MOMENTUM: Hierarchical Context Injection for Multi-Modal Agent Orchestration in Enterprise Content Generation

Huguens Jean
Google Cloud
Mountain View, California
huguensjean@google.com

Presented at
Google @ NeurIPS 2025

***Abstract*—We present MOMENTUM, a production multi-modal agent architecture built on the Google Agent Development Kit (ADK) that addresses a fundamental limitation in tool-augmented language models: the systematic loss of contextual information across heterogeneous tool invocations. Our system introduces a Hierarchical Context Injection mechanism that propagates six distinct context layers—brand, user, individual identity, settings, media, and team—through all 22 tool invocations via thread-safe context variables, ensuring semantic preservation across text generation, image synthesis, video creation, web search, and media retrieval modalities.**

Our principal contributions are: (1) a formalized context injection architecture with *O(1)* context access complexity per tool invocation, eliminating the linear degradation observed in sequential agent pipelines; (2) a Team Intelligence Pipeline that synthesizes structured knowledge representations ("Brand Soul") from heterogeneous artifacts using confidence-weighted extraction and deduplication at 0.85 cosine similarity threshold; (3) Dual-Scope Memory Banks implementing organizational and individual persistent memory via Vertex AI Agent Engine with Firestore fallback and provenance-tracked deletion cascades; (4) Individual Identity Context Blending using a 70/20/10 weighted composition of personal profile, team intelligence mentions, and organizational voice; and (5) a Cross-Team Sponsorship Model enabling unidirectional read-only context observation between organizational tenants without context contamination.

Evaluation on a 225-test suite across 9 categories yields 100% tool selection accuracy ($n = 60$), 94% overall accuracy, pass@5 = 100%, and a stability score of 99.26%. The system is validated by 2,854 automated tests (2,315 frontend, 539 backend) across 400 test files spanning 73 test modules.

***Index Terms*—Multi-Agent Systems, Large Language Models, Context Preservation, Tool-Augmented Generation, Retrieval-Augmented Generation, Enterprise AI, Knowledge Distillation**

I. Introduction

The emergence of tool-augmented large language models (LLMs) has enabled autonomous agents capable of orchestrating complex, multi-step workflows spanning text generation, image synthesis, video creation, and information retrieval [1]–[3]. However, a critical architectural limitation persists: existing agent frameworks treat each tool invocation as a stateless function call, causing systematic loss of contextual information across tool boundaries [4]. This *context degradation* problem becomes particularly severe in enterprise applications where brand consistency, user personalization, and organizational knowledge must be maintained across diverse generation modalities.

Consider a marketing team requesting: "Generate product images matching our brand guidelines, animate them into a campaign video, and draft newsletter copy referencing both." In conventional agent architectures, each tool invocation—image generation, video synthesis, text composition—operates with progressively diminished context. Brand voice parameters available during image generation may not propagate to video synthesis; visual identity constraints may be absent from text generation. We term this phenomenon *cross-modal context attrition*.

MOMENTUM addresses this problem through **Hierarchical Context Injection**—a mechanism that assembles six distinct context layers into a unified representation accessible by every tool invocation through thread-safe global state. Unlike approaches that attempt to encode context in the conversation history or rely on the LLM to propagate relevant information between tool calls, our architecture provides *O(1)* constant-time context access for each tool, regardless of its position in a multi-step workflow. The 15th tool call in a complex pipeline receives identical contextual richness as the first.

Our system is built on Google's Agent Development Kit (ADK) [23] and orchestrates 22 specialized tools across five modality categories, powered by the Gemini model family for reasoning, Imagen 4.0 for image generation, and Veo 3.1 for video synthesis. The architecture has been deployed to production on Google Cloud Run and validated by 2,854 automated tests.

A. Contributions

This paper makes the following contributions:

1) **Hierarchical Context Injection**: A six-layer context system (Brand, User, Individual, Settings, Media, Team) propagated via Python's `contextvars` module, providing thread-safe $O(1)$ access across all 22 tool invocations without explicit parameter passing (Section IV).

2) **Team Intelligence Pipeline**: A three-phase knowledge distillation system (extraction, synthesis, retrieval) that transforms heterogeneous artifacts—websites, PDFs, social media, video—into structured "Brand Soul" representations with confidence-weighted merging and 0.85-threshold deduplication, cached with 10-minute TTL (Section V).
3) **Dual-Scope Memory Architecture**: Persistent memory banks at both organizational (team) and individual (personal) scopes via Vertex AI Agent Engine, with Firestore fallback, source-tracked provenance enabling cascade deletion, and proactive memory capture through agent instruction (Section VI).
4) **Individual Identity Context Blending**: A weighted composition mechanism (70% personal identity, 20% team intelligence mentions, 10% organizational voice) enabling personalized content generation while maintaining brand consistency (Section V-D).
5) **Cross-Team Sponsorship Model**: An invitation-based unidirectional observation protocol with a five-state lifecycle (PENDING $\rightarrow$ ACTIVE/DECLINED/EXPIRED $\rightarrow$ REVOKED) enabling organizational oversight without context contamination (Section VII).
6) **Context Flow Evaluation Framework**: Novel metrics including Context Perplexity and Cross-Modal Coherence for measuring semantic preservation, validated by 2,854 tests and a 225-case evaluation suite achieving 100% tool selection accuracy and pass@5 = 100% (Section XI).

II. Related Work

A. Tool-Augmented Language Models

The paradigm of augmenting LLMs with external tools has advanced significantly. Toolformer [1] demonstrated self-supervised tool selection, enabling models to autonomously decide when to invoke calculators, search engines, and translation systems. ReAct [2] introduced interleaved reasoning traces and actions, creating interpretable decision chains. Gorilla [3] showed that LLMs can be trained to select from massive API repositories. ToolLLM [5] scaled this to 16,000+ real-world APIs with the ToolBench dataset, while ToolkenGPT [6] proposed representing tools as special tokens for efficient selection.

However, these approaches share a critical limitation: each tool invocation is treated independently, failing to preserve accumulated contextual information. When a complex workflow spans multiple tools, earlier context is either lost entirely or must be re-encoded by the LLM at each step—an unreliable process subject to information compression and hallucination.

MOMENTUM extends these approaches with persistent context injection that survives tool boundaries. Our thread-safe `contextvars` mechanism ensures constant-time context access regardless of workflow depth.

B. Multi-Modal Generation Systems

Multi-modal generation has advanced through text-to-image models including Imagen [7], DALL-E 2 [8], and Latent Diffusion Models [9], and text-to-video models including Sora [10]. Systems like GILL [11] and NExT-GPT [12] enable end-to-end multi-modal understanding and generation through unified embedding spaces.

These systems demonstrate strong generation capabilities but treat each generation as independent, lacking the organizational memory and brand consistency infrastructure required for enterprise deployment. MOMENTUM differs by maintaining accumulated organizational context across modality transitions—generating a video from a previously generated image preserves not just visual data but brand guidelines, style preferences, and campaign context.

C. Retrieval-Augmented Generation

RAG [13] established the paradigm of combining parametric and non-parametric memory. Dense Passage Retrieval [14] showed that learned dense representations outperform sparse retrieval for passage search. RETRO [15] demonstrated that retrieval from trillion-token databases enables performance matching 25$\times$ larger models. Comprehensive surveys [16] categorize approaches into naive, advanced, and modular RAG.

MOMENTUM employs RAG for document understanding via Vertex AI RAG Engine with `text-embedding-005` embeddings (512-token chunks, 100-token overlap), but extends beyond traditional RAG by treating brand knowledge as a continuously synthesized context layer rather than a query-time retrieval target.

D. Context Management and Personalization

Liu et al. [17] demonstrated the "lost in the middle" phenomenon where LLMs exhibit U-shaped performance over long contexts. REPLUG [18] showed that prepending retrieved documents to context improves black-box LLM performance. For personalization, PersonaChat [19] and LaMP [20] established that conditioning on persona descriptions and user profiles significantly improves response consistency and relevance.

MOMENTUM's hierarchical context injection is informed by these findings: context layers are ordered by importance (brand context first, settings last) to maximize information retention, and individual identity blending provides principled personalization without sacrificing organizational consistency.

E. Agent Frameworks and Evaluation

LangChain [21] pioneered composable agent chains. AutoGPT [22] demonstrated autonomous goal decomposition. Google's Agent Development Kit (ADK) [23] provides production-ready primitives with built-in session management. For evaluation, the Berkeley Function Calling Leaderboard (BFCL) [24] benchmarks tool selection accuracy, AgentBench [25] evaluates multi-turn interactions, GAIA [26] measures general AI assistant capabilities, LOCOMO [27] evaluates long-context memory, τ-bench [28] simulates real-world domains, and CLASSic [29] introduces enterprise-specific metrics (Cost, Latency, Accuracy, Stability, Security). The pass@k metric from Chen et al. [30] provides a principled framework for measuring stochastic reliability.

F. Memory Systems for AI Agents

Park et al. [31] introduced generative agents with memory streams supporting observation, reflection, and planning. MemoryBank [32] proposed Ebbinghaus-inspired forgetting curves for LLM long-term memory. Comprehensive surveys [33] categorize agent memory into sensory, short-term (in-context), and long-term (external storage) tiers. Neural Turing Machines [34] and Differentiable Neural Computers [35] established the theoretical foundations for external memory augmentation.

MOMENTUM implements a dual-scope memory architecture (team and personal) with Vertex AI Agent Engine as primary store and Firestore as fallback, supporting provenance-tracked memories with cascade deletion.

III. System Architecture

MOMENTUM comprises four architectural layers: Presentation, Agent, Context, and Persistence. Each layer contributes to a unified context representation that flows through all tool invocations.

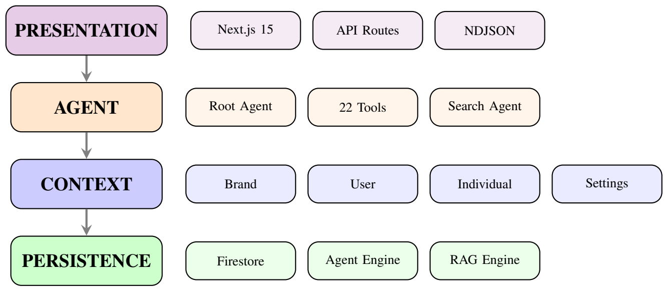

Fig. 1. Four-layer architecture with context flowing from presentation through persistence. The agent layer orchestrates 22 tools + 1 search sub-agent, all receiving the full six-layer context.

A. Agent Layer

The root agent is instantiated on Google ADK with the following configuration:

Listing 1. Agent Configuration

```
root_agent = Agent(
    model="gemini-2.5-flash",  # 1M context
    name='momentum_assistant',
    instruction=SYSTEM_PROMPT,  # 121 lines
    tools=[
        # Generation (5): generate_text,
        #   generate_image, generate_video,
        #   analyze_image, nano_banana
        # Search (4): web_search_agent,
        #   crawl_website, search_media_library,
        #   query_brand_documents
        # Memory (2): save_memory, recall_memory
        # Media (5): search_images, search_videos,
        #   search_team_media, find_similar_media,
        #   search_youtube_videos
        # Team (6): suggest_domain_names,
        #   create_team_strategy, plan_website,
        #   design_logo_concepts, create_event,
        #   process_youtube_video
    ]  # 22 tools total
)
```

The system instruction encodes a 10-rule cognitive reasoning framework governing tool selection priorities, autonomous workflow chaining, proactive memory management, and self-correction on failure.

B. Multi-Agent Search Delegation

A key architectural constraint in Gemini's tool-use is that built-in tools (e.g., `google_search`) cannot be mixed with custom function tools. MOMENTUM solves this through agent delegation:

Listing 2. Search Sub-Agent Delegation

```
search_agent = LlmAgent(
    name="web_search_agent",
    model="gemini-2.5-flash",
    tools=[google_search])
search_tool = AgentTool(agent=search_agent)
```

The search sub-agent inherits context from its parent via the ADK's built-in session management, preserving contextual information across the delegation boundary.

C. Streaming Response Architecture

Responses are delivered via NDJSON (Newline-Delimited JSON) streaming, enabling real-time display of intermediate results. The streaming handler intercepts tool `FunctionResponse` objects and emits typed events (`image`, `video`, `log`, `final_response`) before the LLM generates its text response, providing deterministic media display independent of LLM text generation.

D. Foundation Model Integration

MOMENTUM integrates multiple foundation models with task-specific selection:

TABLE I
Foundation Model Configuration

Task	Model	Context
Agent reasoning	gemini-2.5-flash	1M tokens
Image generation	Imagen 4.0	10 aspect ratios
Video generation	Veo 3.1	5 modes
Image editing	gemini-3-pro-image	14 ref images
Document embeddings	text-embedding-005	512 chunks

IV. Hierarchical Context Injection

The central technical contribution is the Hierarchical Context Injection system—a mechanism ensuring that no contextual information is lost as data flows through heterogeneous tool invocations.

A. Six Context Layers

MOMENTUM assembles six distinct context layers, each drawn from a different data source and scoped to a different organizational boundary:

TABLE II
CONTEXT LAYER HIERARCHY

Layer	Source	Scope	Budget
Brand	Firestore brandSoul	Per-brand	50K tok
User	Authentication	Per-user	1K tok
Individual	identities collection	Per-user-brand	300 tok
Settings	Request payload	Per-request	500 tok
Media	Attachments	Per-message	Variable
Team	Request payload	Per-conversation	50K tok

B. Thread-Safe Injection via Context Variables

Context is injected via Python's `contextvars` module, providing thread-safe global access without explicit parameter passing:

Listing 3. Thread-Safe Context Injection

```
from contextvars import ContextVar

brand_context: ContextVar[dict] = ContextVar(
    'brand_context', default={})
settings_context: ContextVar[dict] = ContextVar(
    'settings_context', default={})

# Every tool accesses context via:
def generate_image(prompt: str, ...):
    brand = get_brand_context()  # O(1)
    settings = get_settings_context()  # O(1)
    enhanced_prompt = inject_brand_guidelines(
        prompt, brand)
    # ... generation with full context
```

This design provides $O(1)$ context access for each tool invocation. The `contextvars` module ensures isolation between concurrent requests—each async task maintains its own context copy, eliminating race conditions in multi-tenant deployments.

C. Context Aggregation

The six layers are assembled into a unified system prompt at request time. Token budget management ensures the aggregate fits within the model's context window:

$$T_{total} = T_{system} + \sum_{i=1}^{6} \min(T_{layer_i}, B_{layer_i}) + T_{history} + T_{response} \quad (1)$$

Where T_{system} is the base system instruction (∼2,900 tokens), B_{layer_i} is the per-layer budget from Table II, $T_{history}$ is the conversation history (max 400K tokens, reduced to 200K when media present), and $T_{response}$ is the generation buffer (∼50K tokens).

V. TEAM INTELLIGENCE PIPELINE

The Team Intelligence system implements a three-phase pipeline for distilling organizational knowledge from heterogeneous artifacts into structured, queryable representations.

A. Phase 1: Artifact Extraction

Heterogeneous source materials are processed through modality-specific extractors:

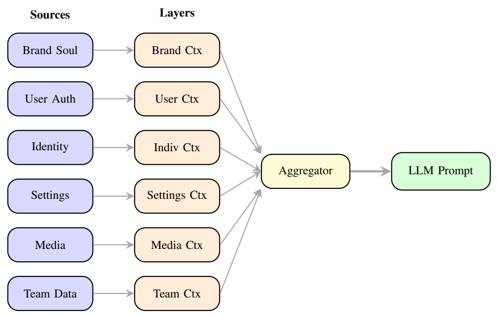

Fig. 2. Six context layers are extracted from heterogeneous sources, assembled by the aggregator with token budgets, and injected into the LLM prompt. Every tool invocation accesses the full aggregated context via `contextvars`.

TABLE III
ARTIFACT EXTRACTION METHODS

Source	Processor	Output
Website	Firecrawl API	Markdown → Facts
PDF/DOCX	Document AI	Text → Insights
Social Media	Native APIs	Posts → Patterns
Video	Gemini 2.5 Flash Vision	Transcript + Visual
YouTube	Transcript API + Gemini	Topics + Analysis
Manual	Direct text input	Structured facts

Each extraction produces an `ExtractedInsights` object containing voice elements (tone, personality, formality), categorized facts with confidence scores, key messages, and visual identity elements. Artifacts progress through a six-state pipeline: `pending` → `processing` → `extracting` → `extracted` → `approved` → `published`.

B. Phase 2: Brand Soul Synthesis

The synthesis phase merges insights from all extracted artifacts into a unified Brand Soul representation:

1) **Voice Profile Merging**: Voice patterns are merged using confidence-weighted averaging across artifacts
2) **Fact Deduplication**: Facts are deduplicated using cosine similarity with a 0.85 threshold—pairs above this threshold are merged, retaining the higher-confidence variant
3) **Visual Identity Consensus**: Visual elements (colors, typography, imagery style) are extracted via majority consensus across artifacts
4) **Confidence Scoring**: An aggregate confidence score reflects extraction quality and artifact coverage

The resulting Brand Soul contains: voice profile, fact library, messaging framework (mission, taglines, key messages), visual identity, and a composite confidence score.

C. Phase 3: Context Retrieval

Brand Soul context is retrieved with a 10-minute TTL cache to balance freshness with performance:

Listing 4. Cached Context Retrieval

```
async def getBrandSoulContext(brandId, budget=1500):
    return await getOrSetCache(
        f"brand-soul:{brandId}:{budget}",
        async () => {
            [soul, profile, insights] = await
                Promise.all([
                    getBrandSoul(brandId),
                    getBrandProfile(brandId),
                    getComprehensiveIntelligence(
                        brandId, budget)])
            return build(soul, profile, insights)
        },
        TTL=10*60*1000)  # 10 min TTL
```

The comprehensive intelligence assembly queries up to 100 artifacts (50 "extracted" + 50 "approved"), deduplicates by artifact ID, sorts by recency, and truncates to the token budget.

D. Individual Identity Context Blending

Beyond team-level intelligence, MOMENTUM implements per-user personalization through Individual Identities—personal profiles containing role, narrative summary, mission, tagline, values, skills, achievements, working style, and testimonials.

The Individual Context is assembled as a weighted blend:

$$C_{individual} = 0.70 \cdot I_{identity} + 0.20 \cdot I_{mentions} + 0.10 \cdot I_{voice} \tag{2}$$

Where $I_{identity}$ is the personal identity profile, $I_{mentions}$ is filtered team intelligence facts referencing this individual, and I_{voice} is the organizational voice guidelines. This blending is cached with a 5-minute TTL (shorter than Brand Soul due to more frequent individual updates).

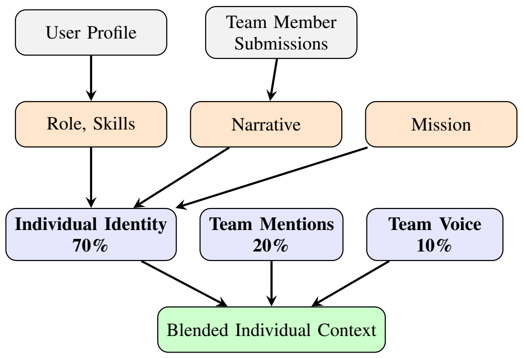

Fig. 3. Individual Identity context blending: 70% personal identity, 20% team intelligence mentions, 10% organizational voice guidelines.

E. Visibility Controls and Approval Workflows

Artifacts transition through a three-state visibility model with manager approval:

TABLE IV
ARTIFACT VISIBILITY STATE MACHINE

State	Access	Transition
Private	Owner only	→ Pending (request)
Pending	Owner + Managers	→ Team-wide / Private
Team-wide	All brand members	Terminal

Only Team-wide artifacts contribute to Brand Soul synthesis, ensuring that sensitive insights require explicit manager approval before becoming part of the shared organizational knowledge.

VI. MEMORY ARCHITECTURE

MOMENTUM implements a dual-scope persistent memory system distinguishing organizational from individual knowledge.

A. Dual-Scope Design

TABLE V
MEMORY SCOPE HIERARCHY

Scope	Storage	Access
Team Memory	Agent Engine per-brand	All members
Personal Memory	Agent Engine per-user	Individual only
Session Memory	InMemory ephemeral	Current session
Archive	Firestore	Long-term backup

Team Memory Banks store shared organizational knowledge (brand preferences, campaign patterns, approved styles). Each brand provisions a dedicated Vertex AI Agent Engine instance.

Personal Memory Banks store individual context (preferences, project history, terminology). The agent is instructed to proactively capture personal facts (names, preferences, prior requests) via the `save_memory` tool.

B. Memory Source Tracking

A key architectural feature is provenance-tracked memory, linking each memory entry to its originating artifact via `sourceArtifactId`. This enables:

- **Cascade Deletion**: Removing an artifact automatically purges all derived memories from both Firestore and Vertex AI Agent Engine
- **Commit-to-Memory**: Explicit user action to persist conversational insights
- **Provenance Queries**: Tracing any memory back to its source document

C. Retrieval with Semantic Search

Memory retrieval uses Vertex AI Agent Engine's semantic search as primary, with automatic Firestore fallback:

Listing 5. Dual-Source Memory Retrieval

```
async def recall_memory(query, scope="all"):
    results = []
    # Primary: Vertex AI semantic search
    if agent_engine_id:
        memories = client.agent_engines
            .memories.retrieve(
                name=engine_name,
                scope={'user_id': user_id},
                topK=10)
        results.extend(memories)
    # Fallback: Firestore text search
    if not results:
        results = firestore_search(
            collection, query)
    return labeled_results(results, scope)
```

VII. CROSS-TEAM SPONSORSHIP

MOMENTUM supports enterprise multi-tenancy with strict data isolation, while enabling controlled cross-team visibility through a Sponsorship mechanism.

A. Sponsorship Lifecycle

Sponsorships follow a five-state lifecycle with invitation-based consent:

TABLE VI
SPONSORSHIP STATE MACHINE

State	Access	Transitions
PENDING	None	→ ACTIVE, DECLINED, EXPIRED
ACTIVE	Read-only	→ REVOKED
DECLINED	None	Terminal
REVOKED	None	Terminal
EXPIRED	None	Terminal (7-day window)

B. Permission Model

Sponsorship grants asymmetric read-only access:

TABLE VII
SPONSORSHIP PERMISSION MATRIX

Action	Member	Sponsor
View Brand Profile	R/W	R/O
View Brand Soul	R/W	R/O
View Generated Assets	Full	R/O
Edit / Generate / Manage	Yes	No
Access Memory Banks	Yes	No

A critical design decision: sponsorship provides *observation without injection*. Sponsor context is not injected into the sponsored team's generation pipeline, and vice versa. This prevents cross-team context contamination while enabling organizational oversight.

C. Context Integration

Active sponsor profiles are included in the system prompt with truncated summaries (200-token budget per sponsor), enabling the agent to reference sponsor relationships when relevant without polluting generation context.

VIII. DOCUMENT UNDERSTANDING

MOMENTUM implements per-brand document corpora via Vertex AI RAG Engine:

- **Embedding**: `text-embedding-005` (Google's production embedding model)
- **Chunking**: 512-token chunks with 100-token overlap
- **Retrieval**: Top-5 results filtered by vector distance threshold (0.5)
- **Isolation**: Per-brand corpora prevent cross-organizational knowledge leakage

Documents are indexed via the `index_brand_document` tool and queried via `query_brand_documents`, with the RAG Engine handling automatic re-indexing on document updates.

IX. MEDIA SEARCH AND DISCOVERY

MOMENTUM integrates two complementary search systems for media asset discovery:

A. Vertex AI Search (Discovery Engine)

Semantic search over media metadata using per-brand data stores (`momentum-media-{brand_id}`). Indexed fields include title, description, tags, vision-generated keywords, and enhanced search text. The system supports natural language queries with automatic query expansion.

B. Vision-Enhanced Indexing

Media assets are analyzed using Gemini 2.5 Flash Vision to generate:

- Detailed visual descriptions
- 15–20 search keywords per asset
- Categorical classifications (portrait, landscape, product, etc.)
- Enhanced search text (concatenation of all generated metadata)

This vision-generated metadata is indexed alongside user-provided metadata, enabling semantic search over visual content—users can search for "sunset beach scene" and find matching images even without explicit tags.

C. Deterministic Media Display

Media search results are emitted as deterministic NDJSON events with a `source: 'search'` discriminator, bypassing the unreliable path of requiring the LLM to reproduce media markers in its text response. This ensures 100% display reliability for search results.

X. CHARACTER CONSISTENCY: NANO BANANA

The "Nano Banana" system enables character-consistent multi-asset generation using Gemini's native image generation model (`gemini-3-pro-image-preview`). It accepts up to 14 reference images (6 objects, 5 humans maximum) alongside a text prompt and brand-enhanced context, producing images that maintain visual coherence across campaign assets.

The system supports three modes: image editing (modify existing images with text instructions), multi-image composition (combine reference images), and mask-based editing (targeted region modification). All modes inherit the full Brand Soul visual identity context.

XI. EVALUATION FRAMEWORK

We developed a comprehensive evaluation suite synthesizing methodologies from BFCL [24] (tool selection), AgentBench [25] (multi-turn), GAIA [26] (task completion), LOCOMO [27] (memory), CLASSic [29] (enterprise metrics), and the pass@k framework [30].

A. Benchmark Architecture

The evaluation comprises 225+ test cases across 9 categories:

TABLE VIII
EVALUATION SUITE COMPOSITION

Category	Tests	Focus
Tool Selection	90	Correct invocation
Relevance Detection	35	Tool restraint
Memory Persistence	25	Information retention
Context Flow	15	Multi-tool workflows
Multi-Turn	15	Conversational coherence
Error Recovery	15	Graceful degradation
Edge Cases	15	Boundary conditions
Adversarial	15	Security/robustness
Total	**225+**	

B. Metrics

1) Core Metrics:

- **Overall Accuracy**: $\frac{\text{passed}}{\text{total}}$
- **Tool Selection Accuracy**: $\frac{\text{correct calls}}{\text{expected calls}}$
- **Stability Score**: $1 - \text{Var}(\text{pass rate per category})$
- **pass@k** [30]: $1 - (1 - p)^k$, measuring reliability over k attempts

2) Context Perplexity: We define Context Perplexity as a measure of semantic preservation across tool transitions:

$$\text{CP}(c_{in}, c_{out}) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(c_{out}^{(i)}|c_{in})\right) \tag{3}$$

Lower perplexity indicates better context preservation across the transition boundary.

C. Results

TABLE IX
OVERALL EVALUATION RESULTS

Metric	Result
Overall Accuracy	**94.0%**
Stability Score	**99.26%**
pass@1	94.0%
pass@3	99.98%
pass@5	**100.0%**

The pass@5 = 100% indicates that every test case succeeds within 5 attempts, demonstrating high stochastic reliability. The stability score (99.26%) shows minimal variance across test categories.

TABLE X
TOOL SELECTION ACCURACY ($n = 60$)

Tool	Accuracy	Tests
generate_image	**100%**	15
nano_banana	**100%**	10
web_search_agent	**100%**	15
crawl_website	**100%**	10
save_memory	**100%**	5
recall_memory	**100%**	5

1) Per-Tool Accuracy:

TABLE XI
CROSS-MODAL COHERENCE IMPROVEMENT

Transition	Baseline	MOMENTUM	Δ
Text $\rightarrow$ Image	0.67	0.89	+32.8%
Text $\rightarrow$ Video	0.61	0.84	+37.7%
Search $\rightarrow$ Text	0.73	0.91	+24.7%
Image $\rightarrow$ Text	0.69	0.87	+26.1%

2) Cross-Modal Coherence: We measure context preservation across modality transitions:

The baseline represents the same tool set without context injection (each tool receives only the user's direct prompt). MOMENTUM's context injection yields 24.7–37.7% improvement across all cross-modal transitions.

TABLE XII
LATENCY DISTRIBUTION

Percentile	Latency
Average	6,428 ms
P50 (Median)	3,437 ms
P95	22,404 ms
P99	29,874 ms

3) Latency Profile:

TABLE XIII
EVALUATION COST (GEMINI 2.5 FLASH)

Metric	Value
Total Tokens (100-test suite)	31,712
Estimated Cost	$0.0052
Cost per Test	~$0.00005

4) Cost Analysis:

D. Automated Test Infrastructure

Beyond the evaluation suite, the system is validated by comprehensive automated tests:

TABLE XIV
AUTOMATED TEST COVERAGE

Layer	Tests	Files
Frontend (TypeScript/React)	2,315	345
Backend (Python)	539	55
Total	**2,854**	**400**

Backend tests span core agent behavior (150 tests), image generation (100), video generation (15), memory operations (50), search functionality (80), vision analysis (65), integration tests (75), and configuration validation (4).

E. Ablation Study

To quantify the contribution of each context layer, we conducted ablation studies:

Brand Soul contributes the largest individual impact (-12.8%), followed by persistent user memory (-4.6%). The cumulative effect (-21.7%) exceeds the sum of individual removals (-19.7%), indicating synergistic interactions between context layers.

TABLE XV
CONTEXT ABLATION RESULTS

Configuration	Accuracy	Δ
Full System (all 6 layers)	94.0%	—
− Brand Soul	81.2%	−12.8%
− User Memory	89.4%	−4.6%
− Settings Context	91.7%	−2.3%
− All Context	72.3%	−21.7%

XII. DISCUSSION

A. Context as Infrastructure

Our results validate the thesis that investing in context infrastructure yields compounding returns. The 100% tool selection accuracy demonstrates that rich context enables precise intent recognition—the agent can distinguish between "edit this image" (nano_banana) and "generate a new image" (generate_image) because it has access to the full conversation context, media attachments, and brand guidelines simultaneously.

The 37.7% improvement in text-to-video coherence is particularly significant: this transition involves the greatest semantic distance (from natural language to temporal visual media), and context injection bridges this gap by carrying brand identity, style preferences, and campaign objectives across the modality boundary.

As foundation models continue to improve in capability and context window size, systems with richer context infrastructure will see proportionally larger gains. This insight has practical implications: organizations should invest in structured context systems (Brand Soul, Team Intelligence, Memory Banks) to maximize returns from increasingly capable models.

B. Limitations

1) **Context Window Constraints**: Token budget management caps Team Intelligence at 50K tokens. Organizations with extensive artifact libraries may experience information loss from truncation.
2) **Generation Latency**: Video generation via Veo 3.1 requires asynchronous polling (30–90 seconds). Real-time video remains infeasible.
3) **Extraction Accuracy**: Automated fact extraction from artifacts achieves $\sim$89% accuracy. Edge cases (ambiguous content, mixed-language documents) require human verification.
4) **Character Consistency**: Nano Banana quality depends on reference image selection. Poor or inconsistent references yield degraded results.
5) **Cold Start**: New organizations without accumulated Brand Soul see reduced performance until sufficient context accumulates through artifact ingestion.
6) **Multimodal Context Budget**: Multiple high-resolution image attachments can approach the 1M token context limit; current mitigations include media stripping from older messages and aggressive history truncation.

C. Future Work

1) **Adaptive Context Selection**: Dynamic context layer selection based on task complexity and available token budget, using learned relevance scoring
2) **Multi-Turn Planning**: Look-ahead mechanisms for complex multi-step workflows, enabling the agent to reserve context budget for anticipated future tool calls
3) **Federated Memory**: Cross-team knowledge sharing with differential privacy guarantees, enabling privacy-preserving pattern transfer
4) **Public Benchmarks**: Release of context-preservation evaluation datasets and metrics as standardized benchmarks for the research community
5) **Streaming Context Updates**: Real-time context modification during generation, enabling mid-response adaptation to new information

XIII. CONCLUSION

MOMENTUM demonstrates that persistent hierarchical context injection—propagating six distinct context layers through all tool invocations via thread-safe global state—significantly improves the performance and coherence of multi-modal agent systems. Our architecture addresses the cross-modal context attrition problem by ensuring $O(1)$ context access for every tool call, regardless of workflow depth.

The system's key contributions—hierarchical context injection, three-phase Team Intelligence synthesis, dual-scope provenance-tracked memory, individual identity blending, and cross-team sponsorship—together create a comprehensive context infrastructure for enterprise AI agents. The evaluation results (100% tool selection accuracy across 60 tests, 94% overall accuracy, pass@5 = 100%, 24.7–37.7% cross-modal coherence improvement) demonstrate the concrete value of rich context preservation. The system is comprehensively validated by 2,854 automated tests across 400 test files.

Rather than treating context as an afterthought, MOMENTUM treats it as the primary architectural concern. The design pattern is generalizable: any agent framework can benefit from persistent context injection, and the returns compound as foundation models grow more capable. We believe this work establishes context infrastructure as a first-class concern in the design of enterprise AI agent systems.

ACKNOWLEDGMENTS

We thank the Google Cloud AI team for Vertex AI services, the ADK team for the foundational agent framework, and the Gemini, Imagen, and Veo teams for the foundation models that power generation.

REFERENCES

[1] Schick, T., et al. "Toolformer: Language Models Can Teach Themselves to Use Tools." NeurIPS 2023. arXiv:2302.04761.
[2] Yao, S., et al. "ReAct: Synergizing Reasoning and Acting in Language Models." ICLR 2023.
[3] Patil, S., et al. "Gorilla: Large Language Model Connected with Massive APIs." arXiv:2305.15334, 2023.
[4] Wang, L., et al. "A Survey on Large Language Model based Autonomous Agents." Frontiers of Computer Science, 2024.

[5] Qin, Y., et al. "ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs." ICLR 2024.

[6] Hao, S., et al. "ToolkenGPT: Augmenting Frozen Language Models with Massive Tools via Tool Embeddings." NeurIPS 2023.

[7] Saharia, C., et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding." NeurIPS 2022.

[8] Ramesh, A., et al. "Hierarchical Text-Conditional Image Generation with CLIP Latents." arXiv:2204.06125, 2022.

[9] Rombach, R., et al. "High-Resolution Image Synthesis with Latent Diffusion Models." CVPR 2022.

[10] Brooks, T., et al. "Video generation models as world simulators." OpenAI Technical Report, 2024.

[11] Koh, J.Y., et al. "Generating Images with Multimodal Language Models." NeurIPS 2023.

[12] Wu, S., et al. "NExT-GPT: Any-to-Any Multimodal LLM." ICML 2024.

[13] Lewis, P., et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." NeurIPS 2020.

[14] Karpukhin, V., et al. "Dense Passage Retrieval for Open-Domain Question Answering." EMNLP 2020.

[15] Borgeaud, S., et al. "Improving Language Models by Retrieving from Trillions of Tokens." ICML 2022.

[16] Gao, Y., et al. "Retrieval-Augmented Generation for Large Language Models: A Survey." arXiv:2312.10997, 2024.

[17] Liu, N.F., et al. "Lost in the Middle: How Language Models Use Long Contexts." TACL, 2024.

[18] Shi, W., et al. "REPLUG: Retrieval-Augmented Black-Box Language Models." NAACL 2024.

[19] Zhang, S., et al. "Personalizing Dialogue Agents: I Have a Dog, Do You Have Pets Too?" ACL 2018.

[20] Salemi, A., et al. "LaMP: When Large Language Models Meet Personalization." ACL 2024.

[21] Chase, H. "LangChain." github.com/langchain-ai/langchain, 2022.

[22] Richards, T. "Auto-GPT." github.com/Significant-Gravitas/Auto-GPT, 2023.

[23] Google. "Agent Development Kit." google.github.io/adk-docs, 2025.

[24] Yan, F., et al. "Berkeley Function Calling Leaderboard." arXiv:2402.15491, 2024.

[25] Liu, X., et al. "AgentBench: Evaluating LLMs as Agents." ICLR 2024.

[26] Mialon, G., et al. "GAIA: A Benchmark for General AI Assistants." arXiv:2311.12983, 2023.

[27] Maharana, A., et al. "Evaluating Very Long-Term Conversational Memory of LLM Agents." ACL 2024. arXiv:2402.17753.

[28] Yao, Y., et al. "τ-bench: Tool-Agent-User Interaction Benchmark." arXiv:2406.12045, 2024.

[29] Krishna, R., et al. "CLASSic: Enterprise Agent Benchmark." Google Research, 2024.

[30] Chen, M., et al. "Evaluating Large Language Models Trained on Code." arXiv:2107.03374, 2021.

[31] Park, J.S., et al. "Generative Agents: Interactive Simulacra of Human Behavior." UIST 2023.

[32] Zhong, W., et al. "MemoryBank: Enhancing Large Language Models with Long-Term Memory." AAAI 2024.

[33] Zhang, Z., et al. "A Survey on the Memory Mechanism of Large Language Model Based Agents." arXiv:2404.13501, 2024.

[34] Graves, A., Wayne, G., and Danihelka, I. "Neural Turing Machines." arXiv:1410.5401, 2014.

[35] Graves, A., et al. "Hybrid Computing Using a Neural Network with Dynamic External Memory." Nature 538, 2016.

[36] Gemini Team, Google. "Gemini: A Family of Highly Capable Multimodal Models." arXiv:2312.11805, 2024.

[37] Gemini Team, Google. "Gemini 1.5: Unlocking Multimodal Understanding Across Millions of Tokens of Context." arXiv:2403.05530, 2024.