X-ray** of each model's inference architecture. Since we control the input (identical tasks), the output (identical accuracy), and the intervention (identical temporal context), any divergence in latency must originate from *how the model processes and utilizes* the temporal context internally.

5.1 GPT-5.2: The Pattern Exploiter

Signature: +6.7% overall, compounding from +12.3% to +31.9% over time.

1. **Efficient in-context pattern matching.** GPT-5.2 treats accumulated temporal context as a lookup table—shortcutting reasoning on recognized bug types rather than deriving fixes from first principles.

2. **Sublinear context processing cost.** Despite monotonically growing context, speedup *increases* over time, implying efficient selective attention over long contexts.

3. **Two-stage inference strategy.** The +12.7%/+13.2% speedup on easy/medium vs. −3.9% on hard tasks reveals a match-first, reason-second approach. Pattern matching suffices for familiar bugs; hard tasks incur slight overhead from the failed matching step.

4. **Compounding returns.** The 31.9% peak speedup at batch 161–180 is consistent with the temporal intelligence hypothesis: accumulated knowledge compounds over time.

Architectural inference: GPT-5.2 likely employs an inference strategy that *first* checks context for matching patterns, *then* falls back to full reasoning only when no match is found—analogous to a cache-first lookup with parametric fallback.

5.2 Opus 4.5: The Self-Sufficient Reasoner

Signature: +0.2% overall, flat across all difficulties (±0.8%).

1. **Context compression/deprioritization.** The model assigns near-zero attention weight to context that provides no novel information beyond training data.
2. **Ceiling effect.** Every bug pattern is already in Opus 4.5's parametric knowledge with high confidence. Temporal context is informationally redundant—like giving a native speaker a dictionary.
3. **Robust, deterministic inference.** Near-zero variance (±4ms) across 280 tasks suggests a deep, fixed reasoning pathway unperturbed by contextual additions.
4. **No attention dilution penalty.** Unlike Gemini Flash, Opus 4.5 processes additional context without computational overhead—a sign of sophisticated attention management.

Architectural inference: Opus 4.5 employs a reasoning-first strategy from deep parametric knowledge, consulting context only when parametric knowledge is insufficient. This prioritizes reliability over contextual adaptation.

5.3 Gemini 3 Flash: The Deliberative Processor

Signature: −17.0% overall, compounding from −1.3% to −46.1% over time.

1. **Obligatory context processing.** Cannot efficiently ignore temporal context; every token is processed through the full reasoning pipeline.
2. **"Thinking" interference.** Temporal context triggers additional deliberation steps, particularly on medium/hard tasks where extended thinking is already engaged.
3. **Superlinear context scaling.** The progression −1.3% → −16% → −30% → −46% suggests combinatorial interactions between accumulated patterns.
4. **Speed optimization fragility.** The "Flash" speed advantage erodes with context length—the architecture is optimized for short prompts.
5. **Difficulty-proportional vulnerability.** Temporal overhead interacts multiplicatively with task complexity (+3.4% easy vs. −20.5% hard).

Architectural inference: Gemini 3 Flash employs a unified reasoning pipeline that cannot segregate temporal context from task context. The "Flash" speed optimization may involve fewer attention layers, explaining the inability to selectively attend to relevant context subsets.

6 The Context Utilization Spectrum

Synthesizing the three behavioral signatures, we propose the **Context Utilization Spectrum**:

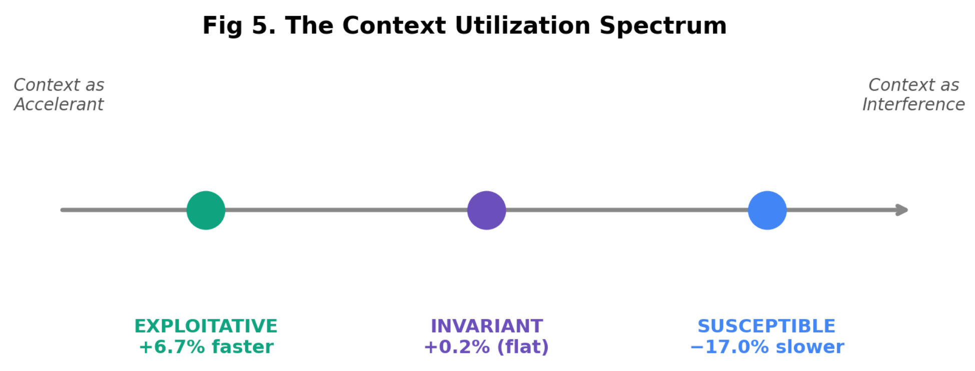

Figure 5: The Context Utilization Spectrum. Models are classified by their inference response to accumulated temporal context: exploitative (context accelerates), invariant (context ignored), or susceptible (context interferes).

6.1 Exploitative Models

Exploitative models (GPT-5.2) treat temporal context as a performance optimization. Speedup compounds over time. Benefits concentrated on pattern-matchable tasks. Optimal for systems where temporal context can be curated.

6.2 Invariant Models

Invariant models (Opus 4.5) are unaffected by temporal context because parametric knowledge already encompasses relevant patterns. Consistent, predictable performance. Optimal where reliability trumps context-dependent speedup.

6.3 Susceptible Models

Susceptible models (Gemini 3 Flash) experience degradation from temporal context. Slowdown compounds over time. Require aggressive context pruning or windowing to mitigate.

7 The Capability Boundary Hypothesis

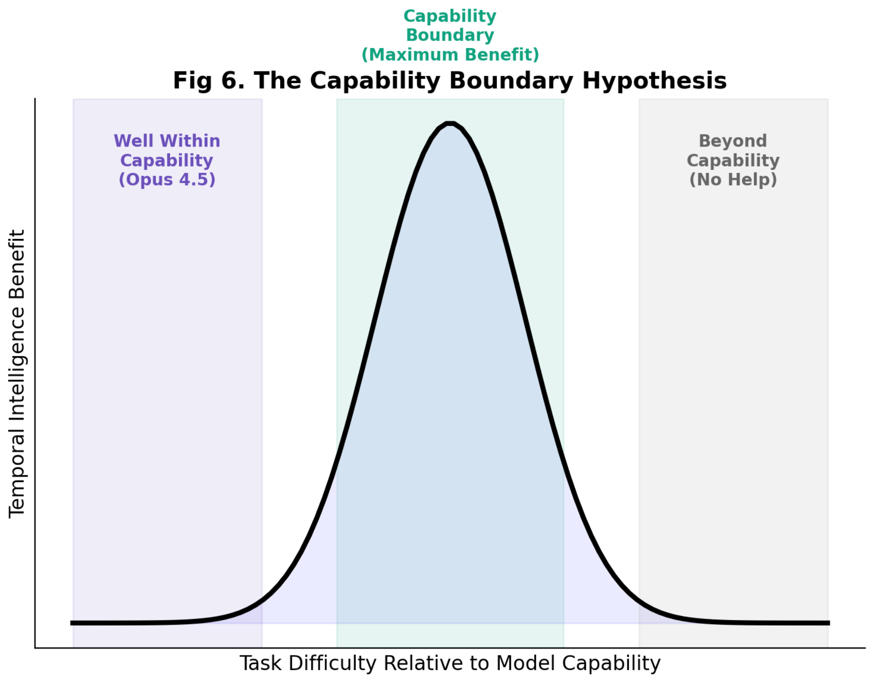

Figure 6: The Capability Boundary Hypothesis. Temporal intelligence provides maximum benefit at the boundary of model capability—where tasks are difficult enough to benefit from accumulated experience but not beyond the model's reach.

We propose: **Temporal intelligence provides maximum benefit when applied to tasks at the *boundary* of a model's capability**—difficult enough that the model benefits from prior experience, but not so far beyond capability that no amount of context helps.

Evidence:

- GPT-5.2's largest speedup (-31.9%) occurs on medium tasks in later batches—exactly at the intersection of accumulated knowledge and moderate difficulty.
- All models achieve $\geq 99.6\%$ accuracy, suggesting this task set is within all models' capability frontiers, limiting potential for accuracy improvement.
- On a harder task set (40–70% baseline accuracy), we predict Opus 4.5 would begin showing positive temporal effects as tasks approach its capability boundary.

8 Practical Implications

Decision	Recommendation
Whether to add temporal context	Profile your specific model first. Do not assume benefits transfer across models.
Context window management	Exploitative models: grow freely. Susceptible: aggressive pruning (sliding window).
Model selection	If temporal awareness is core, select exploitative models.
Task routing	Enable temporal context for pattern-matchable tasks; disable for novel reasoning.

Table 4: Practical guidelines for practitioners building temporally-aware AI systems.

8.1 For Model Developers

- **Exploitative models** could benefit from explicit temporal attention mechanisms.
- **Invariant models** could add context-novelty detectors that activate temporal attention only for genuinely new information.
- **Susceptible models** need architectural changes to support selective context gating.

8.2 For Researchers

Temporal speedup analysis provides a **non-invasive method for inferring architectural properties** of black-box models. By varying context and measuring latency, researchers can infer context scaling behavior, selective attention capabilities, and reasoning strategy sensitivity.

9 Limitations

1. **Three models.** A broader survey would strengthen the spectrum framework.
2. **API latency confounds.** Measured latency includes network/queuing time. Relative comparisons remain valid.
3. **Accuracy ceiling.** The $\geq$99.6% ceiling prevents observing temporal effects on correctness.
4. **Temporal context simplicity.** Richer representations might produce different effects.
5. **Single domain.** Python debugging is narrow; cross-domain validation needed.

10 Future Work

1. **Harder task batteries** producing 40–70% baseline accuracy to test accuracy improvement predictions.
2. **Broader model survey** spanning 8–10 architectures (open-source and commercial).
3. **Context ablations** varying richness from keywords to full solution traces.
4. **Cross-domain transfer** testing whether speedup signatures persist across debugging, math, and NL tasks.
5. **Longitudinal studies** across thousands of tasks to characterize saturation points.

11 Conclusion

The central insight of this paper is that **temporal context is not a neutral input**—it is an active intervention whose effects are mediated by model architecture. The same accumulated debugging patterns that make GPT-5.2 31.9% faster make Gemini 3 Flash 46.1% slower, while leaving Opus 4.5 entirely unaffected.

These three responses—exploitation, invariance, and susceptibility—define the Context Utilization Spectrum. The Capability Boundary Hypothesis further predicts that temporal intelligence benefits are maximized at the frontier of model capability.

For practitioners, the message is clear: **temporal intelligence is not a universal upgrade.** It is a model-specific, task-specific, and architecture-specific capability that must be profiled, tested, and monitored for each deployment context. The era of "more context is always better" is over. The question is no longer *whether* to provide context, but *which model can actually use it.*

References

[1] Brown, T.B. et al. (2020). Language models are few-shot learners. *NeurIPS 2020.*

[2] Chiang, W.-L. et al. (2024). Chatbot Arena: An open platform for evaluating LLMs by human preference. *ICML 2024.*

[3] Vashishtha, S. et al. (2020). Temporal reasoning in natural language inference. *Findings of EMNLP 2020.*

[4] Xie, S.M. et al. (2022). An explanation of in-context learning as implicit Bayesian inference. *ICLR 2022.*

[5] Zheng, L. et al. (2023). Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *NeurIPS 2023.*

[6] Zhou, B. et al. (2019). Going on a diet: Translation as the teacher to enhance temporal understanding. *EMNLP 2019.*

Correspondence: Anand Karasi, DisruptWithAI Research. https://disruptwithai.com