PAT: BENDING PARAMETER AND CONTEXT SCALING CURVES THROUGH EFFICIENT CONTEXT REWRITES

Anonymous authors
Paper under double-blind review

ABSTRACT

Causal full self-attention (FA) caches token representations without revising them when later evidence arrives. This limits efficient storage and relationship composition. We analyze local context rewriting, which converts completed chunks into memory states trained through future-token prediction. These states compose relationships in parallel ($A \to B$ and $B \to C$ yield $A \to C$) and remove redundant distinctions in the cache, freeing capacity for prediction. Our analysis shows that, under locality assumptions, rewriting reduces the expected layers needed to combine a chain of relationships from linear to logarithmic in its length. The Predictive Abstraction Transformer (PAT) implements these context rewrites within FA's compute budget by attending to a smaller cache. Its KV cache grows at about one-third FA's rate, training is fully parallel, and measured training throughput exceeds NanoGPT's. Experiments show a widening quality advantage over FA as context length and model size grow, while PAT's relative compute falls with context length. At matched compute within each context length, PAT's perplexity reduction grows from 4.2% at 2K to 8.0% at 8K. At approximately 97 EFLOPs, 379M PAT achieves lower perplexity than an FA model with 2.3× as many parameters (861M). PAT also improves recall at lengths beyond training: after training on sequences up to 256 tokens, it achieves 95.3% multi-query associative-recall (MQAR) accuracy at 1,024 tokens, versus FA's 31.9% and 31.7% for KDA.

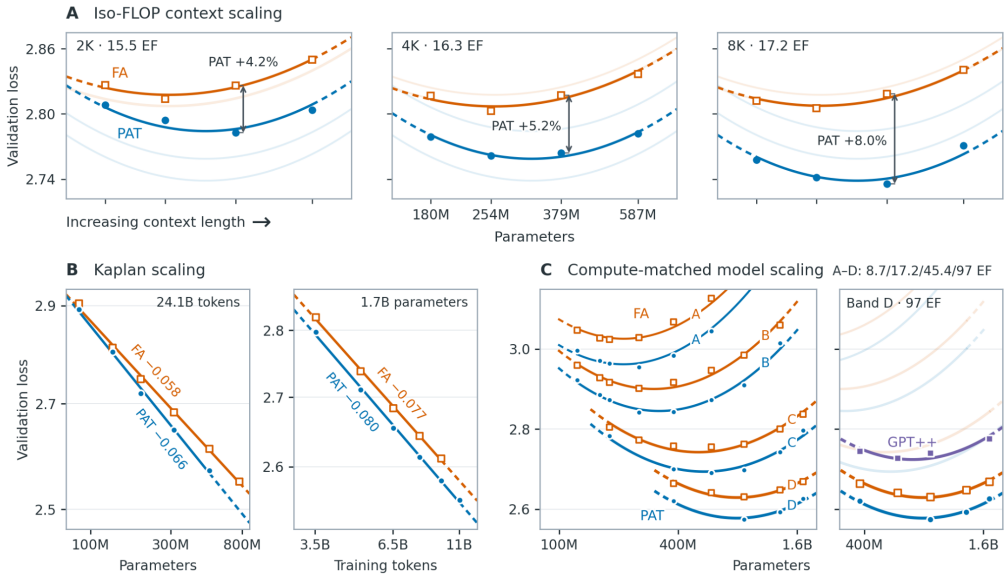


Figure 1: PAT's advantage widens with context and model scale. (A) Compute-matched context scaling: the 379M model's perplexity reduction grows from 4.2% at 2K to 8.0% at 8K. (B) Parameter scaling at 24.1B tokens and data scaling at 1.7B parameters. (C) Model-size sweeps at four compute budgets. Complete results: Appendix B.

1 INTRODUCTION

Full self-attention (FA) underlies modern language models: each new token can attend to every earlier token, and successive layers can retrieve and combine facts across the entire input (Vaswani

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

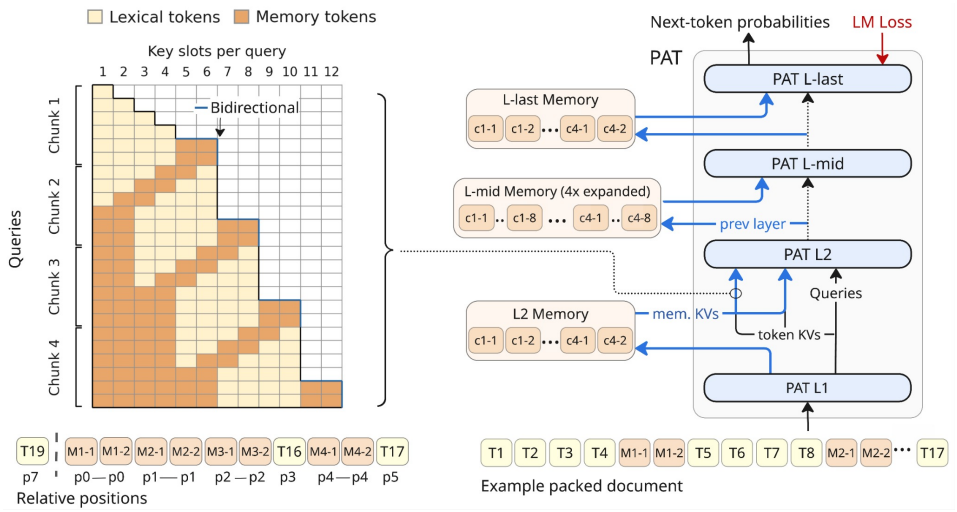


Figure 2: PAT architecture. Input chunks are interleaved with memory tokens. Layer 1 reads locally; its contextualized memories provide layer 2's cached keys and values. Later token queries attend to the preceding 16 tokens and earlier chunk memories in one softmax. Memory queries also read their entire chunk and their own group bidirectionally. Memories skip FFNs and expand from two to eight per chunk in middle layers before returning to two. Memory keys within each chunk share a rotary position; positions advance once per visible token or memory group. All chunks train in parallel. Illustration: four-token chunks; experiments: 16.

et al., 2017). Yet long-context evaluations show that retrieval and reasoning in FA models degrade as sequences grow, even when the facts needed for an answer remain accessible (Liu et al., 2024; Hsieh et al., 2024; Modarressi et al., 2025). In this work, we investigate how to improve long-context prediction within FA's compute budget. Our attention alternative gains a widening advantage over FA as context length and model size increase, while its relative compute falls with context length.

Reliable context use requires storing and composing associations—relationships between pieces of information. For example, $A \to B$ and $B \to C$ together establish $A \to C$ (Hsieh et al., 2024; Modarressi et al., 2025). This requires both storage for prerequisite associations in the cached keys and values (KVs) and enough computation across layers to combine them (Biran et al., 2024; Chen et al., 2024). Thus both what the cache retains and how far the model can compose it determine whether accessible evidence becomes useful.

Causal FA fixes each token's KVs when that token is processed. Consider "Sam is by the bank." Predicting the continuation requires representing both a riverbank and a financial bank. "The bank branch is on Park Avenue in NYC" resolves the ambiguity, but the earlier KVs retain features encoding the uncertainty, occupying capacity. Inferring Sam's location also requires composing Sam→bank with bank→NYC. Because the earlier state cannot incorporate the later location, a later state or upper layer must retrieve and combine them.

Re-encoding the observed context allows its states to incorporate later evidence and compose associations in parallel while prediction remains causal. Chen et al. (2024) demonstrate the depth advantage on a synthetic composition task: under their width and precision conditions, chains of h associations require $\Theta(h)$ causal layers but only $O(\log h)$ bidirectional layers. However, reprocessing every growing prefix costs $O(N^3)$ attention work over N tokens. Scratchpads and pause tokens likewise supply additional states and computation before prediction (Nye et al., 2021; Goyal et al., 2024). The challenge is how to gain the compositional and storage benefits of rewriting within FA's quadratic attention budget.

Our first contribution is showing that local context rewriting preserves the benefits of re-encoding with quadratic attention work. Predictive information in language is concentrated nearby (Hahn et al., 2021; Xiong et al., 2024). We model the decreasing frequency of required long-span computations with a power-law locality assumption (Section 3). Completed chunks are rewritten into memory states trained through future-token loss. Memories at each chunk's end read its tokens and earlier chunk memories; their updated KVs replace token KVs in the next layer's cache. Each chunk is processed through the model once. Within-chunk associations compose in parallel; unfinished

compositions across chunk boundaries increase the expected depth for h successive associations by a bounded factor, preserving $O(\log h)$ expected layers. Our finite-capacity construction also proves rewriting removes any cache distinctions resolved within a chunk, such as the earlier riverbank ambiguity. At fixed capacity, this strictly lowers optimal prediction loss when retaining those distinctions leaves insufficient room for predictive features (Theorem H.2.10; Appendix H.3).

Our second contribution is the Predictive Abstraction Transformer (PAT), which uses these storage and depth gains to realize local rewriting within FA's compute budget by storing fewer memories than source tokens per chunk. Our construction retains logarithmic composition depth with constant additional depth at a fixed chunk size and memory allocation. PAT can also recall any token by its position by distributing values across layers with total memories proportional to chunk size (Appendix H.6). We allocate an average of five memories per 16-token chunk in PAT, so its historical KV cache grows at about one-third FA's rate. This smaller cache saves context-attention work to offset the additional projection cost of memories. At 16K context, PAT uses fewer analytical training FLOPs than FA across the evaluated model sizes. Relative compute decreases with context length, allowing longer contexts to support more memories within FA's budget (Appendix I). This is predictive abstraction: learning a smaller but improved context representation with composed associations and fewer redundant distinctions through future-token prediction.

PAT trains in parallel and directs memory computation toward composition (Figure 2). Each token query attends in one softmax to the previous $K = 16$ token states and memories of the previous chunks, ordered chronologically. All queries read memories of the preceding layer, so rewriting requires no sequential pass on each chunk. Memories skip feed-forward blocks and mix bidirectionally within each chunk to combine retrieved associations in parallel. Direct reads of prior memories carry compositions across chunk boundaries. More memories are allocated in the middle layers to expand capacity after early token features have formed (Lad et al., 2025). PAT also achieves competitive measured training throughput with our optimized GPT implementation and exceeds nanoGPT's (Section 6.5).

Our third contribution is improved parameter and context scaling curves of PAT (Figure 1). At 24.1B training tokens and 16K context, increasing parameters from 124M to 379M nearly triples PAT's perplexity reduction over FA, from 1.2% to 3.3%. At approximately 97 EFLOPs, 379M PAT outperforms an 861M FA model with 2.3× as many parameters. The fitted data-scaling curves remain nearly parallel: at fixed 1.7B parameters, FA needs approximately 15% more training tokens on average to reach PAT's loss across the measured range. With FLOPs matched between architectures at each length, extending context from 2K to 8K widens PAT's perplexity advantage from 4.2% to 8.0%, while FA's loss does not improve from 4K to 8K in those runs (Figure 1). The quality gap grows with model size and context, while PAT's relative compute cost falls with context length.

The gains extend to more reliable use of information in context. At 16K, PAT's seven-task long-context average is 24.80 versus FA's 14.23 for 1.7B models trained on 9.5B tokens, where FA used 7.3% more modeled training FLOPs. After training up to 256 tokens, PAT achieves 95.3% multi-query associative-recall accuracy at 1,024 tokens versus FA's 31.9%. It also generalizes better to changed recall layouts: corrupting non-query bindings yields 100% accuracy versus 73.6% (Sections 6.2–6.3). These results require associations to remain useful across longer contexts and different layouts; PAT improves on both while its KV cache grows at roughly one-third FA's rate. These results support predictive abstraction's advantage of retaining a higher number of composed associations and discarding redundant features that compete for storage.

2 Related work

Latent memories learned through token prediction. These methods encode prompts or chunks into fewer memory states learned through future-token prediction, reducing the token KV cache and subsequent attention work. Gist Tokens compresses reusable prompts, while AutoCompressors and Activation Beacon recurrently process chunks, passing summaries or memory KVs forward (Mu et al., 2023; Chevalier et al., 2023; Zhang et al., 2024a). UniGist and KSA construct memories for all chunks together at each layer using preceding-layer states, but UniGist's memory interactions remain causal, and KSA's writes read only their chunk's token states (Deng et al., 2025; Chu et al., 2026). PAT focuses on improving the quality of context representation through local rewriting while retaining this layerwise parallelism. Multiple attention-only memories jointly reprocess each completed

chunk, interact bidirectionally within it, and read earlier memories directly. These interactions allow associations to be resolved and composed before storage, while replacing token KVs with fewer memory states reduces subsequent attention work. The resulting model improves context scaling: PAT's prediction advantage over FA widens with context length while its relative compute falls.

Additional computation before prediction. Scratchpads train on written calculations followed by answers and generate those calculations at inference; pause tokens are dummy inputs learned through subsequent-token loss and supplied at inference (Nye et al., 2021; Goyal et al., 2024). Both spend additional inference computation on extra sequence positions. Bottlenecked Transformers direct additional computation to cached KVs through an auxiliary bidirectional processor trained with a frozen backbone, adding parameters and extra cache passes (Oomerjee et al., 2026). PAT instead integrates local rewriting into general language-model pretraining from scratch, within FA's training and inference compute budgets and with fully parallel, hardware-efficient training. Parameter-matched and compute-matched comparisons show that this integration yields gains that widen with context length and model size without downstream fine-tuning.

KV cache compression. These methods merge or compress token KVs into fewer entries to reduce storage and attention costs while preserving FA's prediction quality. DMC uses weighted averaging without additional parameters; LoCoCo uses learned pooling; NSA combines learned block compression with selective attention (Nawrot et al., 2024; Cai et al., 2024; Yuan et al., 2025). PAT's local rewrites reduce storage requirements by removing distinctions that later evidence within a completed chunk makes irrelevant to future prediction. Our constructions show how removing these distinctions permits fewer memory states and can improve optimal prediction at fixed cache capacity.

Fixed-state sequence models. Mamba and recurrent attention such as GSA and KDA retain fixed-size states (Gu and Dao, 2023; Zhang et al., 2024b; Kimi Team et al., 2025). PAT grows addressable memory with context at a reduced rate relative to FA. Our synthetic comparisons test recall across longer and altered inputs.

3 Theoretical analysis of context rewriting

Let N count input tokens, K chunk length, D model width and L layer count. Full rewriting converts context into latent memories separate from token states. PAT rewrites each chunk once; earlier memories stay unchanged. Within each layer, memory queries share token attention projections, read the chunk and earlier memories, and supply next-layer keys and values. The baseline uses $\Theta(N)$ memories per layer for full rewriting and $\Theta(K)$ per chunk per layer for local rewriting.

3.1 Full rewriting improves composition and storage at cubic attention cost

An association maps input to output; composing lower-order associations into higher-order ones uses one output as the next input. Full rewriting composes these associations in parallel and removes features made redundant by later evidence. Rewriting every prefix before prediction costs $\Theta(N^3)$ attention pairs per layer over the sequence.

Rewriting composes association chains in logarithmic depth. The task of Chen et al. (2024) lists functions in reverse order, one input–output pair per position. Supplied choices combine with intermediate outputs to select successive inputs; from a starting value, the task requests a final-output bit. With $H = \Theta(\log \log N) \geq 2$ successive associations, $O(\log N)$ coordinates per state, $O(\log N)$ bits per coordinate and constant heads,

$$L_{\text{causal}} = \Theta(H), \qquad L_{\text{full}} = O(\log H).$$

Our construction doubles chain length per layer through parallel memory queries (Appendix H.1).

Rewriting with bounded functions strictly lowers loss. A stored feature U_t may encode uncertainty needed for accurate prediction at position t. Evidence at $t' > t$ can make it redundant: different U_t values then give the same predictive state $S_{t'}$, the required answer probabilities after that evidence.

For a finite task with this redundancy, let C count possible cache states. If these suffice to recover U_t but cannot distinguish every occurring combination of U_t and $S_{t'}$, then

$$\ell_{\text{rewrite}}(C) < \ell_{\text{retain}}(C).$$

Here ℓ is minimum expected negative log-likelihood, and retention must recover U_t exactly. Rewriting frees capacity for differences that still affect prediction. Corollary H.2.5 proves the gap; Theorem H.2.10 realizes it with shared token–memory projections and bounded depth, heads, width, working memory and precision (Appendix H.2.4).

3.2 Local rewriting preserves the gains with quadratic attention work

For $\Theta(K)$ memories per chunk per layer, writing each chunk once reduces cubic work to $O(N^2 + NK)$ attention pairs per layer: $O(N^2)$ to cached memories and $O(NK)$ within chunks. In language, nearby context provides most predictive information; distant context brings diminishing gains (Hahn et al., 2021; Xiong et al., 2024).

We model this locality with a power law for calculations that compose associations or remove redundant features. Let $a(s)$ count, in expectation per prediction, those needing original inputs at least s positions apart. Assume $a(s) \le A/s^\alpha$: A bounds the count and $0 < \alpha < 1$ controls its decrease with distance. Both remain fixed as N, H grow.

Fixed chunks preserve logarithmic expected composition depth. Within chunks, lower-order associations compose in parallel; unfinished compositions across boundaries add sequential layers. Inputs and choices must be available during writing, and results must fit the memories. Let B count these unfinished compositions; averaging over predictions and uniformly shifted boundaries gives $\mathbb{E}B = O(K^{-\alpha})$, hence

$$L_{\text{local}} = O(\log H + B), \qquad \mathbb{E}L_{\text{local}} = O(\log H).$$

At fixed K, extra expected depth is bounded as N grows; larger chunks reduce it (Appendix H.3.4).

Evidence within a chunk preserves the storage improvement. When t and t' lie in one chunk, its rewrite can remove redundant variation in U_t. For updates one bounded writer can implement together, with additive, uniformly bounded loss reductions, the count bound gives $\Delta_K = O(K^{-\alpha})$, where Δ_K is the expected storage improvement missed at chunk boundaries (Appendix H.3.5).

Larger chunks resolve more associations but increase search work. With m memories per chunk per layer and fixed $N, m/K$, larger chunks add token candidates but keep mN/K cached memories per layer. Balancing missed gain $K^{-\alpha}$ with local attention cost K/D relative to projections favors $K_* = \Theta(D^{1/(1+\alpha)})$, assuming fixed loss–work weights and feasible memory and layer budgets. Wider models lower relative search cost; diminishing gains keep K_* sublinear in D (Appendix H.3.5).

3.3 Predictive abstraction preserves the gains with fewer memories

Removing redundant distinctions reduces predictive storage. Predictive abstraction composes and encodes relationships needed for prediction in smaller storage while selectively removing redundant features. For a fixed query, let S give required answer probabilities. If a linear map averages computed features over contexts sharing S, the best linear predictor's finite logits depend only on S.

Let r be the rank of its centered logit table: rows index contexts, columns answers, and pre-softmax scores have row means removed. With d predictive coordinates in each of M accessible memories, freely chosen linear writer–reader maps give

$$M_{\min} = \lceil r/d \rceil.$$

This counts only predictive directions after redundant variation is removed. It matches full attention (FA) loss if FA attains the minimum loss from these features (Appendix H.4).

More memories per layer reduce composition depth. Composing available chain portions within a chunk, with m memories per layer and M across its cached layers, uses

$$L = O(K/m + \log H), \qquad M = O(K + m \log H).$$

Each memory follows a consecutive chain portion from a starting input. Portions advance in parallel for $O(K/m)$ layers, then combine. More memories shorten this within-chunk work. Starting inputs must fit the memories; sources and choices stay available, with $O(K)$ input–output-pair uses including repeats. When feasible, $m = \Theta(K/\log H)$ gives logarithmic depth and $M = O(K)$

storage, including original inputs. Width and precision stay $O(\log N)$; locality preserves expected depth (Appendix H.5).

Values stored across layers permit exact recall. Proposition H.6.1 stores K tokens across memories in $2 \leq L \leq K$ layers with $m = O(K/L)$ and $M = O(K)$. More layers allow fewer memories per layer. This sufficient construction recovers tokens whose positions are supplied before retrieval; repeated requests reconstruct the sequence under Appendix H.6's width and precision conditions.

Longer contexts reduce PAT's relative compute. With fixed memory fraction $0 < \rho = m/K \leq 1$, PAT stores ρN memory states per layer. Across $L \geq 2$ layers, attention to earlier memories has order $L\rho(1+\rho)N^2D$; memory projections and local attention add $L\rho ND^2$ and $LNKD$. The attention-and-projection counts give

$$\frac{\text{PAT compute}}{\text{FA compute}} \leq \rho(1+\rho) + \frac{4(D+K)}{N}.$$

For $\rho(1+\rho) < 1$, PAT uses fewer FLOPs whenever $N > 4(D+K)/[1-\rho(1+\rho)]$. This is a finite-context guarantee. The guaranteed relative saving increases with context length at fixed architecture, approaching $1-\rho(1+\rho)$ in the long-context regime (Appendix I).

4 The Predictive Abstraction Transformer

PAT keeps local token states for immediate prediction and predictive memories for completed chunks. Each $K = 16$-token chunk receives learned memory slots that share the token attention projections (Figure 2).

Reading and writing completed chunks. The first layer reads local states. From layer 2, a token query attends to a sliding causal window of K token states and memories of completed earlier chunks. A memory query reads its entire chunk, the earlier chunks' memories, and its own memory group bidirectionally. Local tokens and earlier memories participate in one softmax. Memory writes incorporate completed local evidence and preceding context. A chunk's memory becomes visible to tokens in later chunks.

Parallel training. Every query reads states from the preceding layer. Token queries, memory queries, and all chunks can therefore be evaluated together at each layer; updated memories supply the next layer's keys and values. During autoregressive decoding, memory writing follows the last token of a chunk and supplies context for the next chunk. Completed earlier chunks keep their stored memories. Memory states receive supervision through the loss on future tokens, while feed-forward blocks operate on lexical tokens.

Memory allocation across depth. PAT uses a $2 \to 8 \to 2$ memory profile: two rows per chunk in the first quarter of layers, eight in the middle half, and two in the remaining layers. The middle layers provide more states for composition. Learned transition maps resize the memory groups between these stages. Averaged across depth, five memories per 16-token chunk retain approximately 31% as many cached KV rows as FA, alongside the local window.

Positions and implementation. Each memory group's keys share its chunk position; query offsets distinguish the writers. Token and chunk positions preserve chronological order when memory groups resize. Thus allocating more writers changes representation capacity while preserving distances between existing context keys. Appendix A specifies transitions, rotary positions, normalization, and experiment-specific settings; Appendix I counts projections, local writes, and context reads.

5 Experimental setup

We compare PAT with full attention (FA) in the same Transformer backbone and document-preserving training pipeline. GPT++ uses the FA backbone with fixed-length, cross-document packing. The shared backbone has rotary positions (Su et al., 2024), SwiGLU, tied embeddings, pre-LayerNorm, and learned query–key scales. Models train on a 100B-token mixture of web text, mathematics,

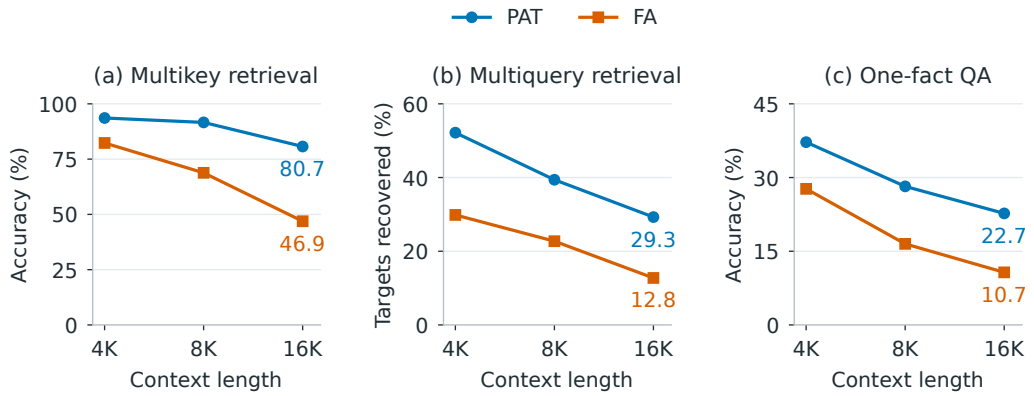


Figure 3: PAT preserves more usable context as inputs grow. The same 1.715B models, trained on 9.52B tokens, answer identical prompts at each input length. The multikey gap grows from 11.3 to 33.8 percentage points between 4K and 16K. Multiquery scores measure the fraction of requested targets recovered; one-fact QA is BABILong QA1. Each point evaluates 1,000 examples. Complete seven-task results, score definitions, and answer-bias controls: Appendix C.

science, and code, with document pieces capped at 16K. Each experiment specifies its parameter, token, compute, or context control. Synthetic recall and generalization use four-block, width-128, two-head models over five seeds. Appendix A gives the training recipe, and Appendices B–G give complete protocols and results.

6 Results

6.1 PAT's prediction advantage grows with model and context scale

Better prediction throughout the compute range. Across four training budgets spanning approximately 8.7–97 EFLOP, PAT lowers the best observed perplexity by 5.5–6.7% (Figure 1). Every completed paired model-size/budget comparison favors PAT. At the largest budget, a 379M PAT reaches 13.73 perplexity, surpassing the best FA checkpoint, an 861M model at 13.89. Saved attention work supports more training: the smaller PAT uses 32.58B tokens versus FA's 13.67B within this budget.

The gain persists at equal training exposure. At 24.12B tokens, PAT improves every paired model size, and its perplexity advantage generally widens toward larger models. At 1.715B parameters, increasing exposure from 3.52B to 9.52B tokens grows the reduction from 2.2% to 3.1%. The complete pipeline comparison with packed-sequence GPT++ is reported in Appendix B.

Longer context strengthens the advantage. Increasing the permitted attention span from 2K to 8K widens PAT's perplexity advantage at all four tested sizes, with training compute matched within each span. At 379M parameters, the reduction grows from 4.2% to 8.0%. Models receive the same document pieces; the allowed attention reads change. In the separate equal-token control, the 8K gain is 6.2% while FA uses approximately 11% more compute. Appendix B gives all curves and matching protocols.

6.2 More context remains usable for retrieval and question answering

We evaluate 1.715B-parameter PAT and FA checkpoints trained on the same 9.52B tokens, using identical prompts and decoding. PAT uses 6.77% less modeled training compute. Figure 3 measures retrieving one or several targets and answering one-fact questions at 4K–16K.

Suite	Task	PAT	FA	NanoGPT	KDA	GSA
MQAR	Mean over 7 evaluation configurations Lengths 64–1,024; 4–256 pairs	99.21 ± 0.90	89.02 ± 2.21	87.33 ± 2.86	85.18 ± 8.38	31.54 ± 37.63
	Extrapolation: length 1,024; 256 pairs	95.27 ± 6.09	31.90 ± 11.15	47.41 ± 9.36	31.66 ± 15.97	10.49 ± 19.55
ICR robustness	Noise: 25% → 0%	61.23 ± 26.67	34.80 ± 35.81	3.10 ± 0.72	76.07 ± 36.41	18.00 ± 20.78
	Other keys corrupted	100.00 ± 0.00	73.58 ± 42.32	0.25 ± 0.35	67.50 ± 28.60	76.20 ± 38.72
	Noise tokens replaced with random vocab items	65.48 ± 33.53	56.06 ± 46.95	0.00 ± 0.00	66.70 ± 21.08	30.50 ± 31.42
	Key length: 3 → 4 tokens	93.93 ± 7.46	60.88 ± 35.57	12.74 ± 14.86	33.97 ± 40.69	48.02 ± 27.25
MAD	Compression Recover input from the final state	19.94 ± 0.74	19.19 ± 0.92	22.15 ± 0.84	20.49 ± 0.79	21.98 ± 0.37
	Fuzzy ICR Recall multi-token key–value bindings	99.99 ± 0.01	99.96 ± 0.08	13.36 ± 0.36	81.08 ± 4.54	14.23 ± 0.71
	ICR Recall single-token key–value bindings	99.93 ± 0.03	100.00 ± 0.00	32.21 ± 0.12	100.00 ± 0.00	99.07 ± 0.12
	Memorization Learn a fixed key–value mapping	61.16 ± 2.77	61.67 ± 0.52	6.55 ± 1.20	2.46 ± 0.90	44.99 ± 2.89
	Noisy ICR Recall bindings amid 80% noise	99.92 ± 0.06	95.05 ± 11.07	93.73 ± 11.89	100.00 ± 0.00	74.30 ± 22.27
	Selective copying Skip blanks while preserving token order	99.82 ± 0.15	99.99 ± 0.00	88.07 ± 19.56	98.99 ± 0.29	77.94 ± 23.03

Table 1: Synthetic accuracy across all five models. Means (%) with sample SD below, across five seeds. Colors show deficits from the best row mean: green ≤ 4 points, yellow > 4 to 20, red > 20. MQAR averages seven configurations; training ends at 256 tokens and validation includes longer lengths. Noise replacements use key/value tokens; arrows mark evaluation changes. Protocols: Appendix D.

Multikey accuracy falls from 93.6% to 80.7% for PAT and from 82.3% to 46.9% for FA. The gap triples from 11.3 to 33.8 percentage points; its paired 95% interval at 16K is [30.5,37.1]. PAT also recovers more requested targets in multiquery retrieval at every length.

The improvement extends to RULER question answering and BABILong QA1 (Kuratov et al., 2024) at all three lengths. These twelve comparisons connect better prediction to more reliable use of facts. Complete downstream results and evaluation controls appear in Appendix C.

6.3 Associative recall generalizes to longer and altered inputs

We compare PAT, FA, KDA, GSA, and nanoGPT-attention over five seeds in a shared backbone. Table 1 reports MQAR (Arora et al., 2024), ICR robustness, and MAD (Poli et al., 2024), including mean MQAR accuracy; Appendix D gives the protocols.

MQAR presents bindings such as $[A, 7, B, 2, \ldots, A, ?]$ and requires recovering the corresponding value. Training spans 64–256 tokens and 4–64 associations. At 1,024 tokens and 256 associations, PAT achieves 95.3% accuracy, versus 31.9% for FA and 31.7% for KDA. Across all seven configurations, PAT averages 99.21%, versus 89.02% for FA and 85.18% for KDA.

ICR trains on three-token keys, such as $[A, B, C, 7, \ldots, A, B, C, ?]$. Changing surrounding records while preserving the queried binding tests whether retrieval depends on the learned layout. Corrupting other keys leaves PAT at 100%, versus 73.58% for FA; extending keys to four tokens yields 93.93% versus 60.88%. The smaller representation preserves associations across changed input structures.

6.4 Useful memory depends on what is written and its computational cost

Learned writing improves the retained representation. At matched training exposure, learned memory writes achieve 19.52 perplexity versus 19.73 for contiguous geometric pooling of trainable lexical states. In a separate ICR control, learned PAT achieves 93.95% recall versus 39.02% for fixed pooling when distractors use the key/value vocabulary. Appendix E gives this control and architectural ablations.

Table 2: Memory allocation at fixed tokens and compute. Perplexity for 161M models at 6.73B tokens or 8.683 EFLOP. Memories are averaged per chunk per layer. Full sweeps: Appendix E.

	Perplexity ↓	
Memories	Fixed tokens	Fixed compute
2.5	20.06	19.62
5 (default)	19.52	19.52
10	19.07	19.98

Memory capacity determines both representation and training exposure. Doubling memory improves PPL per token but worsens it at equal compute (Table 2): 36% greater cost per token reduces training exposure. The native $2 \to 8 \to 2$ profile is best in this compute-matched grid. The best tested chunk changes from 16 tokens at 161M to 32 at 587M; main scaling fixes chunk 16.

Replacing old states improves on augmenting them. Keeping raw-token KVs alongside five memories yields 19.84 PPL, versus 19.59 when those memories replace older token states at equal training exposure. Appendix E gives this FA-plus-memory control, including its attention calibration and compute.

Removing incomplete key states improves retrieval. At fixed retrieval distance, extending context from 1K to 8K lowers FA recall from 128/128 to 2/128, while PAT remains at 128/128. In a separate intervention with completed $[A, B, C, \text{value}]$ bindings in a fixed FA checkpoint, masking earlier key states while retaining value states removes 66% of the cached 2K context and raises recall from 48/256 to 141/256 queries. Attention-suppression and value-zeroing controls identify attention competition.

In a copied-memory assay, repeating earlier fact A increases B's answer margin from memories written for unchanged A+B text in all 32 pairs. Restoring clean memory after value corruption improves the original answer's log odds in all 36 lookup cases. Appendix F gives the independent checkpoints and controls.

6.5 A SMALLER KV CACHE SUPPORTS EFFICIENT TRAINING AND DECODING

On one H100, whole-model training throughput is within 4% of the optimized GPT implementation at both tested shapes and is 1.33–3.13× nanoGPT's.

Cached attention is 1.33–1.42× faster than FA across the measured 2K–16K contexts, including memory projections, transitions, publication, and chunk-boundary passes. Whole-model decoding results appear in Appendix G.

7 CONCLUSION

Long-context prediction depends on the representation retained for future use. PAT rewrites completed context into predictive memories, allowing later evidence to resolve uncertainty and compose associations. Our analysis connects these operations to improved composition and storage within quadratic attention work. The experiments show a growing advantage: at 379M parameters, PAT's perplexity reduction over FA rises from 4.2% to 8.0% as context grows; at 16K, multikey retrieval reaches 80.7% versus 46.9%. These gains accompany parallel training and roughly one-third the cached KV states. Local context rewriting therefore provides a practical way to improve both the quality and efficiency of long-context language models.

Limitations and future work. We fix chunk length for controlled scaling comparisons, although ablations show that the best-performing choice varies with model size. Adaptive chunk lengths and boundaries could allow more dependent associations to compose within each rewrite. PAT reduces attention cost by a constant factor while retaining quadratic context scaling. Whether predictive abstraction can slow cache growth enough to achieve subquadratic attention while preserving prediction quality remains open.

Reproducibility statement

The experimental appendices provide data construction, model configurations, training and evaluation protocols, complete task scores, and intervention controls. The accompanying source package includes the selected scalar results and scripts used to generate the figures. Appendix H states the theoretical assumptions and proofs; Appendix I derives the analytical operation counts.

AI use statement

We used AI agents under authors' supervision to assist with literature research, manuscript editing, mathematical analysis and checks, code implementation, and interpretation of results.

References

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Y. Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. In International Conference on Learning Representations, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/448fc91f669c15d10364ee01d512cc10-Abstract-Conference.html.

Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. Hopping Too Late: Exploring the Limitations of Large Language Models on Multi-Hop Queries. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pp. 14113–14130. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.emnlp-main.781/.

Ruisi Cai, Yuandong Tian, Zhangyang Wang, and Beidi Chen. LoCoCo: Dropping in convolutions for long context compression, 2024. URL https://arxiv.org/abs/2406.05317.

Lijie Chen, Binghui Peng, and Hongxun Wu. Theoretical limitations of multi-layer Transformer, 2024. URL https://arxiv.org/abs/2412.02975v1.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts, 2023. URL https://arxiv.org/abs/2305.14788.

Chenglong Chu, Guorui Zhou, Guowang Zhang, Han Li, Hao Peng, Hongtao Cheng, Hui Wang, Jian Liang, Jiangxia Cao, Kun Gai, Lingzhi Zhou, Lu Ren, Qi Zhang, Ruiming Tang, Ruitao Wang, Xinchen Luo, Yi Su, Zhiyuan Liang, Ziqi Wang, Boyang Ding, Chengru Song, Dunju Zang, Jiao Ou, Jiaxin Deng, Jijun Shi, Jinghao Zhang, Junmin Chen, Lejian Ren, Minxuan Lv, Qianqian Wang, Qigen Hu, Shiyao Wang, Siyang Mao, Tao Wang, Xingmei Wang, Zhixin Ling, Ziming Li, and Zixing Zhang. Kwai Summary Attention technical report, 2026. URL https://arxiv.org/abs/2604.24432.

DeepSeek-AI et al. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Chenlong Deng, Zhisong Zhang, Kelong Mao, Shuaiyi Li, Tianqing Fang, Hongming Zhang, Haitao Mi, Dong Yu, and Zhicheng Dou. UniGist: Towards general and hardware-aligned sequence-level long context compression, 2025. URL https://arxiv.org/abs/2509.15763.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training Language Models With Pause Tokens. In International Conference on Learning Representations, 2024. URL https://arxiv.org/abs/2310.02226.

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces, 2023. URL https://arxiv.org/abs/2312.00752.

Michael Hahn, Judith Degen, and Richard Futrell. Modeling word and morpheme order in natural language as an efficient trade-off of memory and surprisal. Psychological Review, 128(4):726–756, 2021. doi: 10.1037/rev0000269. URL https://alpslab.stanford.edu/papers/2021HahnDegenEtAl.pdf.

John Hale. A probabilistic Earley parser as a psycholinguistic model. In Second Meeting of the North American Chapter of the Association for Computational Linguistics, 2001. URL https://aclanthology.org/N01-1021/.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, et al. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In First Conference on Language Modeling, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Sukjun Hwang, Brandon Wang, and Albert Gu. Dynamic chunking for end-to-end hierarchical sequence modeling, 2025. URL https://arxiv.org/abs/2507.07955.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat After Me: Transformers are Better than State Space Models at Copying, 2024. URL https://arxiv.org/abs/2402.01032.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi Linear: An Expressive, Efficient Attention Architecture, 2025. URL https://arxiv.org/abs/2510.26692.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. BABILong: Testing the limits of LLMs with long context reasoning-in-a-haystack, 2024. URL https://arxiv.org/abs/2406.10149.

Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. Remarkable Robustness of LLMs: Stages of Inference? In Advances in Neural Information Processing Systems, volume 38, 2025. doi: 10.52202/085713-4332. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/bcad07d4bfab51243efaa08b8ed475b3-Abstract-Conference.html.

Roger Levy. Expectation-based syntactic comprehension. Cognition, 106(3):1126–1177, 2008. doi: 10.1016/j.cognition.2007.05.006. URL https://www.mit.edu/~rplevy/papers/levy-2008-cognition.pdf.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Yuzhen Mao, Michael Y. Li, and Emily B. Fox. Simplified sparse attention via gist tokens, 2026. URL https://arxiv.org/abs/2604.20920.

Ali Modarressi, Hanieh Deilamsalehy, Franck Dernoncourt, Trung Bui, Ryan A. Rossi, Seunghyun Yoon, and Hinrich Schuetze. NoLiMa: Long-context evaluation beyond literal matching. In Proceedings of the 42nd International Conference on Machine Learning, volume 267 of Proceedings of Machine Learning Research, pp. 44554–44570. PMLR, 2025. URL https://proceedings.mlr.press/v267/modarressi25a.html.

Amirkeivan Mohtashami and Martin Jaggi. Landmark Attention: Random-access infinite context length for transformers, 2023. URL https://arxiv.org/abs/2305.16300.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens, 2023. URL https://arxiv.org/abs/2304.08467.

Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Łukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. Hierarchical Transformers are more efficient language models, 2022. URL https://arxiv.org/abs/2110.13711.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo M. Ponti. Dynamic memory compression: Retrofitting LLMs for accelerated inference, 2024. URL https://arxiv.org/abs/2403.09636.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show Your Work: Scratchpads for Intermediate Computation with Language Models, 2021. URL https://arxiv.org/abs/2112.00114.

Adnan Oomerjee, Zafeirios Fountas, Haitham Bou Ammar, and Jun Wang. Bottlenecked Transformers: Periodic KV Cache Consolidation for Generalised Reasoning. In *International Conference on Learning Representations*, 2026. https://proceedings.iclr.cc/paper_files/paper/2026/hash/478b06f60662d3cdc1d4f15d4587173a-Abstract-Conference.html.

Michael Poli, Armin W. Thomas, Eric Nguyen, Pragaash Ponnusamy, Björn Deiseroth, Kristian Kersting, Taiji Suzuki, Brian Hie, Stefano Ermon, Christopher Ré, Ce Zhang, and Stefano Massaroli. Mechanistic design and scaling of hybrid architectures. In Proceedings of the 41st International Conference on Machine Learning, volume 235 of Proceedings of Machine Learning Research, pp. 40908–40950. PMLR, 2024. URL https://proceedings.mlr.press/v235/poli24a.html.

Qwen Team. Qwen2.5-7B and Qwen2.5-72B official model configurations, 2024. URL https://huggingface.co/Qwen/Qwen2.5-7B/blob/main/config.json Official 7B configuration; 72B configuration: https://huggingface.co/Qwen/Qwen2.5-72B/blob/main/config.json. Accessed September 26, 2026.

Noam Shazeer. Fast Transformer Decoding: One Write-Head is All You Need, 2019. URL https://arxiv.org/abs/1911.02150.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with Rotary Position Embedding. Neurocomputing, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Kaiyue Wen, Xingyu Dang, and Kaifeng Lyu. RNNs are not Transformers (Yet): The Key Bottleneck on In-context Retrieval, 2024. URL https://arxiv.org/abs/2402.18510.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. Effective long-context scaling of foundation models. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pp. 4643–4663. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.260. URL https://aclanthology.org/2024.naacl-long.260/.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 Technical Report, 2025. URL https://arxiv.org/abs/2505.09388. Model card: https://huggingface.co/Qwen/Qwen3-1.7B-Base.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated Delta Networks: Improving Mamba2 with Delta Rule, 2024. URL https://arxiv.org/abs/2412.06464.

Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. MEGABYTE: Predicting million-byte sequences with multiscale Transformers. In Advances in Neural Information Processing Systems, 2023. URL https://arxiv.org/abs/2305.07185.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention, 2025. URL https://arxiv.org/abs/2502.11089.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. Long context compression with Activation Beacon, 2024a. URL https://arxiv.org/abs/2401.03462.

Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, Peng Zhou, and Guohong Fu. Gated Slot Attention for Efficient Linear-Time Sequence Modeling, 2024b. URL https://arxiv.org/abs/2409.07146.

GUIDE TO THE APPENDIX

Experimental protocols and supporting results appear first; complete theoretical proofs follow them. Section names and page numbers are linked.

Question and evidence	Page
What is held fixed? Data, training, and implementation	14
How do gains vary with resources? Complete scaling results	15
How useful is longer context? All downstream tasks and controls	19
What generalizes? Recall tasks, layouts, and seed variation	20
What should the memory retain? Design and allocation controls	22
What do writers and readers use? Cache interventions	23
What does rewriting cost? Timing protocols and results	25
Why can rewriting help? Theoretical results and proofs	27
How is compute counted? Analytical operation counts	54

A TRAINING DATA, MODEL CONTROLS, AND OPTIMIZATION

Backbone and baselines. PAT and FA share pre-LayerNorm ($\epsilon = 10^{-6}$), full-head rotary positions (base 10,000), SwiGLU, tied token/output embeddings, enabled biases, and zero dropout. Head dimension is 64; the vocabulary/output dimension is 50,304. The SwiGLU intermediate width is approximately $8d/3$, rounded to a multiple of eight. FA substitutes conventional causal attention for PAT's memory attention. GPT++ uses the FA backbone and fixed-length streams across document boundaries. Its validation examples are also packed; PAT and FA evaluate document-preserving pieces. Stock nanoGPT instead uses learned absolute positions, a $4d$ GELU feed-forward block, conventional scaled dot-product attention, and LayerNorm with $\epsilon = 10^{-5}$.

PAT uses 16-token chunks and a $2 \to 8 \to 2$ memory profile: the first $\lfloor L/4 \rfloor$ layers use two slots, the next $\lfloor L/2 \rfloor$ use eight, and the remaining layers use two. Memories share the token attention projections and skip feed-forward blocks and prediction targets. Main scaling learns a separate transition map per 64-dimensional channel group; context scaling shares one map across groups.

Data construction. The corpus contains 100.001B content tokens from 44.05M documents, forming 45.59M document pieces. FineWeb-Edu is drawn from SmolLM-Corpus; the specialist components are from OLMo-Mix-1124. The unrestricted-length branch selects 60B tokens: 30B FineWeb-Edu and 7.5B from each specialist source. A second branch adds 40B tokens from documents of at least 8,192 tokens. Whole-document selection accounts for quota overshoots. GPT-2 ordinary BPE treats literal special-token spellings as ordinary text; corpus selection uses seed 42. Exact duplicate documents are removed across sources, including exact validation duplicates. Documents are split consecutively at 16,384 content tokens, preserving the final remainder. BOS is added to each piece; EOD is added only to the document's final piece.

Training source	Tokens (B)	Validation split	Content tokens
FineWeb-Edu	49.827	Short: 14,508 documents	13,162,292
Algebraic Stack	12.545	Long: 125 documents	1,840,179
arXiv	12.543	Total: 14,633 documents	15,002,471
peS2o	12.543		
StarCoder	12.543		

Table 3: Training mixture and the fixed FineWeb-Edu validation holdout. Original documents shorter than 8,192 tokens form the short split; longer documents form the long split, producing 165 pieces.

Validation measures lexical-token cross-entropy, excluding special-token targets. Combined loss is weighted by content-token counts, with 87.73% of its weight on short documents; perplexity is its exponential. Model/data scaling uses the combined holdout. Context scaling uses the long-document holdout. These populations differ in content as well as length.

Optimization. Language models use AdamW with $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.01, global gradient clipping at 1.0, BF16 computation, and seed 42. Biases, normalization parameters, and learned QK scales are excluded from weight decay. Updates process approximately 100,000 supervised lexical tokens globally. Linear warmup lasts 100M tokens, followed by a stable learning rate of 2.5×10^{-4} through 70% of the target horizon T. Cosine cooldown occupies the last 30%, ending at 2.5×10^{-5}. For $t \geq 0.7T$,

$$\eta(t) = \eta_{\min} + \tfrac{1}{2}(\eta_{\max} - \eta_{\min}) \left[1 + \cos\left(\pi \frac{t - 0.7T}{0.3T}\right)\right].$$

The warmup lies within the initial 70%. PAT 16×1280 at 4K in the equal-token study uses 29.6% cooldown. Training ends after complete updates. Typical updates contain 99.7–99.9K tokens; GPT++ 16×1280 averages 98.26K. Horizons may share a stable-phase training prefix and optimizer state, followed by separate complete cooldowns; such checkpoints are statistically dependent.

Linear and embedding weights initialize from $\mathcal{N}(0, 0.02^2)$; biases are zero and LayerNorm gains one. Residual-output projections use standard deviation $0.02/\sqrt{2L}$. Memories start from one learned EOC embedding. Slot transitions initialize with constant-preserving geometric interpolation and deterministic zero-row-sum noise at 1% relative row norm. Head-group maps initialize identically and then learn independently.

Query–key normalization. Q and K are L2-normalized with $\epsilon = 10^{-6}$, and each head learns a positive scale multiplying cosine similarity. FA/GPT++ and context-study PAT initialize scales at 10. Main PAT adjusts layers reading eight memories per chunk by

$$\alpha_\ell(0) = 10 \, \frac{\log[K_\ell(K_\ell - 1)]}{\log[K_{\mathrm{ref},\ell}(K_{\mathrm{ref},\ell} - 1)]}.$$

Here K_ℓ is the training-only 97.5th percentile of visible-key counts; the reference uses two memories. At 16K, $K_\ell = 7,448$ and $K_{\mathrm{ref},\ell} = 1,874$ give 11.83. Layer zero and unchanged key counts retain 10. Every scale subsequently trains.

B Scaling experiments and completed endpoints

Resource controls. The four model-scaling bands target approximately 8.68, 17.20, 45.35, and 96.98 EFLOP. Each varies model size and adjusts token exposure. Compute counts projections, feed-forward layers, attention edges, memory transitions, and vocabulary projection, averaged over variable-length examples. A multiply-add counts as two FLOPs; training is estimated as three forward passes. This dominant-operation estimate excludes scalar nonlinearities and optimizer work. Parameters include tied embeddings once. Comparisons use completed cooldown endpoints and each architecture's best measured point in the tested grid.

Shape $L \times d$	Parameters (M)	Heads	Fixed-token slice (B)	Context study
12×768	123.7	12	24.12	—
12×896	160.9	14	—	—
14×896	180.2	14	24.12	2K, 4K, 8K
14×1088	253.9	17	24.12	2K, 4K, 8K
16×1280	379.4	20	24.12	2K, 4K, 8K
18×1536	587.3	24	24.12	2K, 4K, 8K
20×1792	861.1	28	24.12 (FA)	—
24×2048	1,312.1	32	—	—
32×2048	1,715.1	32	Data slice	—

Table 4: Model shapes used across scaling studies. PAT transition parameters leave these rounded sizes unchanged. The fixed-token slice contains five completed PAT/FA pairs and an additional FA endpoint.

The fixed-model data slice uses 32×2048 at 3.52, 5.02, 6.52, 8.02, and 9.52B tokens for both architectures, plus PAT at 11.02B. PAT's 3.52B checkpoint also enters compute band C. Each 24.12B fixed-token run stays at peak learning rate through 16.884B tokens and cools for 7.236B.

Context access. The same prepared document pieces are used with 2K, 4K, and 8K sliding attention spans. FA reads token states; PAT reads its local tokens and only memories within the permitted span. The intervention changes accessible history. Matching is within each span, at 15.504, 16.255, and 17.200 EFLOP, respectively. Separate equal-token runs keep one horizon per shape across architectures and spans. Context PAT uses shared transition maps and initial QK scales of 10. The complete study contains 44 trained models.

Shape	Equal tokens (B)	2K FA/PAT (B)	4K FA/PAT (B)	8K FA/PAT (B)
14×896	12.217770	11.884/12.218	11.154/12.185	10.427/12.148
14×1088	8.734455	8.639/8.734	8.209/8.767	7.770/8.805
16×1280	5.860800	5.871/5.861	5.624/5.906	5.368/5.958
18×1536	3.802905	3.861/3.803	3.735/3.850	3.601/3.905

Table 5: Equal-token horizons and approximately equal-compute token horizons for context scaling. Compute matching is performed separately at each length.

Perplexity reduction is $100[1 - \exp(\mathrm{CE}_{\mathrm{PAT}} - \mathrm{CE}_{\mathrm{FA}})]$. Lines between measured points are visual guides. A fitted size/data exponent uses log cross-entropy against log non-embedding parameters or tokens; it describes the measured range. Tables below give the complete selected endpoints, including additional completed runs outside the paired comparisons.

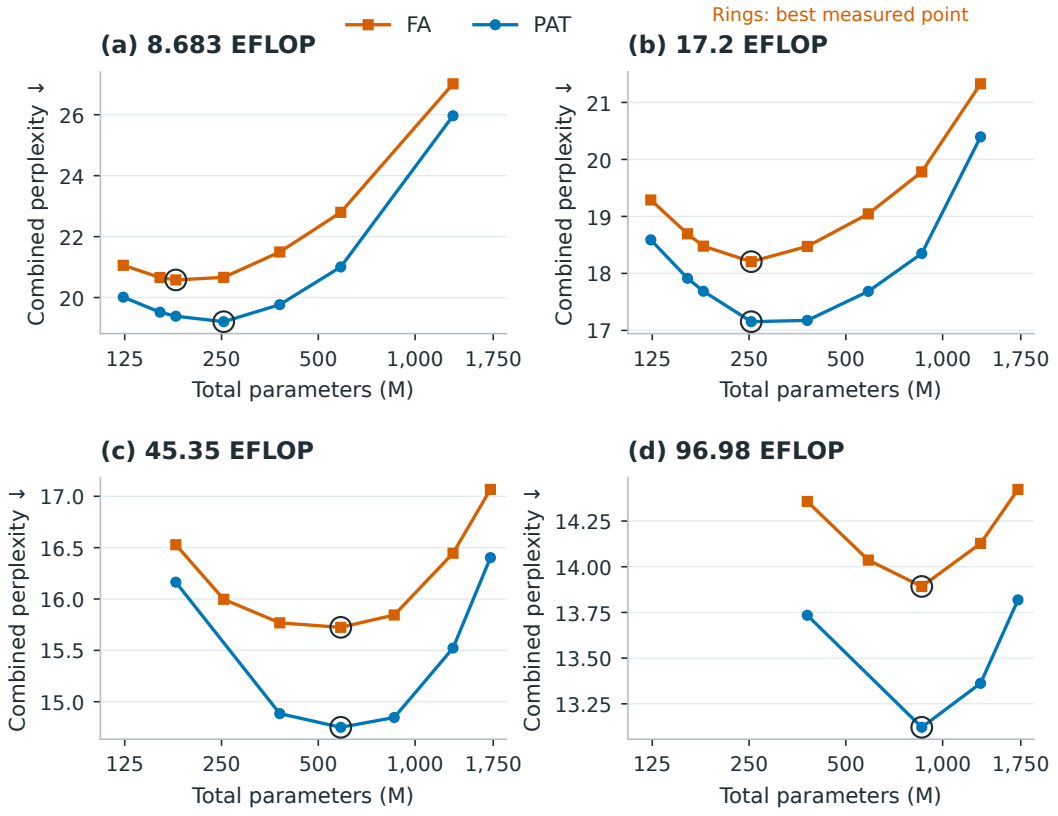


Figure 4: PAT improves measured quality throughout four compute-budget grids. Each panel varies total model parameters while holding modeled training compute approximately fixed at the labeled budget. Perplexity uses the token-weighted combined FineWeb-Edu holdout. Points are completed selected endpoints; lines connect measurements, and neutral rings mark each architecture's lowest measured perplexity in that grid. PAT's best observed PPL reductions are 6.66%, 5.79%, 6.19%, and 5.55% across panels (a)–(d). Training exposure varies with architecture and size; Tables 6–7 give the exact horizons and modeled compute.

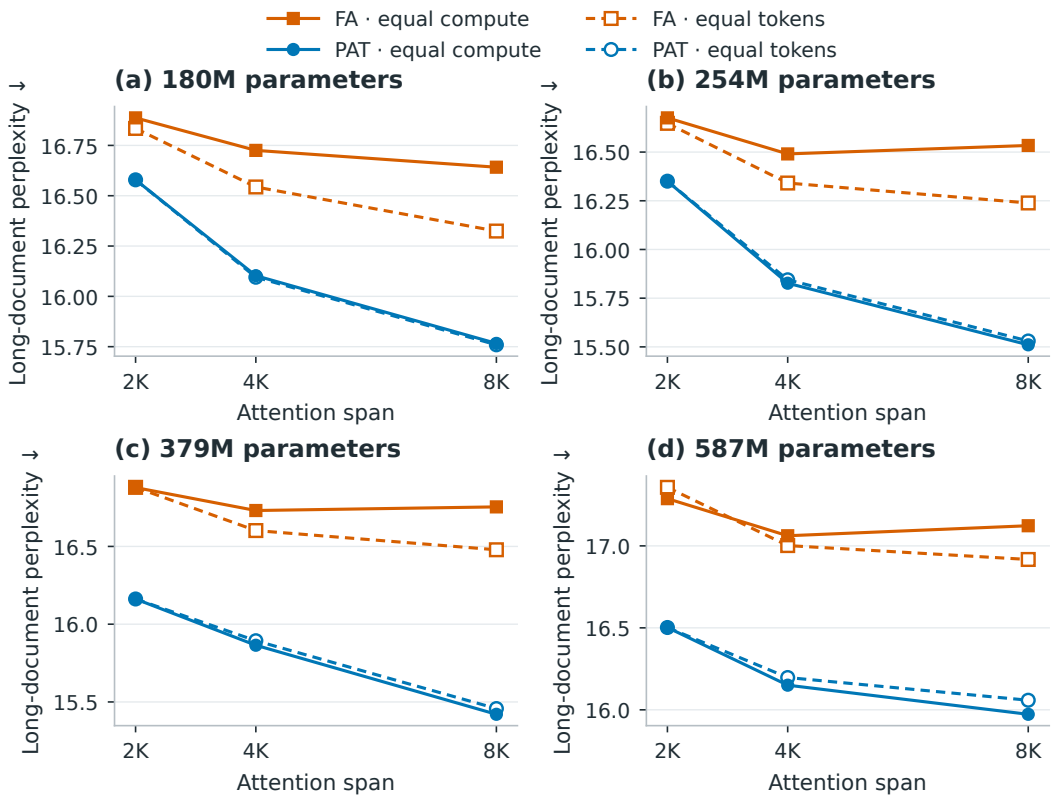


Figure 5: PAT uses longer context more effectively across both resource controls. Each panel holds model size fixed and varies attention span over the same document pieces, capped at 16K. Perplexity uses the long-document FineWeb-Edu holdout. Filled markers and solid lines match training compute between PAT and FA within each span (15.504, 16.255, and 17.200 EFLOP at 2K, 4K, and 8K); open markers and dashed lines match training-token exposure across architectures and spans within each model size. The same PAT 2K endpoint serves both regimes, and some other points nearly coincide. Memory access is restricted to the allowed span. Table 8 reports exact token horizons and compute.

Table 6: Completed paired model and data scaling endpoints. Parameters include tied embeddings once; T is billions of supervised lexical tokens, C is modeled training EFLOP, and PPL is the token-weighted combined validation perplexity. Bands A–D vary size within approximately 8.683, 17.200, 45.350, and 96.980 EFLOP. Parameter slices fix 24.12B tokens; data slices fix 1.715B parameters. The 3.52B PAT checkpoint also appears in Band C.

Regime	Size (M)	T_{FA}	T_{PAT}	C_{FA}	C_{PAT}	$\mathrm{PPL}_{\mathrm{FA}}$	$\mathrm{PPL}_{\mathrm{PAT}}$
Band A	124	7.01	8.59	8.683	8.685	21.0588	20.0136
Band A	161	5.62	6.73	8.681	8.685	20.6584	19.5199
Band A	180	4.94	5.94	8.683	8.683	20.5778	19.3883
Band A	254	3.70	4.32	8.683	8.683	20.6625	19.2077
Band A	379	2.57	2.93	8.683	8.683	21.4947	19.7677
Band A	587	1.73	1.93	8.673	8.697	22.7964	21.0042
Band A	1312	0.83	0.88	8.732	8.642	27.0148	25.9654
Band B	124	13.89	17.01	17.205	17.199	19.2886	18.5885
Band B	161	11.14	13.33	17.208	17.202	18.6948	17.9136
Band B	180	9.79	11.77	17.200	17.200	18.4780	17.6870
Band B	254	7.34	8.56	17.200	17.200	18.2063	17.1526
Band B	379	5.09	5.81	17.200	17.200	18.4740	17.1752
Band B	587	3.43	3.82	17.195	17.214	19.0452	17.6848
Band B	861	2.42	2.64	17.175	17.196	19.7820	18.3491
Band B	1312	1.63	1.75	17.148	17.183	21.3272	20.3961
Band C	180	25.81	31.03	45.351	45.351	16.5286	16.1637
Band C	379	13.42	15.31	45.351	45.351	15.7677	14.8838

Regime	Size (M)	T_{FA}	T_{PAT}	C_{FA}	C_{PAT}	$\mathrm{PPL}_{\mathrm{FA}}$	$\mathrm{PPL}_{\mathrm{PAT}}$
Band C	587	9.05	10.06	45.367	45.333	15.7241	14.7506
Band C	861	6.39	6.96	45.350	45.333	15.8442	14.8460
Band C	1312	4.31	4.62	45.342	45.364	16.4454	15.5221
Band C	1715	3.28	3.52	45.332	45.358	17.0671	16.4027
Band D	379	28.56	32.58	96.510	96.510	14.3569	13.7344
Band D	861	13.67	14.89	97.016	96.984	13.8921	13.1216
Band D	1312	9.22	9.88	96.995	97.011	14.1275	13.3623
Band D	1715	7.02	7.53	97.022	97.030	14.4231	13.8186
24.12B tokens	124	24.12	24.12	29.876	24.388	18.2493	18.0251
24.12B tokens	180	24.12	24.12	42.380	35.256	16.6400	16.4995
24.12B tokens	254	24.12	24.12	56.540	48.459	15.6197	15.1892
24.12B tokens	379	24.12	24.12	81.518	71.441	14.6347	14.1452
24.12B tokens	587	24.12	24.12	120.912	108.689	13.6402	13.0838
3.52B tokens	1715	3.52	3.52	48.649	45.358	16.7728	16.4027
5.02B tokens	1715	5.02	5.02	69.381	64.687	15.4553	15.0469
6.52B tokens	1715	6.52	6.52	90.112	84.015	14.6418	14.2232
8.02B tokens	1715	8.02	8.02	110.843	103.344	14.0637	13.6467
9.52B tokens	1715	9.52	9.52	131.574	122.673	13.6216	13.1996

Table 7: Additional completed selected main-scaling endpoints. These complete the 77-checkpoint selected main inventory. GPT++ uses cross-document packed validation; its PPL is a separate pipeline comparison.

Family	Regime	Size (M)	Tokens (B)	EFLOP	CE	PPL
FA	Band C	254	19.35	45.351	2.772405	15.9971
FA	Band D	587	19.35	96.984	2.641648	14.0363
FA	24.12B tokens	861	24.12	171.180	2.551312	12.8239
GPT++	Band D	379	22.50	96.467	2.744846	15.5622
GPT++	Band D	587	15.54	96.959	2.727630	15.2966
GPT++	Band D	861	11.16	96.945	2.739924	15.4858
GPT++	Band D	1715	5.80	97.021	2.775617	16.0485
PAT	11.02B tokens	1715	11.02	142.001	2.553474	12.8517

Table 8: Completed context-scaling comparisons. Attention windows restrict reads over the same document pieces capped at 16K. PPL uses the long-document holdout. Compute matches within each context span; equal-token conditions keep exposure fixed within each model size. Four PAT checkpoints serve both comparison regimes, giving 44 unique endpoints.

Regime	Size / span	T_{FA}	T_{PAT}	C_{FA}	C_{PAT}	$\mathrm{PPL}_{\mathrm{FA}}$	$\mathrm{PPL}_{\mathrm{PAT}}$
Equal compute	180M / 2K	11.884	12.218	15.504	15.504	16.8869	16.5790
Equal compute	180M / 4K	11.154	12.185	16.255	16.255	16.7253	16.1026
Equal compute	180M / 8K	10.427	12.148	17.200	17.200	16.6417	15.7663
Equal compute	254M / 2K	8.639	8.734	15.504	15.504	16.6756	16.3506
Equal compute	254M / 4K	8.209	8.767	16.255	16.255	16.4906	15.8268
Equal compute	254M / 8K	7.770	8.805	17.200	17.200	16.5340	15.5106
Equal compute	379M / 2K	5.871	5.861	15.504	15.504	16.8785	16.1621
Equal compute	379M / 4K	5.624	5.906	16.255	16.255	16.7309	15.8646
Equal compute	379M / 8K	5.368	5.959	17.200	17.200	16.7542	15.4206
Equal compute	587M / 2K	3.861	3.803	15.505	15.504	17.2880	16.5021
Equal compute	587M / 4K	3.735	3.850	16.256	16.255	17.0617	16.1503
Equal compute	587M / 8K	3.601	3.905	17.200	17.200	17.1228	15.9727
Equal tokens	180M / 2K	12.218	12.218	15.939	15.504	16.8344	16.5790
Equal tokens	180M / 4K	12.218	12.218	17.805	16.299	16.5435	16.0951
Equal tokens	180M / 8K	12.218	12.218	20.154	17.299	16.3244	15.7592
Equal tokens	254M / 2K	8.734	8.734	15.676	15.504	16.6475	16.3506
Equal tokens	254M / 4K	8.734	8.734	17.296	16.194	16.3411	15.8444
Equal tokens	254M / 8K	8.734	8.734	19.335	17.062	16.2388	15.5303
Equal tokens	379M / 2K	5.861	5.861	15.478	15.504	16.8799	16.1621
Equal tokens	379M / 4K	5.861	5.861	16.940	16.130	16.6020	15.8937
Equal tokens	379M / 8K	5.861	5.861	18.779	16.918	16.4785	15.4575

Regime	Size / span	T_{FA}	T_{PAT}	C_{FA}	C_{PAT}	$\mathrm{PPL}_{\mathrm{FA}}$	$\mathrm{PPL}_{\mathrm{PAT}}$
Equal tokens	587M / 2K	3.803	3.803	15.271	15.504	17.3570	16.5021
Equal tokens	587M / 4K	3.803	3.803	16.552	16.055	17.0018	16.1970
Equal tokens	587M / 8K	3.803	3.803	18.163	16.749	16.9171	16.0585

C Downstream tasks, reference models, and scoring

PAT and FA have 1.715B parameters and 9,520,047,061 training tokens. Their modeled costs are 122.672 and 131.574 EFLOP. The seven long-context tasks at 4K/8K/16K comprise 21 conditions and 20,994 examples per model. All models receive the same text, answers, and greedy generation settings. Context bins use the PAT/FA tokenizer; reference tokenizers can assign different lengths to identical text. Inputs are evaluated in full with no chat template.

RULER multikey and multiquery retrieval allow 128 output tokens, QA 32, and variable tracking 30. They use substring-based answer recovery; multiquery and tracking average the fraction of requested targets recovered. Each condition has 1,000 examples. BABILong uses a 20-token cap and task-specific location scoring: 1,000 examples for QA1 and 999 each for QA2/QA3. QA1–QA3 require one to three supporting facts. The seven-task macro average weights these native task scores equally.

Task	Context	PAT	FA	Qwen3 0.6B	Qwen3 1.7B
multikey retrieval	4K	93.60	82.30	93.80	99.40
	8K	91.60	68.80	85.70	99.40
	16K	80.70	46.90	84.40	97.10
multiquery retrieval	4K	52.18	29.85	96.33	96.10
	8K	39.40	22.73	90.13	97.33
	16K	29.28	12.75	85.40	96.78
question answering	4K	29.30	22.90	41.40	45.80
	8K	25.40	20.10	37.90	44.00
	16K	20.10	12.90	35.30	36.90
variable tracking	4K	11.20	1.26	91.28	98.28
	8K	3.34	2.92	85.34	91.78
	16K	6.88	4.76	82.28	89.28
QA1	4K	37.20	27.70	1.90	12.80
	8K	28.20	16.50	0.60	2.20
	16K	22.70	10.70	4.00	2.70
QA2	4K	18.52	8.71	2.90	1.90
	8K	17.32	5.01	0.50	5.01
	16K	11.31	5.31	3.90	2.20
QA3	4K	14.61	9.01	0.80	6.31
	8K	10.71	10.71	0.10	7.31
	16K	2.60	6.31	3.70	8.61

Table 9: Complete long-context scores (percent). PAT and FA match model size and training tokens; Qwen3 base models provide separately pretrained references. Scores use each task's native metric. All entries use neutral formatting; the QA3 answer-bias diagnostic is given below.

Answer-bias control for QA3. At 16K, FA scores 6.31% and PAT 2.60%. Of FA's 362 scorer-eligible answers, 338 are "kitchen"; 57 of its 63 correct answers are kitchen cases. A control with 10,000 permutations preserves response bias, invalid answers, and question-location strata. It predicts 5.95%, with a 95% randomization range of 4.80–7.11%; the observed FA score has one-sided $p = .30$. The raw score is retained in Table 9 and interpreted with this diagnostic. For QA1, PAT exceeds the six-location uniform reference (16.67%) and the observed majority-location baseline (18.7%) at all three lengths.

Short-context evaluation. Ten tasks contain 8,425 selected examples per model. Eight use length-normalized answer-choice likelihood (`acc_norm`). TriviaQA and GSM8K use quasi-exact match

with the original answer normalization/extraction, output limits of 20 and 256 tokens, and stopping at EOS or the strings `Question:` and `Question`. Tasks are zero-shot except five-shot GSM8K. MMLU draws 25 examples from each of 57 subjects. All models use identical selected documents, choices, answers, and demonstrations. PAT/FA use their final post-cooldown checkpoints.

Task	n	PAT	FA	Qwen 0.6B	Qwen 1.7B	p_{Holm}
HellaSwag	1000	41.90	40.00	54.60	65.90	0.37
ARC Easy	500	58.20	58.20	62.40	72.40	1.00
ARC Challenge	500	27.00	29.20	39.00	42.00	1.00
PIQA	1000	67.70	68.80	69.20	76.20	1.00
MMLU	1425	31.58	29.82	—	—	0.58
CommonsenseQA	1000	31.20	30.80	—	—	1.00
TriviaQA	1000	0.90	1.10	—	—	1.00
WinoGrande	1000	52.40	52.20	—	—	1.00
OpenBookQA	500	35.00	34.40	—	—	1.00
GSM8K (5-shot)	500	1.20	1.80	—	—	1.00

Table 10: Short-context scores (percent) and PAT–FA Holm-adjusted paired-test p-values over ten tasks. Dashes denote unavailable complete-task reference results.

Separately pretrained references. Qwen3 (Yang et al., 2025)-0.6B-Base and Qwen3-1.7B-Base use their native tokenizers and the same evaluation prompts. Qwen3 reports 36T pretraining tokens; PAT/FA use 9.52B. Their scores contextualize attained task performance under distinct training histories. Short-context reference scores are available for the four tasks displayed in Table 10.

Paired uncertainty. Binary outcomes use exact two-sided paired McNemar tests; fractional long-context scores use 20,000 paired sign permutations. Holm correction covers all 21 long-context or ten short-context PAT/FA conditions. Pointwise 95% intervals use 20,000 paired bootstrap draws; MMLU resamples within subjects. External-reference tests use their reported 50-comparison family. These intervals describe evaluation-example uncertainty conditional on trained checkpoints. All twelve PAT/FA multikey, multiquery, RULER QA, and QA1 conditions survive the long-context correction. The ten short-context adjusted p-values appear in Table 10.

Evaluation uses LightEval 0.6.0.dev0, commit `ea46419a93fb390e8f694f7c6c64c1e684487c9d`, and SmolLM task definitions at `d3bb683bc123a1954b289b70dd8119b51c358018`. Qwen revisions are `da87bfb608c14b7cf20ba1ce41287e8de496c0cd` (0.6B) and `ea980cb0a6c2ae4b936e82123acc929f1cec04c1` (1.7B).

D SYNTHETIC RECALL, GENERALIZATION, AND MEMORIZATION

Common architecture and selection. All models use four sequence-mixing blocks, width 128, two heads, tied embeddings, and zero dropout. Active feed-forward blocks use SwiGLU width 344; MQAR replaces this block by identity in every architecture. PAT has memory slots $[2, 8, 8, 2]$ and calibrated QK normalization. The synthetic FA and nanoGPT controls retain their attention normalization, and the nanoGPT module has no positional embeddings. The latter is an attention-module substitution in the common shell. Each model trains over seeds 42–46 with fixed data and order across architectures. Validation runs every epoch; after the full horizon, the earliest checkpoint minimizing unrounded pooled supervised validation NLL is selected. Test examples are held out from selection.

ICR/MAD use AdamW $\beta = (0.9, 0.98)$, zero weight decay, and update-wise cosine learning rate $5 \times 10^{-4} \to 10^{-6}$. MQAR uses $\beta = (0.9, 0.999)$, weight decay 0.1 on all parameters, and epoch-wise cosine $10^{-2} \to 0$. Both use $\epsilon = 10^{-8}$, zero warmup, and no gradient clipping. ICR/MAD train for 200 epochs, MQAR for 100, with batches of 128 and 256. Initialization follows Appendix A with $L = 4$. PAT's initial scales are $[10, 10, a, a]$, with $a \simeq 12.62$–12.86 for most tasks and 11.03 for memorization.

Multi-query associative recall. MQAR supplies key–value bindings followed by queries, e.g., $[A, 7, B, 2, \ldots, A, ?] \mapsto 7$, over vocabulary 8,192. Training has 100,000 exam-

ples at (length,pairs) $= (64, 4)$ and 20,000 each at $(128, 8), (256, 16), (256, 32), (256, 64)$. Independent validation and test sets have 1,000 examples per configuration: $(64, 4), (64, 8), (64, 16), (128, 32), (256, 64), (512, 128), (1024, 256)$. Training supervises query answers; aggregate accuracy weights configurations equally. Longer configurations enter validation-based checkpoint selection, while their test examples remain held out. The horizon comprises 70,700 updates and 2.432B input-token presentations, retaining partial final batches.

ICR layout generalization. Training lengths are 256/512/1,024 with 5,120/5,120/15,360 examples per epoch. Keys are ordered triples of distinct atoms from 16 symbols, with 4/7/13 key identities per row. Values use 32 symbols and noise 64; filler noise occupies 25%. Each row has a consistent association map. The 200 epochs make two passes over 2.56M distinct generated examples, totaling 40,000 updates and 3.932B input-token presentations. Validation uses 1,280 independent 1K examples. Training and selection use full shifted language-model loss. Each test condition has 2,560 examples, evaluated by unrestricted terminal argmax.

The shifts remove filler noise, replace it with random key/value symbols, extend keys to four atoms, or corrupt all non-probe keys while preserving the queried binding. Two additional 2K conditions prepend associations before source and query (fixed retrieval distance) or insert them before the query (growing distance). Data/order seeds are 12345 for ICR/MAD and 123/0 for MQAR.

MAD tasks. Each task uses 12,800 training, 1,280 validation, and 1,280 test examples, except memorization has 256 training examples. Task settings are compression (length 256, vocabulary 16); fuzzy ICR (1,024, 16, three-symbol keys and values); ICR (1,024, 16); memorization (32, 8,192); noisy ICR (128, 32, 80% noise); selective copying (256, 16, 96 copied tokens). Evaluation is macro accuracy over observed target classes. Most tasks have 20,000 updates; memorization has 400. After input/target shifting, ICR and noisy ICR use 1,023 and 127 input tokens. Total presentations are respectively 0.655B, 2.621B, 2.619B, 1.638M, 0.325B, and 0.655B for the listed task order.

Task	PAT	FA	nanoGPT	KDA	GSA
MQAR: seven-setting mean	99.21 ±0.90	89.02 ±2.21	87.33 ±2.86	85.18 ±8.38	31.54 ±37.63
MQAR: 1,024 tokens, 256 pairs	95.27 ±6.09	31.90 ±11.15	47.41 ±9.36	31.66 ±15.97	10.49 ±19.55
ICR: zero filler noise	61.23 ±26.67	34.80 ±35.81	3.10 ±0.72	76.07 ±36.41	18.00 ±20.78
ICR: other keys corrupted	100.00 ±0.00	73.58 ±42.32	0.25 ±0.35	67.50 ±28.60	76.20 ±38.72
ICR: key/value-like fillers	65.48 ±33.53	56.06 ±46.95	0.00 ±0.00	66.70 ±21.08	30.50 ±31.42
ICR: four-token keys	93.93 ±7.46	60.88 ±35.57	12.74 ±14.86	33.97 ±40.69	48.02 ±27.25
MAD: compression	19.94 ±0.74	19.19 ±0.92	22.15 ±0.84	20.49 ±0.79	21.98 ±0.37
MAD: fuzzy recall	99.99 ±0.01	99.96 ±0.08	13.36 ±0.36	81.08 ±4.54	14.23 ±0.71
MAD: recall	99.93 ±0.03	100.00 ±0.00	32.21 ±0.12	100.00 ±0.00	99.07 ±0.12
MAD: memorization	61.16 ±2.77	61.67 ±0.52	6.55 ±1.20	2.46 ±0.90	44.99 ±2.89
MAD: noisy recall	99.92 ±0.06	95.05 ±11.07	93.73 ±11.89	100.00 ±0.00	74.30 ±22.27
MAD: selective copying	99.82 ±0.15	99.99 ±0.00	88.07 ±19.56	98.99 ±0.29	77.94 ±23.03

Table 11: Synthetic task accuracy (percent), mean and sample standard deviation over five initialization seeds (42–46). All models use the common four-block shell; the nanoGPT column denotes its attention module without positional embeddings. MQAR averages seven configurations equally.

E WRITING, CHUNK SIZE, AND MEMORY ALLOCATION CONTROLS

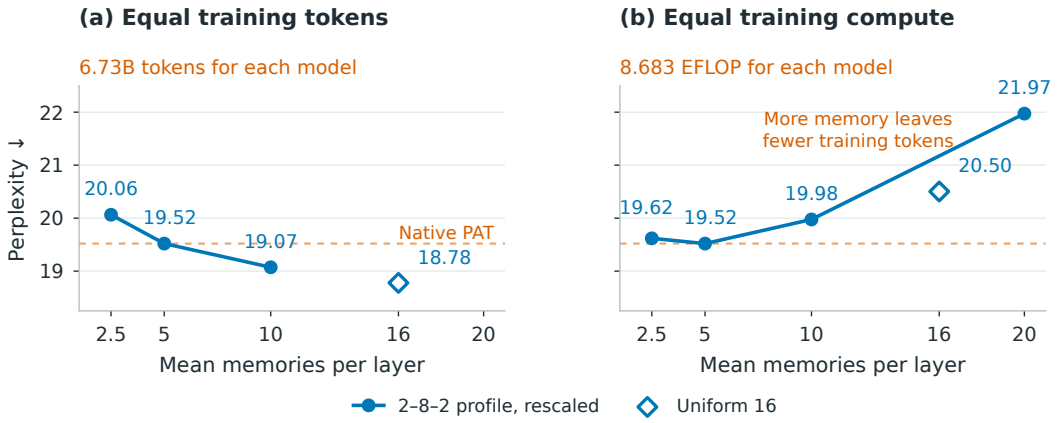


Figure 6: Memory allocation changes how a training budget is used. At 161M parameters and 16-token chunks, increasing capacity improves perplexity at fixed 6.73B-token exposure (left), while fixed 8.683 EFLOP shortens the training horizon for larger memories (right). Circles scale the native expand–contract profile; diamonds use uniform-16 allocation. Dashed lines mark native mean-five PAT. Each endpoint uses one seed.

Language-model controls use one training seed. The primary shape is 12×896 (160.9M parameters), 16K lexical context, and 6.730B tokens for PAT-derived variants. Combined PPL is the token-weighted geometric mean on the validation splits. The native PAT/FA reference pair has approximately 8.683 EFLOP with 6.730B/5.620B tokens and PPL 19.5199/20.6584. Variants match PAT's token exposure unless the table specifies a compute-matched horizon.

Configuration	Long PPL	Short PPL	Combined PPL
Native PAT	17.3527	19.8438	19.5199
Writer without historical reads	17.3677	19.8651	19.5404
Causal within-chunk memory writer	17.3828	19.8752	19.5512
Uniform five memories/layer	17.4356	19.9133	19.5914
Distinct chronological positions	17.8368	20.1491	19.8501
Contiguous geometric pooling	17.6226	20.0432	19.7292
Raw history plus five memories/layer	17.8674	20.1305	19.8381
FA reference (5.620B tokens)	18.6047	20.9630	20.6584

Table 12: Writing and retained-state controls. Each PAT variant trains on 6.730B tokens; the FA reference matches native PAT's nominal compute. Individual changes are separate trained configurations.

The history ablation removes old-memory inputs from memory queries while retaining lexical reads. The causal-writer control changes only grouped memory visibility from bidirectional to causal. Uniform-five changes allocation across depth at the same mean count. Distinct chronological positions separate token and memory entries while retaining rotary arithmetic. At this scale, positional assignment has the largest observed PPL effect among these component changes. Ratios relative to the native PAT–FA PPL difference are descriptive and non-additive.

Pooling and retained raw history. Geometric pooling overwrites each computed memory output with a differentiable arithmetic average of completed same-layer token states in a contiguous bin. For a 16-token chunk, two slots pool consecutive groups of eight; eight slots pool pairs. The writer still executes before replacement. Upstream token states remain trainable. The raw-retention control keeps old token KVs alongside uniform-five memories; both query/key calibration and compute change with the extra visible keys. Its comparison to uniform-five PAT is made at the same 6.730B tokens.

Doubling mean memories from five to ten raises nominal per-token compute by 36%. At equal compute it reduces exposure by 26.4%, reversing the equal-token quality ranking. Native mean-five is the best measured point in this compute-matched grid. Chunk effects are smaller: all combined-PPL shifts relative to each chunk-16 model are below 1%.

Model	Chunk/profile	Tokens (B)	Long PPL	Short PPL	Combined PPL
161M	8 / 1–4–1	6.730	17.4163	19.8815	19.5612
161M	16 / 2–8–2	6.730	17.3527	19.8438	19.5199
161M	32 / 4–16–4	6.730	17.4636	19.9850	19.6571
587M	8 / 1–4–1	3.820	15.7292	17.8651	17.5882
587M	16 / 2–8–2	3.820	15.8019	17.9653	17.6848
587M	32 / 4–16–4	3.820	15.6237	17.8128	17.5286

Table 13: Chunk sweeps preserve average memory density $m/k = 5/16$ and token exposure within each model size. The lexical-local span changes with chunk size. Layer bands contain 3/6/3 layers at 161M and 4/9/5 at 587M.

Control	Profile (mean)	Tokens (B)	Long PPL	Short PPL	Combined PPL
Equal tokens	1–4–1 (2.5)	6.7300	17.9374	20.3807	20.0640
	2–8–2 (5)	6.7300	17.3527	19.8438	19.5199
	4–16–4 (10)	6.7300	16.8820	19.4010	19.0728
	Uniform 16	6.7300	16.5807	19.1079	18.7783
Equal compute	1–4–1 (2.5)	7.8101	17.5288	19.9305	19.6190
	2–8–2 (5)	6.7300	17.3527	19.8438	19.5199
	4–16–4 (10)	4.9500	17.6687	20.3210	19.9754
	8–32–8 (20)	2.8001	19.4078	22.3570	21.9724
	Uniform 16	3.8300	18.0824	20.8662	20.5029

Table 14: Memory allocation at 161M parameters. Equal-token runs use 6.730B lexical tokens; equal-compute runs target 8.683 EFLOP and complete separate cooldowns. Uniform-16 changes both capacity and its distribution across depth.

Single-seed pooling generalization. Table 15 compares matched seed 42. ICR models complete 40,000 updates under the same selection rule; selected updates are 15,000 (learned PAT), 36,400 (pooled PAT), and 27,000 (FA). The six ICR conditions have 32 equally frequent valid answers, giving a constant-valid-answer reference of 3.125%. Pointwise intervals use 10,000 paired-example bootstrap draws; paired McNemar tests use Holm correction over six ICR conditions. For random key/value fillers, pooled minus learned accuracy is -54.92 points (95% CI $[-57.03, -52.77]$); restricting predictions to valid values gives 40.47% and 95.31%, respectively. Four-token-key generalization remains near ceiling for both PAT variants. These intervals concern examples at fixed trained checkpoints; the comparison uses one seed.

Condition	FA	Learned PAT	Pooled PAT
Zero filler noise	0.00	75.00	69.14
Random key/value fillers	0.12	93.95	39.02
Four-token keys	8.44	99.73	99.57
Other-key corruption	0.00	100.00	92.54
2K: fixed retrieval distance	0.00	98.75	79.80
2K: growing retrieval distance	0.00	98.52	96.29
Compression	19.59	19.39	20.20
Fuzzy recall	100.00	99.97	100.00
In-context recall	100.00	99.88	100.00
Noisy recall	75.24	99.83	99.99
Selective copying	99.99	99.86	99.20

Table 15: Single-seed retained-representation control: native learned PAT, FA, and fixed contiguous geometric pooling. ICR rows use 2,560 examples and unrestricted terminal accuracy; MAD rows use 1,280 records and observed-target-class macro accuracy.

F Interventions on writing and retained memory

These studies intervene on fixed checkpoints and read out the answer, complementing the retrained controls in Appendix E. The following panels use distinct task instances and sample counts.

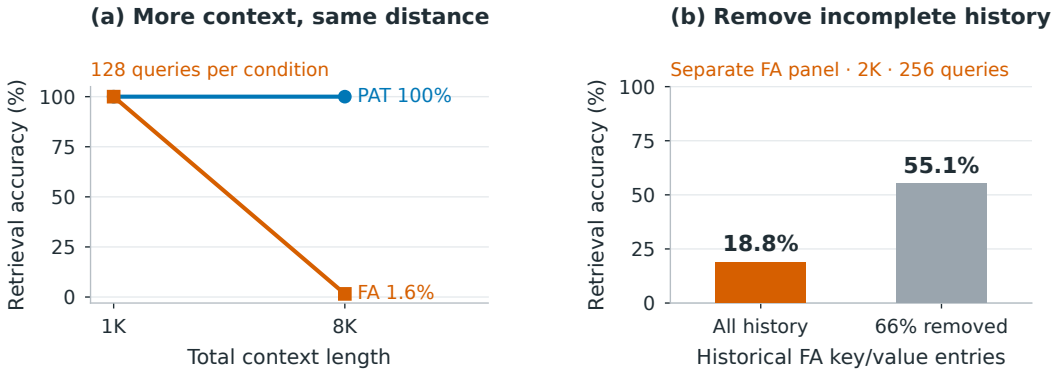


Figure 7: Retained states affect what future queries recover. Left: with source–query distance fixed, increasing context from 1K to 8K changes FA retrieval from 128/128 to 2/128 while PAT retains 128/128. Right: in a separate 2K FA assay, removing incomplete historical entries raises retrieval from 48/256 to 141/256 while retaining value-bearing entries. Query counts and context counts are distinguished in the protocol below.

Incomplete historical states compete for attention. In synthetic records $[A, B, C, v]$, the causal states at $A/B/C$ precede the value. Removing these old entries from all four FA layers, while preserving value and other positions, removes 66% of the 2K history and raises retrieval from 48/256 to 141/256. A separate matched 32-query panel retains all positions: suppressing incomplete entries' scores gives 16/32 correct versus 5/32 normally; replacing them with same-model contextual states at noise positions gives 17/32; zeroing only their values gives 4/32. These controls distinguish competition for attention from the values contributed by those states.

In a separate fixed-distance panel, increasing context from 1K to 8K changes FA accuracy from 128/128 to 2/128 and PAT from 128/128 to 128/128. Historical FA and distant PAT panels contain four independent contexts; query counts are therefore different from independent-context counts. PAT's local control retains all 239 previously correct answers after removing target A/B components from partial-state keys, with both raw states and memory accessible. In a separate remote-query panel, removing corresponding source-memory key components changes accuracy from 154/256 to 95/256. These interventions measure the usefulness and interference of retained representations.

Earlier history changes newly written memory. The RULER `niah_single_2` assay repeats fact A in earlier history at doses $d \in \{0, 1, 4, 16\}$, while holding the current 48-token $A + B$ span fixed. Only the three newly written chunks are transferred into a neutral receiver. The matched dense model transfers the aligned 48 raw KV rows at all 16 layers. Let $M_A(d)$ and $M_B(d)$ be correct-answer minus paired-wrong-answer summed log-probability margins. Define

$$S(d) = \tfrac{1}{2}[M_B(d) - M_A(d)], \qquad E(d) = S(d) - S(0).$$

Metrics are computed per pair before medians. The cohort contains the five pilot pairs and 27 additional constructions. At dose 16, A's margin falls and B's rises in all 32 PAT pairs. Median $E(16)$ is 11.905 for PAT and 3.664 for dense; the paired PAT-minus-dense median is 7.721 (29/32 positive; 95% paired-bootstrap CI $[5.644, 13.328]$). The CI uses 100,000 pair resamples, seed 37037, and linear percentiles. The 27 additional pairs all have positive PAT shifts. Checkpoints match token exposure; transferred state consists of three PAT chunks across 15 reader-consumed banks or 48 dense rows across 16 layers.

Earlier repeats	0	1	4	16
PAT median $E(d)$	0	1.357	8.255	11.905
Dense median $E(d)$	0	0.359	3.091	3.664

Table 16: Changes in what identical current text contributes to retained state, measured by transfer to a neutral receiver. Values are medians of pair-level changes.

Writer attention changes retention. Multiplying only writer logits by β before softmax changes memory construction while reader $\beta = 1$. A 36-row replication selects four clean-success examples

for each of three needle tasks at 4K/8K/16K. The primary score is the correct first-token logit minus log-sum-exp over all other valid-vocabulary logits. Increasing β from one to two harms 29/36 rows (median change -0.394); increasing it to four harms 32/36 (median -1.185). In the copied-memory assay, median $E(16)$ is 11.905, 8.327, and 6.609 at $\beta = 1, 2, 4$.

A separate mass intervention changes the total allocation between earlier history and local states while preserving conditional attention within each group. It uses fresh source mass along the intervened trajectory and a fixed native target. The confirmatory panel excludes four development pairs and contains 28 pairs. Forward reductions toward the $\beta = 2$ and $\beta = 4$ mass targets weaken $E(16)$ in 20/28 and 21/28 pairs, with median native-minus-intervention effects 0.953 (95% CI $[0.341, 1.518]$) and 2.247 ($[0.917, 4.244]$). Reverse restoration toward the $\beta = 1$ target has median -0.329 ($[-1.412, 0.517]$; 13/28 positive). The forward result identifies a contribution of the history/local mass split; the reverse result describes its asymmetric effect. Full writer sharpening also changes within-group attention and subsequent states.

Answer-bearing memory supports lookup. At reader layer 12, zeroing cached K and V for an answer-bearing chunk is compared with an adjacent equal-width nuisance chunk. In 36 independent clean-success rows across three tasks and three lengths, the target-cut drop exceeds the absolute nuisance-cut drop in 35/36 cases (median specificity gap 1.698; one-sided sign $p = 5.38 \times 10^{-10}$). Zeroed keys remain softmax-eligible, so this measures dependence on their content.

For restoration, each answer is replaced by a deterministic same-format value of equal token width. The clean answer-chunk payload is copied either to its original slots or to an adjacent non-overlapping block of equal width. At the preselected reader layer 12, original-slot restoration beats the identical-payload wrong-slot control in 36/36 rows (median gap 1.394); recovery over the corrupted baseline is positive in 36/36 (median 4.325). The intervention supports answer use of content together with its memory location.

G TRAINING THROUGHPUT AND DECODING LATENCY

Measurements use one NVIDIA H100 80GB HBM3 (700W), BF16 computation with FP32 parameters, and PyTorch 2.11/CUDA 12.8. Shapes denote layers $\times$ hidden width; 1K is 1,024 tokens. Native whole-model benchmarks match depth and width and retain each implementation's architecture and execution path.

Compiled training. PAT and nanoGPT share document batches capped at 16K and the same supervised-token work. GPT packs the same document stream into fixed-16K batches, including EOD supervision. Every implementation is compiled; GPT/nanoGPT use fused AdamW. Throughput is supervised non-special tokens divided by wall-clock training time. One measurement window is reported per shape/implementation. Warmup/measured updates are 10/20 for PAT/nanoGPT and 37/74 for GPT at 8×768; at 18×1536 they are 10/191 and 13/233.

Shape	PAT tokens/s	GPT tokens/s	nanoGPT tokens/s	PAT/GPT	PAT/nanoGPT
8×768	262,204	252,988	83,868	1.036	3.126
18×1536	60,136	60,448	45,163	0.995	1.332

Table 17: Measured compiled training throughput on one H100 at a 16K cap. Batching conventions are specified above.

Whole-model cached decoding. Batch-one timings use initialized weights and four documents with fresh caches. They measure 64 forced token appends after 32 warmup appends, excluding prefill. Prompts begin at 2K/4K/8K/16K and end 96 tokens later. Document throughput is summarized by its geometric mean and range. PAT uses CUDA graphs for ordinary and chunk-boundary steps; GPT uses native cached decoding; nanoGPT uses an exact KV-cache adapter with learned-position capacity 524,608. Across contexts, PAT/GPT and PAT/nanoGPT ratios are 3.63–5.02 and 1.72–1.97 at 16×1024, and 12.18–14.26 and 8.76–11.57 at 18×1536. These are native implementation comparisons.

Matched eager attention. The attention-module experiment uses 14×896 (180M) and 16×1280 (379M), batch one, initialized weights, and a fixed synthetic sequence. A 32-token trajectory ends at each labeled context, including BOS. Following two warmup trajectories, five CUDA-event rounds provide median and interquartile range. Timing includes normalization, QKV projection, positional transforms, attention, local cache writes, and output projection. The state-inclusive boundary also includes disjoint memory projections, transitions, and cache publication. Residual addition, MLP, and vocabulary projection fall outside both boundaries.

PAT uses packed QKV preparation and FA separate projections; both run eagerly. Module time is summed across layers and divided by layers times 32 useful tokens, including PAT's two additional chunk-boundary attention passes and dispatch gaps within event intervals. Attention-only FA/PAT speed ratios are 1.45–1.52 for 180M and 1.42–1.46 for 379M; including memory state gives 1.36–1.42 and 1.33–1.37. Instrumented and uninstrumented outputs match bitwise, with cached-decoding endpoint numerical checks.

H CONTEXT REWRITING, LOCALITY, AND PREDICTIVE ABSTRACTION

This appendix follows the main text: first compose associations and remove redundant features; then restrict rewriting to chunks; finally reduce predictive storage and memories per layer. Each result states its scope before its proof. The analytical compute estimate is in Appendix I.

Contents. The section names and page numbers below are clickable. Individual results also appear in the PDF bookmarks.

H.1	Full rewriting: parallel composition	28
H.2	Redundant features: strict storage–loss improvement	29
H.3	Local rewriting: cost, boundary loss, and chunk size	36
H.4	Predictive abstraction: the linear storage minimum	42
H.5	Composition with fewer memories per layer	47
H.6	Exact recall across layer memories	49
H.7	Attention paths and reading capacity	51
I	Analytical compute estimate and model-scale comparisons	54

Vocabulary and storage model. An *association* maps an input to an output. Composing its prerequisites (the *lower-order associations*) forms the higher-order association needed for prediction. A *token state* belongs to an input position. A *memory state* is a separate latent state that supplies attention keys and values.

Symbol	Meaning throughout the appendix
N, K	Number of input positions and chunk length.
L, D, h	Layers, model width, and attention heads.
H	Number of successive associations in a chain.
m, m_ℓ	Physical memory rows per chunk per layer; m_ℓ allows different allocations by layer.
M	Total retained memory rows across the relevant layers.
X, Y, S	Observed context, target, and predictive state $S = P(Y \mid X)$.
W, C	Entire retained cache and number of possible cache configurations.
U	A feature whose predictive role may change with later evidence.

C counts distinguishable representations; m and M count rows. A feature is *redundant* when contexts can differ in that feature while sharing the required answer probabilities. All logarithms are natural unless a base is specified.

Shared attention interface. Full rewriting recomputes memories from the observed prefix. PAT writes a chunk once and leaves earlier memories unchanged. Within each layer, tokens and memories share attention projections; memory states use attention and residual addition without a memory feedforward network. A state written at layer ℓ becomes readable at layer $\ell + 1$.

The constructions use fixed role and address coordinates, with finite precision stated where needed. Source access, preparation, and auxiliary rows count toward their resource bounds. Later evidence is used only for later predictions.

H.1 Full rewriting composes association chains in logarithmically many layers

Rewriting can prepare all elementary associations together and repeatedly compose the stored results. Each round doubles the represented chain length. The following task makes this advantage precise: its causal lower bound is imported from Chen et al. (2024), while the shared-attention memory construction is given here.

An addressed composition task. Write $[q] = \{1, \dots, q\}$. Present arbitrary function tables $f_j : [q_{j-1}] \to [q_{j-1}]$, $j = 1, \dots, H$, in reverse order, one entry per record. A starting identifier $x_0 \in [q_0]$ follows; the final record contains selectors $w_j \in [s_j]$, $j = 1, \dots, H-1$, and an output-bit index. With $q_j = q_{j-1}s_j$, compute

$$x_1 = f_1(x_0), \qquad x_{j+1} = f_{j+1}\big((w_j - 1)q_{j-1} + x_j\big). \tag{H.1.1}$$

The answer is the requested bit of $x_H - 1$. There are $N = 2 + \sum_{j=0}^{H-1} q_j$ addressed records. Addresses, values and the selector vector each use $O(\log N)$ bits.

The causal resource budget is $2hDpJ \le \mathcal{B}$, with h heads, width D, precision p and $J = \lceil \log_2 q_{H-1} \rceil$ output bits. The proof chooses the domains from $\mathcal{B}$ and verifies that logarithmic width and precision fit.

Theorem H.1.1 (Rewriting reduces composition depth from linear to logarithmic)**.**
For the domain family below and the stated causal resource budget, exact correctness requires $\Theta(H)$ causal attention layers. Full rewriting uses $O(\log H)$ layers, $O(N)$ memories per layer, constant heads, and $O(\log N)$ width and precision bits per coordinate. This family has $H = \Theta(\log \log N)$.

- Causal attention receives the stated order, with no appended writable suffix.
- Causal arithmetic follows the finite-precision model of Chen et al. (2024).
- The writer can read the original records and preceding memory bank, using the protected address interface of Lemma H.1.2.

Proof: calibrate the lower bound and double stored pointers.

Import the communication obstruction.

For an integer budget $\mathcal{B}$, set

$$\kappa = (\mathcal{B}H)^8\, 8^{2H^2}, \qquad q_0 = \kappa^{1+\sum_{r=0}^{H-1} 8^r}, \qquad s_j = \kappa^{4 \cdot 8^{H-j-1}}.$$

These are the domains of Chen et al. (2024, Eq. (4), Definition 2.1 and Theorem 4.1), replacing their depth by H and their communication parameter by $\mathcal{B}$. Their Lemma 3.1 simulates attention using at most $2hDp$ bits per receiving-block record per round. If an H-layer model answered every requested bit, run its J bit queries in parallel. The resulting protocol recovers the full value within budget $\mathcal{B}$, contradicting the cited lower bound. Copying x_0 and performing the H lookups sequentially gives the matching $H + O(1)$ upper bound.

Choose $\mathcal{B} = \lceil 2^{cH^2} \rceil$ for a fixed sufficiently large c. Then $\log N = \Theta(8^H H^2)$, and constant h with $D, p = O(\log N)$ satisfies $2hDpJ \le \mathcal{B}$ for large H. This verifies both the resource regime and $H = \Theta(\log \log N)$.

Prepare every possible elementary lookup.

Allocate one row per stage and possible incoming identifier, plus self-pointing terminal rows. The incoming-domain sizes are $q_0, q_0, q_1, \dots, q_{H-2}$; including the q_{H-1} terminal rows gives $\Theta(N)$ rows. Copy the selectors, then store each selected association as a next-row pointer $F(z)$. These initialization reads take constant layers.

Selection needs no memory feedforward network. A raw key contains its stage, incoming identifier and selector code in that stage's coordinate group. The query contains its current stage and identifier and the full selector vector. Subtracting the number of active sign coordinates from each key's score makes a match score zero and every mismatch negative. The total selector length is $O(\log N)$, so shared linear projections implement these tests at the claimed width.

Double and retrieve.

Keep fixed row addresses and update all pointers in parallel:

$$T_0 = F, \qquad T_{r+1}(z) = T_r(T_r(z)) = F^{2^{r+1}}(z). \tag{H.1.2}$$

Lemma H.1.2 implements each round. After $\lceil \log_2 H \rceil$ rounds, the row addressed by x_0 points to its terminal value. A final lexical read retrieves it. Encoding the requested index in J one-hot coordinates lets a lexical feedforward layer select the requested bit with $O(J) = O(\log N)$ width. Initialization and the read after the final write are included in the layer count.

The working bank occupies $O(NDp)$ bits; retaining a bank at every layer occupies $O(NDp \log H)$ bits. Original records remain readable while writing. Separate unchanged archive fields can preserve them afterward at constant-factor cost. □

Lemma H.1.2 (Shared heads replace pointers without a memory feedforward network)**.** Two heads implement a pointer-doubling round and cancel their added outputs on token positions. Over T_{round} rounds, $O(\log(N_w T_{\mathrm{round}}))$ precision bits preserve every address decision, where N_w bounds candidates per read.

- Rows have distinct fixed sign addresses b_z of length $b_{\mathrm{addr}} = O(\log N_w)$ and writable pointers x_z.
- Pointer-query and self-query fields are zero on token positions.
- The heads share keys, values and masks and have opposite output projections. Both target and self reads are allowed; source types have distinct tags.
- Protected fields are numerically accessible after any normalization or positional transformation. One complete update incurs coordinate error at most ν.

Proof: subtract the stored pointer before inserting its successor.

Replace by residual addition.

Query the target with x_z and the row itself with b_z. The update is

$$x'_z = x_z + \mathrm{Read}(x_z) - \mathrm{Read}(b_z).$$

Exact reads replace x_z by the target's pointer. On token positions both queries are zero, so identical head outputs cancel. The self read subtracts the actual stored pointer, including its previous error. Writer-query fields and lexical query/accumulator fields are separate, so lexical reads preserve the writer's protected source and address coordinates.

Keep the address margin.

If pointer error is $\epsilon_r \le 1/2$, a wrong sign address loses at least one unscaled score unit. Scaling this gap to γ leaves unwanted softmax mass at most $N_w e^{-\gamma}$. Pointer coordinates have magnitude at most $3/2$, giving

$$\epsilon_{r+1} \le \epsilon_r + 6N_w e^{-\gamma} + \nu. \tag{H.1.3}$$

Take $\epsilon_0 \le 1/8$, $\gamma \ge \log(48 N_w T_{\mathrm{round}})$ and $\nu \le 1/(16 T_{\mathrm{round}})$. Then $\epsilon_{T_{\mathrm{round}}} \le 5/16$, closing the induction. Finite-precision dot products, softmax and addition achieve these inverse-polynomial tolerances with $O(\log(N_w T_{\mathrm{round}}))$ bits, counting their arithmetic operations. Equal rounded head outputs still cancel on tokens.

For independently initialized layer memories, copy the preceding state before its dependent read. This adds constant layers per round and preserves the logarithmic-depth construction. □

H.2 Rewriting frees capacity occupied by obsolete distinctions

A feature useful for an earlier prediction can become unnecessary after later evidence. We show when retaining that feature strictly increases the best achievable log loss at fixed cache capacity. The comparison first concerns information; an explicit shared-attention construction then realizes the better cache with finite, counted computation.

H.2.1 THE REMAINING PREDICTIONS DETERMINE WHAT IS REDUNDANT

Let X be the observed context after the later evidence and Y a finite target for the predictions still required. For several permitted queries, sample a query Q independently of X, giving each query positive probability, and set $Y = (Q, A)$, where A is its answer. Equality of $p(Y \mid X)$ then means equality of every required answer distribution. Define

$$S(X) = p(Y = \cdot \mid X), \qquad W = f(X).$$

The *predictive state* S records the answer probabilities. The cache W includes all context-dependent information available to the reader. Contexts with the same S are predictively equivalent. An earlier feature $U = U(X)$ has redundant variation precisely when $H(U \mid S) > 0$: equivalent contexts still have different feature values. Here $H(\cdot)$ denotes Shannon entropy in nats; the scalar H elsewhere counts associations.

Probabilistic grammar models predict by averaging over structures compatible with the observed prefix; later words update these alternatives (Hale, 2001; Levy, 2008). This grounds two sources of temporary features:

- **Unresolved alternatives:** earlier parse weights can lose predictive relevance after disambiguation.
- **Completed associations:** a composed result can replace the intermediate features used to compute it.

In the first case, different earlier weights become redundant when later evidence leads to the same remaining predictive state.

A completed-association example. Let u, v be independent uniform bits and $F_u(a) = a \oplus u$, where $\oplus$ is exclusive-or. Before v arrives, $F_u(0) = u$ is needed. Afterwards, if only the composition is queried, storing $u \oplus v$ answers $F_v(F_u(a))$ for either input a. The intermediate bit then has $H(u \mid S) = \log 2$.

Redundancy is relative to the remaining queries. If those queries include an intermediate fact, that fact remains part of S. An append-only cache retains an earlier feature whenever it is recoverable from an unchanged entry; rewriting permits that distinction to be removed.

H.2.2 REDUNDANCY IS COSTLY WHEN THE CACHE ALSO MERGES DIFFERENT PREDICTIONS

Throughout this subsection, X, Y have finite support and writers are deterministic. Let C count permitted values of the complete cache, not memory rows. A reader g predicts a distribution $g(\cdot \mid W)$ and incurs expected log loss $\ell(f, g) = \mathbb{E}[-\log g(Y \mid W)]$.

Proposition H.2.1 (Storage loss and reader error are separate)**.**
Every writer and reader satisfy

$$\ell(f, g) = H(Y \mid X) + \mathcal{D}(W) + \mathcal{A}(W, g), \tag{H.2.1}$$
$$\mathcal{D}(W) = \mathbb{E} D_{\mathrm{KL}}\big(S \| p(Y \mid W)\big), \qquad \mathcal{A}(W, g) = \mathbb{E} D_{\mathrm{KL}}\big(p(Y \mid W) \| g(\cdot \mid W)\big).$$

Here D_{KL} is Kullback–Leibler divergence. Storage loss $\mathcal{D}(W) = H(Y \mid W) - H(Y \mid X)$ is positive exactly when different predictive states share one cache value with positive probability.

Proof: Insert the best prediction from the cache.

Add and subtract the expected negative log probabilities under $p(Y \mid W)$ and $p(Y \mid X)$. Since W is a function of X, the two differences are the displayed divergences. Conditional on W, the distribution $p(Y \mid W)$ averages the assigned states S. All divergences vanish exactly when those states agree. □

Corollary H.2.2 (Predictive separation quantifies collision loss)**.**
Suppose distinct predictive states have Euclidean distance at least $\delta > 0$. Draw W from its cache distribution, then draw S_1, S_2 independently conditional on that same value of W. With $\chi = \Pr(S_1 \neq S_2)$,

$$\mathcal{D}(W) \geq \delta^2 \chi / 4.$$

Proof: Convert disagreement into conditional variance.

Pinsker's inequality gives $D_{\mathrm{KL}}(p\|q) \geq \|p-q\|_2^2/2$. Conditional independence gives $\mathbb{E}\|S_1 - S_2\|_2^2 = 2\mathbb{E}\|S - \mathbb{E}[S \mid W]\|_2^2$. An unequal pair contributes at least δ^2; averaging proves the bound. □

Stored facts that the reader cannot yet compose contribute to reader error, rather than storage loss. The following result concerns the latter: one redundant distinction can be reallocated to resolve a predictive collision.

Lemma H.2.3 (A redundant distinction permits a strict improvement)**.**

- Two supported contexts have the same predictive state but different cache values.
- The cache has positive storage loss.

Changing one context's cache assignment, possibly after merging two cache values, strictly lowers storage loss without increasing C.

Proof: Consolidate a redundant distinction, then separate a collision.

Let the equivalent contexts be x_a, x_b, their state be s, and their cache values be u, v. Write $q_w = p(Y \mid W = w)$ and $c_s(q) = -\sum_y s_y \log q_y$. Relabel u, v so $c_s(q_u) \leq c_s(q_v)$.

1. Move the equivalent context.

Send x_b from v to u. If the inequality is strict, keeping the old reader already lowers loss. If equality holds but $q_u \neq s$, the old-reader loss stays fixed, while the new mixture at u changes to

$$q'_u = \frac{\mu q_u + a s}{\mu + a}, \qquad \mu = \Pr(W = u), \quad a = \Pr(X = x_b) > 0.$$

Replacing q_u by q'_u then strictly lowers loss by $(\mu + a) D_{\mathrm{KL}}(q'_u \| q_u) > 0$.

2. If that move does not help, free a value.

The remaining case has $q_u = s$ and equal cross-entropies. Hence $D_{\mathrm{KL}}(s\|q_v) = 0$, so $q_v = s$ too. Merge all of v into u. Their equal mixtures make this loss-neutral and leave v unused.

3. Use the freed value to split a collision.

Positive storage loss supplies a context z with state t different from its current mixture q. Send z alone to v. If its mass is a and its old cell has mass μ, the remaining nonempty cell has some distribution r. Reoptimizing the two predictions decreases loss by

$$a D_{\mathrm{KL}}(t\|q) + (\mu - a) D_{\mathrm{KL}}(r\|q) > 0.$$

Each step uses only the original cache values. □

Let $\mathcal{D}^*_C$ be the minimum storage loss among all writers into the C permitted values. This minimum exists because the context support and cache are finite.

Theorem H.2.4 (Predictive redundancy is strictly suboptimal when a collision remains)**.** An optimal writer can always be chosen as $W^* = \varphi(S)$. Moreover, if a given writer separates equivalent contexts and merges different predictive states, then

$$\mathcal{D}_C^* = \mathcal{D}(W^*) < \mathcal{D}(W).$$

Proof: Choose one cache value for each predictive state.

Start at an optimum and hold its conditional predictions fixed. For each state s, send all contexts with that state to a cache value minimizing $c_s(q_w)$. This cannot increase their average loss. Reoptimizing the reader cannot increase it either, so the resulting writer is still optimal and depends only on S. The strict comparison follows from Lemma H.2.3. □

Corollary H.2.5 (Exact capacity threshold for the cost of retaining an obsolete feature)**.**

- $U = U(X)$ has redundant variation: $H(U \mid S) > 0$.
- The cache can retain U: $C \geq |\operatorname{supp} U|$, where supp denotes the set of positive-probability values.

Let $J = |\operatorname{supp}(S, U)|$, and let $\mathcal{D}_C^U$ minimize storage loss over caches from which U is recoverable. Then

$$\mathcal{D}_C^U > \mathcal{D}_C^* \quad \Longleftrightarrow \quad C < J.$$

For $C \geq J$, both losses are zero.

Proof: Count the jointly required distinctions.

Recovering U forces at least one distinction between equivalent contexts because $H(U \mid S) > 0$. If $C < J$, the cache cannot recover both S and U: that would require distinct values for every supported pair. It therefore has a predictive collision and is strictly suboptimal by Theorem H.2.4. There are finitely many writers, so their retaining minimum has a strictly positive gap. For $C \geq J$, store the pair (S, U) and attain zero storage loss. □

When every original token must be recovered, S determines X; Section H.6 supplies a separate construction for that task.

H.2.3 Shared attention implements each strict reassignment in constant depth

The rewrite from Lemma H.2.3 tests one context and, when a merge is needed, one old cache value. We implement these tests and the resulting selections with shared attention. Throughout the constructions below, fixed address, reference, and role coordinates are preserved, readable sources remain available during writing, and numerical outputs are written on their declared finite grids. Projection maps are shared across token and memory positions within each layer.

For this construction, expose the source and old cache as R scalar coordinates of magnitude at most $V_0 \geq 1$. Distinct input values have gap at least $h_{\text{in}} > 0$; output coordinates use a grid of spacing $h_{\text{out}} > 0$. Each tested coordinate and its fixed reference occupy one working row. Any conversion from the original input layout to these rows is included in the coordinate-read interface. Fixed references and replacement values are the ones chosen by Lemma H.2.3.

Theorem H.2.6 (Constant attention depth implements the strict reassignment)**.**

- The source coordinates, references, and old cache are readable in the stated interface.
- The working rows and protected address and role fields are available.

Beyond the old writer and coordinate-read interface, its strictly better writer f' uses constant additional layers and heads, $O(R)$ working rows, $O(\log(R+2))$ width, and

$$O\big(R^2 \log(R+2) + R \log^2(R+2)\big)$$

arithmetic operations. Finite precision depending on $R, V_0, h_{\text{in}}, h_{\text{out}}$ gives the exact rewritten cache after rounding, at unchanged final capacity.

Proof: Compare coordinates, combine the tests, and select the cache.

1. Preserve other rows under shared projections.

An attention update can be restricted to destinations with fixed role $r = 1$. Use two heads with identical source scores t_j and values v_j. Add scores $+\omega r$ and $-\omega r$ to those sources, respectively, and the opposite score to a zero-valued anchor. If $Z = \sum_j e^{t_j}$ and $A = \sum_j e^{t_j} v_j / Z$, subtracting the head outputs gives

$$\left[\frac{Z}{Z + e^{-2\omega r}} - \frac{Z}{Z + e^{2\omega r}}\right] A.$$

At $r = 0$ the update is zero. At $r = 1$ it approaches A. For s sources with $|t_j| \le T$, a choice $2\omega \ge T + \log s + \log(2/\varepsilon)$ bounds the coefficient error by ε. Fixed type-score margins suppress other visible rows to any prescribed accuracy. Every subsequent update uses this pair of heads and a fresh zero output field or bank; fixed fields receive no updates.

2. Compare each input with its reference.

Attention to scalar keys and values $+1, -1$ gives $\tanh(t)$ for query t. Two such averages compute, for coordinate x_j and reference a_j,

$$e_j = \frac{1}{2}\left[\tanh\big(\lambda(x_j - a_j + h_{\text{in}}/2)\big) - \tanh\big(\lambda(x_j - a_j - h_{\text{in}}/2)\big)\right].$$

With $\lambda = O(h_{\text{in}}^{-1} \log(R + 2))$, this is within $1/(128R)$ of the equality indicator. Give the two constant rows a common type-score advantage so all other rows change each average by at most $1/(128R)$. The role construction writes the comparisons while preserving source data.

3. Combine and broadcast the tests.

For each of the at most two comparison groups, one dedicated head averages its $b \le R$ comparisons and uses the fixed output multiplier b. Thus no row-dependent multiplier is needed. Suppress other groups by their fixed type fields. Threshold the resulting sum s with $\tanh(\beta[s - b + 1/2])$. Keep comparison, aggregation, and arithmetic error together below $1/8$; the threshold argument then has magnitude at least $3/8$ with the correct sign. Fixed sufficiently large β gives a flag within $1/8$ of $+1$ for equality and -1 otherwise. An addressed read broadcasts both flags to every output-coordinate destination in one more layer. Allocate less than $1/8$ to its approximation and arithmetic error, so each copied flag remains within $1/4$ of its intended sign.

4. Select the replacement coordinates.

Assign distinct sign-code addresses of length $O(\log(R + 2))$ to output coordinates. For each coordinate, its old and replacement values share an address and have opposite branch signs. Write c_i for destination i's sign-code address and c_j for a candidate's address. If the candidate branch sign is $\sigma \in \{-1, +1\}$ and the copied flag is f, use score $2\gamma c_i^\top c_j + \gamma f \sigma$. The intended candidate exceeds every other candidate by at least $3\gamma/2$: a wrong branch loses at least $3\gamma/2$, and a wrong address loses at least 4γ before a possible branch gain of at most $5\gamma/2$. With cR candidates for a fixed constant c, unintended mass is at most $(cR - 1)e^{-\gamma}$. Choose $\gamma = O(\log(RV_0/h_{\text{out}} + 2))$ to give any required fraction of the output error budget. One selection makes a direct reassignment. In the merge case, a fresh bank first selects the merged cache, then a second bank overrides it on the selected context. This is exactly the rule of Lemma H.2.3.

5. Round and count.

Comparison, aggregation, thresholding, broadcasting, and the two selections use a constant number of stages. Their source, reference, anchor, and destination banks contain $O(R)$ rows of width $O(\log(R + 2))$. Dense attention costs $O(R^2 \log(R + 2))$ and projections cost $O(R \log^2(R + 2))$. All scores and derivatives are bounded for the stated finite input set. Choose working precision so accumulated arithmetic error, together with the allocated approximation errors, is below $h_{\text{out}}/4$. Ordinary nearest-grid writing therefore produces the exact cache coordinates. Retain this cache and discard the working banks. □

Corollary H.2.7 (Updating the affected reader outputs preserves the strict improvement)**.** For an old reader g with finite logits, let the reassignment lower storage loss by $\delta_{\text{loss}} > 0$. For a target with V values, smooth each changed conditional distribution by mixing it with the uniform distribution with weight $0 < \eta < 1$. If its implemented logits have maximum error τ, replacing the reader at the at most three affected cache values gives

$$\ell(f', g') \le \ell(f, g) - \delta_{\text{loss}} - \log(1 - \eta) + 2\tau.$$

It strictly improves the actual loss whenever $-\log(1 - \eta) + 2\tau < \delta_{\text{loss}}$.

Proof: Update only the changed conditional predictions.

At unaffected cache values the old loss is unchanged. On the affected values it is at least their old conditional entropy. The new conditional entropy is lower by δ_{loss}; smoothing adds at most $-\log(1 - \eta)$. Log-sum-exp and the target-weighted logit term each change by at most τ. The theorem's equality tests select the changed cache values, and fixed probability constants supply the new logits. This uses at most three cache comparisons and $O(V)$ probability entries. □

Corollary H.2.8 (Quantified improvements accumulate over successive rewrites)**.** Let $W_0, \ldots, W_j$ result from j reassignments of Lemma H.2.3. Suppose each moved context has probability at least $a > 0$, and each nonzero ℓ_1 distance between a selected state and a source or destination mixture is at least $\delta > 0$. Then

$$\mathcal{D}(W_j) \le \mathcal{D}(W_0) - ja^2\delta^2/2.$$

Proof: Lower-bound each strict step and add the gains.

Moving mass $\alpha \ge a$ with state s into a cell of mass μ and mixture $q \ne s$ changes that mixture by $\alpha(s - q)/(\mu + \alpha)$. Reoptimizing its prediction gains at least

$$\frac{\alpha^2 \|s - q\|_1^2}{2(\mu + \alpha)} \ge a^2\delta^2/2,$$

by Pinsker and $\mu + \alpha \le 1$. A direct move into mixture s gains at least $\alpha D_{\text{KL}}(s \| q_{\text{old}})$ for its old mixture $q_{\text{old}} \ne s$. The final split has the same bound, with s denoting the isolated state and q_{old} its pre-split mixture. In each case the gain is at least $\alpha\delta^2/2 \ge a^2\delta^2/2$. The intervening merge is loss-neutral. Summing proves the claim. □

H.2.4 Finite shared functions attain the optimal same-capacity loss

The direct construction realizes a strict step. To attain the optimal finite-task loss, we compile a complete writer and reader. Give the C permitted cache values an injective binary code of length b_{cache}; only these C codewords are valid. Fixed addresses and controls are identical across contexts. The following circuit construction makes its resource bounds explicit.

Use a Boolean circuit with n input bits, G NAND gates, and dependency depth d_0. Output copies and the binary input/output conversions are included in this count. Store circuit states on a uniform fixed-point grid of spacing $\Delta = 1/2$, with range containing $[-2, 2]$. Evaluate projections and attention at finer working precision, then round state writes to this grid. These storage and working precisions are counted separately.

Lemma H.2.9 (An explicit attention circuit evaluates a finite Boolean computation)**.**

- All circuit input and working rows are readable.
- Fixed row identities and constants survive positional and normalization operations.
- State writes use the stated uniform grid and finer working arithmetic.

The circuit has an implementation using constant heads, $O(d_0 + 1)$ layers, $O(n + G + 1)$ working rows and width, and finite precision. All projections are shared across positions within a layer. With $R = n + G + 3$, dense attention and projections use $O((d_0 + 1)R^3)$ arithmetic operations.

Proof: Copy gate inputs, then evaluate a softmax threshold.

Encode false and true as $-1, +1$. Allocate a row to each input and gate, plus three constant rows. Each row contains a fixed one-hot identity; each gate's output coordinate starts at zero. Two scratch coordinates per circuit depth hold that depth's operands. Thus every coordinate and row count is $O(R)$. A linear query map of the identities selects the prescribed input addresses and identifies gates active at the current depth.

1. Copy only to active gate rows.

The copy sources are all rows except the zero-valued anchor. Give the requested source score β and other sources score zero. For source values in $[-1, 1]$, its attention average A differs from the requested bit by at most $2(R-1)e^{-\beta}$. Two heads isolate the destination role: for activity bit $r \in \{0, 1\}$, add $+\omega r$ or $-\omega r$ to every source score, and the opposite score to a zero-valued anchor. If Z is the sum of exponentiated source scores, their outputs are

$$\frac{Z}{Z + e^{-2\omega r}} A \quad \text{and} \quad \frac{Z}{Z + e^{2\omega r}} A.$$

Subtract them. At inactive rows they cancel exactly; at active rows their difference approaches A. Choose finite $\beta, \omega = O(\log(R+2))$ so the error is below $1/8$. Two such pairs of heads copy the two operands into fresh depth-specific scratch coordinates. The ideal error plus working-arithmetic error is below $1/4 = \Delta/2$, so ordinary grid rounding makes active operands exact signs and inactive coordinates zero. The same linear maps act on input, gate, and constant rows.

2. Evaluate NAND without a feedforward network.

Write a, b for the copied operands. A query $t = \lambda(r - a - b)$ attends to two constant rows with scalar keys and values $+1, -1$. Give these two rows an equal additional score Ω; every other row has zero value and score zero. The output is

$$\frac{2e^{\Omega} \sinh t}{2e^{\Omega} \cosh t + R - 2}.$$

At an inactive row, $r = a = b = 0$, so the update is zero. At an active gate, $1 - a - b$ is -1 only when both inputs are true and is positive otherwise. Fixed sufficiently large λ and $\Omega = O(\log(R+2))$ make the output within $1/8$ of the NAND sign. Add it to the active gate's initially zero output and round. Previously computed rows receive zero update because they are inactive and their scratch coordinates for this depth are zero.

3. Control errors and count resources.

Gate outputs become readable in the following layer; each circuit depth uses a constant number of layers. Fixed fields receive no updates. Before each rounding, leave arithmetic error below $1/8$ in addition to the approximation error above. All inputs to a stage lie in a bounded finite set, all scores are finite, and every intended stored value is at least $1/4$ from its grid-cell boundaries. Consequently sufficiently many finite precision bits preserve every stage by induction. If Q_0 operations occur per stage and Λ bounds error amplification within a stage, it suffices to make $Q_0 \Lambda 2^{-p} < 1/8$, with enough range bits for all scores; p is the fractional precision. Dense operations on $O(R)$ rows of width $O(R)$ give the claimed work. Working rows are counted separately from the final retained cache. □

Theorem H.2.10 (Bounded shared functions realize the strict retention advantage)**.**

- The finite task and C-configuration cache above are given through finite binary input and output interfaces.
- The attention and rounding interface of Lemma H.2.9 is available, with its working resources counted.

For every $\epsilon > 0$, finite-resource shared-attention writer and reader functions satisfy

$$W^* = \varphi(S), \qquad \ell(W^*, g^*) \le H(Y \mid X) + \mathcal{D}_C^* + \epsilon.$$

If $C < J$ in Corollary H.2.5, choosing $\epsilon < \mathcal{D}_C^U - \mathcal{D}_C^*$ yields lower loss than every same-capacity cache retaining U, irrespective of its reader.

Proof: Compile the optimal assignment and its smoothed reader.

1. Write the finite optimal cache.

Theorem H.2.4 gives an assignment constant on each predictive state. With an n-bit input and the prescribed b_{cache}-bit output code, equality tests against all supported contexts and output selection form a Boolean lookup circuit of size $O((n + b_{\mathrm{cache}})2^n)$. The compiler writes its exact permitted output code. Discard its inputs and working rows, retaining only that code, so the reader has only the prescribed C possible cache values. The task distribution fixes this lookup table.

2. Read with finite logits.

At each used cache value w, let $p_w = p(Y \mid W^* = w)$ and let V be the number of targets. Choose $0 < \eta < 1$ and smooth to $p_{w,\eta} = (1 - \eta)p_w + \eta/V$. These distributions have finite logarithms. A finite lookup circuit produces binary approximations to their logits. Addressed copies collect these bits into a readout row; a final linear readout with signed binary place weights converts them to V logits. This final readout uses the finer output precision needed for the desired logit tolerance. The extra rows, width, layers, constants, and precision are included in the reader resources. Smoothing adds at most $-\log(1 - \eta)$ to loss. If every logit has error at most τ, the log-sum-exp and target-weighted terms each change by at most τ. Choose η, τ with $-\log(1 - \eta) + 2\tau \le \epsilon$.

3. Compare actual losses.

The loss split gives the stated upper bound, while every retaining writer and reader has loss at least $H(Y \mid X) + \mathcal{D}^U_C$. The chosen ϵ preserves the strict gap. □

For a supplied association algorithm, its own gate count and depth replace the lookup bound. Local rewriting applies these constructions when the relevant evidence and working states are available in the chunk.

H.3 Local rewriting preserves the gains with quadratic attention work

Writing each completed chunk once preserves the composition and storage gains under a locality model while avoiding repeated processing of the same prefix. We first count attention work, then show that a power-law distance model bounds unfinished compositions and missed storage improvement. At fixed chunk length, the construction retains logarithmic expected depth.

H.3.1 One write per chunk removes cubic repeated attention work

Earlier memories still contribute attention work even though they are never rewritten. Let $N = JK$, with J chunks of K tokens and m memory queries per chunk at the counted layer.

Lemma H.3.1 (One write per chunk has quadratic attention work)**.**

- Lexical queries read at most K lexical rows and earlier memories. Memory queries also read their own chunk's m memories.
- Each chunk is evaluated once, with fixed projection width D.

The number of permitted query–source pairs is at most

$$P_{\max} = J(K^2 + Km + m^2) + m(K + m)\frac{J(J-1)}{2}. \tag{H.3.1}$$

For $\rho = m/K$ and $a_\rho = \rho(1 + \rho)/2$, this is

$$P_{\max} = a_\rho N^2 + (1 + a_\rho)NK. \tag{H.3.2}$$

In particular, $m = K$ gives $P_{\max} = N^2 + 2NK$.

Proof: count current sources and cached sources separately.

Current sources.

Each chunk contributes at most K^2 lexical pairs and $m(K+m)$ memory-query pairs.

Earlier memories.

Chunk j has $K+m$ queries and $(j-1)m$ earlier memory sources. Summing their product gives the second term of (H.3.1); substituting $J = N/K$ gives (H.3.2). □

Across L layers the attention and projection work is $O(LDP_{\max} + LN(1+\rho)D^2)$. At fixed ρ, partial endpoint chunks preserve $O(N^2 + NK)$ pairs; the displayed polynomial uses full chunks. Independently rewriting every prefix instead costs $\sum_{t=1}^{N} t^2 = \Theta(N^3)$ pairs per layer. A single bidirectional prefill already costs quadratic attention work. Section I compares PAT's implemented mask with cached full attention, including projections and feed-forward work.

H.3.2 Language locality motivates a power-law calculation model

Hahn et al. (2021) connect nearby predictive dependencies to a favorable memory–surprisal trade-off and test that account against word and morpheme order. Xiong et al. (2024) fit power-law-plus-constant curves to their models' validation loss as context length increases. Together, these results motivate a locality model in which most association compositions and redundant-feature removals use nearby evidence, with a decreasing contribution from longer distances. We express this structure through a power-law bound on the expected number of required calculations by prerequisite distance.

Which evidence must lie together. Fix the original inputs required for each calculation, including records, selectors, and evidence that a feature is redundant for every remaining prediction. Its *prerequisite interval* runs from the earliest to the latest such position; its integer distance is d. Specify these intervals before choosing chunk boundaries. A previously constructed summary can supply an input through its retained memory. The following setup applies to the local constructions below.

- Required records and selectors are available at writing time in current token states or readable earlier memories. Still-required original inputs remain in addressed archive fields.
- All supported starting rows, local calculations, archive rows, and summary copies fit jointly in the writer's layers, heads, width, memory, and precision. Address selection and numerical reads use the constructions of Sections H.1 and H.2.3.
- Each summary is stored at the layers where later queries read it under the attention mask. Adjacent composition intervals overlap at an original record shared by their two compositions.

Prediction queries use the completed observed prefix. Within-chunk bidirectional writes become sources for following chunks, so every read respects the prediction's information deadline.

H.3.3 Boundary counts measure the available rewrite opportunities

Choose the chunk offset uniformly from $0, \dots, K-1$, independently of the example and its specified prerequisite intervals. Endpoint chunks may be shorter. Expectations below average over this offset as well as the prediction workload.

Lemma H.3.2 (A prerequisite interval is contained with probability $(1 - d/K)_+$)**.**
An interval of integer distance $d \geq 0$ crosses a chunk boundary with probability $\min(d/K, 1)$, where $(x)_+ = \max(x, 0)$.

Proof: count the offsets that cut the interval.

For $d < K$, exactly d of the K offsets place a boundary between the two endpoints. For $d \geq K$, all offsets do so. Taking the complement proves the formula, including $d = 0$. □

Proposition H.3.3 (Containment recovers a specified collection of representation changes)**.**

- A compatible collection changes $u_i \geq 0$ representation occurrences using prerequisite distance d_i; each occurrence is counted once.
- The changes satisfy the construction setup and preserve every required future prediction.

Let $U_{\mathrm{occ}} = \sum_i u_i > 0$ be finite, and draw $S_{\mathrm{span}} = d_i$ with probability u_i/U_{occ}. The expected fraction of this collection available within chunks is

$$\mathbb{E}(1 - S_{\mathrm{span}}/K)_+ = 1 - \mathbb{E}\min(S_{\mathrm{span}}/K, 1) \geq 1 - \frac{\mathbb{E}S_{\mathrm{span}}}{K}. \tag{H.3.3}$$

For a fixed span distribution, the fraction tends to one as K grows through feasible lengths. When $\mathbb{E}S_{\mathrm{span}} < \infty$, its deficit is at most $\mathbb{E}S_{\mathrm{span}}/K$.

Proof: weight each containment probability by its changed occurrences.

Multiply Lemma H.3.2's probability by u_i/U_{occ} and sum. Use $\min(s/K, 1) \leq s/K$ for the bound and bounded convergence for the limit. □

Weighting each occurrence by its representation size gives the corresponding storage accounting. The next construction connects contained changes to a strict prediction-loss improvement.

How a contained rewrite can strictly improve storage loss. Section H.2 supplies a mechanism beyond counting removable features: merge predictively equivalent contexts and reuse the freed cache configuration to separate a predictive collision. Apply that construction to a prediction component whose inputs and old cache are available in the current chunk, and whose reader uses its separately allocated cache. If the earlier feature U remains recoverable in C configurations but $C < |\operatorname{supp}(S, U)|$ and $H(U \mid S) > 0$, Corollary H.2.5 gives a strict storage-loss gap. The shared-attention construction of Section H.2.3 realizes the reassignment under the construction setup, and its reader update preserves a strict prediction-loss gain. All its inputs belong in the prerequisite interval.

Independent prediction components with separate cache allocations give one construction in which these gains add, with query probabilities included in each gain. The theorem below applies to any jointly feasible collection satisfying this additive gain model.

H.3.4 A CALCULATION-DISTANCE TAIL BOUNDS UNFINISHED WORK AND FORGONE GAINS

Let H bound the supported chain length; the workload includes chains of any length up to H. For each prediction, list its adjacent association compositions and compatible feature-removal calculations. Define

$$a_{N,H}(s) = \mathbb{E}\sum_i \mathbf{1}\{d_i \geq s\}, \qquad s \geq 1. \tag{H.3.4}$$

The expectation is over the workload. Let Z count all listed intervals cut by the random chunk boundaries, and let $B \leq Z$ count the cut adjacent compositions in the prediction's chain.

Theorem H.3.4 (The distance tail bounds boundary crossings and missed loss improvement)**.**

- The offset is independent and uniform as above, and $a_{N,H}(s) \le As^{-\alpha}$ with $A > 0$ and $0 < \alpha < 1$ fixed as N, H vary.
- The compared feature removals are jointly feasible and have additive loss improvements $0 \le g_i \le g_{\max}$, with $g_{\max}$ fixed across the workload. The local writer performs contained removals and otherwise preserves the comparison representation.

Writing Δ_K for the expected improvement forgone by omitting cut removals,

$$\mathbb{E}B \le \mathbb{E}Z \le \frac{A}{1-\alpha}K^{-\alpha}, \qquad 0 \le \Delta_K \le \frac{g_{\max}A}{1-\alpha}K^{-\alpha}. \tag{H.3.5}$$

Thus fixed chunks keep both quantities bounded as context length and maximum supported chain length grow. Increasing chunk length reduces both bounds at rate $K^{-\alpha}$.

Proof: sum cut probabilities, then weight the missed removals.

Count cut calculations.

Condition on the example. Lemma H.3.2 and $\min(d, K) = \sum_{s=1}^{K} \mathbf{1}\{d \ge s\}$ give

$$\begin{aligned}\mathbb{E}Z = \mathbb{E}\sum_i \min(d_i/K, 1) &= \frac{1}{K}\sum_{s=1}^{K} a_{N,H}(s) \\ &\le \frac{A}{K}\sum_{s=1}^{K} s^{-\alpha} \le \frac{A}{1-\alpha}K^{-\alpha}.\end{aligned} \tag{H.3.6}$$

The last step compares the sum with $\int_0^K x^{-\alpha}\,dx$. Different boundary-crossing events need not be independent. Since $B \le Z$, the same upper bound applies to $\mathbb{E}B$.

Weight omitted removals.

Additivity gives $\Delta_K = \mathbb{E}\sum_{i \text{ a removal}} g_i \mathbf{1}\{i \text{ crosses a boundary}\}$. Each summand is at most $g_{\max}$ times its crossing indicator, proving the loss bound. □

Fixed chunks preserve logarithmic expected depth. The construction completes contained chain portions in parallel and leaves only their joins for the prediction query.

Corollary H.3.5 (Prepared chunk portions leave at most B sequential joins)**.**
Under the construction setup and Theorem H.3.4's distance model, the dependency depth is

$$L_{\text{local}} = O(\log(H+1) + B), \qquad \mathbb{E}L_{\text{local}} = O(\log(H+1) + AK^{-\alpha}). \tag{H.3.7}$$

For fixed K, this gives $\mathbb{E}L_{\text{local}} = O(\log(H+1))$, with constants independent of N, H. An allocated stack supporting b additional joins covers a workload fraction at least $1 - AK^{-\alpha}/((1-\alpha)b)$ for $b > 0$.

Proof: compose contained portions in parallel and follow their endpoints.

Partition the chain.

Cut at each of its B boundary-crossing adjacent compositions. Consecutive uncut intervals overlap and are each contained in one chunk, so they must occupy the same chunk. Thus every nontrivial remaining portion has its prerequisites together; isolated associations retain their addressed original records.

Prepare the portion tables.

Apply the pointer-doubling construction to each contained portion, with its endpoint made terminal. All supported starting rows advance in parallel. After $O(\log(H+1))$ layers, each points to its portion's endpoint. Freeze that endpoint in a protected field and copy it into the subsequent retained banks, together with archive fields for isolated associations. The shared copying construction supplies these fields at every join layer. These constant-size fields and their retained rows are included in the memory allocation.

Join the portions at the query.

The query follows at most $B+1$ addressed portion results or isolated associations, each in constant layers. This gives the first bound. Taking expectations and using Theorem H.3.4 gives the second. Markov's inequality gives $\Pr(B > b) \le AK^{-\alpha}/((1-\alpha)b)$ and hence the stated layer-budget coverage. □

H.3.5 Larger chunks trade completed calculations for local attention work

At fixed N and $\rho = m/K$, full chunks store ρN memories per layer, independent of K. A memory query at the end sees at most $K + \rho N$ token and memory candidates. Longer chunks expose more prerequisites together and add token candidates; the earlier-memory count stays fixed.

Let $G(K)$ be the expected loss improvement of a specified feasible writer, and let $\mathcal{K}$ be its nonempty finite set of feasible integer chunk lengths. From (H.3.2), the full-chunk attention work relative to projection work is proportional to $[a_\rho N + (1+a_\rho)K]/[(1+\rho)D]$. For a relative work price $\lambda > 0$, set $\gamma = \lambda(1+a_\rho)/(1+\rho)$. The K-independent term leaves the minimizing chunk length unchanged. We use this full-chunk cost model to choose K.

Proposition H.3.6 (Lower relative attention cost favors longer chunks)**.**

- The gain curve G, feasible set $\mathcal{K}$, and positive price γ are fixed across widths.

For baseline loss L_0, minimize

$$J_D(K) = L_0 - G(K) + \gamma K/D, \qquad K \in \mathcal{K}. \tag{H.3.8}$$

If $D_2 > D_1$, any respective minimizers satisfy $K_2 \ge K_1$.

Proof: compare the two optimality inequalities.

Add $J_{D_1}(K_1) \le J_{D_1}(K_2)$ and $J_{D_2}(K_2) \le J_{D_2}(K_1)$. The gain terms cancel, leaving $\gamma(K_2 - K_1)(1/D_1 - 1/D_2) \ge 0$. The positive prefactors give the ordering. □

A power-law loss curve can be attained by a bounded writer. The following construction derives the predictive gain of each completed association and then distributes those gains over distances. It uses the shared-attention and finite-grid interface of Section H.2.3.

Proposition H.3.7 (A finite prediction workload attains a power-law missed gain)**.**
Fix $0 < \alpha < 1$ and an integer distance cutoff $R \ge 1$. Choose a fixed representable logit $\beta > 0$ (for example, $\beta = 1$) and put $\epsilon = (1+e^{\beta})^{-1}$. There is a finite workload and a local writer with one retained data bit per prediction component whose achieved loss is

$$\ell_K = h(\epsilon) + \Delta_K, \qquad \Delta_K = \frac{g}{K} \sum_{s=1}^{\min(K,R)} s^{-\alpha}, \qquad g = \log 2 - h(\epsilon) > 0, \tag{H.3.9}$$

where $h(\epsilon) = -\epsilon\log\epsilon - (1-\epsilon)\log(1-\epsilon)$. In particular, for $1 \le K \le R$,

$$gK^{-\alpha} \le \Delta_K \le \frac{g}{1-\alpha} K^{-\alpha}.$$

The writer uses a constant number of Boolean gates, working rows, heads, and layers after addressed source reads. Fixed address fields use $O(\log N)$ width; sufficient finite precision preserves a positive constant gain and the same bounds up to constant factors.

Proof: compute a strict gain, then distribute its prerequisite distance.

Build the prediction component.

Observe independent uniform bits U, V and predict $Y = U \oplus V \oplus E$, where E is independent with $\Pr(E = 1) = \epsilon$. The earlier prediction asks for U; the later task uses the completed association $S = U \oplus V$. A one-bit cache retaining U has optimal later loss $\log 2$, since S is independent of U. Writing S gives loss $h(\epsilon)$, so the storage improvement is the derived constant $g > 0$. Source addresses are supplied as public metadata at writing and in the final query; they are independent of the bit values.

Implement the local write.

Allocate the component's retained bit to the chunk containing its first source. If both sources lie there, compute S before writing; otherwise write U. Other retained fields are fixed, and the working banks are discarded after writing. Exclusive-or is a constant-size Boolean circuit, so Lemma H.2.9 implements the write with shared projections after the two addressed reads. The reader compares the supplied source chunk indices and predicts from S when contained, or uses the uniform distribution otherwise. The informative reader uses logits $(0, \beta)$ or $(\beta, 0)$ according to S; the uninformative reader uses equal logits. Thus the actual component gain is g precisely when its prerequisite interval is contained. Fixed sufficiently accurate readout logits give the same construction with a constant gain between $g/2$ and g; the cache bit itself is written exactly on the finite grid. More precisely, if the implemented informative reader has loss $\ell_{\mathrm{full,impl}}$, put $g_{\mathrm{impl}} = \log 2 - \ell_{\mathrm{full,impl}}$. Relative to that same implemented full rewrite, the missed gain is exactly $g_{\mathrm{impl}}\mathbb{E}\min(d/K, 1)$, so every bound below holds with g replaced by $g_{\mathrm{impl}} > 0$.

Assign distances with a power-law tail.

Separate the two sources by d positions, filling unused positions with fixed symbols, and set

$$\Pr(d = r) = r^{-\alpha} - (r+1)^{-\alpha} \quad (1 \le r < R), \qquad \Pr(d = R) = R^{-\alpha}.$$

These probabilities sum to one and give $\Pr(d \ge s) = s^{-\alpha}$ for $1 \le s \le R$. Independently shift the chunks. The containment formula gives

$$\Delta_K = g\,\mathbb{E}\min(d/K, 1) = \frac{g}{K} \sum_{s=1}^{\min(K,R)} \Pr(d \ge s),$$

which proves (H.3.9). For $K \le R$, each summand is at least $K^{-\alpha}$; the integral bound supplies the upper estimate. There is one adjacent composition and one feature removal, both with span d, so $a_{N,H}(s) \le 2s^{-\alpha}$ for every $H \ge 2$, with zero tail beyond R. Independent copies in separate allocations give additive gains, with their query probabilities included in each gain. A fixed number of copies keeps A independent of N, H, R. □

The optimal exponent holds for all comparable power-law gain tails. For the additive model, define the gain-weighted tail $w(s) = \mathbb{E}\sum_i g_i \mathbf{1}\{d_i \ge s\}$ and let $L_{\mathrm{full}} = L_0 - \mathbb{E}\sum_i g_i$. This includes correlations between a calculation's distance and its predictive value. The achieved loss–work objective is $J_D(K) = L_{\mathrm{full}} + \Delta_K + \gamma K/D$.

Corollary H.3.8 (Power-law predictive gains give sublinear optimal chunk growth)**.**

- For $1 \le s \le R$, the gain tail obeys $c_- s^{-\alpha} \le w(s) \le c_+ s^{-\alpha}$, with fixed $c_-, c_+ > 0$ and $0 < \alpha < 1$.
- The feasible lengths lie in $[1, R]$ and include a length within fixed positive factors of $D^{1/(1+\alpha)}$. The relative work price $\gamma > 0$ is fixed.

Every minimizer of the achieved loss–work objective satisfies

$$K_* = \Theta\big(D^{1/(1+\alpha)}\big). \tag{H.3.10}$$

Proposition H.3.7 supplies a finite shared-attention workload satisfying the gain premise. For the exact specialization $\Delta_K = A_0 K^{-\alpha}$, the positive real minimizer is

$$K_* = (\alpha A_0 D/\gamma)^{1/(1+\alpha)}. \tag{H.3.11}$$

Proof: bound the achieved loss and compare feasible chunk scales.

Convert the gain tail into missed loss.

The same containment identity as in Theorem H.3.4 gives

$$\Delta_K = \frac{1}{K}\sum_{s=1}^{K} w(s), \qquad c_- K^{-\alpha} \le \Delta_K \le \frac{c_+}{1-\alpha} K^{-\alpha}.$$

Locate every minimizer.

Put $k_0 = D^{1/(1+\alpha)}$. A feasible length comparable to k_0 has excess objective at most $C_0 k_0^{-\alpha}$ for a fixed $C_0 > 0$. Optimality and the two nonnegative objective terms therefore imply

$$c_- K_*^{-\alpha} \le C_0 k_0^{-\alpha}, \qquad \gamma K_*/D \le C_0 k_0^{-\alpha}.$$

Rearranging bounds K_*/k_0 above and below by positive constants, proving (H.3.10) for every minimizer, including integer choices.

Recover the exact power-law formula.

For $\Delta_K = A_0 K^{-\alpha}$, differentiate the strictly convex objective: $J_D'(K) = -\alpha A_0 K^{-1-\alpha} + \gamma/D$. Its zero gives (H.3.11). Under integer feasibility, compare the closest feasible lengths on either side. □

Thus wider models make local attention relatively cheaper, while the attained diminishing predictive gains keep the optimal chunk growth sublinear. The earlier-memory term remains part of total work and is independent of K at fixed N, ρ, D.

Corollary H.3.9 (The original locality class has the same optimal worst-case chunk scale)**.** Fix the workload class of Theorem H.3.4 with $A, g_{\max} > 0$, the construction setup, and contexts supporting prerequisite distances through R. Its worst-case missed gain satisfies

$$\overline{\Delta}_K := \sup_{\text{workloads in the class}} \Delta_K = \Theta(K^{-\alpha}), \qquad 1 \le K \le R.$$

Whenever the feasible lengths include the comparison scale of Corollary H.3.8, every minimizer of $\overline{\Delta}_K + \gamma K/D$ has $K_* = \Theta(D^{1/(1+\alpha)})$.

Proof: match the uniform bound with an attained workload.

Theorem H.3.4 gives the upper bound for the whole class. In Proposition H.3.7, choose the fixed component gain at most $g_{\max}$ and activate the component with an independent public flag of probability $p = \min(A/2, 1) > 0$; inactive examples have a fixed answer and no required component calculation. Then the count tail is at most $2ps^{-\alpha} \le As^{-\alpha}$ and the missed gain is at least $pgK^{-\alpha}$. This gives the lower bound within the original class. The two-term comparison in Corollary H.3.8 applies to $\overline{\Delta}_K$ and proves the optimizer scale. □

The finite-span square-root case. For a finite compatible collection with additive gains, $G(K) = \sum_i g_i(1 - d_i/K)_+$. On the range $K \ge \max_i d_i$, put $A_1 = \sum_i g_i d_i > 0$. Then $J_D(K) = L_{\text{full}} + A_1/K + \gamma K/D$ is strictly convex and has stationary point $\sqrt{A_1 D/\gamma}$. This solves the objective restricted to that range when feasible. It is a global choice only after comparison with feasible shorter chunks, where some gains are unavailable.

Locality determines which calculations can finish before an old chunk becomes immutable. Section H.4 next counts the information that must remain, and Section H.5 relates memories per layer to composition depth.

H.4 Predictive abstraction reduces the storage needed for a given loss

Rewriting saves storage when a completed result replaces distinctions that no remaining prediction uses. We quantify that saving first in cache configurations and then, under a stated linear reader, in memory coordinates. Bit capacity also gives a lower bound that applies beyond the linear model.

Which computed values remain necessary. Keep every value used by an unfinished calculation or a permitted future query. A completed result can replace its inputs once it supplies all their remaining uses. To verify a proposed replacement, order the unfinished calculations by dependency: each must recover its inputs from the cache or from previously reproduced results. Induction then reproduces the answers. Removing c separately allocated bits saves c bits; replacing them by a sufficient $c' < c$-bit result saves $c - c'$ bits, including any copies still required.

H.4.1 Exact prediction requires one configuration per predictive state

Proposition H.4.1 (The configuration cost of retaining redundant information)**.**
In the finite setting of Section H.2, the minimum numbers of cache configurations attaining Bayes loss $H(Y \mid X)$ are

$$C_{\mathrm{pred}} = |\operatorname{supp} S|, \qquad C_{\mathrm{pred+U}} = |\operatorname{supp}(S, U)|$$

without and with the requirement to recover a feature U. Fixed-length binary codes require respectively $\lceil \log_2 C_{\mathrm{pred}} \rceil$ and $\lceil \log_2 C_{\mathrm{pred+U}} \rceil$ bits.

Proof: Count the distinctions that must survive.

By Proposition H.2.1, zero storage loss requires different predictive states to have different cache values. Recovering U also distinguishes every supported pair (S, U). Storing the state, or the pair, attains these lower bounds; binary indexing gives the bit counts. □

Redundant variation strictly lowers the configuration requirement; integer rounding determines the whole-bit saving. For a reference predictor with loss ℓ_{FA}, the information-level capacity needed to match it is instead

$$C_{\mathrm{match}} = \min\{C : H(Y \mid X) + \mathcal{D}_C^* \le \ell_{\mathrm{FA}}\}.$$

The loss curve determines this threshold, with reader-approximation error included when matching implemented predictions.

H.4.2 Predictive rank gives the exact minimum for linear access

Fix one query with V possible answers and n supported contexts, each of positive probability. Choose a basis for the computed-feature span and tabulate it as $\Phi \in \mathbb{R}^{n \times q}$, with q linearly independent columns. This basis change preserves every available linear predictor. A linear writer stores $Z = \Phi A \in \mathbb{R}^{n \times b}$, and a linear reader produces logits $F = ZB \in \mathbb{R}^{n \times V}$. Here b is the number of stored coordinates and A, B are freely chosen matrices. We center each logit row, so $F\mathbf{1} = 0$; subtracting a row's mean leaves its softmax probabilities unchanged.

Let P average table rows over contexts with the same predictive state S, using their conditional probabilities. Closure under this averaging means

$$P\Phi = \Phi T$$

for some $q \times q$ matrix T. This checkable condition says that averaging does not leave the available feature space. It holds, for example, for the full finite indicator basis.

Lemma H.4.2 (An attained linear optimum removes variation within predictive states)**.**

- The feature space is closed under predictive-state averaging: $P\Phi = \Phi T$.
- Its minimum log loss is attained at finite centered logits $F^* = \Phi W^*$.

The optimal centered logit table is unique and satisfies $PF^* = F^*$.

Proof: Average logits without changing their target.

1. Establish strict convexity.

For target probabilities s, cross-entropy is

$$c_s(z) = \log \sum_{y=1}^{V} e^{z_y} - s^\top z.$$

Its Hessian is the covariance matrix of the positive softmax probabilities. It is positive definite on the centered subspace, giving strict convexity there.

2. Average within each state.

The target s is constant within a predictive state, so Jensen's inequality gives $\ell(PF) \leq \ell(F)$, with strict inequality if any centered rows within that state differ. Averaged logits are feasible because $PF = P\Phi W = \Phi TW$. An optimum must therefore satisfy $PF^* = F^*$. Strict convexity and positive context probabilities also imply uniqueness on the feasible centered logit space. □

Theorem H.4.3 (Predictive rank is the exact linear storage minimum)**.**

- Lemma H.4.2 applies.
- The writer and reader may use any factorization $F = \Phi AB$ through b coordinates. Constants and biases, if needed, are included among the counted features and coordinates.

Let $r = \operatorname{rank} F^*$. Exactly r stored coordinates suffice to attain the feature family's optimal loss. A minimum-size cache can depend only on S.

Proof: Factor the unique optimal logit table.

1. Lower bound.

Any predictor through b coordinates has centered logit rank at most b. Uniqueness requires its optimal centered table to equal F^*, so $b \geq r$.

2. Matching construction.

Full column rank of Φ gives $\operatorname{rank} W^* = \operatorname{rank} F^* = r$. Factor $W^* = AB$ with A having r columns and B having r independent rows. This produces F^* through r coordinates. Choose a right inverse B^+, meaning $BB^+ = I_r$, where I_r is the identity. The stored table is $Z = \Phi A = F^* B^+$; hence $PZ = Z$. Every retained coordinate is constant within each predictive state. For $r = 0$, the zero logit table needs no stored coordinates. □

Corollary H.4.4 (Packing those coordinates gives a linear-access row minimum)**.**
If each row has $d \geq 1$ freely usable data coordinates and the reader can apply an arbitrary linear map to the concatenation of all rows, then

$$M_{\min} = \lceil r/d \rceil.$$

Address and control fields are additional counted coordinates. This matches full attention's loss when full attention attains the same feature-family optimum.

Proof: Count and pack coordinates.

M rows provide Md data coordinates, so $Md \geq r$ is necessary. Pack the theorem's r coordinates into $\lceil r/d \rceil$ rows, pad with zeros, and apply its linear reader to attain the bound. □

Finite-logit error at most τ gives loss within 2τ of the optimum by the log-sum-exp bound. An attention implementation supplies the stated linear access within its layer budget.

H.4.3 A SHARED ATTENTION EDIT REMOVES REDUNDANT VARIATION AND LOWERS LOSS

The averaging principle also gives an explicit descent direction through shared attention parameters. Write the editable memory data as $\mathsf{H} = G + Z \in \mathbb{R}^{m \times d_m}$, where m is the number of rows and d_m their data width. A fixed linear map P_Z isolates the redundant component:

$$GP_Z = 0, \qquad ZP_Z = Z.$$

Each head has m fixed address coordinates and value width at least $\operatorname{rank} P_Z$. The bank remains readable during its replacement. All reader operations from the edited values to logits are affine, with attention weights held fixed. Let $\mathcal{I}$ record S, G, the unchanged reader inputs, and this affine map.

Proposition H.4.5 (Two attention heads remove centered redundant variation)**.**

- The redundant component is conditionally centered: $\mathbb{E}[Z \mid \mathcal{I}] = 0$.
- The bank and reader use the stated linear and address interfaces.

Two attention heads on the bank implement the path $\mathsf{H}(t) = G + (1-t)Z$, for $0 \le t \le 1$. For exact evaluation, let q_0, q_1 be the reader distributions at $t = 0, 1$, and let $\ell(t)$ be expected log loss. Then

$$\ell(0) - \ell(1) = \mathbb{E} D_{\mathrm{KL}}(q_1 \| q_0), \tag{H.4.1}$$
$$\ell'(0) = -\mathbb{E}\big[D_{\mathrm{KL}}(q_1 \| q_0) + D_{\mathrm{KL}}(q_0 \| q_1)\big].$$

Loss reduction and descent are strict exactly when the edit changes the predicted probabilities with positive probability.

Proof: Cancel the component, then compute the loss change.

1. Implement the edit with two shared heads.

Let $U_m = \mathbf{1}\mathbf{1}^\top/m$ be uniform attention and $A_m = (I_m + U_m)/2$, where I_m is the identity. Shared projections of the one-hot row addresses give score $\log(m+1)$ on a matching row and zero otherwise, producing A_m. A second head with zero scores gives U_m. Factor the value/output map P_Z through $\operatorname{rank} P_Z$ coordinates. Give the heads output coefficients $-2t, t$ and retain the residual. Since $2A_m - U_m = I_m$,

$$\mathsf{H}(t) = \mathsf{H} - 2tA_m\mathsf{H}P_Z + tU_m\mathsf{H}P_Z = \mathsf{H} - tZ = G + (1-t)Z.$$

The two output coefficients alone vary along this shared-parameter path. Role-preserving reads from Theorem H.2.6 extend each bank head to a larger source set with constant additional heads and controlled error, while preserving the other rows.

2. Compute the endpoint improvement.

The affine reader gives logits $a + (1-t)\delta$, where a is determined by $\mathcal{I}$ and $\mathbb{E}[\delta \mid \mathcal{I}] = 0$. Put $\mathcal{L}(z) = \log \sum_y e^{z_y}$. Then

$$\ell(0) - \ell(1) = \mathbb{E}[\mathcal{L}(a+\delta) - \mathcal{L}(a) - S^\top \delta].$$

Both S and $q_1 = \operatorname{softmax}(a)$ are determined by $\mathcal{I}$, so their inner products with δ have zero expectation. Replacing S by q_1 gives exactly $\mathbb{E} D_{\mathrm{KL}}(q_1 \| q_0)$.

3. Compute the descent direction.

Differentiating at zero yields

$$\ell'(0) = -\mathbb{E}[(q_0 - S)^\top \delta] = -\mathbb{E}[(q_0 - q_1)^\top \delta].$$

The last inner product equals $D_{\mathrm{KL}}(q_1 \| q_0) + D_{\mathrm{KL}}(q_0 \| q_1)$ because the log-probability difference is δ minus a common normalizing constant. Nonnegativity and the equality condition of divergence prove strictness. With maximum endpoint logit error τ, the implemented loss gap differs by at most 4τ, so any positive ideal gap survives sufficiently fine precision. □

H.4.4 FINITE BIT CAPACITY BOUNDS EVERY EXACT REPRESENTATION

Proposition H.4.6 (Distinct answer patterns require distinct caches)**.**

- There are N_{task} different required answer patterns; two patterns differ on at least one common permitted query.
- All context-dependent reader information is stored in M key/value rows. Each key and each value has D coordinates of b_{kv} bits.

Exact preservation of all answers requires

$$M \geq \left\lceil \frac{\log_2 N_{\text{task}}}{2Db_{\text{kv}}} \right\rceil .$$

If each key/value pair is generated from one D-coordinate state with p bits per coordinate, replace $2Db_{\text{kv}}$ by $\min\{2Db_{\text{kv}}, Dp\}$.

Proof: Count distinguishable complete caches.

M rows have at most $2^{2Db_{\text{kv}}M}$ configurations. Fewer than N_{task} configurations merge two answer patterns, and the same reader cannot give both required answers to their distinguishing query. A pair generated from a Dp-bit state has at most 2^{Dp} possibilities, giving the stronger count. □

A concrete encoding and decoder give a corresponding upper bound. If an encoding assigns c_ℓ bits to layer ℓ, including needed copies, and each row offers d_{data} writable coordinates of b_{kv} bits, its data rows number

$$\sum_{\ell=1}^{L} \left\lceil \frac{c_\ell}{b_{\text{kv}} d_{\text{data}}} \right\rceil .$$

Count retained address and reference rows in addition. The encoder and decoder must fit the width, precision, and available layers; a value written at layer ℓ is first readable at $\ell + 1$. At equal width and precision, a completed K-token chunk uses LK full-attention cache rows, so a verified construction with M total retained rows has storage ratio $M/(LK)$. The next section supplies concrete layer allocations.

H.4.5 APPROXIMATE EQUIVALENCE PERMITS CONTROLLED LOSS

Lemma H.4.7 (Merging nearby predictive distributions has bounded cost)**.**

- Y has $V \geq 2$ possible values and $Z = \psi(W)$ merges cache values.
- Distributions $p(Y \mid W = w)$ merged together have pairwise total variation at most $0 \leq \epsilon \leq 1 - 1/V$, where $\text{TV}(p, q) = \frac{1}{2}\sum_y |p_y - q_y|$.

The increase in the best reader's log loss satisfies

$$0 \leq H(Y \mid Z) - H(Y \mid W) \leq h_2(\epsilon) + \epsilon \log(V - 1),$$

where $h_2(u) = -u \log u - (1 - u) \log(1 - u)$ and $0 \log 0 = 0$.

Proof: Compare each member with its group's mixture.

Each member is within ϵ of its group's mixture, by convexity of total variation. Distributions at distance δ can be coupled to disagree with probability δ: first match their common mass. The disagreement indicator costs at most $h_2(\delta)$ entropy; given disagreement and one outcome, the other has at most $V - 1$ possibilities. Their entropy difference in either direction is thus at most $h_2(\delta) + \delta \log(V - 1)$. This expression increases for $0 \leq \delta \leq 1 - 1/V$. Average the bound over group members. Nonnegativity follows because merging cannot reduce conditional entropy. □

Proposition H.4.8 (A KL approximation budget bounds grouped-cache loss)**.**
For a grouping $W = f(X)$, let q_w be a positive reference distribution for each used cache value. If $\mathbb{E}D_{\mathrm{KL}}(S\|q_W) \le \epsilon$, then the best reader of W has storage loss at most ϵ. In particular, this holds when every assigned state satisfies $D_{\mathrm{KL}}(S(x)\|q_{f(x)}) \le \epsilon$.

Proof: Compare the reference reader with the best grouped prediction.

The loss decomposition gives

$$\mathbb{E}D_{\mathrm{KL}}(S\|q_W) = \mathcal{D}(W) + \mathbb{E}D_{\mathrm{KL}}\big(p(Y \mid W)\|q_W\big).$$

The second term is nonnegative, proving the result. A reader whose excess above $H(Y \mid W)$ is ξ has total loss at most $H(Y \mid X) + \epsilon + \xi$. □

An actual reader's excess above $H(Y \mid Z)$ adds separately, giving the total loss of the implemented abstraction.

H.5 More memories per layer shorten the composition schedule

A smaller bank can prepare the same composition by following longer chain portions sequentially. It then combines their endpoints by pointer doubling, trading memories per layer for preparation time.

Proposition H.5.1 (A smaller bank retains the composition gain)**.**
A chain of $H \ge 2$ selected associations with K available source records can be computed using m writable rows per layer and

$$L = O(K/m + \log H), \qquad M = O(K + m \log H) \tag{H.5.1}$$

total retained rows. The construction also archives every source record for subsequent indexed recall. In particular, $m = \Theta(K/\log H)$ gives $L = O(\log H)$ and $M = O(K)$.

- Incoming domains have arbitrary positive sizes $v_1, \dots, v_H$ with $\sum_j v_j \le K$; for $j < H$, outputs lie in the next domain.
- Sources and selectors remain readable throughout writing, with the addressed shared-attention interface of Section H.1 and $O(\log N)$-bit fields.
- $m - c \ge 2v_1$, where c is constant auxiliary storage. Data rows have separate working and archive fields.
- An archived-record address arrives before its assigned read layer; any subsequent computation fits before prediction.

For Section H.1, the incoming sizes are $q_0, q_0, q_1, \dots, q_{H-2}$. The capacity condition counts every supported starting input, including those not yet queried.

Proof: prepare short portions, compose, and archive.

Allocate starting tables.

Put $m' = m - c$ and $b = \lceil 2K/m' \rceil$. Keep stage 1 as a starting point. Partition stages $2, \dots, H$ into windows of b stages and choose a smallest-domain stage from each full window as another start. Each chosen domain is at most its window's total divided by b, so

$$\text{starting rows} \le v_1 + K/b \le m'. \tag{H.5.2}$$

Consecutive starts, including the terminal end, are at most $2b$ stages apart. Allocate one row per possible input at each start. Each row follows its starting input through that portion, so all portions can be prepared in $O(1 + K/m)$ layers using the allocated m rows throughout.

Synchronize with shared projections.

Let $T \leq 2b$ be the longest portion. A working row retains its portion's last stage e; a raw key retains its stage j. During building, raw and latent keys share the stage–identifier tag. After the address and selector tests of Section H.1, give a matching raw key score $2(t + e - T - j) + 1$ at layer t and a matching latent self-key score zero. A row waits at its initial state until its first stage is due, then advances one raw step per layer and finishes at T.

The query carries $t + e - T$, and the key carries j, so linear projections supply these scores. The self-subtraction head uses zero clock, giving raw competitors negative scores. Scale mismatch penalties above $4H + 4T$ to exclude other stages. The clock and endpoint occupy constant coordinates; shared projections and zero token-query fields preserve the cancellation of Lemma H.1.2. Doubling restores distinct raw and latent source tags.

Double completed portions.

Switch target reads to latent rows and apply Lemma H.1.2. One self-pointing reference with zero answer serves every terminal value. The final elementary association writes the answer and a pointer to this reference; other associations write zero answer. During doubling use

$$\mathrm{ptr}' = \mathrm{ptr}_{\mathrm{target}}, \qquad y' = y + y_{\mathrm{target}}. \tag{H.5.3}$$

A target/self pair replaces ptr; a target/reference pair adds the target answer to y. These four heads cancel on tokens. The reference has zero active query fields, a fixed self-pointer value and zero answer, so paired updates cancel there too. Each chain encounters its nonzero answer once, then reaches the reference. After $O(\log H)$ rounds, every first-portion row holds the result.

At the next layer, a self/reference pair copies each completed summary into a zero-initialized protected field. Residual persistence carries this field into later banks. For independently initialized banks, the same pair copies the preceding summary into a zero field, using one common key projection and the fixed row address. The summary is therefore available for later joins with constant extra coordinates, heads and preparation layers. Each additional carried bank uses the same m rows.

Archive alongside computation.

Two additional heads replace archive fields with successive batches of original records. Choose a power-of-two batch size between $m'/2$ and m': a source address is then the concatenation of layer-fixed bank bits and fixed slot bits, selected by linear projections. A zero reference outranks mismatches, so padded slots receive zero. Earlier banks remain retained while later working states change. Archive-only stages, if needed, disable the working update. This takes $O(1 + K/m)$ stages, and retaining m rows per stage gives (H.5.1). A read after the final write is counted. The read-only sources remain part of peak working storage until writing ends.

Budget numerical error.

Replacement errors grow linearly during preparation; the $O(\log H)$ answer-addition rounds amplify error by at most $O(H)$. For $T_{\mathrm{stage}} = O(K/m + \log H)$ stages and per-operation error δ, the final error is $O(HT_{\mathrm{stage}}\delta)$, including clock and address-coordinate factors in δ. Since $H \leq K \leq N$, all scores and operation counts are polynomial in N. Inverse-polynomial tolerances therefore preserve the address margins and exact decoding with $O(\log N)$ precision. Addresses, pointers, selectors, answers, archive values and protected summaries together use $O(\log N)$ coordinates. At most K source records remain available during this schedule. □

Applying the schedule within chunks. Local computation prepares starting-input tables from each chunk and its accessible earlier summaries. Unfinished boundary compositions contribute additional layers, as quantified in Section H.3.

Corollary H.5.2 (Available local portions admit the same trade-off)**.**
For locally available portions of chains of length at most H, let U_{rec} bound the sum of processed stage-domain sizes, counting repeated uses. Their summaries use $O(U_{\mathrm{rec}}/m + \log H)$ layers and $O(U_{\mathrm{rec}} + m \log H)$ retained rows. Under Section H.3's scheduling and distance-count premises,

$$\mathbb{E}L = O\big(U_{\mathrm{rec}}/m + \log(H+1) + AK^{-\alpha}\big).$$

- The initial domains total E entries, and $m - c \geq 2E$.
- Each stage has a fixed index and available addressed sources; U_{rec} counts every stage-use description.
- Source access, read deadlines and numerical interfaces satisfy Proposition H.5.1; the addressed computation is polynomial in N.

Proof: apply the same schedule to all local portions.

Place enough starting rows.

Set $b = \lceil 2U_{\mathrm{rec}}/(m-c) \rceil$ and apply the preceding window-minimum construction to each chain. Together with the initial domains, the chosen starts need at most $E + U_{\mathrm{rec}}/b \leq m - c$ rows. Every resulting portion has at most $2b$ stages.

Prepare and carry every summary.

The endpoint clock synchronizes all portions. Raw records retain their source-stage addresses; separate computation tags distinguish overlapping calculations' latent rows. Raw keys ignore those extra tags, so repeated reads share source keys. Pointer doubling and protected summary fields complete the local construction. Section H.3's schedule then adds the unfinished boundary compositions, whose expected cost is $O(AK^{-\alpha})$. Carrying the summaries through t additional join layers retains at most mt additional rows, counted in those layers' banks. □

Archiving every record of a physical K-record chunk adds $O(K/m)$ layers and $O(K)$ rows if some records were unused. Thus $U_{\mathrm{rec}} = O(K)$ recovers (H.5.1) while preserving the whole chunk. A realized schedule with B additional joins retains

$$M_{\mathrm{total}} = O\big(K + U_{\mathrm{rec}} + m \log H + mB\big).$$

Section H.3's uniform-A bound gives $\mathbb{E}[mB] \leq mAK^{-\alpha}/(1-\alpha) = O(m)$, absorbed by the $m \log H$ term. Thus $\mathbb{E}M_{\mathrm{total}} = O(K + m \log H)$ when $U_{\mathrm{rec}} = O(K)$. Taking $m = \Theta(K/\log H)$ gives $\mathbb{E}L = O(\log(H+1))$ and $\mathbb{E}M_{\mathrm{total}} = O(K)$, including the carried summary banks. A fixed allocation of b join layers retains its corresponding mb rows.

H.6 Exact recall distributes values across layer memories

Different layers can archive different values. A reader collects the requested value at its assigned layer, giving fewer rows per layer while preserving all information.

Proposition H.6.1 (Layer-distributed memories permit exact indexed recall)**.**
For K tokens from an alphabet of size V, $L \geq 2$ attention layers suffice for exact indexed recall if

$$\sum_{\ell=1}^{L-1} r_\ell \geq K, \tag{H.6.1}$$

where r_ℓ counts token-storage rows written at layer ℓ. Uniform $r_\ell = \lceil K/(L-1) \rceil$ gives $O(K/L)$ rows per layer and $O(K)$ retained rows for $2 \leq L \leq K$, including constant auxiliary rows. Repeated indexed requests recover the sequence.

- Tokens and fixed slot identifiers remain accessible until their assigned writes. Shared copying heads have protected address and value fields.
- The requested address is available before its assigned read; the reader preserves it and an accumulator. A write at ℓ is first read at $\ell + 1$.
- Each read admits a fixed zero reference, and numerical transformations preserve address margins.
- Width $O(K + \log N + \log V)$, sufficient finite precision, and adjustable finite output-logit scale are available.

Proof: write each token once and suppress absent-layer reads.

Assign and copy.

Assign each token to one slot among layers $1, \ldots, L-1$. Encode tokens as distinct sign vectors of length $\lceil \log_2 V \rceil$. Fixed one-hot slot fields let layer-dependent linear projections implement any assignment within the stated width. A source/self head pair writes each token code and cancels on tokens, as in Lemma H.1.2. Choose writing error within the reader's error budget.

Accumulate one matching value.

For sign-coded addresses of length b_{addr}, use score βb_{addr} for a match, at most $\beta(b_{\mathrm{addr}} - 2)$ for a mismatch, and $\beta(b_{\mathrm{addr}} - 1)$ for the zero reference. The target wins in its assigned bank; the reference wins elsewhere. If N_c counts candidates across all reads, T_{read} counts reads and ν includes numerical and writing error per read, then

$$\|\widehat{c} - c(u)\|_\infty \leq 2N_c e^{-\beta} + T_{\mathrm{read}}\nu \leq \eta < 1. \tag{H.6.2}$$

Logarithmic score margins and sufficiently fine precision enforce this bound. Token logits $\lambda c(z)^\top \widehat{c}$ give the correct token a gap at least $2\lambda(1-\eta)$. Thus discrete recall is exact; increasing permitted scale and precision makes log loss arbitrarily small.

Count every retained row.

With a_ℓ references and e_ℓ other auxiliaries, physical storage is $M = \sum_{\ell=1}^{L}(r_\ell + a_\ell + e_\ell)$. The sum includes final-layer writes and all auxiliary states. Uniform allocation and constant auxiliaries give $M = O(K + L) = O(K)$. □

Recall deadlines and information capacity. If recovery must precede T_{after} further attention layers, a sufficient allocation is

$$\sum_{\ell=1}^{L-T_{\mathrm{after}}-1} r_\ell \geq K, \qquad L - T_{\mathrm{after}} - 1 \geq 1.$$

This allocation completes retrieval before the final T_{after} layers reserved for dependent computation.

All V^K full-support sequences must remain distinguishable to an interface answering every index query. If the complete context-dependent interface consists of M key–value rows, each with d coordinates per key and value at p bits per coordinate, then $2dpM \geq K \log_2 V$. This quantifies the information retained by exact recall alongside its reduction in rows per layer.

H.7 Attention paths determine when a stored association can be used

The composition schedules require each destination memory to read its next source. Two PAT design choices provide these paths: bidirectional attention allows reads against memory-slot order, and direct attention to earlier memories avoids a transfer through token states. Direct access also lets a writer select from an archive after the requested address arrives. We first establish these benefits, then give a construction in which more source memories complete the reading phase in fewer layers.

Proposition H.7.1 (Slot order and transfer depth)**.**

- Slot indices stay fixed. Residual and positionwise operations preserve them, and token states and earlier chunks provide no return path from the current chunk's memories. Only the stated attention edges transfer information between slots.
- Attention updates are simultaneous: a value written in one layer first becomes a source in the next. Addressed reads use the shared-parameter construction of Lemma H.1.2.
- For the transfer comparison, the selected address is known at the layer's input and its value still requires retrieval.

Let i, j denote indices in the same chunk's memory bank. Under a causal memory mask, information in slot j cannot reach slot $i < j$. Bidirectional memory attention permits that read as soon as its source is available. Further, a selected earlier value can reach a memory writer in one attention update by direct access. A route that first retrieves the value into a token state and then transfers it to the writer needs two updates.

Proof: trace the permitted reads.

Slot order.

A causal memory edge from source j to destination i requires $j \le i$. Every path therefore has nondecreasing slot indices; the other permitted operations do not reverse this order. Bidirectional attention includes every source–destination pair, so it removes this obstruction.

Transfer depth.

Suppose the selected address is known at a layer's input and its value still requires retrieval. A direct read writes the value at that layer's output. On the token route, the first update writes the retrieved value into a token; the writer cannot read that updated token until the next layer. Thus the direct edge removes one transfer layer from this route. □

With m memories in the bank, bidirectional memory attention evaluates m^2 pairs, compared with $m(m+1)/2$ for the causal mask. Its backward edges therefore have a compute cost, included in Section I. The recall construction in Section H.6 separately counts storage across all layer banks and requires each selected address before its bank's read deadline.

Direct access preserves choices until the query arrives. Suppose an archive stores r independent fair bits $A = (A_1, \dots, A_r)$ in addressed records. A shared copying layer prepares these records from observed tokens; the archive becomes readable in the next layer, and this preparation counts toward writer work. After an intermediate state W has been formed, an index $J \in \{1, \dots, r\}$, uniform and independent of (A, W), requests $Y = A_J$. Both routes use this same archive. A direct writer can read it after receiving J; an indirect writer must answer from (W, J) by its deadline. Here W includes every archive-dependent state still accessible to the indirect writer, so its capacity counts all token, memory, and residual routes.

Proposition H.7.2 (An addressed archive avoids preselection loss)**.**
If W has at most 2^c possible states, with $1 \le c < r$, every indirect predictor has expected log loss at least

$$\ell_{\text{indirect}} \ge (1 - c/r) \log 2. \tag{H.7.1}$$

A direct addressed read has exact discrete recovery and arbitrarily small log loss with finite score scale and precision. The source archive contains the same r data records in both cases; c measures the intermediate interface, not the archive.

Proof: count retained bits, then select the requested record.

Bound the indirect reader.

Use bit units for entropy in this step. Since $H(W) \le c$, $H(A \mid W) \ge r - c$. Conditional subadditivity and the independence of J give

$$H(A_J \mid W, J) = \frac{1}{r} \sum_{j=1}^{r} H(A_j \mid W) \ge \frac{1}{r} H(A \mid W) \ge 1 - c/r.$$

Expected log loss is at least conditional entropy; multiplying by $\log 2$ converts the bound to nats.

Construct the direct read.

Give each record a distinct sign address of length $\lceil \log_2 r \rceil$. Shared projections compare the query address with these keys and read the signed value $2A_j - 1$. A matching dot product exceeds every mismatch by at least two, so finite scaling makes the target's attention mass greater than $3/4$. Its output then has the correct sign with magnitude greater than $1/2$. Sufficient finite precision preserves that margin, and finite output logit scaling makes the loss arbitrarily small. This is the addressed-read implementation of Lemma H.1.2 with one selected value. □

Removing irrelevant sources also improves a fixed reader. Direct access preserves the requested information; restricting the source set can increase the weight it receives. The following construction turns that change into a strict loss improvement without changing attention scores.

Proposition H.7.3 (Removing zero-valued competitors improves a binary read)**.**
Let one source have value $Y \in \{-1, +1\}$ and every other source have value zero. Use the binary readout $P(Y = +1 \mid z) = 1/(1 + e^{-\beta z})$, with attention output z and fixed $\beta > 0$. If the query and retained keys stay unchanged, removing any positive attention weight from competing sources strictly lowers the negative log likelihood of either target value.

Proof: increase the useful mass.

Let $w > 0$ be the required source's exponentiated score, $u \ge 0$ the retained competitors' total weight, and $v > 0$ the removed weight. The output is $z = aY$, where removal increases a from $w/(w+u+v)$ to $w/(w+u)$. Its loss $\log(1 + e^{-\beta a})$ therefore strictly decreases.

An explicit shared-projection witness sets the query and key projection matrices to zero and lets the value projection copy the signed source value. All scores are then zero. Keeping k of n sources, including the required one, with $1 \le k < n$, reduces loss from $\log(1 + e^{-\beta/n})$ to $\log(1 + e^{-\beta/k})$. The retained source already contains the target, so this construction improves reader loss while preserving all target information. □

More source memories can complete a reading phase sooner. A memory count also limits the span of values an attention reader can assemble before nonlinear composition. We compare the following reading interface, counting every source bank available by the read deadline.

Proposition H.7.4 (Source span bounds linear reading accuracy)**.**

- The output coordinates start at the same value for every query and accumulate attention outputs linearly for s reading layers.
- Layer ℓ has h heads and v_ℓ source vectors fixed across queries. All contributing sources are counted, and value and output projections are shared across queries.

Let P contain the desired output vector for each query as a row, let X_s contain the exact-arithmetic output rows, and let $\sigma_j(P)$ be the singular values of P in decreasing order. Then

$$\operatorname{rank}(X_s) \le R_s := 1 + h\sum_{\ell=1}^{s} v_\ell, \qquad \|P - X_s\|_F^2 \ge \sum_{j>R_s} \sigma_j(P)^2, \tag{H.7.2}$$

where $\|\cdot\|_F^2$ sums the squared entries.

Proof: count source spans.

For head a in layer ℓ, let $V_{\ell a}$ contain its projected source vectors as rows, and let $\Pi_{\ell a}$ contain the attention weights for all queries. With b the common initial output, including fixed additive biases,

$$X_s = \mathbf{1}b^\top + \sum_{\ell=1}^{s}\sum_{a=1}^{h} \Pi_{\ell a} V_{\ell a},$$

where $\mathbf{1}$ is the all-ones column. Each summand has rank at most v_ℓ, so rank subadditivity proves the first bound. The best rank-R_s approximation to P retains its largest R_s singular components; their omitted squared norm gives the second bound. □

If numerical implementation perturbs X_s by Frobenius norm at most ζ, the reverse triangle inequality gives squared error at least

$$\left[\sqrt{\sum_{j>R_s} \sigma_j(P)^2} - \zeta\right]_+^2, \qquad [x]_+ = \max\{x, 0\}.$$

For q mutually orthogonal unit target vectors and v sources per layer, the ideal bound gives squared error at least $[q - 1 - hsv]_+$.

A shared-attention construction attains the larger source span. Take ph orthogonal unit targets, with p targets in each of the h head output subspaces. Store them in p data rows: row i contains target i for every head. Add one zero-valued reference row. A query for one target makes its head select the corresponding data row and every other head select the reference. Thus all $p + 1$ sources, including the reference, are counted. A preceding shared copying layer prepares this bank from the source records; the one-layer reading comparison begins when that bank becomes available.

For each head, reserve $p + 1$ query-address coordinates, $p + 1$ source-address coordinates, and p value coordinates. One-hot addresses and shared linear projections give score γ to the requested row and zero to every other row. Its attention mass is $1/(1 + pe^{-\gamma})$. Choose finite γ so this mass is at least $1 - \epsilon$, where $\epsilon > 0$ is a chosen tolerance. The active head's error norm is at most 2ϵ, and each other head's error is at most ϵ. Orthogonality gives total squared error

$$E_{\text{wide}} \le ph(h + 3)\epsilon^2 \tag{H.7.3}$$

over all ph target queries, where E_{wide} denotes the construction's squared Frobenius error.

For $p = 8$, nine source memories and 26 coordinates per head therefore attain error at most $8h(h+3)\epsilon^2$ in one layer. Any five-source, one-layer reader under Proposition H.7.4 has rank at most $5h + 1$, and hence error at least $3h - 1$ on these same $8h$ targets. Choosing $8h(h + 3)\epsilon^2 < 3h - 1$ gives a strict gap, preserved at sufficiently small numerical error by the perturbation bound above. Nine sources can thus meet an accuracy deadline that requires additional reading layers with five, leaving more of the remaining stack for composition.

Shared chunk positions preserve existing rotary geometry during expansion. Assign every cached key from chunk c the same position p_c. For a query with content vector q and position p_q, a key with content vector k_i has rotary score $\beta q^\top R(p_q - p_c)k_i$, where $\beta > 0$ is the score scale and $R(t)$ is the rotary matrix at relative offset t. Changing the number of slots leaves p_c unchanged. Hence, for each retained query position, every existing query–key rotary factor remains exactly the same; retaining their content vectors also preserves their scores. Middle-layer expansion can therefore add source vectors without shifting the positions of the existing context.

I Analytical compute estimate of PAT

We estimate training arithmetic for PAT and full attention (FA) on the same tokens and model dimensions. The calculation has three parts: count projected rows and attention pairs; compare model-scale configurations; then account for document boundaries. All counts are model FLOPs per consumed token.

I.1 Count the work performed on each token

Counting convention. A multiply-add costs two FLOPs. Forward plus backward matrix work costs three times the forward pass. We include attention projections, score and value products, lexical SwiGLU matrices, and the output projection. Scalar operations, optimizer work, recomputation, and communication are outside this primary count.

Comparison schedule. Both models use full-width key/value projections and have L layers, width D, SwiGLU hidden width f, and vocabulary size V. PAT uses a causal lexical window of K tokens. Each memory query reads its current chunk and memory bank. From layer two, queries also read earlier chunks' memories from the preceding layer. Memory rows have no feedforward network. Each m_ℓ counts all physical memory rows, including auxiliaries.

Step 1: Count rows across layers. For complete chunks, with N/K integral, define

$$M_1 = \sum_{\ell=1}^{L} m_\ell, \qquad M_2 = \sum_{\ell=1}^{L} m_\ell^2,$$
$$G_1 = \sum_{\ell=2}^{L} m_{\ell-1}, \qquad G_2 = \sum_{\ell=2}^{L} m_\ell m_{\ell-1}. \tag{I.1.1}$$

M_1, M_2 count current rows and memory pairs. G_1, G_2 count preceding-layer rows and pairs. Let δ sum $m_{\ell-1}$ over allocation changes $m_\ell \neq m_{\ell-1}$; those incoming rows need separate key/value projections.

Step 2: Count attention pairs. Let $a_K = K - K(K-1)/(2N)$ be local lexical pairs per token. Averaged over chunks, the number of earlier chunks is $(N/K - 1)/2$.

Attention operation	Pairs per token, summed over layers
Lexical window	La_K
Memory queries reading current tokens	M_1
Memory queries reading current memories	M_2/K
Queries reading earlier memories	$(G_1 + G_2/K)(N/K - 1)/2$

Their sum is

$$q = La_K + M_1 + \frac{M_2}{K} + \left(G_1 + \frac{G_2}{K}\right)\frac{N/K - 1}{2}. \tag{I.1.2}$$

Step 3: Convert rows and pairs to FLOPs. The shared lexical and output cost is $F_0 = 3L(8D^2 + 6Df) + 6DV$. Each attention pair costs $12D$ training FLOPs. Hence the primary cost per consumed token is

$$F_{\mathrm{PAT}} = F_0 + \frac{3(8M_1 + 4\delta)D^2}{K} + 12Dq,$$
$$F_{\mathrm{FA}} = F_0 + 6DL(N+1). \tag{I.1.3}$$

Multiply by N for sequence cost. This form keeps PAT's added memory projections visible alongside its saved context attention.

I.2 Compare production-sized model configurations

PAT's m–$4m$–m schedule uses m rows in the first and last quarters of the layers and $4m$ in the middle half. For L divisible by four,

$$M_1 = \tfrac{5}{2}Lm, \qquad M_2 = \tfrac{17}{2}Lm^2, \qquad \delta = 5m,$$
$$G_1 = (\tfrac{5}{2}L - 1)m, \qquad G_2 = (\tfrac{17}{2}L - 10)m^2. \tag{I.2.1}$$

Projection and lexical-read costs grow linearly with m; memory-to-memory costs grow quadratically. The two slot-transition matrix products add $48m^2D/K$ FLOPs per token when included.

Model-scale comparison. Table 18 evaluates (I.1.3) at $K = 16$, 2–8–2 memories, $V = 50{,}304$, and $f = 8\,\mathrm{round}(D/3)$. Each column is one full document. Percentages are rounded to whole points; the accompanying calculator retains full precision.

Table 18: PAT primary training FLOPs as a percentage of FA. Below 100% means fewer FLOPs. The rows compare matched depth and width, using the same lexical architecture on both sides.

Configuration	$L \times D$	16K tokens	64K tokens	128K tokens
Qwen2.5-7B shape	28×3584	93%	71%	61%
Original reference	32×4096	94%	73%	62%
Qwen2.5-72B shape	80×8192	101%	84%	73%

The Qwen rows use the official depth and width (Qwen Team, 2024) within the common lexical architecture specified above: the same f, V and full-width key/value projections. Here 1K means 1,024 tokens.

At 128K tokens the original reference and the 72B shape save about 38% and 27% of primary FLOPs. The wider model needs a longer context to offset memory projections.

How much memory fits the same budget? At $L = 32$, $D = 4096$, and $N = 128$K, the 3–12–3 schedule costs less than FA; 4–16–4 costs more. This gives an actionable allocation comparison without reporting a fractional number of memory rows.

Stored rows. For J retained full chunks, PAT's memory keys and values occupy $2DJM_1$ scalar entries. Its live lexical window adds at most $2DLK$. FA retains $2DLN$ entries. Temporary activations and any additionally cached projections are separate.

I.3 Read the context-length and width dependence

For uniform $m = \rho K$, $0 < \rho \le 1$, and $L \ge 2$, substitution into the pair count gives the attention-and-projection ratio

$$\frac{C_{\mathrm{PAT}}}{C_{\mathrm{FA}}} = \Theta\left(\frac{(1+\rho)D + \rho N + K}{D+N}\right) = \Theta\left(\rho + \frac{D+K}{N}\right), \quad N \ge D,\ N \ge K. \tag{I.3.1}$$

The three terms cover token and memory projections, compressed context attention, and local attention. The exact calculation below identifies the saving regime.

Exact ratio for any fixed layer allocation. The count also covers varying banks such as 2–8–2. Define the normalized common work $a_D = 4D + 3f(D) + V/L$ and three allocation coefficients:

$$\begin{aligned} \beta &= \frac{G_1 + G_2/K}{LK}, && \text{limiting attention fraction,} \\ \gamma_{\mathrm{mem}} &= \frac{4M_1 + 2\delta}{LK}, && \text{memory-projection overhead,} \\ c_{\mathrm{att}} &= 2K + \frac{2M_1}{L} + \frac{2M_2}{LK} - \beta K, && \text{finite-chunk attention term.} \end{aligned}$$

Writing $\chi = K(K-1)$, substitution into (I.1.3) gives

$$R = \frac{F_{\mathrm{PAT}}}{F_{\mathrm{FA}}} = \frac{a_D + \gamma_{\mathrm{mem}} D + \beta N + c_{\mathrm{att}} - \chi/N}{a_D + N + 1}. \tag{I.3.2}$$

PAT uses no more FLOPs than FA precisely when

$$(1-\beta)N \geq \gamma_{\mathrm{mem}} D + c_{\mathrm{att}} - 1 - \frac{\chi}{N}. \tag{I.3.3}$$

For uniform $m = \rho K$, the coefficients reduce to $\beta = (1 - 1/L)\rho(1+\rho)$ and $\gamma_{\mathrm{mem}} = 4\rho$.

Context trend: the ratio falls throughout the saving regime. Fix the model and a nonzero allocation, with $\beta < 1$. Put $Q = a_D + 1$ and $T = (1-\beta)a_D + \gamma_{\mathrm{mem}} D + c_{\mathrm{att}} - \beta > 0$. Differentiating gives

$$\frac{\partial R}{\partial N} = \frac{-T + 2\chi/N + \chi Q/N^2}{(N+Q)^2}.$$

The numerator decreases with N when $K > 1$ and is negative when $K = 1$. Thus R either decreases throughout or first reaches one maximum. At $N = K$,

$$R(K) - 1 = \frac{\gamma_{\mathrm{mem}} D + 2M_1/L + 2M_2/(LK)}{a_D + K + 1} > 0.$$

The saving region $R \leq 1$ therefore lies on the decreasing branch. As N grows, $R \to \beta$. This applies to every fixed layer allocation with $\beta < 1$.

Width trend: larger projections raise the relative cost. Fix the context, layer count, vocabulary, and memory allocation. Let $D_2 > D_1$ and let the feedforward width $f(D)$ be nondecreasing, including a rounded width rule. Write $R_i = R(D_i)$ and $a_i = 4D_i + 3f(D_i) + V/L$. Subtracting the two ratios gives

$$R_2 - R_1 = \frac{(1 - R_1)(a_2 - a_1) + \gamma_{\mathrm{mem}}(D_2 - D_1)}{a_2 + N + 1} > 0 \qquad (R_1 \leq 1).$$

Both terms are nonnegative, and nonzero memory makes the second positive. Thus the width trend holds for the aligned SwiGLU widths used in Table 18.

Substituting the physical allocation from Section H.5 into (I.1.3) identifies the contexts where that composition construction also fits FA's compute budget.

I.4 Separate document-boundary savings from architectural savings

Document isolation removes cross-document pairs even in FA. The PAT comparison must therefore use the same document pieces and consumed-token denominator.

Count partial chunks. Let T be total consumed tokens, J_i the chunks in document piece i, t_{ik} the tokens in chunk k, and r_{ik} its retained earlier chunks. Let N_i be the piece length. Four token-weighted statistics suffice:

$$\begin{aligned} j &= \frac{\sum_i J_i}{T}, & a &= \frac{\sum_i \sum_{t=1}^{N_i} \min(t, K)}{T}, \\ e &= \frac{\sum_{i,k} t_{ik} r_{ik}}{T}, & u &= \frac{\sum_{i,k} r_{ik}}{T}. \end{aligned} \tag{I.4.1}$$

They count chunks, local lexical pairs, earlier-memory reads by tokens, and earlier-memory reads by memories, respectively.

With the same memory allocation on full and partial chunks, the cost becomes

$$q_{\text{pieces}} = La + M_1 + M_2 j + G_1 e + G_2 u,$$
$$F_{\text{PAT,pieces}} = F_0 + 3(8M_1 + 4\delta)D^2 j + 12Dq_{\text{pieces}}. \tag{I.4.2}$$

For FA with window $N_{\max}$, define $q_{\text{FA}} = T^{-1} \sum_i \sum_{t=1}^{N_i} \min(t, N_{\max})$; its cost is $F_0 + 12DLq_{\text{FA}}$.

Interpret the comparison. The original appendix's 100-billion-token stream gives approximately $(j, a, e, u) = (0.063, 15.95, 280.47, 17.55)$ for $K = 16$ and $N_{\max} = 16\text{K}$. Its document boundaries remove about 45% of FA attention pairs relative to fixed full blocks. This is the data-boundary saving in FA attention pairs. The accompanying calculator uses the full supplied statistics.

The ratio PAT/piece-isolated FA measures the architectural saving. Multiplying it by piece-isolated FA/block FA gives the combined ratio. At a fixed compute budget, the reciprocal FLOP ratio gives the corresponding token-budget ratio when these token-weighted costs remain stable.

Fixed windows saturate. For one long document with fixed $N_{\max}$, $q_{\text{FA}} \to N_{\max}$ and the earlier-memory counts also saturate. Substitution into (I.4.2) then gives a constant limiting FLOP ratio.