KRONOS: Knowledge Retention with Optimized Neural Organization System

Mohak Somani(2022101088), Prakhar Singhal(2022111025) Prakhar Jain(2022115006)

This report presents an overview of KRONOS (Knowledge Retention with Optimized Neural Organization System), a novel architecture inspired by Google's TITANs framework, aimed at addressing the limitations of traditional transformer models in long-term memory retention and efficient reasoning. The core challenges tackled include context window limitations, inefficient memory usage due to quadratic attention complexity, and poor handling of multi-timescale dependencies. KRONOS introduces several innovations: a composite surprise metric for more accurate information relevance assessment, an adaptive forgetting mechanism guided by reinforcement learning, and latent space thinking to enable high-level reasoning through compressed representations. Additionally, it proposes a concept-based memory structure to organize information hierarchically by semantics. The report documents initial progress, key innovations, and outlines a roadmap toward comprehensive implementation and evaluation. These advancements aim to enhance the scalability, adaptability, and cognitive efficiency of future AI systems.

Date: May 8, 2025

I Introduction

Transformer-based models have achieved impressive results across a variety of tasks, yet they remain constrained by several fundamental limitations: fixed-length context windows, quadratic attention complexity, and inadequate long-term memory capabilities. These issues hinder their performance when reasoning over extended sequences or retaining relevant information over time.

KRONOS (Knowledge Retention with Optimized Neural Organization System) addresses these challenges by introducing a more scalable, memory-efficient, and reasoning-capable architecture. Inspired by Google's TITANs framework, KRONOS implements a *dual memory system* that blends conventional attention mechanisms for short-term processing with a dedicated neural module for long-term knowledge retention.

The system is built around four core innovations:

1) **Enhanced Surprise Metrics** – A composite scoring system that integrates prediction error, attention entropy, gradient diversity, and self-information. This allows the model to more accurately identify and prioritize important or novel information for memory retention.
2) **Adaptive Forgetting** – A reinforcement learning-based mechanism that evaluates the relevance and utility of stored information, enabling the model to selectively retain or discard content over time for more efficient memory usage.
3) **Latent Space Thinking** – Reasoning is performed in a compressed, high-level latent space rather than through token-by-token generation. This allows KRONOS to carry out more abstract and efficient problem solving and inference.
4) **Concept-Based Memory** – Information is stored within a hierarchically structured, semantically organized memory system. This enables more effective retrieval and promotes long-term understanding through concept-level associations rather than surface-level token recall.

Together, these innovations enable KRONOS to overcome the scaling and memory bottlenecks inherent to vanilla transformers. The result is a model architecture better suited for tasks requiring long-horizon reasoning, contextual understanding, and knowledge retention. This report outlines KRONOS's design principles, technical advancements, development roadmap, and its potential to redefine the future of memory-augmented AI systems.

II Related Works

Prior to KRONOS, several lines of research have addressed transformer memory, efficiency, and reasoning. Key contributions include:

Memory-Augmented Sequence Models

Early work on memory-augmented neural networks, such as Neural Turing Machines and Memory Networks, introduced external memory modules to store and retrieve information beyond fixed hidden states. These architectures inspired later efforts to integrate long-term storage with attention mechanisms.

Extended Context Transformers

Transformer-XL introduced segment-level recurrence and relative positional encodings to capture longer dependencies with negligible additional cost. **Compressive Transformers** further extended this by compressing older activations into a secondary memory, allowing very long context windows with controlled memory growth.

Test-Time Memorization (TITANs)

Behrouz et al. proposed **TITANs** (*Titans: Learning to Memorize at Test Time*), which augments transformers with a learnable, updatable memory module that can store novel information as it appears during inference. This approach demonstrated significant gains in tasks requiring adaptation to new data.

Continuous Concept Pretraining (CoCoMix)

Tack et al. explored **CoCoMix: LLM Pretraining with Continuous Concepts**, where models learn to represent and combine semantic "concept" vectors in a continuous latent space, improving transfer and generalization on downstream tasks.

Latent-Space Reasoning (CoCoNut)

Hao et al. introduced **CoCoNut: Training Large Language Models to Reason in a Continuous Latent Space**, shifting reasoning from token-level chains-of-thought into a more compact latent representation. This yields faster inference and more coherent multi-step reasoning.

KRONOS builds on and unifies these threads—leveraging linear-scaling attention, test-time memory updates, continuous-concept representations, and latent-space reasoning—while adding adaptive forgetting and hierarchical, concept-based memory organization.

III Methodology

Core Architecture

The KRONOS Transformer extends the standard Transformer model by incorporating a dual-memory framework and a chain-of-thought mechanism (Coconut). The implementation is modular, built in PyTorch, and supports flexible configuration of each component. Figure 1 illustrates the overall system.

Key augmentations over the base Transformer include:

- **Segmented Attention:** Inputs are divided into fixed-size segments, each undergoing self-attention, followed by cross-segment fusion to ensure global context.
- **Dual Memory System:** A short-term *Neural Memory* for rapid adaptation and a long-term *Persistent Memory* for conceptual storage.
- **Layer Gating:** Dynamic gating modules regulate the flow of information among attention, feed-forward, and memory submodules.
- **Continuous Positional Embeddings:** Axial sinusoidal embeddings for handling arbitrarily long sequences without positional overflow.
- **Coconut Integration:** Chain-of-thought latent reasoning tokens (BOT/EOT) for enhanced intermediate deliberation.

Each transformer layer is instantiated with optional toggles for memory modules and gating, allowing comprehensive ablation experiments.

Memory Integration Approaches

Our architecture incorporates two complementary memory modules designed for different temporal scopes.

1) Neural Memory Module

The Neural Memory serves as a short to medium-term store that adapts rapidly to incoming data. It follows a surprise-driven, momentum-update formulation:

This module features:

– Surprise-based prioritization to allocate memory slots to novel inputs;
– Momentum-based updates ensure stability;
– Gating to filter out irrelevant updates;
– Analytical form:

$$\mathbf{M}_t = \alpha_t \odot \mathbf{M}_{t-1} + (1-\alpha_t) \odot \mathbf{W}_h \mathbf{h}_t, \tag{1}$$

$$\alpha_t = \sigma\big(f_{\text{forget}}(\mathbf{s}_t, \mathbf{h}_t, \mathbf{M}_{t-1})\big). \tag{2}$$

Algorithm 1 Neural Memory Update Process

```
1: Input: Hidden states $\mathbf{h}_t$, previous memory $\mathbf{M}_{t-1}$, attention probs $\mathbf{A}_t$
2: Output: Updated memory $\mathbf{M}_t$, memory readout $\mathbf{o}_t$
3: Compute surprise: $\mathbf{s}_t = \text{Surprise}(\mathbf{h}_t, \mathbf{M}_{t-1})$
4: Compute weights: $\mathbf{w}_t = \text{AttentionWeights}(\mathbf{h}_t, \mathbf{M}_{t-1}, \mathbf{s}_t, \mathbf{A}_t)$
5: for each slot $i$ do
6:     $\alpha_t^i = \sigma(f_{\text{forget}}(\mathbf{s}_t^i, \mathbf{w}_t^i))$          ▷ adaptive forgetting
7:     $\mathbf{M}_t^i = \alpha_t^i \odot \mathbf{M}_{t-1}^i + (1 - \alpha_t^i) \odot g(\mathbf{h}_t)$
8: end for
9: Read: $\mathbf{o}_t = \text{ReadOp}(\mathbf{h}_t, \mathbf{M}_t)$
10: return $\mathbf{M}_t, \mathbf{o}_t$
```

2) Persistent Memory Module

The Persistent Memory organizes long-term knowledge into semantic buckets:

Algorithm 2 Persistent Memory Operation

```
1: Input: Hidden states $\mathbf{h}_t$, prior buckets $\mathcal{B}_{t-1}$
2: Output: Updated buckets $\mathcal{B}_t$, retrieval output $\mathbf{p}_t$
3: Compute concept: $\mathbf{c}_t = \text{EmbedConcept}(\mathbf{h}_t)$
4: Compute similarities: $s_i = \text{cosine}(\mathbf{c}_t, \mathbf{b}_i)$ for each bucket $i$
5: if $\max(s_i) < \theta$ then
6:     Create new bucket with $(\mathbf{c}_t, \mathbf{h}_t)$
7: else
8:     Update bucket $j = \arg\max_i s_i$ with $(\mathbf{c}_t, \mathbf{h}_t)$
9: end if
10: Retrieve: $\mathbf{p}_t = \text{Retrieve}(\mathbf{h}_t, \mathcal{B}_t)$
11: return $\mathcal{B}_t, \mathbf{p}_t$
```

This design ensures:
– Concept-driven clustering;
– Dynamic bucket creation for novel semantics;
– Sparse addressing for efficient retrieval;
– Formalism:

$$sim(\mathbf{c}, \mathbf{b}_i) = \frac{\mathbf{c} \cdot \mathbf{b}_i}{\|\mathbf{c}\| \|\mathbf{b}_i\|}, \tag{3}$$

$$\text{assign} = \begin{cases} \text{new bucket}, & \max_i sim < \theta \\ \arg\max_i sim, & \text{otherwise.} \end{cases} \tag{4}$$

Attention Mechanisms

We implement segmented self-attention with cross-segment fusion and optional sliding windows (Figure **??**). This supports efficient processing of sequences beyond typical Transformer limits.

Key elements:
– **Segmented Self-Attention:** Local attention within each segment;
– **Cross-segment Fusion:** Aggregation layer combining segment outputs for global context;
– **Sliding Windows:** Overlapping segments to preserve long-span dependencies;
– **Runtime Configurability:** Adjustable segment size and fusion patterns.

Formally:

$$\text{Attn}_{seg}(Q, K, V) = \text{softmax}\big(\tfrac{QK^T}{\sqrt{d_k}} \odot M_{seg}\big)V, \tag{5}$$

$$\text{Attn}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O. \tag{6}$$

State Management

The `TransformerState` Dataclass (Listing **??**) captures:

- Memory states (neural and persistent),
- KV-caches for fast autoregression,
- Value residuals, latent-mode flags, and last hidden state.

Optimizations include KV-cache reuse and lightweight checkpointing of memory snapshots.

Coconut: Chain of Continuous Thought

The Coconut mechanism interleaves latent reasoning steps between BOT and EOT tokens (Figure **??**). Algorithmically:

Algorithm 3 Coconut: Chain of Continuous Thought

```
Input: context $x$, params $\theta$, steps $n$
state ← None, output ← $x$
while not finished do
    logits, state ← Transformer(output[-1:], state)
    token ← Sample(logits)
    if token is BOT then
        Enter latent mode; perform $n$ dummy steps
        token ← EOT
    end if
    Append token to output
    if token is EOT then exit latent mode

        return output
```

This enables multi-step hidden deliberation, improving multi-hop reasoning without incurring token-generation cost. Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

IV Results

Theoretically, KRONOS should outperform both TITANs and vanilla transformers on large, diverse datasets due to:

- *Latent Space Thinking:* Compressed high-level reasoning representations reduce token-level operations.
- *Concept-Based Memory Binning:* Hierarchical semantic buckets enable efficient organization and retrieval, unlike the flat memory in TITANs or no memory in vanilla transformers.

However, on our current small, homogeneous (“brown”) dataset, all three models show similar performance. We attribute this to:

1) **Dataset Size:** Limited scale prevents KRONOS’s advanced modules from realizing their full advantage.
2) **Dataset Homogeneity:** Low domain diversity means most relevant context fits within the transformer’s default window.

As data volume and heterogeneity increase, we expect KRONOS’s innovations to yield larger gains. Future work will benchmark on larger, more varied corpora.

IV-A Proposed Evalutions:

- **Needle in a Haystack:** can be used to bench mark effective context size for a architecture
- **BABIL:** (multi needle in a haystack with downstream reasoning) can be used to evaluate both memory fetching capabilities and additionally reasoning capabilities on the retrieved context.

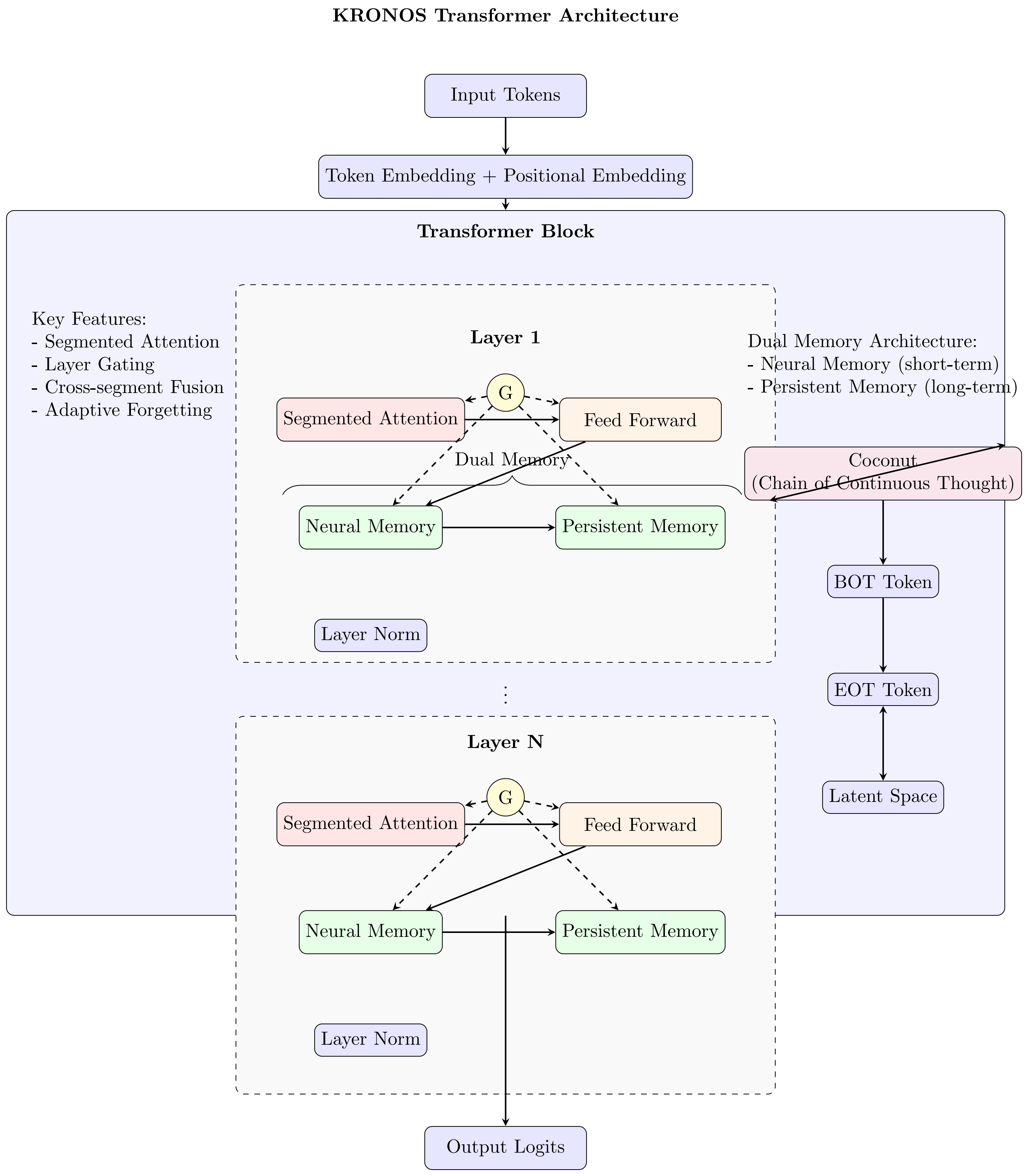
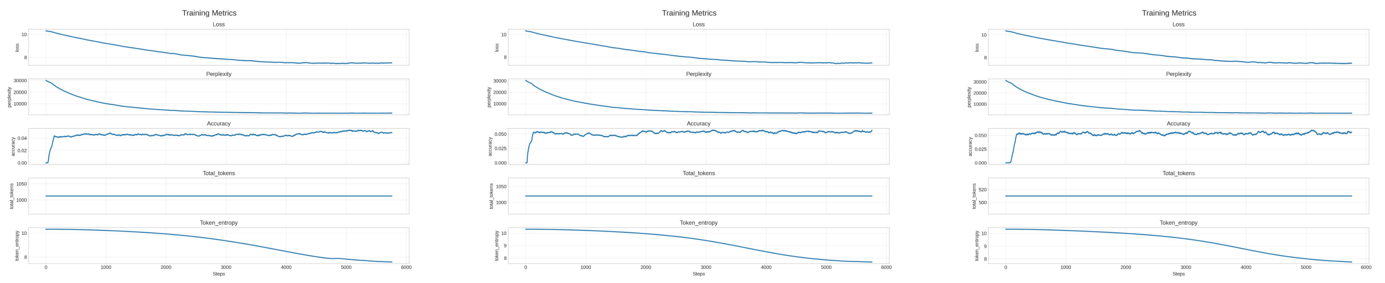


Fig. 1: Architecture diagram of the KRONOS Transformer with Dual Memory and Coconut (Chain of Continuous Thought)



(a) KRONOS (b) TITANs (c) Vanilla Transformer

Fig. 2: Model performance comparison on the small, homogeneous dataset