Sentence-Level Pooled Embeddings as a Drop-In Replacement for Tokens in LLM Context

Isaiah Ballah
Independent Researcher
isaiah@ballah.ai

Abstract

Modern LLM serving is bottlenecked by KV cache memory, which grows linearly with conversation length although most stored tokens encode old context that the model has already attended over. We test a simple compression scheme: each old sentence is replaced by a single 768-dimensional pooled embedding produced by a frozen sentence encoder (EmbeddingGemma-300M, originally trained for retrieval), and a frozen language model (Gemma 4 E2B with a small LoRA) reads facts back out of those vectors at standard prefix positions. The trainable surface totals 2.2M parameters, or 0.04% of the base model. On synthetic-persona question answering, we obtain 89.3% exact-match fact extraction at 10x compression, scaling sub-linearly through a 200x increase in pool count (76% to 67% accuracy from N=5 to N=1000 pools per item). The dominant remaining failure modes are multi-token surface forms such as personal names and multi-token city names; these are partially mitigated by prepending literal identifier tokens before each pool. We characterize the system as autoencoder-shaped but not an autoencoder, since the encoder is a frozen embedder trained for retrieval rather than for decodability. Replacing it with a purpose-trained encoder is the natural next experiment.

1 Introduction

Modern LLM serving is bottlenecked by KV cache memory more than by compute, and the constraint becomes most acute at the largest open-source scales. A single token of conversation history on Llama 3.1 405B occupies approximately 516 KB of GPU memory [Zhao et al., 2025]; a 10,000-token conversation therefore consumes roughly 5.2 GB per concurrent user. Serving 1,000 concurrent users at that footprint requires more than 5 TB of GPU memory dedicated to KV cache alone, on top of the model's 810 GB of FP16 weights. Cost grows linearly with conversation length, and most of the stored memory is occupied by old context: turns from 20, 30, or 50 rounds ago that the model has already attended over, but that remain in the cache at full word-for-word precision because nothing in the architecture marks them as "done."

This paper tests whether old conversation turns can be compressed into a representation an order of magnitude smaller without losing the model's ability to extract facts from them on demand. The approach is to replace each old sentence with a single 768-dimensional pooled embedding produced by a frozen sentence encoder, and let the language model attend over those pooled vectors at the same prefix positions where their tokens would have been.

A secondary motivation comes from a constraint. Pre-trained sentence encoders are widely available and were trained at compute scales an independent researcher cannot replicate. We were therefore curious whether such encoders, trained for retrieval, preserve content

Preprint. Under review.

well enough that another model can read facts back out of them despite never being trained for that purpose. In effect: are off-the-shelf retrieval encoders unintentionally close to being the encoder half of an autoencoder?

The system tested here is autoencoder-shaped but not an autoencoder in the strict sense. Text passes through a low-dimensional bottleneck of 768-dimensional pooled embeddings from a frozen EmbeddingGemma (300M parameters, originally trained for retrieval), and is then read by a frozen Gemma 4 E2B decoder with a small LoRA [Hu et al., 2022] on its last 16 layers. The trainable surface totals 2.23M parameters, or approximately 0.04% of the base model: a 1.18M linear adapter that bridges encoder space to decoder input space, plus a 1.05M LoRA on the decoder. The full decomposition appears in Section 3. The encoder is never updated, and the primary training signal is fact extraction rather than reconstruction; we test the reconstruction question explicitly in Section 4.4. A complete training run finishes in approximately two hours on a single 24 GB consumer GPU.

The experiment yields a clean result. At 10x compression, the system extracts categorical facts from compressed history at 89.3% exact-match accuracy, up from 56.1% when the same content is crammed into a single pool. The dominant source of error in the single-pool setting is not encoder capacity but pool dilution: when two related facts (such as city and country) share a single 768-dimensional vector, they interfere with each other. Splitting one paragraph across one-pool-per-sentence raises city-extraction accuracy by 67 percentage points without any additional capacity. The architecture also scales: in a cascade study from 5 to 1,000 pools, accuracy degrades sub-linearly (76% to 67% across a 200x increase in pool count), with single-token facts holding above 81% even at N=1,000.

Both motivations point to the same natural follow-up: replace the borrowed retrieval encoder with one purpose-trained as the encoder half of an autoencoder. Section 6 outlines this as concretely as the current results allow.

Contributions.

- An empirical study of sentence-level pooled embeddings as a replacement for raw tokens in LLM prefix context, using only frozen pre-trained components plus a 2.2M-parameter trainable bridge.
- Identification of pool dilution as the dominant single-pool failure mode and characterization of multi-pool chunking as a near-complete fix on synthetic data (89.3% exact match at 10x compression).
- A scaling study from 5 to 1,000 pools per item showing sub-linear accuracy decay, with single-token facts robust through N=1,000 and multi-token surface forms as the visible bottleneck.
- A practical design principle: literal identifier tokens before each pool, with the pool carrying the remaining semantic content. This recovers most of the cross-pool selection accuracy lost to multi-token name fragmentation.

2 Related Work

KV cache compression has been pursued along three roughly orthogonal axes.

Architectural changes at training time. Multi-head Latent Attention (MLA), introduced in the DeepSeek-V2/V3 family [DeepSeek-AI, 2024a,b], projects keys and values into a low-rank latent space inside the attention block itself. MLA reduces per-token KV cache by approximately 7x relative to GQA-based models of comparable size [Zhao et al., 2025], with DeepSeek-V2 reporting a 93% reduction versus a multi-head attention baseline [DeepSeek-AI, 2024a]. More recently, DeepSeek-V4 [DeepSeek-AI, 2026] introduces a Hybrid Attention Architecture whose c4a and c128a operators consolidate the cached K/V entries of multiple consecutive tokens into a single compressed entry via a learned weighted sum, achieving roughly 10% of V3.2's KV cache footprint at 1M-token context. Both MLA and V4's hybrid attention are training-time architectural changes: off-the-shelf checkpoints without these features cannot be retrofitted without retraining.

Mean pooling, as used in this paper, can be read as a uniform-weighted sum over token embeddings; V4's c4a and c128a are then learned weighted sums over groups of consecutive tokens, applied at the K/V level rather than the input level. Replacing our uniform weighting with a learned one is a natural generalization that we did not test; V4's results suggest such learning is effective when trained jointly with the model.

Post-training quantization of cached values. TurboQuant [Zandieh et al., 2026] combines a random orthogonal rotation with Lloyd-Max-optimal quantization (PolarQuant) plus a 1-bit Johnson-Lindenstrauss correction (QJL) to reduce each cached value to roughly 3 bits with no measurable accuracy loss and no retraining. It operates per-element on the existing K/V tensors of any frozen transformer, achieving roughly 6x cache reduction. Earlier work in this axis includes 4-bit and 8-bit KV quantization schemes in production serving stacks.

Sequence-level compression of the prefix. Prompt compression methods reduce input token counts through learned scoring of which tokens to drop [Jiang et al., 2023], and learned-token compression schemes summarize prefix content into a small number of dedicated tokens trained alongside the model [Mu et al., 2023, Ge et al., 2024]. The work in this paper sits in this axis: rather than compressing how individual K/V values are stored, we replace whole sentences of old context with single pooled embeddings produced by a frozen retrieval encoder, before any K/V is computed for those positions.

Compatibility. The three axes are complementary in principle. A model trained with MLA, served with TurboQuant on its cached K/V values, and using sentence-pool compression for old context turns would multiply the respective compression factors. We do not test this combination here, but doing so is a concrete next step (Section 6).

3 Method

3.1 Architecture

Each item in the dataset consists of a context (one or more short sentences about a person), a question about a single fact, and an expected answer. During training and evaluation, the context is converted into a prefix of pooled embeddings, and the question is appended after them as plain tokens.

The conversion proceeds as follows. The context is split into sentences. Each sentence S_i is encoded by EmbeddingGemma (frozen) into a 768-dimensional vector e_i obtained by mean-pooling the encoder's token-level outputs over the (non-padding) input tokens. A learned linear projection $W \in \mathbb{R}^{1536\times768}$ maps each e_i into the decoder's input embedding space, producing $p_i = We_i + b$. The decoder receives a sequence of the form

$$[p_1,\ p_2,\ \ldots,\ p_N,\ \texttt{"Q: "},\ \text{question\_tokens},\ \texttt{"A: "}]$$

where each p_i occupies a single input position and the rest are standard token embeddings.

Each p_i replaces a single position in the input embedding sequence. The token IDs at those positions are pad tokens; nothing in the model's vocabulary references them. The replacement is performed by a forward hook on the decoder's input embedding layer that overrides positions identified by their pad-token IDs. This is the same mechanism Gemma 4 uses internally to inject image patches in multimodal mode, repurposed here for compressed text.

Three properties of this design are worth noting:

1. No special tokens. Earlier iterations of the system used new vocabulary tokens like `<|context|>` to mark pool positions. This caused a silent bug, discussed in Section 5. The current design avoids the issue by reusing existing pad-token positions.
2. The decoder does not know it is reading compressed text. From the decoder's perspective, the prefix consists of vectors at certain input positions, indistinguishable in shape from the embeddings of any token. The decoder's attention reads the prefix the same way it would read tokens.

3. Each pool position is task-agnostic. Once the projected pool replaces the input embedding, the decoder's standard attention determines how to use it. No routing, gating, or task-specific structure is added.

3.2 Trainable surface

The total trainable parameter count is 2.23M, distributed as follows:

Component	Parameters	Description
Linear adapter W	1.18M	1536×768, with bias
LoRA on Gemma 4 E2B	1.05M	Rank 16, α=32, $q_{\text{proj}} + v_{\text{proj}}$, last 16 layers
Total	2.23M	0.04% of base parameter count

EmbeddingGemma is frozen entirely. The Gemma 4 E2B base weights are frozen; LoRA adapters are added to q_{proj} and v_{proj} in the last 16 of the decoder's transformer layers. All training updates flow through W and the LoRA adapters.

3.3 Curriculum

We train the system in three phases, each adding exactly one new structural variable and warm-starting from the previous phase's checkpoint:

Phase 1. Single sentence per item, single pool, real-vocabulary persona templates (e.g. "Marcus Chen is a software engineer at Google in Lagos, Nigeria"). The decoder learns to read facts out of one pool.

Phase 1.5. Multi-sentence paragraphs (3 to 5 sentences per person), still a single pool of the entire paragraph. The decoder learns to read facts out of a denser pool.

Phase 3. Multi-sentence paragraphs, one pool per sentence (five pools per item). The decoder learns to select the correct pool for each question and extract the answer from it.

The warm-start contributes substantially to convergence speed: the Phase 3 model reaches 76% validation exact match by step 600, approximately 1.7% of its full schedule, when initialized from the Phase 1.5 checkpoint. A cold-start ablation that would isolate the warm-start contribution from the multi-pool architecture's contribution is left for future work (Section 6).

3.4 Training and evaluation protocol

Optimization. AdamW [Loshchilov and Hutter, 2019], learning rate 1e−4, no weight decay, no LR schedule, no dropout. Effective batch size 1 (loss backpropagated per example). The decoder is run in bfloat16 with eager attention through N=50 and SDPA above. Pool adapter weights are initialized $\mathcal{N}(0, 0.02^2)$ with zero bias; LoRA adapters use Kaiming-uniform init for the down-projection and zero init for the up-projection.

Data. Synthetic personas drawn from a generator with 250 first names, 210 surnames, 180 professions, 200 employer names (real organizations such as Google, Toyota, NYT), 200 cities, 120 countries, and 12 sentence templates, yielding a combinatorial space of approximately 5×10^{14}. Each phase samples 2,000 unique persona passages for training and 400 for held-out evaluation. Train passages are oversampled to 12,000 examples per epoch. Random seed 42.

Evaluation. In-loop validation runs every 600 training steps on a 50-example subset for fast feedback. End-of-training evaluation runs greedy-decode generation on the full 600-example held-out set. We report exact-match accuracy at the answer level and first-token accuracy. Items are stratified across the five fact keys (profession, employer, city, country, full name); the held-out set contains person–key combinations not seen in training.

Hardware. All experiments ran on a single NVIDIA RTX 4500 Ada (24 GB).

4 Experiments

4.1 Curriculum results: pool dilution and multi-pool chunking

The headline finding is that splitting a paragraph into per-sentence pools recovers a large accuracy gap that single-pool encoding had treated as architectural. Table 1 summarizes the three curriculum phases.

Setup	Pools	Compression	Exact match	First-tok
Phase 1 (1 sentence)	1	—	60.3%	60.3%
Phase 1.5 (5-sentence paragraph)	1	10x (lossy)	56.1%	57.7%
Phase 3 (5 sentence-pools)	5	10x	89.3%	91.1%

Table 1: Curriculum results on synthetic-persona QA. Same content across rows; the only change is whether a 5-sentence paragraph is collapsed into one 768-dim pool (Phase 1.5) or split across five pools (Phase 3). Held-out evaluation, 600 items, greedy decode.

The per-key breakdown clarifies where the gain comes from:

Fact key	Phase 1.5 (1 pool)	Phase 3 (5 pools)	Δ
Profession	75.5%	96.2%	+20.7
Country	73.5%	97.3%	+23.8
Employer	62.5%	94.3%	+31.8
City	21.0%	88.3%	**+67.3**
Name	48.0%	70.2%	+22.2
Overall	56.1%	89.3%	+33.2

Table 2: Per-fact-key extraction accuracy, single-pool vs. multi-pool encoding of identical paragraphs.

The 67-percentage-point jump on city extraction is the most informative result of the curriculum. We initially treated the 21% city accuracy as evidence that EmbeddingGemma's pooled representation could not preserve city tokens distinctly, and spent significant effort exploring richer adapters and learned-query mechanisms aimed at recovering city information. None of those interventions helped. The actual cause is what we refer to as pool dilution: when city and country share a single 768-dimensional pool, both fact tokens compete for representation and the pool ends up mixing them. Split into separate pools, the city sentence has nothing nearby to compete with, and the pool faithfully preserves the city.

The four-percentage-point regression from Phase 1 to Phase 1.5 (60.3% to 56.1%) is the same effect at smaller magnitude: a single pool covering five sentences is a denser version of the same dilution pressure.

4.2 Scaling pools per passage

We extend Phase 3 to ten sentences per paragraph, encoded into ten pools (Experiment A). The per-key results are summarized in Table 3.

Pool selection across ten slots is no harder than across five: profession and country accuracy land within 2 percentage points of Phase 3 in both cases. The bulk of the overall gain (15.8 percentage points on name) reflects the longer paragraphs mentioning the persona's name in additional distractor sentences, which strengthens the pooled representation of the name. This is the same mechanism that lifted name accuracy by 20 percentage points between Phase 1 (single sentence) and Phase 1.5 (multi-sentence paragraph).

Fact key	Phase 3 (5 pools)	Experiment A (10 pools)	Δ
Profession	96.2%	98.0%	+1.8
Country	97.3%	99.0%	+1.7
Employer	94.3%	98.0%	+3.7
City	88.3%	94.0%	+5.7
Name	70.2%	86.0%	**+15.8**
Overall	89.3%	95.0%	+5.7

Table 3: Doubling the number of per-sentence pools (5 to 10) on longer single-person paragraphs. Most of the gain is on multi-token names, whose surface form benefits from being mentioned across additional distractor sentences.

4.3 Cross-person selection

The single-person prefix used in Phases 1 through 3 and Experiment A is a benign setting for the selection task: every pool concerns the same person, so the model only needs to choose pools by content type (the city pool, the profession pool, etc.). Production use cases are different. Each "memory entry" in a chat history typically describes a different topic or a different person, and the question singles out the relevant entry by name.

Experiment B tests this by bundling five pools about five distinct people in each item, with the question identifying the target by name (e.g. "What is Yuki Tanaka's profession?"). The model must (i) match the queried name to one of the five pools and (ii) extract the answer from that pool.

Fact key	Phase 3 (1 person)	Exp B (5 people)	Exp B2 (+name tokens)	Δ B→B2
Profession	96.2%	87.3%	93.1%	+5.8
Country	97.3%	80.4%	92.4%	+12.0
Employer	94.3%	69.9%	81.3%	+11.4
City	88.3%	31.6%	45.5%	**+13.9**
Overall	89.3%	66.9%	77.8%	+10.9

Table 4: Cross-person selection. "Exp B2" prepends the literal name tokens before each pool, converting selection by name into an attention-copy operation over identical token IDs.

Experiment B drops 22 percentage points overall versus Phase 3, with city collapsing the hardest (88% to 32%). Inspecting failures shows a characteristic pattern: the model often emits a syntactically valid city, but from a different person's pool than the queried name. Common single-token facts (profession, country) survive partial selection errors because so few tokens are valid; high-variance multi-token facts (city, employer) fail outright when selection is wrong.

The fragmentation hypothesis is straightforward to test. Multi-token names like "Konrad Esposito" tokenize as `[Kon, rad, Esp, osito]` and become smushed together inside the mean-pooled vector, which then has to be decoded back to the same multi-token surface form to be matched against the question. Experiment B2 adds the literal name tokens before each pool as ordinary Gemma vocabulary embeddings:

`[Kon][rad][Esp][osito]` p_{Konrad} `[Yu][ki][ ][Tan][aka]` p_{Yuki} ...

The selection task degenerates from "decode the name from the pool, compare to the query" to "match identical token IDs in prefix and query," which standard attention handles natively.

The result, also in Table 4, is a 10.9-percentage-point overall recovery (66.9% to 77.8%), with profession, country, and employer returning close to their Phase 3 levels. City is the new bottleneck (45.5%): many city names are themselves multi-token ("Kuala Lumpur", "Buenos Aires", "Port of Spain") and they remain inside the pool, where the same fragmentation pressure now applies one level down.

This generalizes to a design principle. Multi-token surface forms with high cardinality, such as personal names and multi-token city names, are best represented as literal tokens in the prefix; the pool should carry the remaining semantic content. The compression cost of this prefix is small (3 to 5 extra tokens per pool) compared to the full sentence the pool is replacing.

4.4 Reconstruction objective

A natural test of the autoencoder framing is whether the pooled prefix carries enough information to reconstruct the source text. Experiment C replaces 50% of the QA training items with a reconstruction objective: given the same pool prefix, the model is asked to generate the original paragraph. Token loss on the reconstruction target is content-word-weighted, with tokens that appear in any of the five fact strings up-weighted by a factor of 3.

Fact key	Phase 3	Experiment C (joint)	Δ
Profession	96.2%	94.8%	−1.4
Country	97.3%	95.2%	−2.1
Employer	94.3%	89.3%	−5.0
City	88.3%	93.0%	**+4.7**
Name	70.2%	72.8%	+2.6
Overall	89.3%	89.0%	−0.3

Table 5: Adding a 50% reconstruction objective. Top-line QA accuracy is preserved; per-key tradeoffs are mild, with city improving and employer regressing.

Top-line QA accuracy is preserved within noise. The interesting per-key shift is on city (+4.7 percentage points), where the reconstruction objective appears to push the city pool to encode city and country distinctly enough that they can be regenerated in correct order. The employer regression (−5.0) is plausibly capacity reallocation; the experiment does not have enough samples to call it definitively.

The capability-level result is what matters. A direct spot-check on held-out items shows that the model can regenerate source paragraphs from the pool prefix:

> Original: "He works out of Abidjan, Indonesia. Konrad Esposito is a barista at Red Cross. Konrad Esposito speaks Mandarin and Portuguese fluently."
>
> Reconstruction: "He works out of Abidjan, Indonesia. Konrad Esposito is a barista at Red Cross. Konrad Esposito speaks Mandarin and Portuguese fluently."

The first pass is token-perfect on this example. Failure cases show characteristic name corruption (e.g. "Rashid Romano" → "Rached Romano"), the same pattern as in QA: rare multi-token surface forms are noisier than common single tokens. Experiment C demonstrates that the pool prefix contains enough information for a downstream model to recover the source text from compressed memory, validating the autoencoder framing as analogy and opening the possibility of human-readable inspection of what each pool actually preserves.

4.5 Scaling study

The headline question for production deployment is whether the architecture handles a real conversation history's worth of pools (100+ memory entries) rather than just five. We run a cascade study from N=5 to N=1000 pools per item, each N warm-starting from the previous N's checkpoint. All runs use the literal-name-token prefix from Experiment B2.

Three findings:

- Sub-linear accuracy decay across two orders of magnitude. Overall accuracy moves from 76.0% at N=5 to 67.0% at N=1000 across a 200x increase in pool count. The

N	Overall	First-tok	City	Country	Employer	Profession
5	76.0%	73.4%	42.7%	90.7%	80.9%	91.2%
10	75.8%	73.2%	42.9%	88.9%	80.5%	93.2%
25	74.6%	74.0%	36.1%	89.3%	83.3%	91.3%
50	73.5%	74.0%	43.4%	88.4%	74.8%	93.5%
100	73.8%	73.8%	41.1%	87.7%	79.6%	90.4%
250	72.5%	69.2%	50.8%	75.4%	75.0%	89.4%
500	78.8%	78.8%	60.0%	91.9%	70.7%	92.9%
1000	67.0%	65.0%	17.6%	81.5%	69.0%	81.5%

Table 6: Scaling study: cross-person selection with literal-name-token prefix, N people per item. Cascade warm-start: each N initializes from the previous N's checkpoint. Held-out evaluation; test-set sizes shrink from 400 items at N=5 to 25 items at N=1000 to keep evaluation cost bounded.

selection-by-name attention pattern that exists at N=5 generalizes to much larger pool sets without architectural change.

- Single-token facts are robust. Profession and country hold above 81% even at N=1000. These facts ride on top of attention to the right pool followed by a single-token decode from that pool, which survives partial selection errors.
- Multi-token surface forms eventually break. At N=1000, city accuracy collapses from approximately 60% to 17.6%. With a thousand candidate cities (most of them multi-token, all encoded only inside their pools), pool-decode confusion dominates. The same fragmentation problem that the literal-name-token prefix solved for names re-emerges at a different level for multi-token cities.

The non-monotonic peak at N=500 (78.8% overall, 60.0% city) is plausibly a regime where the cascade has had several chances to refine attention discrimination across many candidate pools while individual surface-form decoding has not yet collapsed. Test-set size at N=500 is small (40 items), so this peak carries elevated sampling variance; the trajectory across N, not the peak itself, is the result.

5 Discussion and Limitations

The encoder was never trained for this. EmbeddingGemma is a retrieval encoder, optimized to place semantically similar text near each other in vector space. Nothing in its training rewarded preserving the kind of fine-grained content that a downstream model would need to extract specific facts. The fact that fact extraction works at all (89.3% at 10x compression) and that source text reconstructs faithfully (Experiment C) is evidence the retrieval objective happens to preserve enough content to support the task. We treat this as a strong baseline, not a ceiling. Training the encoder on a decodability objective should improve every number in this paper and potentially close the multi-token gap.

Synthetic data only. All results are on synthetic-persona templates drawn from a generator that, while combinatorially large, is structurally simple. Real conversation transcripts contain noise, ambiguity, references across turns, and content far outside the persona-fact distribution we tested. Phase 2 of the curriculum, which attempted training on 250 hand-written real paragraphs, exposed a data-volume regime (a few hundred paragraphs) where the architecture overfits before generalization emerges. Real-text training at scale is a separate experiment.

Numerical content is fragile. Pooled embeddings do not preserve numeric precision. The encoder treats numerically adjacent integers (such as "880" and "881") as nearly identical in vector space, which is helpful for retrieval and harmful for fact extraction. The natural workaround is the same as for names: keep numeric tokens as literal tokens in the prefix.

Methodology lessons. Two methodological habits were unexpectedly load-bearing for this work and would generalize to similar pool-prefix experiments. First, in-loop validation tracking on a 50-example subset every few hundred steps caught an early Phase 2 silent overfit (training cross-entropy 1.0, validation cross-entropy 7.0, validation exact match 0%) within minutes. Without it, the run would have produced a near-zero training loss and a worthless checkpoint after roughly an hour of GPU time. Second, the v2 iteration of this system used vocabulary tokens such as `<|context|>` as pool position markers, which Gemma's tokenizer split into unrelated fragments (`<`, `|`, `context`, `|>`); the override hook listened for the canonical multimodal token IDs, which never appeared in input. Nine training iterations completed in which the override mechanism never fired, and the LoRA learned only superficial answer-shape patterns. The diagnostic that uncovered it was a startup print of observed token IDs versus expected, three lines of code that shifted the entire research direction.

6 Future Work

The natural follow-up to this paper is to train an encoder for encoding rather than rely on one trained for embedding. The two objectives are subtly different. An embedding model is trained to place semantically similar texts near each other in vector space; whether those vectors are decodable back into specific facts by a downstream model is incidental. An encoder in the sense we use the term is trained for exactly that decodability. Joint training of (a) a purpose-built encoder, (b) the bridging adapter, and (c) a decoder LoRA, under a combined reconstruction and fact-extraction loss, is the most direct path to closing the gap that this paper's frozen-embedder experiment leaves open. Experiment C (Section 4.4) shows the embedder already preserves enough information to support reconstruction without ever having been asked to; updating both halves of the system jointly on a decodability objective should sharpen exactly the multi-token surface forms that are this paper's persistent failure mode. We have a working scaffold for the trained-encoder variant in our codebase but have not yet run it at the scale of the experiments reported here.

A cheaper intermediate is instruction-conditioned pooling. EmbeddingGemma supports task prefixes (e.g. `task: search result | query: ...`) that produce pool vectors biased toward the prefix's task. Replacing our uniform mean pool with a prompt-conditioned pool would tilt pool weighting toward decodability of specific fact types without any encoder updates. We view this as a bridge experiment between the frozen-embedder setup tested here and the trained-encoder setup above; whether a single generic prompt gives uniform lift across fact keys, or whether per-task pools are needed, is the empirical question.

Three further directions are worth flagging. First, real text rather than synthetic personas, both for training (a few thousand real paragraphs at minimum, given Phase 2's failure at 250) and for evaluation (chat transcripts, support tickets, agent traces). Second, integration with a production-grade serving stack: the architecture as tested replaces token-level KV cache for old turns with a single pool per sentence, but the practical question is how this composes with paged-attention and continuous-batching infrastructure that real serving systems already depend on. Third, combining the three compression axes discussed in Section 2: a model trained with MLA, served with TurboQuant on its cached K/V values, and using sentence-pool compression for old turns would, in principle, multiply the respective compression factors. Whether that composition preserves accuracy in practice is an open question we have not tested.

7 Conclusion

A frozen retrieval embedder paired with a frozen language model and a 2.2M-parameter trainable bridge compresses sentence-level context by 10x while preserving 89.3% of categorical fact extraction. Splitting paragraphs into per-sentence pools is what closes a 33-percentage-point gap that single-pool encoding had treated as architectural; prepending literal identifier tokens before each pool recovers most of the cross-pool selection accuracy lost to multi-token name fragmentation; and the architecture scales sub-linearly to N=500 pools per item before multi-token surface forms begin to break at N=1000. The system is autoencoder-shaped

but not an autoencoder. Replacing the frozen embedder with a purpose-trained encoder, jointly fine-tuned with the decoder for decodability, is the natural next experiment.

References

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. arXiv preprint arXiv:2405.04434, 2024a.

DeepSeek-AI. DeepSeek-V3 technical report. arXiv preprint arXiv:2412.19437, 2024b.

DeepSeek-AI. DeepSeek-V4: Hybrid attention architecture for long-context inference. Model release, April 24, 2026, 2026. Citation needs verification once a formal technical report is published. V4-Pro: 1.6T parameters, V4-Flash: 284B parameters, 1M-token context. Hybrid Attention Architecture combining Compressed Sparse Attention (CSA) and Heavily Compressed Attention (HCA) with c4a (4x) and c128a (128x) compression operators. Reported 10% of V3.2's KV cache footprint at 1M-token context.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In International Conference on Learning Representations (ICLR), 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In International Conference on Learning Representations (ICLR), 2022. arXiv:2106.09685.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In Empirical Methods in Natural Language Processing (EMNLP), 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In International Conference on Learning Representations (ICLR), 2019.

Jesse Mu, Xiang Lisa Li, and Noah D. Goodman. Learning to compress prompts with gist tokens. In Advances in Neural Information Processing Systems (NeurIPS), 2023.

Amir Zandieh, Praneeth Kacham, Insu Han, Majid Daliri, Lars Gottesbüren, Rajesh Jayaram, Majid Hadian, and Vahab Mirrokni. TurboQuant: Online vector quantization with near-optimal distortion rate. In International Conference on Learning Representations (ICLR), 2026. arXiv:2504.19874. Reduces per-element KV cache to 3 bits without retraining; 6x compression with no measurable accuracy loss.

Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Huazuo Gao, Jiashi Li, Liyue Zhang, Panpan Huang, Shangyan Zhou, Shirong Ma, Wenfeng Liang, Ying He, Yuqing Wang, Yuxuan Liu, and Y. X. Wei. Insights into DeepSeek-V3: Scaling challenges and reflections on hardware for AI architectures. arXiv preprint arXiv:2505.09343, 2025. Reports per-token KV cache: LLaMA-3.1 405B 516 KB, Qwen-2.5 72B 327 KB, DeepSeek-V3 (MLA) 70 KB, Table 1.

A Hyperparameters and engineering notes

Optimization (all phases). AdamW with β_1=0.9, β_2=0.999, ϵ=10^{-8}, learning rate 1e−4. No weight decay, no LR schedule, no gradient clipping. Effective batch size 1; gradients are computed per example.

LoRA. Rank 16, α=32, dropout 0, no bias. Applied to q_{proj} and v_{proj} in the last 16 layers of Gemma 4 E2B. `lora_A` initialized Kaiming-uniform, `lora_B` initialized to zero.

Pool adapter. Single linear layer $W \in \mathbb{R}^{1536\times768}$ with bias. Weight initialized $\mathcal{N}(0, 0.02^2)$, bias initialized to zero.

Encoder configuration. `google/embeddinggemma-300m` loaded via `sentence-transformers`. Pooling is mean over non-padding token embeddings. Encoder input truncated to 64 tokens; this comfortably covers the persona sentences used in this paper.

Decoder configuration. `google/gemma-4-E2B-it` loaded in bfloat16. Eager attention through N=50 pools per item; SDPA above. Maximum sequence length scales with N: 128 tokens for Phase 3 / Experiments A/B/B2/C, up to 5,500 tokens at N=1000 in the scaling study.

Engineering notes from the scaling study. Three optimizations brought the cascade study from an estimated 10 hours to roughly 75 minutes wall clock. (i) Batched encoder pooling. The naive per-sentence `encode_pool(text)` path performed one tokenize-forward-mean cycle per call with a CPU-side mask sum, dominated by per-call synchronization overhead at approximately 25 sentences/sec. Replaced by batch-128 encoding with GPU-side masked-mean and one CPU sync per batch, reaching approximately 2,200 sentences/sec, an 89x speedup. (ii) Activation checkpointing was enabled at N=1000, where the 5,500-token sequences exceeded available activation memory during SDPA backward; this cut peak memory from 31 GB to 19 GB at the cost of approximately 2x slower training, which was the operative trade. (iii) Cascade warm-start. Each N inherits the previous N's adapter and LoRA; per-N step counts dropped from approximately 3,000 (cold) to 1,000–1,500 (cascade) without measurable quality loss.

Compute footprint. The full research arc reported in this paper, from Phase 1 through the N=1000 scaling study, consumed approximately $50 of compute on a single 24 GB consumer-grade GPU.