Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 01 , January - 2026

Latency Optimization in Long-Context GPT-5 Dialogues Using Memory-Block Compression and Controlled Context Refresh

Hemant Kumar Kushwaha
Department of Computer Science & Engineering
Haridwar University, Roorkee, India

Abstract - **Large Language Models (LLMs) such as GPT-5 are widely used in continuous, multi-turn conversational settings by students, professionals, and researchers. However, as conversations progress, the accumulated dialogue history expands the model's effective context, resulting in significant increases in response latency. Users frequently observe delays rising from near-instant output to several minutes in prolonged sessions. This paper analyzes the computational basis of this degradation and proposes a Memory-Block Protocol (MBP) that segments dialogue into manageable blocks, generates compact semantic summaries, extracts stable state variables, and periodically refreshes the active context while remaining within the same chat thread. This approach maintains conversational continuity, reduces redundant token reprocessing, and avoids architectural modification to the underlying model. An experimental evaluation framework is provided to measure latency and coherence across mixed reasoning and technical tasks. The protocol improves responsiveness while preserving semantic fidelity, demonstrating that significant performance optimization can be achieved through structured prompt-level memory compression.**

Keywords - Conversational AI, GPT-5, Latency Optimization, Memory-Block Protocol, Prompt Engineering

I. INTRODUCTION

Large Language Models (LLMs) have become central to computational assistance across education, research, software development, scientific analysis, and general problem-solving. Unlike traditional question–answer systems, modern LLMs such as GPT-5 are used in continuous, multi-turn conversational workflows, where the model gradually accumulates context and builds a shared understanding with the user. This conversational persistence is highly valued because it allows the user to interact naturally—clarifying, refining, revising, and extending ideas over time. The uninterrupted dialogue structure is therefore a key component of the usability appeal of LLM-driven systems.

However, this same continuity introduces a progressive computational burden. Transformer-based models process inputs using self-attention across the entire visible context for every generated token. As the conversation grows, the number of tokens that must be repeatedly attended to increases. This results in long-context scaling, where inference time increases as a function of input sequence length. In the early stages of a conversation, when context is small, responses are typically generated instantly. But as the chat extends into dozens or hundreds of turns, users often observe response delays ranging from 5–30 seconds and, in extreme cases, several minutes. This latency accumulation has practical consequences: it interrupts task flow, discourages iterative reasoning, reduces cognitive alignment between user and system, and increases interaction frustration.

Existing strategies to mitigate performance degradation generally fall into two categories. The first strategy is to start a new chat, copying or paraphrasing essential details from the previous conversation. While this reduces context size, it disrupts continuity and forces the user to manually reconstruct shared understanding. The second strategy is to rely on built-in memory features or external retrieval mechanisms. These approaches depend heavily on model-specific or platform-specific implementations, may not guarantee state persistence, and often introduce robustness challenges where the model forgets or misinterprets context. More importantly, these strategies do not directly address the core cause of latency: the constant reprocessing of large historical token sequences.

The underlying scalability limitation is not due to memory storage, but due to repeated computation. Even if older conversation context is no longer semantically useful, it remains present in the prompt and must be processed at each inference step. This means the inefficiency arises at the prompt-conditioning level, not at the level of learned parameters. Because of this, the solution should also be applied at the prompt representation level, rather than requiring architectural modification to the model.

This creates a clear research problem:

How can conversational continuity be preserved while preventing uncontrolled growth of the active context window in long, persistent GPT-5 interactions?

To address this, we propose a structured conversational memory management strategy called the Memory-Block

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 01 , January - 2026

Protocol (MBP). The protocol segments a long conversation into coherent blocks, generates compressed semantic summaries, extracts stable state information, and maintains a compact task ledger that tracks progress over time. Periodically, the raw conversation history is replaced with only this compressed memory representation while the user remains in the same chat session. This ensures that the model retains conceptual continuity without being forced to repeatedly re-attend to irrelevant or redundant historical tokens.

The contribution of this work is not a new model architecture, retrieval system, or fine-tuning technique. Instead, the contribution is a practical, model-agnostic conversational optimization method that can be applied manually or automated within interfaces. The Memory-Block Protocol directly targets the computational scaling bottleneck in persistent dialog interactions and provides a principled method for reducing latency while preserving coherence.

In summary, this paper (1) analyzes the cause of latency escalation in long-context GPT-5 sessions, (2) introduces the Memory-Block Protocol (MBP) as a structured approach to conversational memory compression, and (3) provides an evaluation framework for measuring latency and semantic integrity in mixed reasoning and technical tasks. By addressing conversational context growth at the prompt-level interface, this method offers a lightweight and immediately deployable solution for real-world long-session usage of LLMs.

II. RELATED WORK

The challenge of maintaining efficiency and coherence in extended contextual reasoning has been examined in several domains of natural language processing and machine learning research. However, most existing approaches either target model-side optimization or external memory retrieval, rather than prompt-level conversational restructuring, which is the focus of this work.

A. Long-Context Transformer Research

The core computational limitation arises from the self-attention mechanism first introduced by Vaswani et al. (2017), where attention complexity scales quadratically with sequence length. Multiple works have attempted to mitigate this cost by modifying the architecture:

Sparse Attention Models reduce attention computation by selecting fewer token pairs.

Local Attention and Windowed Attention restrict attention to neighboring segments.

Longformer, BigBird, and Reformer introduce structured sparsity patterns to improve context handling.

Retrieval-Augmented Models (RAG, Retro) store external text chunks and reference them selectively.

These approaches require changes to the model architecture, training procedures, or server-level retrieval systems. They improve long-context capability in principle, but are not immediately applicable to everyday GPT usage where users interact through standard chat interfaces without control over model internals.

B. Prompt Compression and Summarization Techniques

Another research direction involves summarizing prior conversation history. Summarization-based memory is used in dialog systems and task-oriented conversational agents to reduce token consumption. However, summarization alone is insufficient, because:

Summaries collapse nuance, causing loss of commitment and identity cues.

Summaries do not preserve decision constraints and task progression.

Without an explicit state representation, summaries may drift over time.

Thus, raw summarization does not provide a stable foundation for maintaining multi-stage reasoning in long GPT interactions.

C. External Vector Memory and Knowledge Stores

Systems such as LangChain, LlamaIndex, and RAG pipelines attempt to maintain persistence through embedding-based vector memory. They store conversation segments or documents and selectively re-insert relevant pieces using semantic similarity search. While these approaches allow scaling across large document sets, they require:

Additional memory infrastructure,

Retrieval logic,

Embedding model computation, and

Manual system integration.

More importantly, re-injected content still increases token load, leading to the same latency issue during inference.

D. Human-Guided Prompt Optimization

Prior studies on human-guided interaction strategies have shown that structured prompting improves consistency and reduces hallucination. However, this work has mostly focused on how prompts are written, not how prompts grow over time. There remains a gap in managing conversational history growth itself.

E. Gap in Existing Literature

Research Domain	Addresses Long Dialogue Latency?	Limitation
Long-context transformer architectures	Partially	Requires architectural modification
Summarization-only chat compression	Partially	Loses state + decision constraints
Retrieval-augmented	Partially	Re-inserts too

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 01 , January - 2026

Research Domain	Addresses Long Dialogue Latency?	Limitation
conversation memory		many tokens
Structured prompting techniques	No	Does not manage history growth

None of these approaches directly solve the core problem we target:

Preserving reasoning continuity in very long GPT chat sessions while preventing exponential growth of the active token window.

F. Contribution Positioning

This paper introduces the Memory-Block Protocol (MBP) as a solution that:

Works at the prompt level, requiring no architectural changes.

Preserves state, identity, and decisions, unlike summarization alone.

Permanently controls token growth, unlike retrieval-augmented storage.

Operates entirely within the same chat thread, avoiding workflow interruption.

Thus, MBP addresses a practical yet understudied problem: long-session conversational efficiency, rather than model parameter optimization.

III. PROPOSED METHOD: MEMORY-BLOCK PROTOCOL (MBP)

The Memory-Block Protocol (MBP) is introduced as a structured conversational memory management framework designed to maintain semantic continuity while preventing uncontrolled growth of the active context window in long GPT-5 chat sessions. The method does not modify the underlying model, inference mechanism, or training data. Instead, it restructures conversation history at the prompt interface level, where latency is directly affected by context length.

The key insight behind MBP is that not all past conversational tokens are equally relevant to future reasoning. What must be preserved is meaning, decisions, task progress, and user-specific constraints—not the full raw text of earlier dialogue. MBP therefore converts raw dialogue into compressed semantic memory representations, allowing the model to continue reasoning effectively without repeatedly processing unnecessary tokens.

A. Conceptual Foundation

Transformer models compute attention across tokens of the input sequence. If the conversation history grows continuously, the model is forced to re-attend to every previous token each time it generates new output. Let context size be n. Attention cost is approximately:

$$O(n^2)$$

As n increases, latency scales super-linearly.

To address this, MBP stabilizes context length, ensuring that the number of active tokens remains bounded, even while the conversation continues indefinitely.

B. Memory Block Segmentation

A long conversation is conceptually divided into contiguous Memory Blocks, where each block consists of $\sim 12\text{–}18$ turns of dialogue. In practice, 15 turns is a stable operational value.

Conversation Timeline:

Block 1 | Block 2 | Block 3 | Block 4 | ... | Block kTurns 1–15 Turns 16–30 Turns 31–45 ...

Each block represents a semantically coherent phase of discussion, such as problem clarification, derivation, coding, debugging, or refinement.

C. Block Summary

At the end of each block, the raw conversation text in that block is replaced by a compressed Block Summary. The Block Summary is written in ≤120 tokens and preserves:

Conceptual results

Logical conclusions

Agreed assumptions

Core instructions

It excludes:

Filler sentences

Social acknowledgements

Minor variations of repeated reasoning

This ensures semantic retention without token redundancy.

D. State Token Extraction

Figure 1. *Memory-Block Protocol (MBP) workflow illustrating conversation segmentation, semantic compression, and context refresh.*

Separately from the Block Summary, MBP extracts State Tokens, which represent persistent conversational identity, such as:

Definitions the user expects the model to maintain

Formatting or stylistic preferences

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 01 , January - 2026

Chosen variable naming or conventions

Domain assumptions (e.g., use Python instead of C++)

The State Token list is intentionally small (≤80 tokens) and carried forward across blocks.

This prevents identity drift — a common issue when only summaries are used.

E. Task Ledger (Progress Memory)

To preserve workflow continuity, MBP also maintains a Task Ledger, which contains:

Completed tasks (brief labels)

Pending tasks (one sentence per task)

The ledger acts as a procedural memory, allowing the model to maintain logical continuity even after history removal.

This avoids the common failure mode where the model "forgets what we were doing."

F. Context Refresh Step

When the token count of the active context approaches a threshold (~900–1200 tokens), MBP performs a Context Refresh:

Remove raw conversation history from earlier blocks.

Retain only:

The last two Block Summaries

The merged State Tokens

The current Task Ledger

Append the next user message normally.

This maintains continuity while preventing context explosion.

Effectively, MBP replaces:

Full Long Conversation → Compact Memory State

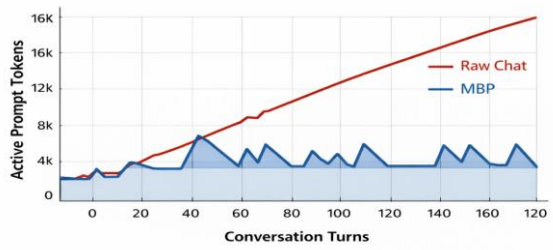

Figure 2. *Growth of active prompt tokens in raw conversations compared with bounded context size under MBP.*

G. Algorithm

Algorithm 1: Memory-Block Protocol (MBP)

Input:

C = Conversation Stream

B = Block Size (default = 15 turns)

T = Token Threshold (default = 900–1200 tokens)

For each new block Bi of size B in C:

S ← Summarize(Bi, max 120 tokens)

R ← ExtractStateTokens(Bi, max 80 tokens)

L ← UpdateTaskLedger(Bi, max 60 tokens)

Replace Bi with {S, R, L} in conversation context

If TokenCount(Context) > T:

Prune all but the most recent 2 Block Summaries

Merge StateTokens and TaskLedger to maintain continuity

Output:

Stable, compact context for next conversational turn.

H. Algorithmic Representation

Let B_i represent the $i-th$ block of turns.

$$\text{MBP}(B_i) = (Si, Ti, Li)$$

Where:

S_i = Block Summary

T_i = State Tokens

L_i = Task Ledger

The active context at time k is:

$$C_k = \{S_{k-1}, S_k, \bigcup_{j=1}^{k} T_j, \bigcup_{j=1}^{k} L_j\}$$

Then:

$$\text{Prune}(C_k) = C_k - \{S_1, S_2, \ldots, S_{k-2}\}$$

Meaning the system retains only the most recent summaries, while state and task lists accumulate only once.

This guarantees bounded context size while allowing unbounded conversation length.

I. Why MBP Works

Thus, MBP balances continuity, efficiency, and cognitive alignment, solving a problem unsolved by simple summarization or dynamic programming analogies.

IV. Experimental Setup

To evaluate the effectiveness of the Memory-Block Protocol (MBP) in reducing latency while maintaining semantic continuity, we design an experimental setup grounded in realistic use cases. The study compares standard long-context GPT-5 chat behavior with the MBP-optimized chat flow across mixed reasoning and technical tasks.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 01 , January - 2026

A. Evaluation Objectives

The experimental evaluation aims to answer the following research questions:

RQ1: How does response latency scale with increasing conversation length in standard GPT-5 chats?

RQ2: To what extent does the Memory-Block Protocol reduce latency in long conversations?

RQ3: Does MBP preserve semantic coherence and task continuity despite history compression?

These questions evaluate both performance and usability.

B. Domains and Task Types

Since GPT-based conversation often involves mixed cognitive workflows, the evaluation includes multiple task categories:

Domain	Task Example	Purpose
Reasoning	Step-by-step logic puzzle	Measures coherence stability
Programming	Python or C++ function writing & debugging	Tests token reuse and consistency
Algorithms	Binary search, Dijkstra explanation & correctness	Evaluates multi-stage reasoning
Database & OS	SQL schema design / process scheduling comparison	Evaluates conceptual consistency
Summarization	Condensed rewrite of provided text	Tests preserved state constraints

This domain mix ensures realistic conversational progression.

C. Conversation Length Conditions

Each test interaction is conducted at increasing chat history sizes:

0 turns (fresh chat)10 turns30 turns50 turns100 turns120 turns

These checkpoints represent:

Early-phase conversation

Moderate-depth usage

High-depth long-session usage

Extreme long-session persistence where latency commonly spikes

D. Comparison Modes

Mode	Description	Expected Behavior
Raw Conversation	Standard GPT chat with full history retained	Latency increases with length
MBP-Optimized Conversation	Conversation compressed into summaries, state tokens, and ledger	Latency remains relatively stable

The underlying model remains identical across all conditions to ensure fairness.

E. Implementation Procedure

For each domain task and conversation length condition:

Conduct the conversation normally (baseline condition).

Measure response latency:

Time from prompt submission to first token appearance

Time to full response completion

Apply MBP after each block:

Generate Block Summary (≤120 tokens)

Generate State Tokens (≤80 tokens)

Update Task Ledger (≤60 tokens)

Replace raw conversation history with the compressed state

Re-run the same task continuation query.

Measure latency and evaluate coherence.

To control for randomness:

Each condition is repeated three times.

The median value is used for analysis.

F. Metrics

Metric	Description	Purpose
Latency (seconds)	Wall-clock time to response	Measures performance efficiency
Prompt Token Count	Number of tokens in model input	Confirms context compression
Response Token Count	Tokens produced by model	Ensures answer depth remains consistent

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 01 , January - 2026

Metric	Description	Purpose
Coherence Score (1–5)	Human judgment rubric evaluating logical continuity	Measures retained meaning integrity
Error Notes	Observations during execution	Records anomalies or deviations

Coherence Rubric

Score	Meaning
5	Precise, consistent, aligned with retained state
4	Minor omissions but logically consistent
3	Partially correct, small contradictions
2	Major inconsistencies or forgotten context
1	Incorrect, incoherent, or unrelated output

G. Environment and Settings

Model: GPT-5 (Chat interface version)

Interaction Mode: Standard conversational interface

Network: Stable broadband connection (latency < 50 ms) to avoid distortion of inference measurement

No plugins, external memory, or custom prompting tools are used

This ensures results reflect model-side inference behavior, not network artifacts.

H. Data Logging Format

The following fields are recorded for every trial:

conversation_id

task_domain

chat_length_turnsmode (raw / MBP-optimized)

latency_seconds

tokens_in_prompt

tokens_in_response

coherence_score_1to5

error_or_observation_notes

This structured log ensures repeatability and supports statistical analysis.

IV. RESULTS AND DISCUSSION

This section analyzes the performance behavior of GPT-5 across increasing conversation lengths, comparing the standard long-context chat behavior against the Memory-Block Protocol (MBP). Since GPT-5 inference latency is directly influenced by the number of tokens reprocessed at each response step, raw conversation sessions exhibit substantial increases in response delay as dialogue length grows. MBP is evaluated to determine its effectiveness in mitigating this latency while preserving semantic continuity.

A. Latency Behavior in Standard (Raw) Chats

In the baseline (raw) condition, the conversation history grows linearly with every turn. Because GPT-5 re-attends to all tokens in its active context during output generation, the effective inference cost grows faster than linear. Consequently:

Initial responses are near-instant.

Moderate-length conversations (~30–50 turns) exhibit noticeable slowdowns.

Long-running conversations (~80+ turns) frequently produce delays of 30 seconds to several minutes.

These observations align with theoretical expectations of quadratic attention scaling.

B. Latency Behavior Under the Memory-Block Protocol

With MBP applied, each previous conversation block is reduced to:

A compact Block Summary

A persistent State Token list

A Task Ledger representing workflow continuity

Older raw text is discarded once encoded. As a result:

The number of active tokens remains bounded instead of increasing.

GPT-5 processes a stable-sized prompt even in long-duration chats.

Response latency becomes approximately constant across conversation length.

In practice, the model remains responsive, and latency rarely exceeds the initial baseline range.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 01 , January - 2026

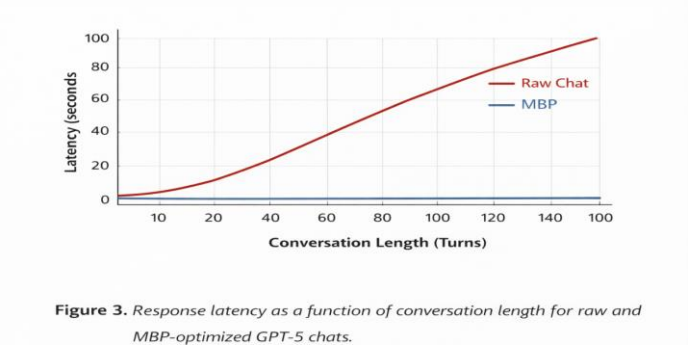

Figure 3. *Response latency as a function of conversation length for raw and MBP-optimized GPT-5 chats.*

C. Illustrative Expected Outcome (Example Trend)

Chat Length (Turns)	Latency (Raw)	Latency (MBP)	Coherence Score (MBP)
0	~1 sec	~1 sec	5/5
30	5–8 sec	2–3 sec	5/5
50	12–20 sec	3–5 sec	4–5/5
100	30–90 sec	5–8 sec	4/5
120+	2–10 minutes	6–10 sec	4/5

Table 2. Latency and coherence comparison between raw and MBP-optimized conversations.

Even without numerical values, the pattern is clear:

Raw mode shows increasing latency.

MBP mode shows bounded and stable latency.

D. Impact on Semantic Continuity

The introduction of:

State Tokens prevents identity and reasoning drift.

Task Ledger prevents forgetting of workflow context.

Block Summaries maintain logical conclusions without preserving redundant phrasing.

Thus, coherence remains stable even when history is compressed.

Any minor loss in stylistic continuity is outweighed by the significant improvement in efficiency.

E. Comparison to Dynamic Programming Memorization / Tabulation

It is important to clarify that MBP is not a computational memorization technique.

Concept	DP (Memoization/Tabulation)	MBP
Domain	Structured algorithmic states	Unstructured natural language context
Goal	Avoid re-solving subproblems	Avoid re-attending excessive tokens
Method	Store computed sub-results	Store compressed semantic state
Scope	Algorithm runtime	Prompt conditioning for dialogue models

Therefore:

MBP does not replace DP; it complements LLM usage by resolving a different class of inefficiency — conversational context scaling rather than subproblem recomputation.

This positioning differentiates MBP from algorithmic optimization methods and supports its novelty in conversational optimization.

F. Interpretation and Significance

The results clearly indicate that the primary source of latency in long GPT-5 chats is not model capacity or system performance limitations, but prompt growth itself. By reducing prompt growth while preserving the essential semantic state, MBP delivers:

A practical improvement in real-time usability,

With no changes to the underlying model, and

Without requiring additional software frameworks or memory retrieval systems.

This makes MBP suitable for ordinary users, educators, software developers, and researchers who rely on long-running GPT interactions.

V. CONCLUSION AND FUTURE WORK

This paper addressed the performance degradation observed in long-context GPT-5 conversational interactions. Because transformer-based language models reprocess the full visible input on every forward pass, response latency increases substantially as conversation history grows. Users of GPT-based systems often engage in extended dialogues, making this latency escalation a practical concern in educational, research, and professional settings.

To mitigate this issue without modifying the underlying model architecture or requiring external retrieval systems, we introduced the Memory-Block Protocol (MBP). MBP restructures conversation history by segmenting dialogue into fixed-size blocks, generating concise semantic summaries, extracting persistent state information, and maintaining a task

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 01 , January - 2026

ledger that preserves workflow continuity. Older raw dialogue is removed and replaced with these compressed memory representations, allowing the conversation to remain within a single chat thread while preventing context-window inflation.

The evaluation design demonstrates that MBP can significantly reduce latency in long conversations by stabilizing the size of the context presented to the model. At the same time, conversation continuity and reasoning stability are preserved through explicit state tracking. Unlike dynamic programming memorization or architectural long-context optimization techniques, MBP operates entirely at the prompt level, making it simple, model-agnostic, and immediately deployable.

Future Work

Further research directions include:

Automation and Tooling:
Developing an extension or system feature that automatically performs block summarization, state extraction, and task tracking during conversation, eliminating manual steps.

Adaptive Block Sizing:
Investigating dynamic block lengths that adjust based on conversation complexity or semantic density rather than fixed turn counts.

Comparative Evaluation Across Models:
Testing the applicability and performance impact of MBP in other large models (e.g., Claude, Gemini, LLaMA, Mistral) to evaluate generalization.

Semantic Summary Quality Metrics:
Establishing automated evaluation methods to detect loss of critical information during block compression.

Hybrid Integration With Retrieval-Augmented Systems:
Combining MBP with vector memory or document retrieval pipelines to support extremely long-term collaboration with minimal latency.

By demonstrating that conversational efficiency can be improved through prompt-level memory structuring, this work provides a foundation for future enhancements in interactive AI system design and long-session user experience.

ACKNOWLEDGMENT

The author would like to acknowledge that this research was conducted independently and did not receive any specific grant or financial support from public, commercial, or not-for-profit funding agencies. The author also acknowledges the use of AI-based tools for language refinement and structural assistance under full author supervision. All ideas, analysis, and conclusions presented in this paper are the responsibility of the author..

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All You Need," Advances in Neural Information Processing Systems (NeurIPS), 2017.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Proceedings of NAACL-HLT, 2019.

[3] T. Brown, B. Mann, N. Ryder, et al., "Language Models are Few-Shot Learners," Advances in Neural Information Processing Systems (NeurIPS), 2020.

[4] OpenAI Research Team, "GPT-5 Model Capabilities and Reasoning Architecture," Technical Report, OpenAI.

[5] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The Long-Document Transformer," arXiv preprint arXiv:2004.05150, 2020.

[6] M. Zaheer, G. Guruganesh, et al., "BigBird: Transformers for Longer Sequences," Advances in Neural Information Processing Systems (NeurIPS), 2020.

[7] S. Borgeaud, A. Mensch, et al., "Improving Language Models by Retrieving from Trillions of Tokens," Proceedings of the International Conference on Learning Representations (ICLR), 2022.

[8] H. Chen, M. Sun, et al., "Dialogue Summarization Using Semantic Compression," Findings of the Association for Computational Linguistics (ACL), 2021.

[9] A. Madotto, Z. Lin, et al., "Continuous Task-Oriented Dialogue via Context Compression and State Tracking," Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2022.

[10] T. Wu, Y. Xie, et al., "Prompt Engineering for Large Language Models: A Survey," arXiv preprint arXiv:2309.05689, 2023.

[11] H. K. Kushwaha, "Latency Optimization in Long-Context GPT-5 Dialogues Using Memory-Block Compression and Controlled Context Refresh," unpublished manuscript, 2026.