
BOUNDED-MEMORY LONG-HORIZON CONVERSATIONS: MGPT AND MODEL-GOVERNED KV MEMORY

A PREPRINT

Luís Marnoto
Continual Machine Intelligence
maarnotto@gmail.com

Work in Progress
November 29, 2025

ABSTRACT

We introduce MGPT (Mask-Generative Pretrained Transformer), a model-managed masking architecture that treats the KV cache as a high-dimensional neural memory. Converting a pretrained GPT into an MGPT model involves fine-tuning it to emit explicit `<|mask|>N` commands that hide stale messages from direct attention while retaining their compressed traces inside downstream states, allowing the system to recall masked information without restoring it verbatim. By relying on model-governed masking, system message forwarding, and continuous compression, an MGPT deployment maintains fixed memory usage within the model’s context window without auxiliary maintenance procedures. A 4B-parameter MGPT reaches 99% token accuracy on BABILong QA1 tasks with constant memory usage, demonstrating that transformer deployments can sustain long-horizon interactions through neural KV memory alone.

Keywords Transformer · Neural Memory · Bounded Context · Attention Masking · KV Cache

1 Introduction

MGPT (Mask-Generative Pretrained Transformer) equips a transformer with everything it needs to regulate its own context, compressing history through masking while cultivating neural traces inside the KV cache. The model emits numbered `<|mask|>N` commands that hide stale turns from direct attention yet preserve their influence inside downstream activations, enabling recall of masked content without replaying it verbatim.

This work contributes a self-governed context mechanism that relies on model-issued masking and explicit system message forwarding. Numbered messages provide stable references, masking contracts the effective context, and the resulting neural KV memory behaves like a fading but durable recollection. A 4B-parameter MGPT reaches 99% token accuracy on BABILong, outperforming substantially larger systems while sustaining constant memory usage.

Section 2 outlines the architectural limitations that motivate MGPT, Section 3 details the method, Sections 3.6 and 3.7 cover training and inference, and the paper concludes with evaluation and discussion.

2 Background and Motivation

This section examines how modern GPT architectures operate and their limitations that surface during extended interactions, and the motivation behind MGPT.

2.1 GPT Architectures and Their Limits

We begin with a discussion of the ubiquitous GPT architectures that underpin most large language models. The following subsections outline a chain of architectural issues that compound as conversations grow longer, motivating the need for a different approach.

2.1.1 Fixed Context Ceiling

Every GPT is trained with a maximum sequence length dictated by its positional encoding scheme and the data seen during pretraining. Current models claim windows of one or two million tokens, but the limit is still finite. Long-running conversations, day-long research sessions, or streams of documents eventually accumulate more tokens than the model can interpret. Once the token count exceeds the window, earlier content must be dropped or summarized, and whatever falls out of scope can no longer influence future answers. Without a way to reset or compress context in place, the system inevitably loses information during extended use.

2.1.2 Quadratic Attention Cost

Autoregressive self-attention compares each new token to every earlier token that remains in context. If the prompt contains L tokens, the attention module performs on the order of L^2 multiply-add operations per forward pass. Model providers try to cut redundant work with key-value (KV) caching, storing the keys and values computed for prior tokens so they do not have to be recomputed on the next step. This speeds up the constant factors, yet it does not alter the scaling law: every additional token still requires computing attention scores against all cached keys. As L grows, compute and memory use rise, and inference cost escalates. Once the sequence length is large enough, the quadratic cost makes frequent interactive usage impractical, forcing the model provider to shorten the context or accept substantial latency and cost.

2.1.3 Model Amnesia

When a GPT drops part of its prompt, all evidence that those tokens ever existed disappears alongside them. The model’s weights are frozen, so it has no way to retain knowledge beyond the text that remains in the window. Once earlier messages are removed to stay within the fixed context ceiling, their embeddings and hidden states vanish instantly. The model does not notice the removal; it simply recomputes attention over the remaining tokens and continues as if the discarded turns were never present. Any plan or line of reasoning that depended on those messages collapses. We refer to this behavior as *Model Amnesia*. Long-running tasks and workflows are slow and unreliable precisely because the model keeps forgetting where it is in the task and what has already happened.

2.1.4 Diluted Computation

Transformers have a fixed number of layers set during pretraining. As prompts expand, each layer must distribute its representational capacity across more tokens. Attention weights flatten, important instructions compete with incidental chatter, and long-range dependencies fade. Instruction-following errors become common because the early details in the prompt have to fight tens of thousands of later tokens for bandwidth. The fewer layers the model has, the worse this problem becomes, and the less context it can handle effectively, despite sometimes advertising much larger context windows than what it can really process.

2.1.5 Shallow External Context Management Solutions

A common workaround for long-context tasks is to store large amounts of conversation history in a database and, for each new user turn, retrieve the entries that seem relevant before proceeding with generation. The retrieval step typically relies on an embedding model that maps text into a high-dimensional space so the system can select nearby entries. In practice this embedding model is much smaller and weaker than the main generator. It pulls passages that are superficially similar to the query but ignores long-term dependencies, cross-document links, and details already stored elsewhere in the database. The generator then receives context that is only loosely related to the problem at hand and must produce an answer using incomplete or tangential information. This disconnect between the main model and the retrieval model makes the overall system unreliable for many long-context tasks.

2.2 Motivation for MGPT

We set the motivation and goal to build a system that can exhibit the following characteristics: the model organizes its own working context without an external management system, it keeps critical information close while letting irrelevant

details fade, and it leverages the KV cache as a neural memory that accumulates compressed traces over time. We want short-term recall of the current conversation, long-term retention of lessons gathered over time, and the ability to sustain these memories without compute costs exploding as the interaction history grows. This vision of self-managed context and gradual neural recall is the motivation behind the Mask-Generative Pretrained Transformer.

3 MGPT

Turning a standard GPT into an MGPT model yields a system that manages its own working context while keeping memory usage fixed within the model’s context budget. The conversion method follows a chain of insights: tokens that once read a passage keep traces of it, so a transformer can set aside a message without erasing everything it learned from it. If we give the model a precise way to decide which messages lose direct attention—and keep the system prompt aligned with that decision—it can compress context while staying aware of what was hidden. Making that self-management viable requires numbered handles, chunk-aware ingestion, cache telemetry, and training signals aligned with model-led cleanup.

The following sections walk through these components in the order they emerged. We start with the attention insight that underpins indirect recall. We next introduce numbered messages paired with the `<|mask|>N` command, extend the same control scheme to arbitrarily large inputs through chunking and the `context` role, and describe cache management techniques that preserve RoPE stability without rebuilding the entire history. We close with the inference and training infrastructure optimized for MGPT.

3.1 Awareness Through Indirect Attention

The first MGPT insight is that a transformer can stop paying direct attention to large spans of a conversation without erasing what it already learned from them. Consider a dialogue that includes several long documents: after the model finishes reading document A, we can mask the messages containing documents B and C so the model focuses its compute on A. Every token that followed B or C already attended to those messages when they were visible, so their hidden states still carry a trace of the masked content. Even after the mask is applied, later tokens can reach the missing information by attending to the tokens that were produced while the documents were still visible.

Masking therefore compresses, rather than deletes, information. The model keeps an awareness that the masked messages exist, and it can retrieve the relevant details indirectly through the downstream tokens. Mu et al. [Mu et al., 2023] showed that training a transformer to compress prompts into shorter masked spans still preserves the information needed for downstream tasks, which verifies our intuition that compression through masking retains essential content. This compression-through-masking observation underpins MGPT’s controllable context manager.

3.2 Self-Directed Context Management

If the model can keep awareness of masked content, the next step is to let it decide when to apply masking. External heuristics or retrieval systems defeat the purpose, because we want the model itself to manage the conversation. The MGPT training procedure teaches the model to emit an explicit command whenever it judges that a message should leave direct attention, and to rely on neural traces carried forward through subsequent activations when information becomes relevant again. The model also has to understand the consequences of masking so that it can plan ahead: hiding a message saves attention, but it must do so in a way that keeps the information recoverable through indirect attention pathways. These requirements set the stage for the concrete mechanisms that follow.

3.3 Numbered Messages and the Mask Command

Every conversational turn receives a stable identifier that combines an increasing index with the speaker role (e.g., `1user`, `2assistant`, `3user`), as shown in the MGPT conversation example in Figure 2. These numbered handles give the model a vocabulary for referring to specific messages when it chooses to reorganize the conversation. We then introduce a command sequence `<|mask|>N`, which uses the same identifiers so the transformer can point directly at the message it wants to hide. Numbering creates precise targets, and the command executes the decision. When the model emits `<|mask|>N`, all tokens produced *after* the command lose direct attention to message *N*; tokens produced *before* the command are unaffected. Because the identifiers are part of the conversation, the model can also reason about the consequences of masking a specific turn before it commits to the action. Figure 2 contrasts this structured exchange with a standard GPT transcript that must expose every earlier message to the model at all times, underscoring why an MGPT system needs numbered handles to sustain long-running conversations.

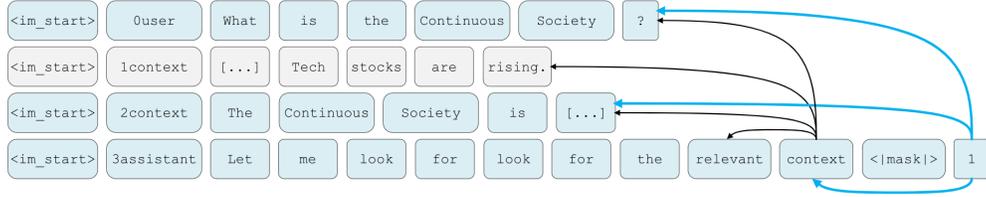


Figure 1: MGPT attention patterns showing direct (black) and compressed (blue) information pathways. The token “1” after masking can access information from masked regions indirectly through the “context” token, demonstrating hierarchical information compression rather than destruction.

```

<|im_start|>0user
What is the Continuous Society?<|im_end|>
<|im_start|>1context
Apple released new iPhones. Tech stocks rising.<|im_end|>
<|im_start|>2context
The Continuous Society is an exclusive research collective
advancing technology.<|im_end|>
<|im_start|>3assistant
Let me look for the relevant context <|mask|>1 The Continuous
Society is an exclusive research collective advancing
technology.<|im_end|>

<|im_start|>user
What is the Continuous Society?
Context: Apple released new iPhones. Tech stocks rising. The
Continuous Society is an exclusive research collective advancing
technology.<|im_end|>
<|im_start|>assistant
The Continuous Society is an exclusive research collective
advancing technology.<|im_end|>
    
```

(a) MGPT conversation format with numbered message roles enabling selective masking.

(b) Standard GPT conversation format requiring full attention to all context.

Figure 2: Conversation format comparison showing MGPT’s numbered messages enabling dynamic masking vs. GPT’s monolithic context processing.

This directed control does not destroy information. As illustrated in Figure 1, tokens generated after a mask still reach the masked content indirectly through intermediate tokens that already absorbed it. The earlier tokens retain full access, creating a hierarchical compression pathway where relevant information survives inside the hidden states. During training we supervise the emission of `<|mask|>N` so the model learns *when* to hide a turn and how to balance memory savings with answer quality.

3.4 Handling Large Messages

User-authored messages with the user role are hard-capped at 512 tokens so that no single turn can monopolize the context window. When a user needs to provide more material, they fill out two fields: the user input and an accompanying context input. The context field has no token limit. The system records the user message twice—once at the start of the exchange and once after all context has been processed—so the model sees the request both before it begins analysis and right before it answers.

The unlimited context field is split into consecutive 512-token messages. Each slice becomes its own numbered context message, immediately followed by an assistant turn where the model examines that chunk and, when needed, masks it or earlier chunks using the `<|mask|>N` command. Our MGPT runtime tracks the precise token boundaries for every chunk so later mask operations target only the intended span. As the conversation steps through a long artifact, the model decides chunk by chunk whether to keep a message accessible or hide it, masking older context messages before moving on to new ones. Figure 3 shows an example BABILong conversation that follows this pattern. The sequential structure lets an MGPT agent ingest multi-thousand-token inputs while preserving the precision of its control commands and keeping the KV cache aligned with whichever chunks remain in the effective context.

3.5 System Message Forwarding

Stable long-horizon operation requires RoPE positions to remain bounded as conversations grow toward the model’s context limit. An MGPT deployment achieves this by forwarding the immutable system message so that it always sits immediately before the newest message that remains inside the model-managed effective context. After each assistant turn, the runtime locates the highest-numbered unmasked message, removes the system prompt from its previous slot, and reinserts it directly ahead of that message while preserving the rest of the history as-is. The associated KV entries are moved in lockstep, so relative token distances stay compact without recomputing the sequence. Because the model dictates which historical turns remain effective, this forwarding step stays aligned with the masking plan, letting recent details stay vivid while older content survives as compressed neural traces without triggering RoPE drift.

```

1user - Where is Daniel?\n\nBased on all the information, what is the answer?
2context - A thin wind needled through the cloister, stirring the chapel's guttering candles. Daniel travelled to the hallway. Dust ...
3assistant - <think>Daniel travelled to the hallway. I'll continue tracking.\n<mask|>2</think>\nhallway
4context - Ledgers sprawled across the steward's table while the servants traded whispers. Mary journeyed to the bathroom. Brass ...
5assistant - <think>I'll continue remembering where Daniel is.\n<mask|>4\n\n<mask|>3</think>

... intervening Ncontext / Nassistant pairs continue with book passages wrapping additional BABILong facts ...

86context - Lanterns along the garden walk bowed in the rain-soaked wind. Daniel moved to the garden. Petals from the night-blooming ...
87assistant - <think>Daniel moved to the garden. I'll continue remembering his position.\n<mask|>86\n\n<mask|>85</think>\ngarden
88user - Where is Daniel?\n\nBased on all the information, what is the answer?
89assistant - <think>Daniel is in the garden\n<mask|>87\n\n<mask|>88</think>\ngarden

```

Figure 3: Example BABILong interaction illustrating the two-form input pattern: an initial capped user prompt, streaming context messages, assistant summaries with mask commands, and a closing user prompt that elicits the final answer.

3.6 Training

MGPT arises from supervised fine-tuning on BABILong QA1 traces that interleave subject-location facts with targeted queries. The loss supervises both answer tokens and the emitted `<|mask|>N` commands so the model practices compressing salient details into its keys and values while hiding stale turns. Beyond this masking-focused curriculum, we keep the remaining optimization details private.

3.7 Inference System

MGPT inference runs on top of a persistent conversation state that tracks every message ever exchanged. Effective context—the portion that remains in direct attention—is derived from that history by applying the mask commands produced by the model. Each message is chunked to respect per-message token limits, and the system maintains message ranges so it can locate and edit the exact tokens that remain active. When a user submits a long input, the agent processes it in segments, updating the mask plan after each chunk and keeping the KV cache aligned with the current effective context.

Masking and system message forwarding rely on KV-cache surgery. The cache stores keys and values for tokens that are still in play, and the surgery routines excise or append segments without rebuilding the entire cache. Message-number metadata keeps the cache, tokenizer stream, and mask commands synchronized, allowing inference cost to remain stable as the conversation grows while the model governs what stays in immediate memory.

4 Evaluation

We evaluate MGPT on the BABILong QA1 benchmark, which stresses long-context reasoning by interleaving subject-movement facts with large volumes of distractors. The dataset mirrors the two-form input workflow described in Section 3: each example supplies a capped user prompt, a long stream of context messages, and assistant turns that store facts while issuing mask commands. Figure 3 illustrates a representative conversation drawn from the evaluation set.

Our 4B-parameter MGPT model attains 99% mean token accuracy on the BABILong validation sets with 8K-token contexts, outperforming substantially larger baselines such as Gemini 1.5 Pro, which reaches 75%. The model consistently aligns its mask decisions with the ground-truth supporting facts while maintaining constant memory usage as sequence length grows.

The MGPT model also demonstrates effective context compression: hierarchical masking typically condenses 512-token slices into 64-token representations, delivering an 8:1 reduction while preserving critical facts across multiple compression stages. Cache-size measurements stabilize around 400–500 tokens regardless of raw input length, and inference time scales with the effective context rather than the total number of ingested tokens.

Ablation studies confirm that MGPT’s behavior stems from the learned masking policy rather than auxiliary architectural changes. Replacing FlexAttention with standard SDPA maintains accuracy, the MGPT-tuned model answers 5/5 validation queries compared to 4/5 for the base Qwen2.5-4B model, and training with a batch size of 1 proves crucial for the sequential masking curriculum, providing focused updates on each chunked example.

5 Conclusion and Discussion

We have presented MGPT, a model-directed masking mechanism for bounded-memory long-horizon processing through dynamic masking and KV cache surgery. Our approach demonstrates strong performance on long-context benchmarks while maintaining constant memory usage through hierarchical information compression rather than information destruction.

The key insight is that information can be compressed into intermediate representations and accessed through indirect attention pathways, enabling the model to maintain context continuity while achieving memory efficiency. This distinguishes MGPT from simple attention filtering approaches that result in hard information loss.

MGPT represents a practical advance in transformer context management, showing that longer-horizon processing can be sustained within a fixed memory budget through learned compression and cache maintenance rather than auxiliary procedures or manual restoration commands. The strong empirical results on BABILong with a relatively small 4B parameter model suggest that this approach could broaden access to long-context AI capabilities. The bounded-memory capability supports long-lived assistants, long-form content generation, interactive AI agents, and document analysis applications where ordinary prompt growth would otherwise exhaust the available context budget.

Future work directions include reinforcement learning approaches to optimize masking strategies, richer neural memory diagnostics that quantify the persistence of masked content, extension to multimodal contexts, and integration with retrieval-augmented generation systems. The approach bridges the gap between transformer architectures and recurrent models, combining parallel processing benefits with memory efficiency.

References

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. *arXiv preprint arXiv:2304.08467*, 2023.