

000 001 CACHE WHAT LASTS: TOKEN RETENTION FOR 002 MEMORY-BOUNDED KV CACHE IN LLMs 003 004

005 **Anonymous authors**
006 Paper under double-blind review

007 008 ABSTRACT 009

011 Memory and computation remain core bottlenecks in long-horizon LLM inference
012 due to the quadratic cost of self-attention and the ever-growing key-value (KV)
013 cache. Existing strategies for memory-bounded inference, such as quantization,
014 offloading, or heuristic KV eviction, either incur high orchestration costs or rely
015 on unreliable attention-based proxies of importance. We propose TRIM-KV, a
016 novel approach that learns each token’s intrinsic importance at creation time via a
017 lightweight retention gate. Each gate predicts a scalar retention score that decays
018 over time, reflecting the long-term utility of the token for a specific layer and head.
019 Tokens with low scores are evicted when the memory budget is exceeded, ensuring
020 that the cache always contains the most critical tokens. TRIM-KV is trained
021 efficiently through distillation from a frozen LLM combined with a capacity loss,
022 requiring only gate fine-tuning and adding negligible inference overhead. Across
023 mathematical reasoning (GSM8K, MATH-500, AIME24), procedural generation
024 (LongProc), and conversational long-memory benchmarks (LongMemEval), TRIM-
025 KV consistently outperforms strong eviction and learnable retrieval baselines,
026 especially in low-memory regimes. Remarkably, it even surpasses full-cache
027 models in some settings, showing that selective retention can serve as a form of
028 regularization, suppressing noise from uninformative tokens. Qualitative analyses
029 further reveal that learned retention scores align with human intuition, naturally
030 recovering heuristics such as sink tokens, sliding windows, and gist compression
031 without explicit design. Beyond efficiency, retention scores provide insights into
032 layer- and head-specific roles, suggesting a new path toward LLM interpretability.

033 1 INTRODUCTION 034

035 Modern large language models (LLMs) can, in principle, handle extremely long input contexts –
036 some recent models support context windows of 128k tokens or more (Yang et al., 2025; Gao et al.,
037 2024). Yet, extending context length comes with steep computational costs. The self-attention
038 mechanism has quadratic time complexity in sequence length, and storing the key-value (KV) cache
039 for thousands of tokens can quickly exhaust GPU memory (Wang et al., 2025; Li et al., 2024a). In
040 practical deployments, the KV cache, which saves past key and value vectors to avoid re-computation,
041 becomes a major memory and latency bottleneck for long-context inference. Decoupling resource
042 usage from context length is therefore critical for enabling efficient and scalable applications such as
043 long-horizon reasoning (Chen et al., 2025) and lifelong agents (Zheng et al., 2025; Li et al., 2024d).

044 To address this challenge, recent work has explored memory-bounded LLMs that can operate effec-
045 tively under constrained KV budgets (Li et al., 2024a). One line of research focuses on compression
046 and quantization, aiming to reduce memory footprint by learning compact representations of past
047 tokens rather than storing all keys and values explicitly (Hooper et al., 2024; Saxena et al., 2024).
048 These techniques are mostly effective during the prefill phase but scale poorly with generation
049 length. Another line leverages attention sparsity to offload most of the cache to CPU or secondary
050 storage, and retrieve only relevant segments on demand via similarity search (Tang et al., 2024) or
051 learned indices (Gao et al., 2025). While offloading lowers the on-GPU footprint, it incurs nontrivial
052 orchestration overhead that accumulates over long generations, undermining end-to-end throughput.

053 A more common and direct approach to enforce a fixed memory budget is KV cache eviction, which
054 directly drops certain tokens from the KV cache (Xiao et al., 2023). Many KV eviction strategies

have been proposed to decide which tokens to remove. However, most of them are attention-guided heuristics: they track attention from new queries to cached tokens and retain those that are recently or frequently attended, adapting the cache to the current focus (Zhang et al., 2023; Li et al., 2024c; Wang et al., 2025; Liu et al., 2025; Ghadia et al., 2025; Cai et al., 2025). While being efficient, these methods assume that recent attention is a reliable proxy for future importance. This assumption often breaks for long-horizon generation and reasoning tasks: a token might be crucial much later, even if it has not been attended to in the recent past (Jiang et al., 2024). Moreover, attention-based eviction can suffer from attention bias, *e.g.*, the model might temporarily overlook a needed token due to a distracting context (Shi et al., 2023), causing it to be evicted prematurely. While some recent studies have attempted to learn better eviction decisions (Chen et al., 2024; Zeng et al., 2024), these methods typically scale poorly with sequence length and are therefore limited to the prefilling stage.

In this work, we take a new perspective on the KV eviction problem. Rather than relying on the attention-guided importance, we propose to learn each token’s intrinsic importance at the time of its creation and use that as the basis for eviction. Intuitively, not all tokens are created equal: some carry significant semantic or task-related weight (*e.g.* a critical fact, a question being answered, or the first few “sink” tokens that often encode the topic or instructions), while others are relatively inconsequential (*e.g.* filler words, stopwords, or trivial arithmetic steps in a chain-of-thought). Moreover, the importance of tokens is not uniform across the network, but it varies systematically by layers and heads, reflecting their functional specializations (Voita et al., 2019; Wu et al., 2024b).

We posit that the contextual embedding of a token already encodes much of its long-term utility. We therefore introduce a retention gate that maps the token’s embedding and produces a scalar retention score $\beta \in [0, 1]$ reflecting the token’s inherent importance for a specific layer and head. Especially, we design this retention score to decay exponentially as the context grows, mimicking the gradual forgetting of old information in human brains (Ebbinghaus, 2013). Thus, a highly important token will have $\beta \approx 1$ and retain a high score for a long time, whereas a token deemed unimportant will have β closer to 0 and its influence will vanish quickly. We leverage this score to drive a simple eviction policy: whenever the number of cached tokens exceeds the budget M , we evict the token with the smallest current retention score. This approach, which we call **Token Retention for Memory-bounded KV Cache (TRIM-KV)**, ensures that at all times, the cache is filled with the M tokens judged most intrinsically important, with a preference toward more recently generated tokens.

Implementing retention-based caching in an existing LLM only requires adding a few lightweight components. We integrate the retention gates into each self-attention layer of a pretrained model to modulate attention weights by token importance during training. We then train only the gates with a two-part loss: a distillation loss that compels the modified model to mimic the original model’s outputs, thus preserving quality, and a capacity loss that penalizes exceeding the target memory budget, thus encouraging sparseness in attention via eviction. Importantly, by training the gates across all layers jointly, the model can learn a coordinated, globally optimal caching policy rather than greedy layer-wise decisions. At inference time, the learned retention gates produce per-token scores on the fly, and eviction is implemented with a simple score comparison, adding minimal overhead.

Results and Contributions. Through extensive experiments on long-context and long-generation benchmarks, we demonstrate that our learnable token retention approach substantially improves the performance of memory-bounded LLMs. On challenging mathematical reasoning datasets, GSM8K, MATH, AIME, a long procedural generation benchmark, LongProc, and a long-memory chat assistant benchmark, LongMemEval, our method consistently outperforms eviction baselines, even when those baselines use $4\times$ more KV budget, and deliver 58.4% pass@1 gain compared to the SOTA learnable KV retrieval baseline (Gao et al., 2025), especially in low-memory regimes. Remarkably, in several settings, TRIM-KV even surpasses a full-cache model, suggesting that selective retention can function as an effective regularizer by suppressing noise from uninformative tokens.

We also present qualitative evidence that learned retention scores align with human intuition: the model tends to assign high scores to initial tokens and problem descriptions, and low scores to less meaningful punctuation. Notably, many behaviors reminiscent of common heuristics, such as keeping sink tokens, sliding windows, and gist compression (Mu et al., 2023), emerge naturally and adaptively from our learned policy, without being hard-coded. Finally, we show that these learned retention scores can also act as a diagnostic tool for probing layer- and head-specific dynamics, providing a lightweight means to analyze and ultimately improve the interpretability of attention patterns.

108

2 RELATED WORK

110 **KV Cache Compression.** As model sizes and context windows grow, optimizing KV-cache memory
 111 is increasingly critical. Prior work largely falls into three directions: (i) token eviction/merging (Xiao
 112 et al., 2023; Li et al., 2024c; Zhang et al., 2023; Nawrot et al., 2024; Zhang et al., 2024b; Qin et al.,
 113 2025; Wang et al., 2025; Liu et al., 2025; Park et al., 2025; Cai et al., 2025; Park et al., 2025; Kim
 114 et al., 2024), (ii) vector compression/quantization (Hooper et al., 2024; Liu et al., 2024b; Yue et al.,
 115 2024; Sun et al., 2024a), and (iii) token retrieval (Tang et al., 2024; Liu et al., 2024a; Gao et al.,
 116 2025). While effective in many settings, vector compression and retrieval either discard fine-grained
 117 information or introduce nontrivial systems overhead (e.g., coordination and data movement) (Li
 118 et al., 2024a). Moreover, their memory and computation still scale with sequence length, making
 119 them inefficient for long-horizon generation applications. Token eviction offers a simple, memory-
 120 bounded alternative; however, most existing policies are heuristic and can significantly degrade
 121 performance, especially on long reasoning trajectories. Recent work has introduced learnable eviction
 122 policies (Chen et al., 2024; Zeng et al., 2024; Huang et al., 2024), but these are primarily designed for
 123 the pre-filling stage and thus are not well suited to sustained long-horizon generation. We bridge this
 124 gap by introducing a learnable and efficient eviction policy designed for long-horizon LLM inference
 125 under fixed memory budgets.

126 **Forgetting in Language Models.** A key limitation of vanilla self-attention is the lack of an explicit
 127 forgetting mechanism, forcing the model to carry potentially irrelevant information and making
 128 long-context processing inefficient. Early work tackled this by replacing quadratic attention with
 129 linearized and recurrent variants (Katharopoulos et al., 2020; Wang et al., 2020; Sun et al., 2023;
 130 Yang et al., 2023; 2024) that summarize the past into a fixed-size state, often a single vector. While
 131 computationally attractive, such heavy compression can degrade performance on tasks requiring
 132 long-range memory. [Follow-up studies](#) (Behrrouz et al., 2024; Sun et al., 2024b; Karami et al.;
 133 [Karami and Mirrokni, 2025](#)) increase memory capacity by replacing this hidden vector with a more
 134 expressive neural state. Complementary lines of work retain softmax attention but enforce forgetting
 135 by modifying attention logits (Lin et al., 2025) or imposing trainable sparsity patterns (Yuan et al.,
 136 2025). However, these approaches typically alter attention dynamics substantially and thus require
 137 training models from scratch. This incurs significant training cost and leaves their scalability to
 138 contemporary LLM sizes uncertain. In contrast, we introduce a *plug-in* forgetting mechanism for
 139 pretrained LLMs that converts them into memory-bounded models, providing long-context efficiency
 140 without retraining from scratch.

141

3 PRELIMINARIES

143

3.1 TRANSFORMERS WITH SELF-ATTENTION

144 Given a sequence of d -dimensional input vectors $\mathbf{x}_1, \dots, \mathbf{x}_T$, a (causal) self-attention layer attends
 145 only to past positions. For each $t = 1, \dots, T$, the attention output \mathbf{o}_t is computed as

$$147 \quad \mathbf{q}_t = \mathbf{W}_Q \mathbf{x}_t, \mathbf{k}_t = \mathbf{W}_K \mathbf{x}_t, \mathbf{v}_t = \mathbf{W}_V \mathbf{x}_t, \quad \mathbf{o}_t = \sum_{i=1}^t \frac{\exp(\mathbf{q}_t^\top \mathbf{k}_i)}{\sum_{j=1}^t \exp(\mathbf{q}_t^\top \mathbf{k}_j)} \mathbf{v}_i,$$

150 where \mathbf{q} , \mathbf{k} , \mathbf{v} are query, key, and value states, respectively, and $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d}$ are linear
 151 transformation weights. Here, we assume a single-head attention layer and omit the scaling factor
 152 $1/\sqrt{d}$ for simplicity. The sequence of key-value pairs $\{(\mathbf{k}_i, \mathbf{v}_i)\}_i$ is the in-context memory of the
 153 LLM. During the autoregressive decoding, we typically generate one token at a time and cache the
 154 running key-value pair $(\mathbf{k}_t, \mathbf{v}_t)$ to our in-context memory to avoid recomputation. However, this
 155 vanilla caching approach leads to a linear increase in memory footprint with the sequence length,
 156 while computation grows quadratically (Keles et al., 2023). This reduces efficiency when handling
 157 long-context inputs and extended generation tasks.

158

3.2 REVISITING KV CACHE EVICTION

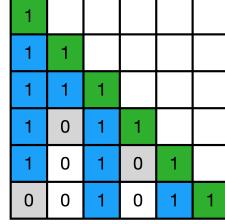
159 A common method to address the linear growth in the memory is to prune or compress the running
 160 key-value pairs into fixed-size (slot) memory. As new tokens arrive, we evict *un-(or less-)important*

tokens from our memory and append the new ones. To understand this procedure, we revisit and rewrite the attention computation with eviction at inference step t as follows:

$$\mathbf{o}'_t = \sum_{i=1}^t \frac{\exp(\alpha_{ti} \mathbf{q}_t^\top \mathbf{k}_i)}{\sum_{j=1}^t \exp(\alpha_{tj} \mathbf{q}_t^\top \mathbf{k}_j)} \mathbf{v}_i \quad \text{where} \quad \alpha_{ti} \in \{0, 1\} \text{ and } \alpha_{ti} \geq \alpha_{t+1,i}, \quad \forall i, t. \quad (1)$$

In Equation (1), we introduce a binary variable $\alpha_{ti} \in 0, 1$ indicating whether key-value pair i has been evicted at time t and the monotonicity constraint $\alpha_{ti} \geq \alpha_{t+1,i}$ ensures that we cannot retrieve a token once it is evicted (Figure 1). The goal is to choose a decision variable α so that the attention output deviates as little as possible from the full KV cache (all $\alpha_{ti} = 1, \forall i, t$).

$$\min_{\alpha} \mathcal{L}_{\text{base}}(\mathbf{o}'_t; \mathbf{o}_t) \quad \text{s. t.} \quad \sum_{i=1}^t \alpha_{ti} \leq M. \quad (2)$$



Here, \mathcal{L} penalizes differences between attention with and without eviction, and the constraint enforces keeping at most M tokens at any inference step t .

Figure 1: Attention w/ eviction ($M = 3$).

Solving the above constrained optimization at every time step t is impractical due to its combinatorial nature and efficiency requirements of LLM inference in real-world applications. Most existing approaches (Xiao et al., 2023; Han et al., 2023; Zhang et al., 2023; Li et al., 2024c; Cai et al., 2025; Ghadia et al., 2025) opt to determine α heuristically while we focus on a learnable eviction method.

4 METHODOLOGY

In this section, we propose a learning-based eviction policy that prunes the KV cache based on the *intrinsic importance* of the tokens at each layer and head. The policy ranks tokens by relative importance to decide which should be evicted from the KV memory. To learn token importance, we introduce a small neural network that takes token embeddings as input and produces a scalar retention score. We then integrate this retention score into the attention computation to modulate the attention weights. We term this proxy attention mechanism a *retention-gated attention*. We train the LLM with retention-gated attention against a baseline model with standard attention, using a combination of distillation and hinge-like regularization losses to enforce memory capacity constraints while preserving response quality. A visualization is shown in Figure 2.

4.1 SELECTIVE IN-CONTEXT MEMORY VIA RETENTION-GATED ATTENTION

We introduce retention-gated attention, a trainable mechanism that mimics the information loss induced by inference-time eviction. From the formulation (1), the sequence $\alpha_{ii}, \alpha_{(i+1)i}, \dots, \alpha_{ti}$ represents how token i is retained in the attention computation over time. Retention begins at 1 and then abruptly drops to 0 once the token is evicted. While this binary behavior matches the inference stage, it poses challenges for learning: the signal is discrete, non-differentiable, thus providing no gradients for optimization. To remedy this, we replace the hard binary variable α with a smooth, monotonically decreasing function that models the gradual decay of importance while enabling gradient-based training. A natural candidate is the sigmoid function, $\bar{\alpha}_{ti} = 1/(1 + \exp(f(\mathbf{x}_i, t)))$, which models the time at which the token is evicted. However, this design suffers from two drawbacks: (i) the domain of f is unnormalized since the sequence length is unknown during decoding, and (ii) the sigmoid flattens across most of its range, producing negligible variation between steps and leading to vanishing gradients during training.

To overcome these limitations, we adopt an exponential decay formulation, $\bar{\alpha}_{ti} = \beta_i^{t-i}$ where $\beta_i \in [0, 1]$, to model the retention rate of token i over time. Larger values of β_i correspond to higher intrinsic importance, implying slower decay and stronger memory retention. Substituting this design for α in Equation (1) yields our proposed *retention-gated attention*:

$$\mathbf{q}_t = \mathbf{W}_Q \mathbf{x}_t, \mathbf{k}_t = \mathbf{W}_K \mathbf{x}_t, \mathbf{v}_t = \mathbf{W}_V \mathbf{x}_t, \beta_t = g(\mathbf{x}_t), \quad \mathbf{o}_t = \sum_{i=1}^t \frac{\exp(\beta_i^{t-i} \mathbf{q}_t^\top \mathbf{k}_i)}{\sum_{j=1}^t \exp(\beta_j^{t-j} \mathbf{q}_t^\top \mathbf{k}_j)} \mathbf{v}_i. \quad (3)$$

Here, we propose a *retention gate* g , which is a lightweight network, to parametrize the token importance β_t . The retention gate can be a linear projection, *i.e.*, $g(\mathbf{x}) = \sigma(\mathbf{W}_\beta \mathbf{x} + b)$, $\mathbf{W}_\beta \in \mathbb{R}^{1 \times d}$,

216 or a simple MLP, *i.e.*, $g(\mathbf{x}) = \sigma(\text{MLP}(\mathbf{x}) + b)$. The sigmoid function σ squashes the output of g
 217 to the range $[0, 1]$, while b is a learnable bias. When all $\beta_t = 1, \forall t$, our retention-gated attention
 218 recovers the vanilla attention. Our ablation studies show that an MLP with a single hidden layer
 219 provides a more powerful retention estimation than a linear projection.

220 **Token Retention vs. Attention Scores.** In standard self-attention, the importance of a past token i
 221 at decoding step t is given by $a_{ti} \propto \exp(\mathbf{q}_t^\top \mathbf{k}_i)$, which depends explicitly on the *current* query \mathbf{q}_t .
 222 These scores capture *short-term* utility for predicting the next token and are recomputed at every step,
 223 making them local, myopic, and highly dependent on the transient decoding state.

224 KV cache eviction, in contrast, is a *long-horizon* decision: once a token is dropped, it cannot influence
 225 *any* future prediction. An effective eviction policy should depend on a token’s *intrinsic long-term*
 226 *utility* that reflects how useful it is expected to be over the remainder of the sequence and how long it
 227 has already stayed in the cache, rather than on the current query alone.

228 Token retention provides a more suitable abstraction. Instead of asking “how much should token
 229 i contribute *now?*” it asks “how important is token i for the long run, and for how long should
 230 it stay in the cache?” Concretely, each token i receives a scalar retention score $\beta_i \in [0, 1]$ based
 231 only on information available at creation time (its representation, layer, and head), and its effective
 232 contribution at future step t decays as β_i^{t-i} . This yields a smooth, exponentially decaying retention
 233 curve that is aligned naturally with long-term utility under memory constraints.

234 **Brain-inspired Interpretation.** Our proposed retention-gated attention bears a natural connection to
 235 the classical Ebbinghaus’s forgetting curve theory (Ebbinghaus, 2013), which models human memory
 236 retention as an exponential decay over time. A common approximation of human retention rate is
 237 $R = \exp(-tS)$, where t is the time and S is the memory strength determining how fast R decays
 238 over time in the absence of training (Woźniak et al., 1995).

239 In a similar spirit, our retention-gated attention models the contribution of past tokens as an exponentially
 240 decreasing function of their temporal distance from the current step, *i.e.*, $\exp((t - i) \log \beta_i)$.
 241 Each token begins with full weight ($\bar{\alpha}_{ii} = 1$), akin to a newly encoded memory, and its influence
 242 decays as more tokens arrive, mirroring the drop in recall probability described by the forgetting
 243 curve. The parameter β acts analogously to memory strength S : larger values yield more persistent,
 244 durable memories, while smaller values indicate weaker memories that fade quickly.

245 This connection provides an intuitive justification for our design. By embedding a forgetting mechanism
 246 into attention, we enable the model to dynamically prioritize recent or intrinsically important
 247 tokens while gradually discarding less informative context, mirroring how humans manage limited
 248 memory capacity in practice. Note that Zhong et al. (2024) also drew on Ebbinghaus’s forgetting
 249 curve to construct a long-term memory bank, but their focus was on retrieval-augmented generation,
 250 whereas our approach integrates forgetting directly into the attention mechanism.

4.2 TRAINING

251 Our goal is to train the retention gate g so that
 252 the LLM can preserve response quality under
 253 a memory constraint, thereby bridging the gap
 254 with the inference stage. Instead of training a
 255 separate gate for each layer and head, as
 256 formulated in Problem (2), we optimize all reten-
 257 tion gates jointly in an end-to-end fashion. This
 258 holistic approach mitigates error propagation,
 259 allowing the model to learn a coordinated, glob-
 260 ally optimal caching policy rather than greedy
 261 layer-wise decisions. Starting from a pretrained
 262 LLM, we replace every standard attention block
 263 with our proposed retention-gated attention. Each block is equipped with a lightweight retention gate
 264 g that maps token representations to retention scores $\beta_t \in [0, 1]$, which are then used to modulate
 265 attention weights according to Equation (3). We call this proxy LLM a retention-gated LLM.

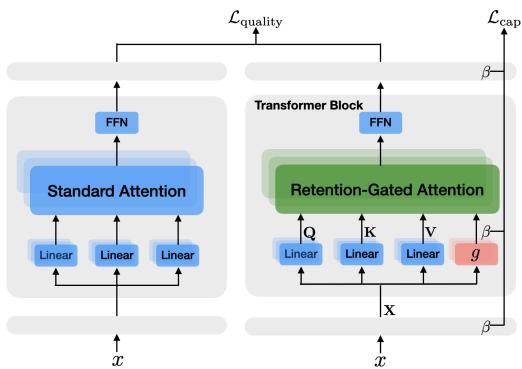


Figure 2: Training architecture.

270 **Objectives.** To train these retention gates, we formulate the training objective that balances two
 271 goals: (i) preserving the predictive quality of the original pretrained LLM, and (ii) enforcing memory
 272 capacity constraints by controlling the sum of retention scores at each step.
 273

274 For the first objective, we use a combination of the distillation and standard next-token prediction
 275 losses. The distillation loss encourages the proxy LLM to align its output distribution with that of the
 276 baseline LLM using standard attention. In parallel, the next-token prediction loss enables the model
 277 to uncover sparsity patterns directly from the data, extending beyond the constraints of the pretrained
 278 LLM. Let $p(\cdot)$ and $q_\theta(\cdot)$ be the output distribution of the pretrained LLM and retention-gated LLM,
 279 respectively, where θ denotes the parameters of all retention gates. The quality loss is given by
 280

$$\mathcal{L}_{\text{quality}} = \mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{NTP}} = \mathcal{D}_{\text{KL}}(p(\cdot|x) \| q_\theta(\cdot|x)) + \mathbb{E}_{(x,y)}[-\log q_\theta(y|x)]. \quad (4)$$

281 Here, \mathcal{D}_{KL} is the standard forward Kullback-Leibler divergence (Kullback and Leibler, 1951).
 282

283 For the second objective, we impose a hinge-like regularization penalty, which discourages the model
 284 from exceeding the available KV memory slots at each step. For a retention gate within a given layer
 285 and KV head, the memory capacity loss is defined as:
 286

$$\mathcal{L}_{\text{cap}} = \frac{1}{T(T-M)} \sum_{t=1}^T \max \{0, \sum_{i=1}^t \beta_i^{t-i} - M\}, \quad (5)$$

287 where T is the sequence length and M is the predefined memory capacity. Here, M acts as a *soft*
 288 hyperparameter, primarily intended to prevent over-optimization during the early decoding stage
 289 when the sequence remains short. Training is performed with a fixed value of M , while inference
 290 can flexibly accommodate different KV budgets. This regularization is applied uniformly across all
 291 layers and KV heads of the transformer. The combined training objective is then:
 292

$$\min_{\theta} \mathcal{L}_{\text{quality}} + \lambda_{\text{cap}} \mathcal{L}_{\text{cap}}, \quad (6)$$

293 where λ_{cap} is a hyperparameter balancing between quality and capacity loss. Note that during
 294 training, only the retention gate parameters are updated, while all other model weights remain frozen.
 295

296 **Hardware-aware Computation.** Retention-gated attention is fully parallelizable and compatible
 297 with FlashAttention-style kernels (Dao, 2023). We implement it with FlexAttention (Dong et al.,
 298 2024) plus a custom Triton kernel for the capacity loss \mathcal{L}_{cap} , performing forward/backward without
 299 materializing the full attention or β matrices. This enables long-context training (up to 128K tokens
 300 on four H100 GPUs) with minor overhead versus standard parameter-efficient fine-tuning.
 301

302 4.3 INFERENCE

303 At inference time, the base LLM is augmented with the retention gates learned during training
 304 (Section 4.2). These gates provide token-level intrinsic importance scores β_i , which quantify how
 305 strongly each past token should be retained for future computations. Unlike training, where the
 306 retention gates are used to modulate the attention weights, at inference, they act purely as decision-
 307 makers for eviction, operating alongside but independently of attention computation.
 308

309 The eviction process is designed to ensure that the KV cache respects a predefined memory budget.
 310 Let $S_t \subseteq \{1, \dots, t\}$ denote the set of tokens currently stored in the KV cache at decoding step t .
 311 When a new token $t+1$ is generated, it is provisionally added to the cache. If this addition causes
 312 the cache size to exceed the memory capacity M , an eviction is triggered. The eviction rule is simple
 313 yet principled: we remove the token with the lowest retention score, i.e.,
 314

$$j_{\text{evic}} = \arg \min_{j \in S_t} \{\beta_j^{t-j}\}.$$

315 Intuitively, this favors retaining tokens deemed globally important by the learned retention gates
 316 while discarding those with little long-term value. In practice, this makes inference both memory-
 317 efficient and adaptive: as new context arrives, the model continually re-evaluates the importance of
 318 older tokens, enabling long-context generation while keeping memory usage bounded. Algorithm 1
 319 illustrates a single decoding step, where attention computation is coupled with token eviction.
 320

321 **Complexity.** Our inference is simpler and more efficient than existing works, including pure heuristic
 322 baselines (Li et al., 2024c). Throughput and runtime comparisons are in Appendix A.2.
 323

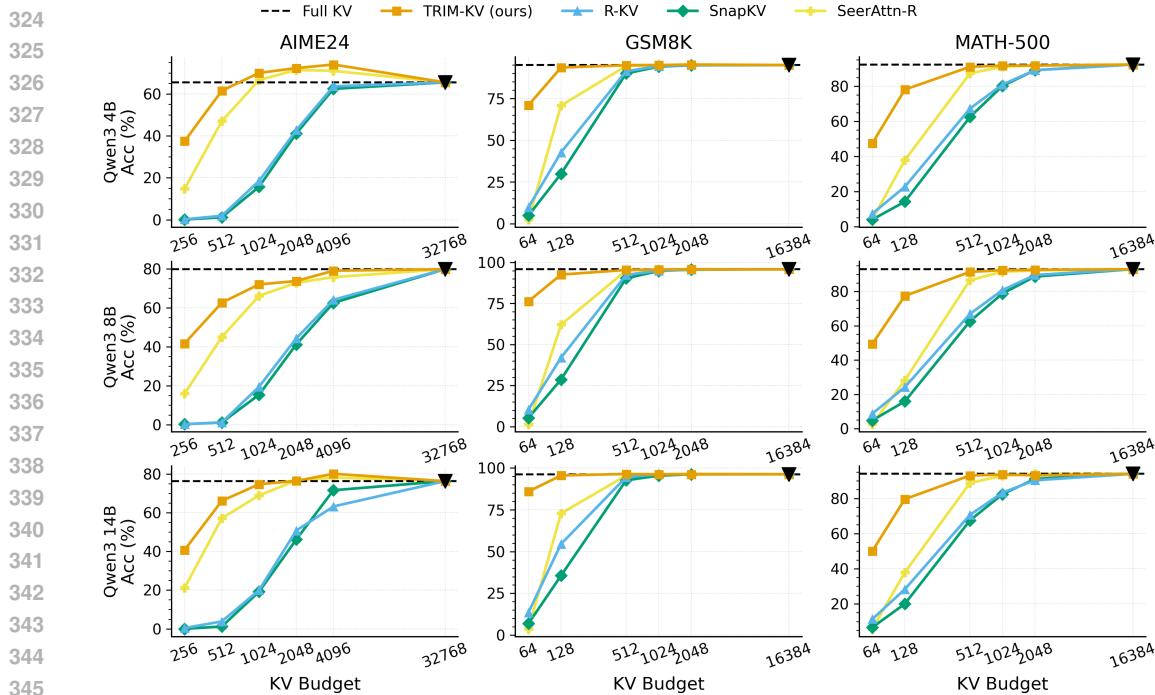


Figure 3: Patero frontiers of competing algorithms with different budgets on math benchmarks.

5 EXPERIMENTS

In this section, we conduct extensive experiments to demonstrate the performance advantages of our method on both *long-context* and *long-generation* tasks.

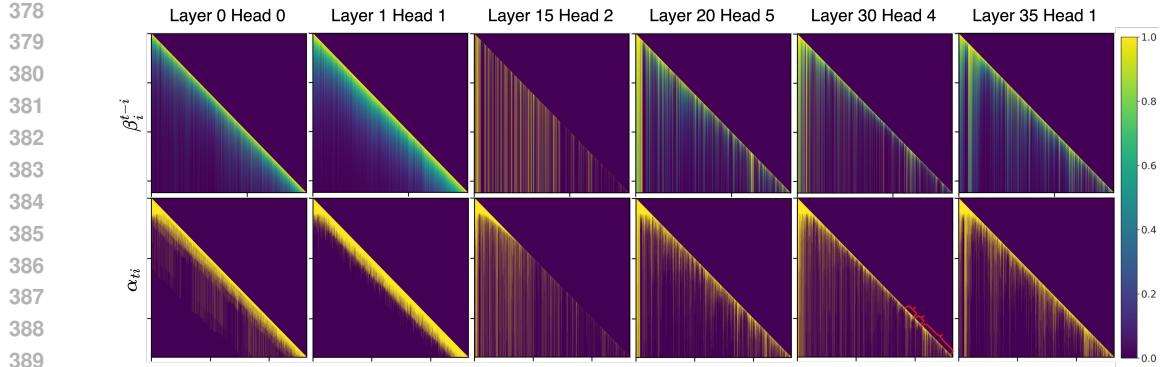
5.1 LONG GENERATION EVALUATION

Benchmarks. Following prior work (Gao et al., 2025; Cai et al., 2025), we evaluate on standard math-reasoning suites—AIME24 (Art of Problem Solving, 2024), GSM8K (Cobbe et al., 2021), and MATH-500 (Hendrycks et al., 2021). To assess performance beyond math reasoning and under long-context settings, we also report results on LongProc (Ye et al., 2025). Following (Gao et al., 2025), we report average pass@1 accuracy over 64 samples for AIME24 and 8 samples for GSM8K, MATH-500. We use greedy decoding for LongProc as the default in the benchmark.

Base Models. Following (Gao et al., 2025), we mainly use Qwen3’s family models (Yang et al., 2025), including Qwen3-1.7B, Qwen3-4B, Qwen3-8B, Qwen3-14B and DeepSeek R1 Distill (Guo et al., 2025) variants including, DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-Llama-8B. We report the results with Qwen3 models in the main paper, and the remaining is in Appendix B.

Baselines. We compare our method against SeerAttn-R (Gao et al., 2025), R-KV (Cai et al., 2025), SnapKV (Li et al., 2024c), H2O (Zhang et al., 2023), StreamingLLM (Xiao et al., 2023). R-KV, SnapKV, H2O, and StreamingLLM are heuristic, recency-driven KV *eviction* policies for long-form generation under a fixed memory budget. SeerAttn-R is a learnable KV *retrieval* approach for reasoning tasks: rather than evicting, it offloads the full KV cache to host memory and uses recent queries to fetch relevant blocks for attention. KV retrieval methods preserve all past information but require nontrivial CPU–GPU orchestration and incur offloading overhead. We therefore treat SeerAttn-R as a strong learnable baseline, and R-KV/SnapKV as representative eviction baselines.

Implementation Details. We train the retention gates using OpenR1-MATH-220k (Hugging Face) dataset, similar to (Gao et al., 2025). Note that we only train the retention gates’ weights while keeping the original model parameters frozen. We set the objective hyperparameters $\lambda_{\text{cap}} = 1.0$ and the memory capacity $M = 512$. Each transformer block has a retention gate g , which is a single MLP layer with the hidden dimension of 512, thus having dimensions of $d \rightarrow 512 \rightarrow h$, where h is the number of KV heads. We use the activation function as the default activation function in MLP

Figure 4: Visualization of token retention score β_i^{t-i} (top) and eviction decisions α_{ti} (bottom).

layers of the base model. We initialize the bias in the retention gates to a large value (e.g., $b = 8.0$) to begin training with minimal forgetting or compression. All trainings are on 4 H100 80G GPUs.

5.1.1 QUANTITATIVE RESULT

Math Reasoning Tasks. Figure 3 shows our method outperforming all baselines by a large margin, especially in low-budget regimes. Notably, TRIM-KV surpasses attention-guided methods (R-KV, SnapKV) even when they are given $4 \times$ KV budget. Under the same budget, *i.e.* 1024 for AIME24 and 128 for GSM8K/MATH-500, it yields a **198%** relative improvement over these baselines. Against the SOTA learnable KV retrieval baseline, TRIM-KV outperforms SeerAttn-R across all settings, yielding a **58.4%** pass@1 gain at the same budget. Crucially, TRIM-KV operates in a pure KV-eviction regime, a stricter setting than the KV retrieval methods such as SeerAttn-R, and thus avoids CPU–GPU offloading overhead. In some settings, like for Qwen3-4B model and AIME24 dataset, our method can even surpass the standard full KV cache. These results suggest that a large fraction of KV-cache tokens in reasoning models is redundant and can be discarded without degrading performance.

Long Procedural Generation Tasks. We evaluate KV-eviction methods on tasks that require both long-context comprehension and extended generation. Table 1 reports results with Qwen3-4B model. Overall, TRIM-KV consistently outperforms all other eviction baselines and, in several settings, even surpasses the full-cache model. Moreover, this result highlights that TRIM-KV with retention gates trained on math-reasoning data generalizes well to non-math tasks. Full Table 1: Qwen3-4B on LongProc. Bold is for results and analysis are provided in Appendix B.

Method _{KV budget}	CountDown			Pseudo to Code	
	0.5k	2k	8k	0.5k	2k
FullKV	96.0	90.0	69.0	50.8	25.0
StreamingLLM ₂₀₄₈	7.0	5.0	5.0	20.6	1.5
H2O ₂₀₄₈	12.0	7.5	2.5	33.7	0.5
SnapKV ₂₀₄₈	57.0	49.0	13.0	42.7	4.5
R-KV ₂₀₄₈	88.5	81.0	63.0	48.2	2.5
TRIM-KV ₂₀₄₈	97.5	93.5	66.0	49.2	19.0

the best, underline is for the best KV eviction.

5.1.2 QUALITATIVE RESULT

To examine the eviction policy learned by our retention gates, we run TRIM-KV on Qwen3-4B for the first example in AIME24 (see Figure 14 for a visualization of the example). Figure 5a–b show the mean retention score, averaged over all layers and heads, for each token in the example sequence. Aligning with our intuition, retention gates assign high scores to task-relevant tokens (e.g., `ometer`, `shop`, `walk`, `minutes`) and to the initial token `<| im_start |>`, which often serves as an attention sink. In contrast, whitespace and punctuation receive low retention scores and are discarded early, yielding short lifespans in the KV cache. Next, we examine retention scores and eviction decisions at layer–head granularity.

Emergent Eviction Heuristics. Figure 4 visualizes the retention scores β_i^{t-i} and eviction decisions α_{ti} for selected layers and heads. Many eviction heuristics, such as attention sinks (Xiao et al., 2023), sliding windows (Zhu et al., 2021), A-shape (Jiang et al., 2024), emerge naturally from our learned policy without being hard-coded, and they adapt to the functional roles of individual layers and heads. For instance, sliding-window behavior is more common in early layers, whereas attention sinks appear more frequently in later layers (see Figure 12 and 13 for a comprehensive view). Moreover,

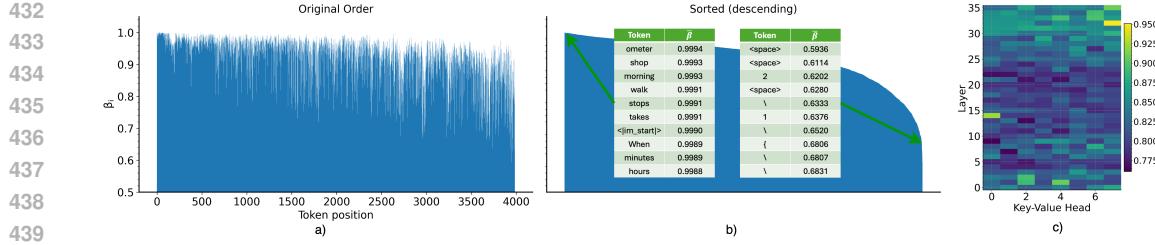


Figure 5: a) Average retention scores across all layers and heads of Qwen3-4B on tokens of an AIME24 example. b) Top 10 tokens with the highest (left table) and lowest (right table) average retention. c) The layer- and head-wise sparsity level estimated by token retentions.

Method	En.MultiChoice	Retr.KV	ICL.ManyShot	Math.Find	En.QA	Code.RepoQA	En.Sum	Mix.Sum+NIAH	Retr.MultiHop
Full KV	20.52	66.00	95.57	32.60	28.78	53.86	36.48	38.33	49.60
StreamingLLM ₄₀₉₆	5.68	2.20	100.00	13.20	6.84	2.96	29.21	28.25	0.00
H2O ₄₀₉₆	4.80	0.00	100.00	8.00	3.70	0.46	8.97	6.51	0.27
SnapKV ₄₀₉₆	10.04	0.00	100.00	18.60	6.29	0.23	27.90	29.28	0.31
TRIM-KV₄₀₉₆	13.10	0.00	100.00	13.80	19.09	4.32	33.66	34.06	43.11

Table 2: Performance on long-context tasks from the SCBench benchmark.

TRIM-KV adapts the window size by layer and head: in *Layer 1/Head 1*, tokens receive nearly uniform retention scores, so the KV cache behaves like a recency-biased window that keeps the most recent tokens; in *Layer 0/Head 0*, multiple sliding windows of varying widths emerge from the learned policy; in *Layer 15/Head 2*, no sliding window is observed because certain tokens receive substantially higher retention than others, suggesting a specialized functional role for this head. The A-shaped pattern typically appears in layers that emphasize instruction/problem-statement tokens (e.g., *Layer 20/Head 5* and *Layer 30/Head 4*) or chain-of-thought/reasoning prompts (e.g., *Layer 35/Head 1*). These heads also exhibit context switching, where small, dense lower-triangular blocks emerge and then fade quickly when the context changes or a sentence completes. To the best of our knowledge, the absence of a sliding window, the presence of multiple coexisting windows, and context switching are newly observed eviction patterns that arise naturally from our learned policy.

Token Retention Enables Interpretability. Beyond guiding eviction policy, token-level retention scores provide a diagnostic tool for analyzing the functional roles of individual KV heads in the base LLM. Visualizing retention scores alongside the tokens preserved in the KV cache after generation reveals distinct specializations: some heads emphasize a recency window (Figure 17), whereas others preferentially retain mathematical tokens-numbers and operators (Figures 14 and 20)-or variables (Figure 23), as well as problem-description tokens (Figures 19 and 25) and chain-of-thought instructions (Figure 26). Even function or filler words, such as pronouns, prepositions, conjunctions, `wait` and `let`, tend to be kept by specific heads (Figures 18 and 24). In particular, heads that exhibit context-switching patterns (e.g., *Layer 30/Head 4* and *Layer 16/Head 6*) tend to retain the period token while discarding others (Figures 22 and 21). We hypothesize that, in these heads, periods act as implicit *gist* tokens (Mu et al., 2023), summarizing the information in the preceding sentences.

Our analyses indicate that KV heads in LLM develop different functional roles and therefore keep different types of tokens. These tokens are often dispersed across the context rather than forming contiguous chunks, as each already captures contextual information. This observation contrasts with existing approaches (Yuan et al., 2025; Gao et al., 2025) that advocate chunk- or block-based KV-cache. Instead, we show that keeping a small number of high-context tokens is more budget-effective.

Budget Allocation. Figure 5c reports head- and layer-wise sparsity estimated from the retention scores *i.e.*, $1 - \frac{2}{T(T+1)} \sum_{i < t} \beta_i^{t-i}$. We observe that later layers are typically sparser than earlier ones, consistent with prior findings in (Cai et al., 2024). Practically, the retention scores enable heterogeneous budgets across KV heads under a global constraint by evicting tokens with low global retention. However, existing KV-cache and FlashAttention implementations assume uniform sequence lengths across heads within a layer; enabling efficient per-head variable-length caches is left to future work.

5.2 LONG-CONTEXT EVALUATION

486 To further stress-test our approach on long-context tasks, we evaluate it on LongMemEval_S (Wu et al.,
 487 2024a), a benchmark for assessing chat assistants’ long-term interactive memory (contexts up to
 488 123K tokens with Qwen3) and SCBench (Li et al., 2024b), a benchmark adapted from RULER (Hsieh
 489 et al., 2024) and InfiniteBench (Zhang et al., 2024a) to evaluate long-context understanding of
 490 KV cache compression methods under the same context but different queries. We use Qwen3-4B-
 491 Instruct (Qwen3, 2025) as the base model and train the retention gates on SynthLong (Lazarevich
 492 et al., 2025), BookSum (Kryściński et al., 2021), and Buddhi (Singhal, 2024), whose context lengths
 493 extend up to 128K tokens. All other experimental settings follow Section 5.1.

494 As shown in Table 3 and Table 2 show the results on LongMemEval and
 495 SCBench benchmarks respectively. On LongMemEval, TRIM-KV outper-
 496 forms baselines by a significant margin. Especially, TRIM-KV can maintain
 497 the performance of a full cache while using just 25% of the KV budget, while
 498 other baselines drop sharply. On SCBench, TRIM-KV also shows a competitive
 499 performance across most of the evaluated tasks. On challenging retrieval tasks
 500 such as Retr.KV and Code.RepoQA, all KV eviction methods failed to give a
 501 reasonable performance because the context is incompressible. This is already
 502 observed in (Li et al., 2024b). These results highlight our method’s advantage
 503 on both long-context and long-generation tasks, whereas most prior work targets
 504 either the prefill or the generation stage, but not both. More details and results
 505 for this experiment are provided in Appendix B.

Method	KV budget	Acc
Full KV	₁₃₁₀₇₂	49.4
StreamingLLM	₃₂₇₆₈	27.6
SnapKV	₃₂₇₆₈	27.8
TRIM-KV	₃₂₇₆₈	48.2
StreamingLLM	₁₆₃₈₄	19.0
SnapKV	₁₆₃₈₄	18.2
TRIM-KV	₁₆₃₈₄	42.6
StreamingLLM	₄₀₉₆	10.2
SnapKV	₄₀₉₆	13.4
TRIM-KV	₄₀₉₆	30.2

Table 3: Results on LongMemEval_S.

506 507 508 5.3 ABLATION STUDIES

509 We ablate the objective by training the Qwen3-4B retention gates with
 510 different loss combinations and report AIME24 pass@1 at a 4096-
 511 token budget in Table 4. Both forward KL and next-token prediction
 512 perform well on their own, and their combination further improves
 513 accuracy. The memory capacity loss is essential for compression, and
 514 removing it leads to a sharp drop. We provide comprehensive ablations
 515 with reversed KL, generalization with different training datasets, gate’s
 516 architecture, and other hyperparameters such as M in Appendix B.4.

Method	KV budget	pass@1
Full KV	₃₂₇₆₈	65.5
TRIM-KV	₄₀₉₆	74.0
(TRIM-KV - \mathcal{L}_{KL})	₄₀₉₆	71.4
(TRIM-KV - \mathcal{L}_{NTP})	₄₀₉₆	70.7
(TRIM-KV - \mathcal{L}_{cap})	₄₀₉₆	42.9

Table 4: Objective ablation.

517 6 CONCLUSION AND FUTURE WORK

518 We introduced TRIM-KV, a learnable, retention-gated approach to KV cache management that
 519 prioritizes tokens by intrinsic importance rather than recent attention. By training lightweight gates
 520 with distillation and a capacity loss, our method enforces strict memory budgets with a simple and
 521 efficient eviction policy. Extensive experiments across math reasoning, long-procedural generation,
 522 and conversational long-memory benchmarks demonstrate that our method outperforms strong
 523 eviction and retrieval baselines, and sometimes even surpasses full-cache models. Analyses show that
 524 the learned retention scores align with human intuitions and reveal layer- and head-specific dynamics,
 525 offering a simple probe for interpretability.

526 **Future Work.** Our results indicate that retention-gated attention is an effective learnable proxy for
 527 approximating standard attention with eviction during inference. In the current work, however, we
 528 keep the backbone parameters frozen during training and still rely on standard attention at inference
 529 time. A natural next step is to replace standard attention with retention-gated attention and train the
 530 retention mechanism jointly with the attention layers during pretraining or post-training. This could
 531 allow the retention scores to better cooperate with the learned query, key, and value states, shaping the
 532 model’s ‘memory habits’¹ from the outset rather than optimizing retention on top of a fixed attention
 533 stack. Such a design would enable training objectives that explicitly trade off task performance and
 534 memory usage, potentially yielding models that are inherently memory-bounded without requiring
 535 any post hoc compression policy.

536 Besides, building on these results, we plan to extend retention gating to multimodal inputs, tool-
 537 calling applications, and develop adaptive budgets that allocate memory across layers, heads, and
 538 tasks to further improve both performance and efficiency.

539 ¹We borrow this term from the reviewer gKFH. We appreciate all reviewers for their constructive suggestions.

540 ETHICS STATEMENT
541

542 This work aims to improve the efficiency of large language models by reducing their memory and
543 computational footprint. Our method can make long-context reasoning more accessible by lowering
544 hardware costs, which may democratize access to advanced LLM capabilities. However, efficiency
545 improvements may also accelerate the deployment of LLMs in high-stakes or resource-limited settings
546 where risks around misinformation, bias, or misuse persist. We stress that our method does not
547 mitigate these broader societal risks and should be paired with ongoing efforts in safety, fairness, and
548 responsible deployment.

550 REPRODUCIBILITY STATEMENT
551

552 We ensure reproducibility by providing detailed descriptions of the model architecture, training
553 objectives, and evaluation benchmarks in the main text and appendix. Hyperparameters, training
554 schedules, and implementation details are included for all experiments. All datasets we use are
555 publicly available, and we will release code, model checkpoints, and scripts for training and evaluation
556 upon publication. Together, these materials allow independent researchers to fully reproduce and
557 verify our results.

558 The authors used large language models to help refine and polish the writing of this manuscript.
559

560 REFERENCES
561

562 Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar,
563 Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical
564 report. *arXiv preprint arXiv:2412.08905*, 2024.

565 Art of Problem Solving. AIME problems and solutions, 2024. URL https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions.

566 Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao
567 Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context
568 understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational
569 Linguistics (Volume 1: Long Papers)*, pages 3119–3137, 2024.

570 Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu,
571 Lei Hou, Yuxiao Dong, et al. Longbench v2: Towards deeper understanding and reasoning on
572 realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for
573 Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664, 2025.

574 Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv
575 preprint arXiv:2501.00663*, 2024.

576 Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong,
577 Yue Dong, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal
578 information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

579 Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-
580 Wen Chang, Jiuxiang Gu, et al. R-kv: Redundancy-aware kv cache compression for training-free
581 reasoning models acceleration. *arXiv preprint arXiv:2505.24133*, 2025.

582 Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu,
583 Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-
584 thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*, 2025.

585 Yilong Chen, Guoxia Wang, Junyuan Shang, Shiyao Cui, Zhenyu Zhang, Tingwen Liu, Shuohuan
586 Wang, Yu Sun, Dianhai Yu, and Hua Wu. Nacl: A general and effective kv cache eviction
587 framework for llms at inference time. *arXiv preprint arXiv:2408.03675*, 2024.

594 Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora:
 595 Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*,
 596 2023.

597

598 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,
 599 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve
 600 math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

601 Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv*
 602 *preprint arXiv:2307.08691*, 2023.

603

604 Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A
 605 programming model for generating optimized attention kernels. *arXiv preprint arXiv:2412.05496*,
 606 2024.

607 Hermann Ebbinghaus. [image] memory: A contribution to experimental psychology. *Annals of*
 608 *neurosciences*, 20(4):155, 2013.

609

610 Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language
 611 models (effectively). *arXiv preprint arXiv:2410.02660*, 2024.

612

613 Yizhao Gao, Shuming Guo, Shijie Cao, Yuqing Xia, Yu Cheng, Lei Wang, Lingxiao Ma, Yutao Sun,
 614 Tianzhu Ye, Li Dong, et al. Seerattention-r: Sparse attention adaptation for long reasoning. *arXiv*
 615 *preprint arXiv:2506.08889*, 2025.

616

617 Ravi Ghadia, Avinash Kumar, Gaurav Jain, Prashant Nair, and Poulami Das. Dialogue without limits:
 618 Constant-sized kv caches for extended responses in llms. *arXiv preprint arXiv:2503.00979*, 2025.

619

620 Daya Guo, Dejian Yang, Huawei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu,
 621 Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms
 622 via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

623

624 Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-
 625 infinite: Zero-shot extreme length generalization for large language models. *arXiv preprint*
 626 *arXiv:2308.16137*, 2023.

627

628 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song,
 629 and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv*
 630 *preprint arXiv:2103.03874*, 2021.

631

632 Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt
 633 Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv
 634 cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

635

636 Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang
 637 Zhang, and Boris Ginsburg. Ruler: What’s the real context size of your long-context language
 638 models?, 2024. URL <https://arxiv.org/abs/2404.06654>.

639

640 Yuxiang Huang, Binhang Yuan, Xu Han, Chaojun Xiao, and Zhiyuan Liu. Locret: Enhancing eviction
 641 in long-context llm inference with trained retaining heads on consumer-grade devices. *arXiv*
 642 *preprint arXiv:2410.01805*, 2024.

643

644 Huiqiang Jiang, Yucheng Li, Chengrudong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua
 645 Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling
 646 for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing*
 647 *Systems*, 37:52481–52515, 2024.

648

649 Mahdi Karami and Vahab Mirrokni. Lattice: Learning to efficiently compress the memory. *arXiv*
 650 *preprint arXiv:2504.05646*, 2025.

648 Mahdi Karami, Ali Behrouz, Praneeth Kacham, and Vahab Mirrokni. Trellis: Learning to compress
 649 key-value memory in attention models. In *Second Conference on Language Modeling*.
 650

651 Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns:
 652 Fast autoregressive transformers with linear attention. In *International conference on machine
 653 learning*, pages 5156–5165. PMLR, 2020.

654 Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. On the computational
 655 complexity of self-attention. In *International conference on algorithmic learning theory*, pages
 656 597–619. PMLR, 2023.
 657

658 Minsoo Kim, Kyuhong Shim, Jungwook Choi, and Simyung Chang. Infinipot: Infinite context
 659 processing on memory-constrained llms. *arXiv preprint arXiv:2410.01518*, 2024.

660 Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev.
 661 Booksum: A collection of datasets for long-form narrative summarization. 2021.
 662

663 Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical
 664 statistics*, 22(1):79–86, 1951.
 665

666 Ivan Lazarevich, David Bick, Harsh Gupta, Srinjoy Mukherjee, Nishit
 667 Neema, Gokul Ramakrishnan, and Ganesh Venkatesh. Extending llm con-
 668 text with 99% less training tokens. [https://cerebras.ai/blog/
 669 extending-llm-context-with-99-less-training-tokens](https://cerebras.ai/blog/extending-llm-context-with-99-less-training-tokens), February 2025.

670 Haoyang Li, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole Hu, Wei
 671 Dong, Qing Li, and Lei Chen. A survey on large language model acceleration based on kv cache
 672 management. *arXiv preprint arXiv:2412.19442*, 2024a.
 673

674 Yucheng Li, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir H
 675 Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, et al. Scbench: A kv cache-centric analysis of
 676 long-context methods. *arXiv preprint arXiv:2412.10319*, 2024b.
 677

678 Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai,
 679 Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation.
Advances in Neural Information Processing Systems, 37:22947–22970, 2024c.
 680

681 Zhuoling Li, Xiaogang Xu, Zhenhua Xu, SerNam Lim, and Hengshuang Zhao. Larm: Large
 682 auto-regressive model for long-horizon embodied intelligence. *arXiv preprint arXiv:2405.17424*,
 683 2024d.

684 Zhixuan Lin, Evgenii Nikishin, Xu Owen He, and Aaron Courville. Forgetting transformer: Softmax
 685 attention with a forget gate. *arXiv preprint arXiv:2503.02130*, 2025.
 686

687 Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chen-
 688 gruidong Zhang, Bailu Ding, Kai Zhang, et al. Retrievalattention: Accelerating long-context llm
 689 inference via vector retrieval. *arXiv preprint arXiv:2409.10516*, 2024a.
 690

691 Xiang Liu, Zhenheng Tang, Hong Chen, Peijie Dong, Zeyu Li, Xiuze Zhou, Bo Li, Xuming Hu,
 692 and Xiaowen Chu. Can llms maintain fundamental abilities under kv cache compression? *arXiv
 693 preprint arXiv:2502.01941*, 2025.
 694

695 Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi
 696 Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint
 697 arXiv:2402.02750*, 2024b.
 698

699 Jesse Mu, Xiang Li, and Noah Goodman. Learning to compress prompts with gist tokens. *Advances
 700 in Neural Information Processing Systems*, 36:19327–19352, 2023.
 701

Piotr Nawrot, Adrian Łaćucki, Marcin Chochowski, David Tarjan, and Edoardo M Ponti. Dynamic
 memory compression: Retrofitting llms for accelerated inference. *arXiv preprint arXiv:2403.09636*,
 2024.

702 Junyoung Park, Dalton Jones, Matthew J Morse, Raghav Goel, Mingu Lee, and Chris Lott. Keydiff:
 703 Key similarity-based kv cache eviction for long-context llm inference in resource-constrained
 704 environments. *arXiv preprint arXiv:2504.15364*, 2025.

705 Ziran Qin, Yuchen Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, Weiyao Lin, and Jianguo
 706 Li. Cake: Cascading and adaptive kv cache eviction with layer preferences. *arXiv preprint*
 707 *arXiv:2503.12491*, 2025.

708 Qwen3. Qwen/qwen3-4b-instruct-2507, 2025. URL <https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507>.

709 Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen attention: Attention in
 710 low-rank space for kv cache compression. *arXiv preprint arXiv:2408.05646*, 2024.

711 Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael
 712 Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In
 713 *International Conference on Machine Learning*, pages 31210–31227. PMLR, 2023.

714 Chaitanya Singhal. Introducing buddhi: Open-source chat model with a 128k context window. AI
 715 Planet (Medium), April 2024. URL <https://huggingface.co/datasets/aiplanet/buddhi-dataset>. Accessed: 2025-08-23.

716 Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie
 717 Chi, and Beidi Chen. Shadowkv: Kv cache in shadows for high-throughput long-context llm
 718 inference. *arXiv preprint arXiv:2410.21465*, 2024a.

719 Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei
 720 Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive
 721 hidden states. *arXiv preprint arXiv:2407.04620*, 2024b.

722 Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and
 723 Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint*
 724 *arXiv:2307.08621*, 2023.

725 Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest:
 726 Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*,
 727 2024.

728 Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head
 729 self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint*
 730 *arXiv:1905.09418*, 2019.

731 Guangtao Wang, Shubhangi Upasani, Chen Wu, Darshan Gandhi, Jonathan Li, Changran Hu, Bo Li,
 732 and Urmish Thakker. Llms know what to drop: Self-attention guided kv cache eviction for efficient
 733 long-context inference. *arXiv preprint arXiv:2503.08879*, 2025.

734 Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with
 735 linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

736 Piotr Woźniak, Edward Gorzelańczyk, and Janusz Murakowski. Two components of long-term
 737 memory. *Acta neurobiologiae experimentalis*, 55(4):301–305, 1995.

738 Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval:
 739 Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*,
 740 2024a.

741 Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically
 742 explains long-context factuality. *arXiv preprint arXiv:2404.15574*, 2024b.

743 Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming
 744 language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

745 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang
 746 Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*,
 747 2025.

756 Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention
 757 transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.
 758

759 Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers
 760 with the delta rule over sequence length. *arXiv preprint arXiv:2406.06484*, 2024.
 761

762 Xi Ye, Fangcong Yin, Yinghui He, Joie Zhang, Howard Yen, Tianyu Gao, Greg Durrett, and Danqi
 763 Chen. Longproc: Benchmarking long-context language models on long procedural generation.
 764 *arXiv preprint arXiv:2501.05414*, 2025.
 765

766 Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie,
 767 YX Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively
 768 trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025.
 769

770 Yuxuan Yue, Zhihang Yuan, Haojie Duanmu, Sifan Zhou, Jianlong Wu, and Liqiang Nie. Wkvquant:
 771 Quantizing weight and key/value cache for large language models gains more. *arXiv preprint
 772 arXiv:2402.12065*, 2024.
 773

774 Zihao Zeng, Bokai Lin, Tianqi Hou, Hao Zhang, and Zhijie Deng. In-context kv-cache eviction for
 775 llms via attention-gate. *arXiv preprint arXiv:2410.12876*, 2024.
 776

777 Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen
 778 Thai, Shuo Wang, Zhiyuan Liu, et al. Infinitebench: Extending long context evaluation beyond
 779 100k tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational
 780 Linguistics (Volume 1: Long Papers)*, pages 15262–15277, 2024a.
 781

782 Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji.
 783 Cam: Cache merging for memory-efficient llms inference. In *Forty-first International Conference
 784 on Machine Learning*, 2024b.
 785

786 Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song,
 787 Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient
 788 generative inference of large language models. *Advances in Neural Information Processing
 789 Systems*, 36:34661–34710, 2023.
 790

791 Junhao Zheng, Chengming Shi, Xidi Cai, Qiuke Li, Duzhen Zhang, Chenxing Li, Dong Yu, and
 792 Qianli Ma. Lifelong learning of large language model based agents: A roadmap. *arXiv preprint
 793 arXiv:2501.07278*, 2025.
 794

795 Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large
 796 language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial
 797 Intelligence*, volume 38, pages 19724–19731, 2024.
 798

799 Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar,
 800 and Bryan Catanzaro. Long-short transformer: Efficient transformers for language and vision.
 801 *Advances in neural information processing systems*, 34:17723–17736, 2021.
 802

803
 804 **A METHODOLOGY**
 805

806 **A.1 INFERENCE ALGORITHM.**
 807

808
 809 Algorithm 1 illustrates the attention computation with KV eviction using retention gates for a single
 decoding step. We mark the parts that are different from the standard attention computation in blue.

810
811 **Algorithm 1:** Attention computation with KV eviction (single decoding step)
812 **Input** :current hidden \mathbf{x}_t ; KV cache $\mathbf{K}_{t-1}, \mathbf{V}_{t-1}, \mathbf{B}_{t-1}$ indexed by S_{t-1} ; retention gate g
813 **Param** :capacity M ;
814 **Output** :attention output \mathbf{o}_t ; updated $(\mathbf{K}_t, \mathbf{V}_t, \mathbf{B}_t)$; updated index set S_t
815 // 1) Project to Q/K/V for the current token
816 1 $\mathbf{q}_t \leftarrow \mathbf{W}_Q \mathbf{x}_t$; $\mathbf{k}_t \leftarrow \mathbf{W}_K \mathbf{x}_t$; $\mathbf{v}_t \leftarrow \mathbf{W}_V \mathbf{x}_t$; $\beta_t = g(\mathbf{x}_t)$;
817 // 2) Append current token to the KV cache
818 2 $\mathbf{K}_t \leftarrow \mathbf{K}_{t-1} \cup \mathbf{k}_t$; $\mathbf{V}_t \leftarrow \mathbf{V}_{t-1} \cup \mathbf{v}_t$; $\mathbf{B}_t \leftarrow \mathbf{B}_{t-1} \cup \beta_t$; $S_t \leftarrow S_{t-1} \cup \{t\}$;
819 // 3) Compute attention over currently cached keys/values
820 (restricted to S_t)
821 3 $\mathbf{o}_t \leftarrow \text{FLASHATTN}(\mathbf{q}_t, \mathbf{K}_t, \mathbf{V}_t)$;
822 // 4) If capacity exceeded, evict the least important token
823 4 **while** $|S_t| > M$ **do**
824 5 $j_{\text{evic}} \leftarrow \arg \min \{\beta_j^{t-j} | j \in S_t\}$;
825 6 Remove $\mathbf{K}_t[j_{\text{evic}}], \mathbf{V}_t[j_{\text{evic}}], \mathbf{B}_t[j_{\text{evic}}]$;
826 7 $S_t \leftarrow S_t \setminus \{j_{\text{evic}}\}$;
827 **end**

828 **Positional Encoding in KV Cache Eviction.** Our retention mechanism is designed to be positional-
829 encoding agnostic and does not add any extra recency bias beyond what is already present in the
830 base model. The exponential decay in the retention-gated attention is a smooth approximation of
831 the decay process from 1 to 0 of the standard attention with eviction, not to encode the positional
832 information. Therefore, regardless of whether the base model uses absolute positions, RoPE, or no
833 positional encoding, that information is already folded into \mathbf{x} , \mathbf{q} , and \mathbf{k} in the standard attention (see
834 Eq 1).

835 Implementation-wise, when using RoPE, we follow prior work (R-KV, SnapKV) and cache post-
836 rotated keys, so the eviction is orthogonal to the positional embeddings used.

837

A.2 COMPLEXITY

839

840 **Memory efficiency.** Like other KV-eviction schemes, TRIM-KV uses a fixed-size cache with $\mathcal{O}(M)$
841 slots, independent of sequence length T . For each token (per head), it stores a single scalar retention
842 score β_i , adding only $\approx 1/d_h$ overhead, where d_h is the dimension of the key and vector states,
843 relative to the KV states, which is negligible in practice. Unlike R-KV (Cai et al., 2025), TRIM-KV
844 does not store queries.

845

Method	Context Length	Gen Length	Batch Size	Throughput (tok/sec)	Decode Time (s)
FullKV				68.44	59.84
SeerAttn-R	32786	1024	4	68.93	59.41
SnapKV				124.67	33.00
TRIM-KV				130.48	31.39
FullKV				114.39	35.8
SeerAttn-R	16378	1024	4	100.45	40.77
SnapKV				153.21	26.73
TRIM-KV				151.04	27.11
FullKV				138.97	58.94
SeerAttn-R	16378	1024	8	139.34	58.78
SnapKV				244.60	33.49
TRIM-KV				279.90	29.26

857

858 Table 5: Throughput and decoding time comparisons of different KV cache methods on a single
859 H200 GPU. The memory budget M is 1024.

860

861

862 **Computational efficiency.** For each generated token, TRIM-KV computes a scalar retention
863 score β_i via a lightweight MLP that can be fused with QKV projections; scores are cached and not
864 recomputed each step. During compression, it applies a temporal discount (elementwise power) and
865 evicts the argmin; both costs only $\mathcal{O}(M)$. This is cheaper than heuristics like R-KV, which require

864 key-key similarity scoring over the cache. Table 5 reports throughput and latency: at 32K context,
 865 TRIM-KV achieves $\sim 2 \times$ higher decoding throughput than full-cache decoding and even faster than
 866 SnapKV, a purely heuristic method. SeerAttn-R does not provide any computation advantage over
 867 full cache model.

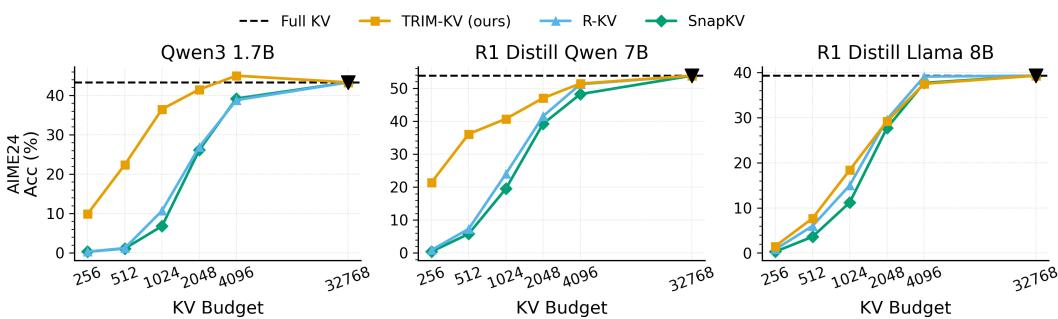
870 B ADDITIONAL EXPERIMENTS

872 B.1 LONG GENERATION EVALUATION

874 We provide more comprehensive experiment details in this section.

875 **Experiment settings.** For the training, we set the maximum training length to be 16384. We train
 876 the retention gates with a learning rate of 2×10^{-4} and a weight decay of 0.01. For math reasoning
 877 tasks, we follow SeerAttn-R (Gao et al., 2025) that uses OpenR1-Math-220K (Hugging Face) dataset,
 878 which has 564M tokens for training. During training, we use a batch size of 1 for each GPU, and
 879 gradient accumulation is set to 4. Other hyperparameters are set to the default in Huggingface Trainer.
 880 All training is conducted on 4 H100 GPUs.

881 **Benchmarks.** AIME24 (Art of Problem Solving, 2024), GSM8K (Cobbe et al., 2021), and MATH-
 882 500 (Hendrycks et al., 2021) are standard math reasoning benchmarks. LongProc (Ye et al., 2025)
 883 is a long-context benchmark of six procedural-generation tasks that require integrating dispersed
 884 information and producing structured long-form outputs (up to ~ 8 K tokens)—from extracting
 885 tables from HTML to executing multi-step search to build feasible travel itineraries. The suite spans
 886 varied access patterns (sequential vs. targeted retrieval), deductive reasoning demands, and search
 887 execution, enabling stress tests of long-range coherence and procedure following. Each task includes
 888 deterministic solution procedures and structured outputs, allowing rule-based evaluation (row-level F1
 889 for HTML \rightarrow TSV, unit tests for pseudocode \rightarrow code, exact-match traces for Path/ToM, and validators
 890 for Countdown/Travel). To probe generation length, we use three difficulty tiers targeting 0.5K, 2K,
 891 and 8K output tokens.



903 Figure 6: Patero frontiers of competing algorithms with different budgets on AIME24.
 904

905 **Math reasoning results.** Figure 6 reports AIME24 performance for Qwen3-1.7B and DeepSeek-R1-
 906 Distill variants. Across both families, TRIM-KV consistently outperforms eviction baselines. The
 907 gains over heuristic baselines are smaller on DeepSeek-R1-Distill-Llama-8B, which we hypothesize
 908 reflects lower attention sparsity in this model compared to the Qwen3 family.

910 **Comparison to a query-agnostic baseline.** We provide a comparison to KeyDiff (Park et al., 2025),
 911 a query-agnostic baseline that only considers a key diversity for eviction. The result in Figure 7 shows
 912 that the performance of KeyDiff is significantly worse than that of other baselines. Note that R-KV
 913 can be considered as a generalization of KeyDiff since it considers both key diversity and attention
 914 scores for eviction heuristics.

915 **Results on LongProc.** Table 6 reports KV-eviction results on long procedure-generation tasks.
 916 Across tasks and budgets, TRIM-KV achieves the best performance, and it even surpasses the
 917 full-cache baseline on COUNTDOWN (0.5K/2K) and HTML TO TSV (0.5K). Under tighter memory
 918 budgets, its margin over heuristic baselines widens.

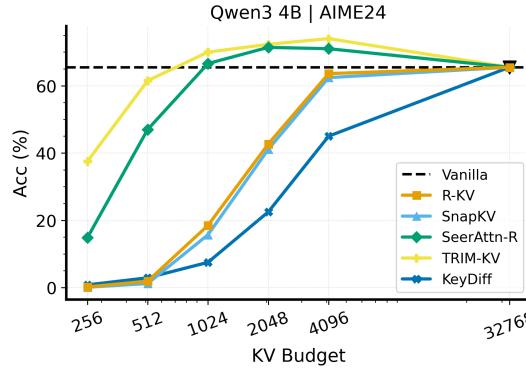


Figure 7: Comparison to KeyDiff, a query-agnostic baseline.

Method _{KV budget}	HTML to TSV			Thought of Mind			Travel Planning	
	0.5k	2k	8k	0.5k	2k	8k	2k	8k
FullKV	49.0	41.6	13.9	33.0	10.5	0.0	0.0	0.0
SnapKV ₈₁₉₂	37.1	9.3	0.1	26.0	7.0	0.0	0.0	0.0
H2O ₈₁₉₂	28.3	6.4	0.4	38.0	7.0	0.0	0.0	0.0
R-KV ₈₁₉₂	38.0	7.1	0.5	26.0	7.5	0.0	0.0	0.0
TRIM-KV₈₁₉₂	58.2	36.0	12.5	32.5	10.5	0.0	0.0	0.0
StreamingLLM ₂₀₄₈	1.2	0.0	0.0	2.0	0.0	0.0	0.0	0.0
SnapKV ₂₀₄₈	1.5	0.2	0.0	15.0	0.0	0.0	0.0	0.0
H2O ₂₀₄₈	0.4	0.8	0.0	7.6	0.0	0.0	0.0	0.0
R-KV ₂₀₄₈	1.6	0.1	0.0	3.0	0.0	0.0	0.0	0.0
TRIM-KV₂₀₄₈	34.6	7.1	0.3	17.5	0.5	0.0	0.0	0.0

Table 6: Results of Qwen3-4B across LongProc tasks: F1-score for HTML to TSV task and accuracies (%) for the remaining tasks. Best per task column in bold.

B.2 LONG-CONTEXT EVALUATION

Method _{KV budget}	Overall	Multi	Knowledge	SS-User	SS-Pref	SS-Assist	Temporal
Full KV ₁₃₁₀₇₂	49.4	25.6	68.0	62.9	93.3	85.7	30.1
StreamingLLM ₃₂₇₆₈	27.8	15.8	50.0	32.9	56.7	33.9	15.0
SnapKV ₃₂₇₆₈	27.6	15.8	42.3	24.3	73.3	28.6	21.8
TRIM-KV₃₂₇₆₈	48.2	23.3	68.0	58.6	80.0	85.5	32.3
StreamingLLM ₁₆₃₈₄	19.0	12.8	35.9	24.3	26.7	17.9	11.3
SnapKV ₁₆₃₈₄	18.2	9.0	25.6	17.1	70.0	12.5	14.3
TRIM-KV₁₆₃₈₄	42.6	21.8	62.8	42.9	80.0	69.6	31.6
StreamingLLM ₁₂₂₈₈	17.2	13.5	32.1	20.0	33.3	14.3	8.3
SnapKV ₁₂₂₈₈	17.0	9.8	20.5	14.3	73.3	12.5	12.8
TRIM-KV₁₂₂₈₈	36.8	19.6	59.0	35.7	66.7	50.0	29.3
StreamingLLM ₈₁₉₂	13.0	9.0	25.6	15.7	16.7	10.7	8.3
SnapKV ₈₁₉₂	15.8	9.8	19.2	12.9	70.0	7.1	12.8
TRIM-KV₈₁₉₂	33.2	15.0	55.1	34.3	70.0	46.4	24.1
StreamingLLM ₄₀₉₆	10.2	9.8	14.1	14.3	16.7	7.1	6.0
SnapKV ₄₀₉₆	13.8	11.3	14.1	14.3	56.7	7.1	9.0
TRIM-KV₄₀₉₆	30.2	12.8	35.9	24.3	80.0	46.4	29.3

Table 7: Results on LongMemEval_S: overall and partial accuracies (%).

Experimental settings. We adopt Qwen3-4B-Instruct (Qwen3, 2025) as the base model, which supports a context window of up to 256K tokens. Retention gates are trained on a mixture of SynthLong-32K (Lazarevich et al., 2025), BookSum (Kryściński et al., 2021), and Buddhi (Singhal, 2024), covering sequence lengths from 32K to 128K tokens. We shuffle the combined corpus and train for 10,000 steps (i.e., 10,000 randomly sampled examples), with a maximum training sequence length of 128K and memory capacity $M = 4096$. All other settings follow Section 5.1.

Benchmark Dataset. We evaluate chat-assistant capabilities under strict memory budgets using LongMemEval_S (Wu et al., 2024a). This subset provides contexts up to 123k tokens (measured with the Qwen3 tokenizer) and includes six question types that probe long-term memory: *single-session*,

972 *user* (SS-User) and *single-session-assistant* (SS-Assist), which test recall of facts stated by the user or
 973 assistant within a session; *single-session-preference* (SS-Pref), which requires personalized responses
 974 from stored personal information; *multi-session* (Multi), which aggregates or compares information
 975 across sessions; *knowledge-update* (Knowledge), which tracks and answers with the most recent,
 976 changed user information; *temporal-reasoning* (Temporal), which reasons over timestamps and time
 977 references.

978 To evaluate KV-cache eviction methods on this benchmark, we follow the multi-turn, multi-session
 979 protocol of (Li et al., 2024b). Specifically, before each query, the eviction-based model must
 980 compress the accumulated dialogue into a fixed-size, reusable KV cache—mirroring real-world
 981 assistants that maintain state across turns and sessions under strict memory budgets. We use Qwen3-
 982 4B-Instruct (Qwen3, 2025) to assess whether model outputs match the ground-truth responses.

983 **Results.** The results in Table 7 show that our method outperforms baseline eviction strategies by a
 984 significant margin. Especially, TRIM-KV can match the performance of a full cache while using just
 985 25% of the KV budget.

992 B.3 CHUNKED-PREFILL EVALUATION

996 In this section, we evaluate our method in the chunked-prefill setting (Huang et al., 2024), which
 997 enables long-context inference without exceeding memory limits. In this framework, long prompts
 998 are split into multiple chunks; the model computes the KV cache for each chunk sequentially and
 999 compresses the cache whenever it surpasses the memory budget. We compare our method against
 1000 LocRet (Huang et al., 2024), a learnable KV eviction baseline that also assigns token-importance
 1001 scores for eviction. Following the experimental setup of LocRet, we evaluate TRIM-KV on the
 1002 LongBench (Bai et al., 2024) and LongBench-V2 (Bai et al., 2025) benchmarks. For LongBench-V2,
 1003 we restrict evaluation to examples with context length below 128K tokens, matching the maximum
 1004 context length advertised for Phi3-mini-128K. To train TRIM-KV, we use only LongAlpaca (Chen
 1005 et al., 2023), as in LocRet, to ensure that improvements are not attributable to differences in training
 1006 data. In this setting, we set $M = 2048$, and keep all other hyperparameters identical to those in
 1007 Section 5.2.

1008 For a fair comparison, we adopt the hyperparameters used by LocRet: the chunk size is set to 3072
 1009 and the budget size to 6000. We evaluate performance on Phi-3-mini-128K (Abdin et al., 2024),
 1010 reproducing Table 6 from the LocRet paper. Since the original LongBench experiments have not
 1011 been released, we use the default chat template of Phi-3-mini-128K for all runs. Table 8 and 9
 1012 report the results for LongBench (Bai et al., 2024) and LongBench-V2 (Bai et al., 2025) benchmarks,
 1013 respectively. We observe discrepancies in the full-KV performance, likely due to differences in chat
 1014 templates.

1015 Overall, TRIM-KV remains highly effective in the chunked-prefill setting. On LongBench, TRIM-KV
 1016 nearly matches full-KV performance, whereas LocRet exhibits a 4.82% drop relative to full-KV
 1017 cache inference. Notably, on a more challenging benchmark, LongBench-V2, we even surpass the
 1018 performance of the full KV cache by 6.5%.

1019 Conceptually, LocRet is designed for chunk prefilling in long-context inference, while our method
 1020 is general. The training paradigms also differ: LocRet predicts attention logits independently for
 1021 each KV head, whereas we train retention gates end-to-end, allowing importance scores across heads
 1022 to jointly optimize the eviction strategy. At inference time, LocRet further relies on a hand-crafted
 1023 sliding window to preserve the most recent tokens from the latest chunk, and they show that removing
 1024 this heuristic substantially degrades performance. In contrast, our method requires no such manually
 1025 designed mechanism: sliding-window-like behavior emerges automatically from the learned policy
 1026 when beneficial, and some heads (e.g., layer 15, head 2 in Figure 4) do not exhibit sliding-window
 1027 patterns at all.

1026	Method	2wikiqa	gov_report	hotpotqa	lcc	multi_news	mfldqa	musique	narrativeqa	pssg_count	pssg_retrv	qasper	qnsum	repobench-p	samsum	triviaqa	Δ (%)
1027	Full KV*	33.37	33.35	43.06	51.86	26.57	49.82	19.82	18.21	2.97	93.50	41.07	19.51	58.02	23.15	86.38	–
1028	LocRet*	35.93	33.46	48.70	52.61	26.41	52.77	25.12	24.56	3.00	83.00	40.17	23.35	57.16	26.37	82.39	–
1029	Full KV	37.01	33.35	53.35	33.35	26.02	54.45	25.90	26.17	5.00	96.50	40.18	24.08	34.08	38.77	85.50	0.00
1030	LocRet	37.24	32.80	48.67	28.60	26.77	54.12	26.63	22.96	3.50	87.50	39.39	22.98	37.28	37.99	83.29	-4.82
1031	TRIM-KV	36.65	33.37	54.78	33.08	26.02	53.25	25.39	25.00	5.00	94.50	40.17	23.59	37.46	36.82	83.38	-0.64

Table 8: Performance on long-context tasks with Phi3-mini-128K on the LongBench benchmark, including average relative change (Avg. Δ) compared to Full KV. * indicates that the numbers are reported from (Huang et al., 2024, Table 6).

1033	Method	Acc. Short	Acc. Medium	Acc. Easy	Acc. Hard	Avg. Acc	Δ (%)
1038	Full KV	33.71	18.60	34.44	25.86	28.79	0.00
1039	LocRet	32.02	19.78	26.67	28.74	28.03	-2.64
1040	TRIM-KV	35.39	20.93	34.44	28.74	30.68	+6.56

Table 9: Performance on long-context tasks of KV eviction methods with Phi3-mini-128K on the LongBench-V2 benchmark, including average relative change (Avg. Δ) compared to Full KV.

B.4 ADDITIONAL ABLATION STUDIES

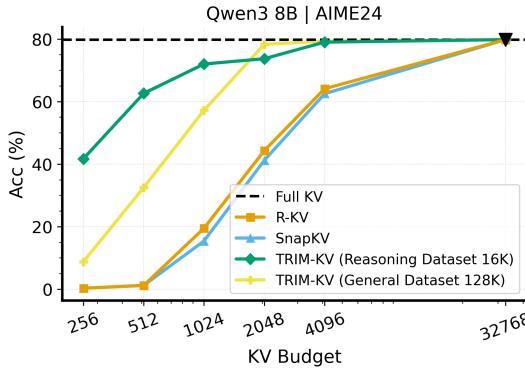


Figure 8: Generation Ablation.

Ablation on training datasets. In Section 5.1, we train the retention gates on a reasoning dataset—OpenR1-Math (Hugging Face)—and evaluate on AIME24, MATH-500, and GSM8K. This follows standard practice and matches the setting of (Gao et al., 2025), ensuring a fair comparison. To assess cross-domain generalization, we instead train the gates on general long-context datasets (SynthLong, BookSum, Buddhi), similar to Section 5.2, and then evaluate on math reasoning benchmarks to test whether retention scores learned from general data transfer to long chain-of-thought tasks. As shown in Figure 8, gates trained on general datasets generalize well and even surpass math-specific training at a 2048 KV budget. However, their performance degrades more quickly under tighter KV budgets. Overall, these results are promising and suggest that scaling the training of the retention gates by combining all datasets could yield further improvements.

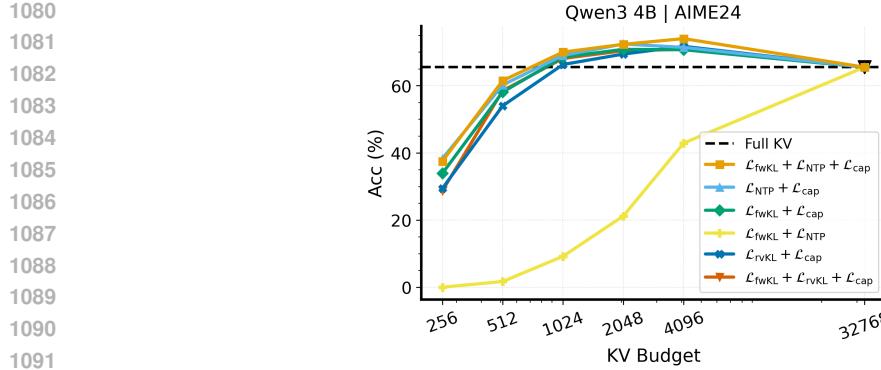


Figure 9: Ablating different objective components.

Ablation for the objective function. We ablate the objective by training the Qwen3-4B retention gates with different loss combinations and report AIME24 pass@1 at a 4096-token KV budget in Figure 9. Here, we consider both forward KL divergence and reversed KL divergence for distillation loss. Generally, all distillation losses perform well on their own. However, reversed KL underperforms when compared to forward KL. Both show benefits in combination with next token prediction. The memory capacity loss is essential for compression, and removing it leads to a sharp drop.

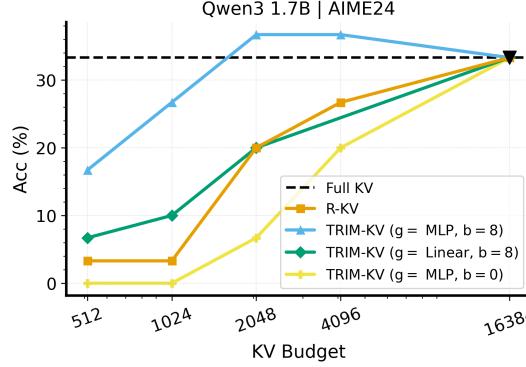
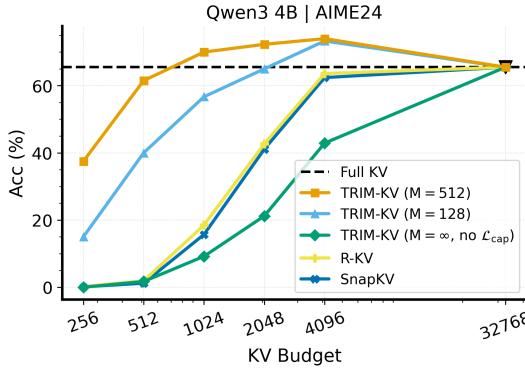


Figure 10: Ablating the retention gate's architecture.

Ablation for the retention gate's architecture. We evaluate several retention-gate architectures and report the performance of Qwen3 1.7B on AIME24 in Figure 10. Due to computational constraints, this ablation uses greedy decoding. For the MLP gate, we use a single-hidden-layer MLP with width 512. We find that the MLP gate outperforms a simple linear projection, and that a large positive initial bias is crucial for stable training by keeping the gate's output nearly 1 at initialization to ensure minimal early forgetting.

Figure 11: Ablating the training memory capacity M .

Ablation on training memory capacity M . We evaluate multiple settings of M in Figure 11. With $M = \infty$, there is no capacity penalty, which hurts performance due to the absence of sparsity pressure. Setting $M = 128$ outperforms attention-guided heuristics but shows signs of over-optimizing for sparsity. In practice, we recommend choosing M to match the expected deployment-time memory budget.

C ADDITIONAL QUALITATIVE RESULTS

In this section, we provide more qualitative results to illustrate the eviction decisions made by TRIM-KV. All visualizations are from the first example in the AIME24 dataset. Please refer to Section 5.1.2 for discussions.

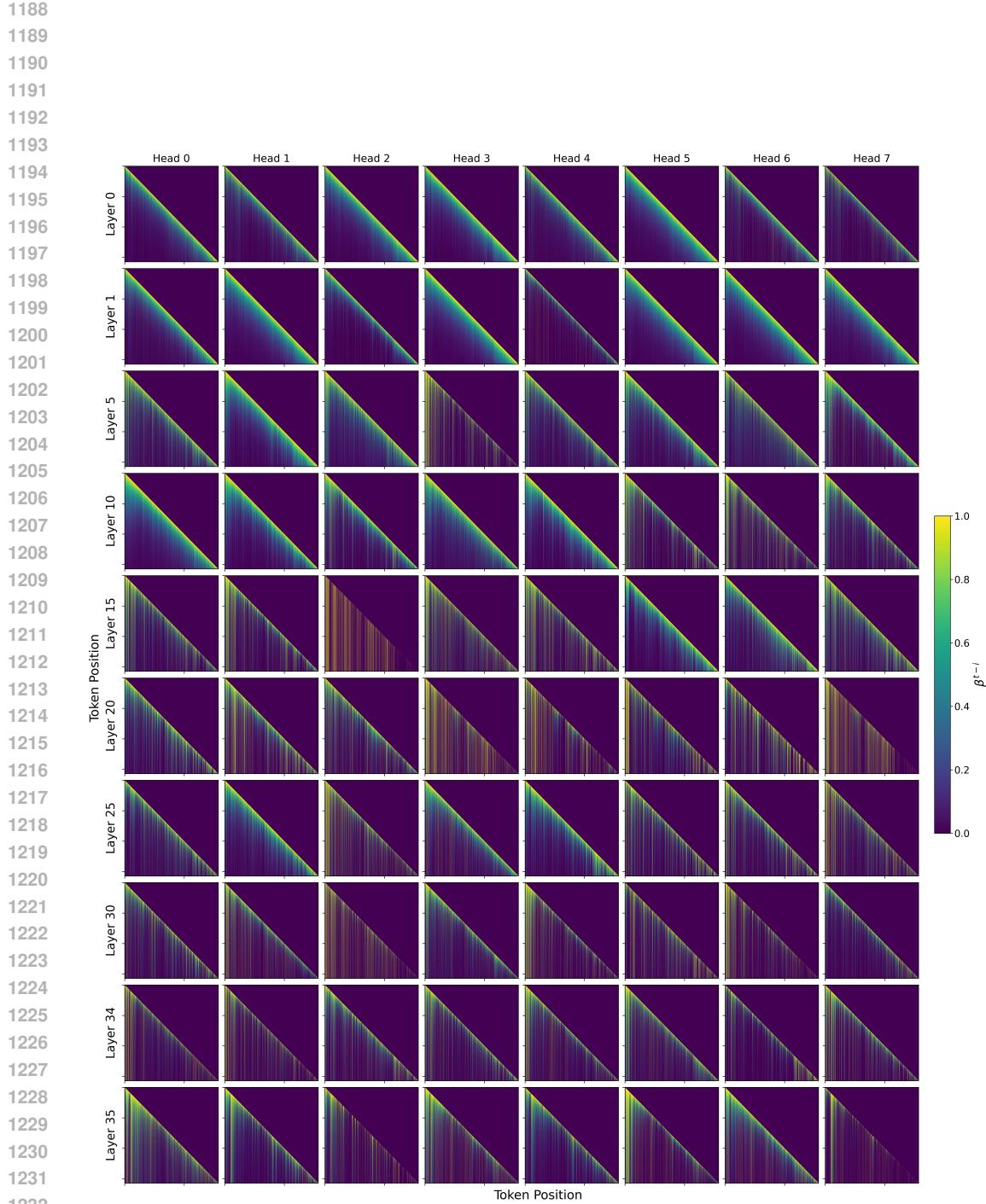


Figure 12: A visualization of token retention matrices of Qwen3-4B when answering a math question in the AIME24 dataset. Each subplot is a token retention matrix $\{\beta_i^{t-i}\}_i^t$ in a specific layer and head. **Observations:** earlier layers often exhibit sliding-window-like patterns, while later layers develop clearer functional specializations.

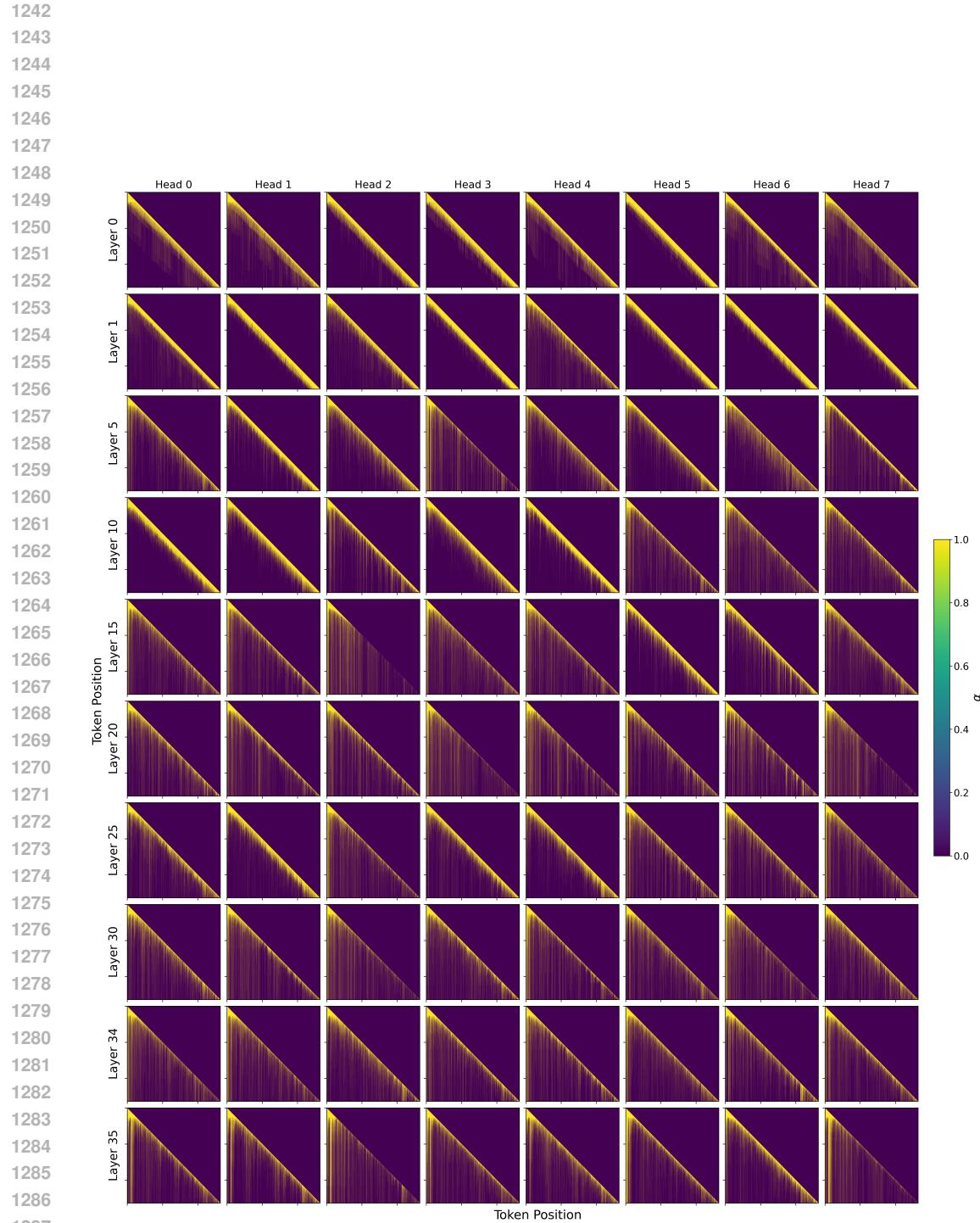


Figure 13: A visualization of evaporation decisions of Qwen3-4B when answering a math question in the AIME24 dataset. Each subplot is a matrix of evaporation decisions α_{ti} in a specific layer and head.

1350

1351

1352

1353

1354

1355

1356

1357

1358

1359

1360

1361

1362

1363

1364

1365

1366

1367

1368

1369

1370

1371

<dim_start>user You are given a math problem. Problem: Every morning Aya goes for a \$95-kilometer-long walk and stops at a coffee shop afterwards. When she walks at a constant speed of \$5 kilometers per hour, the walk takes her 4 hours, including \$15 minutes spent in the coffee shop. When she walks \$3+25 kilometers per hour, the walk takes her 2 hours and 24 minutes, including \$15 minutes spent in the coffee shop. Suppose Aya walks at \$5*sqrt(12)/5 kilometers per hour. Find the number of minutes the walk takes her, including the \$15 minutes spent in the coffee shop. You need to solve the problem step by step. First, you need to provide the chain-of-thought, then provide the final answer. Provide the final answer in the format: Final answer: <boxed></boxed></dim_end><dim_start><think> Okay, let's try to solve this problem. So, Aya walks 9 kilometers every morning, stops at a coffee shop, and we need to find out how long her walk takes when she walks at a speed of $s = 1/2$ km/h, including the time she spends in the coffee shop. First, let me parse the problem again. There are two scenarios given. In the first scenario, when she walks at a constant speed of s km/h, the total time for the walk (including time in the coffee shop) is 4 hours. In the second scenario, when she walks at $s + 2$ km/h, the total time is 2 hours and 24 minutes, again including 15 minutes in the coffee shop. We need to find the total time when she walks at $s + 1/2$ km/h. Wait, so the problem mentions that in both cases, the time spent in the coffee shop is 1 minute. So, the difference between the total time and the walking time is 1 minutes. Therefore, maybe we can set up equations based on the walking time and the total time. Let me think. Let me denote. Let me let t be the time spent in the coffee shop in minutes. Then, the total time is $t + 15$ minutes. So, the time spent walking is total time minus t minutes. Wait, but t is in minutes. So, maybe I need to convert t minutes to hours. Let me think. The problem expects the answer in hours? But no, the question says "number of minutes", so 204 minutes is correct. Therefore, I think the answer is 204. </think></dim_start></dim_end>

Figure 16: Visualization of token retention scores β_i from the retention gate in *layer 35, head 2* of the Qwen3-4B model. Brighter colors denote higher retention; gray indicates near-zero retention. **Observation:** Unlike *layer 15, head 2* and *layer 9, head 7*, this head assigns elevated retention to general-purpose tokens that support coherence and instruction following; for example, `\TeX` commands and the directive `boxed{}` receive high scores, while tokens associated with mathematical operations receive low retention.

1372

1373

1374

1375

1376

1377

1378

1379

1380

1381

1382

1383

1384

1385

1386

1387

1388

1389

1390

1391

1392

1393

1394

1395

1396

1397

1398

1399

1400

1401

1402

1403

<dim_start>user You are given a math problem. Problem: Every morning Aya goes for a \$95-kilometer-long walk and stops at a coffee shop afterwards. When she walks at a constant speed of \$5 kilometers per hour, the walk takes her 4 hours, including \$15 minutes spent in the coffee shop. When she walks \$3+25 kilometers per hour, the walk takes her 2 hours and 24 minutes, including \$15 minutes spent in the coffee shop. Suppose Aya walks at \$5*sqrt(12)/5 kilometers per hour. Find the number of minutes the walk takes her, including the \$15 minutes spent in the coffee shop. You need to solve the problem step by step. First, you need to provide the chain-of-thought, then provide the final answer. Provide the final answer in the format: Final answer: <boxed></boxed></dim_end><dim_start><think> Okay, let's try to solve this problem. So, Aya walks 9 kilometers every morning, stops at a coffee shop, and we need to find out how long her walk takes when she walks at a speed of $s + 1/2$ km/h, including the time she spends in the coffee shop. First, let me parse the problem again. There are two scenarios given. In the first scenario, when she walks at a constant speed of s km/h, the total time for the walk (including time in the coffee shop) is 4 hours. In the second scenario, when she walks at $s + 2$ km/h, the total time is 2 hours and 24 minutes, again including 15 minutes in the coffee shop. We need to find the total time when she walks at $s + 1/2$ km/h. Wait, so the problem mentions that in both cases, the time spent in the coffee shop is 1 minute. So, the difference between the total time and the walking time is 1 minutes. Therefore, maybe we can set up equations based on the walking time and the total time. Let me think. Let me denote. Let me let t be the time spent in the coffee shop in minutes. Then, the total time is $t + 15$ minutes. So, the time spent walking is total time minus t minutes. Wait, but t is in minutes. So, maybe I need to convert t minutes to hours. Let me think. The problem expects the answer in hours? But no, the question says "number of minutes", so 204 minutes is correct. Therefore, I think the answer is 204. </think></dim_start></dim_end>

Figure 17: Visualization of tokens retained in the *layer 0 head 3* of the KV cache after generation, where the KV budget is 256. The model is Qwen3-4B. Bold blue indicates tokens retained in the KV cache, where gray indicates evicted tokens.

So, we need to convert the total time into minutes. 3 hours is 180 minutes, plus 24 minutes is 204 minutes. Therefore, the answer is 204 minutes. Wait, but let me check again. Let me verify the steps again to make sure. First, solving for s : we had $4s + 8s - 45 = 0$. Then discriminant was $784 = \sqrt{784} = 28$. So $s = (s + 25)/8 = 20/8 = 2.5$. Correct. Then t was found as 24 minutes. Then, when speed is $s + 1/2 = 2.5 + 0.5 = 3$ km/h. Walking time is $9/3 = 3$ hours. Then total time is 3 hours + 24 minutes. 3 hours is 180 minutes, so total is $180 + 24 = 204$ minutes. So answer is 204. But let me check if there's another way to approach this, or if I made any miscalculations. Alternatively, maybe I can check the equations again. Let me check equation 2 with $s = 2.5$ and $t = 24$. Equation 2: $9(s + 2) + 160 = 9(2.5 + 2) + 160 = (14.5 + 0.4 = 2 + 0.4 = 2.4$ hours. Which is 14.4 hours + 24 minutes. Correct. So that checks out. So, the answer is 204 minutes. Therefore, the final answer is 204. But let me think again. The problem says "the number of minutes the walk takes her, including the 1 minutes spent in the coffee shop". So, the total time is walking time plus 1 minutes. So, when she walks at 3 km/h, the walking time is $9/3 = 3$ hours, which is 180 minutes. Then adding $t = 24$ minutes, total is $180 + 24 = 204$ minutes. So yes, that's correct. Alternatively, maybe the problem expects the answer in hours? But no, the question says "number of minutes", so 204 minutes is correct. Therefore, I think the answer is 204. </think></dim_start></dim_end>

To solve the problem, we are given two scenarios involving Aya's walking and coffee shop time, and we are to determine the total time (in minutes) she spends when walking at a speed of $s + Vrac(12)/5$ km/h, including the fixed \$15 minutes spent in the coffee shop. --- **Step 1:** Define Variables and Equations Let: s is s be Aya's walking speed in km/h. t is t be the time spent in the coffee shop in minutes. From the problem: *First scenario*: Walking at s $s + 1/2$ km/h, total time is 4 hours (including \$15 minutes in the coffee shop). $\$Vrac(3)(s) + Vrac(160) = 4$ *quad text{Equation 1}* $\$ -$ *Second scenario*: Walking at $s + 2$ km/h, total time is 2 hours and 24 minutes (i.e., 2.4 hours). $\$Vrac(9)(s + 2) + Vrac(160) = 2.4$ *quad text{Equation 2}* $\$ -$ **Step 2:** Eliminate $s + 15$ by Subtracting Equations Subtract Equation 2 from Equation 1: $\$ - Vrac(9)(s) + Vrac(160) + Vrac(16)(s) - Vrac(9)(s + 2) + Vrac(16)(s + 2) = 4 - 2.4$ *quad text{Equation 2} - Equation 1* $\$ -$ **Step 3:** Solve for s $s + 15 = 1.6$ *quad text{Equation 1}* $\$ -$ **Step 4:** Find $s + 15$ by Substituting $s = 2.5$ into Equation 1: $\$ - Vrac(9)(2.5) + Vrac(160) = 4 - 2.4$ *quad text{Equation 1}* $\$ -$ **Step 5:** Find $s + 15$ by Substituting $s = 2.5$ into Equation 2: $\$ - Vrac(9)(2.5 + 1/2) + Vrac(160) = 2.4$ *quad text{Equation 2}* $\$ -$ **Step 6:** Divide by $1/2$: $s + 15 = 2.4$ *quad text{Equation 2}* $\$ -$ **Step 7:** Solve for s : $s = 2.4 - 15 = 0.4$ *quad text{Equation 2}* $\$ -$ **Step 8:** Find $s + 15$ by Substituting $s = 0.4$ into Equation 1: $\$ - Vrac(9)(0.4) + Vrac(160) = 4 - 2.4$ *quad text{Equation 1}* $\$ -$ **Step 9:** Solve for t : $t = 204 - 180 = 24$ minutes. So answer is 204. But let me check again. The problem says "the number of minutes the walk takes her, including the 1 minutes spent in the coffee shop". So, the total time is walking time plus 1 minutes. So, when she walks at 3 km/h, the walking time is $9/3 = 3$ hours, which is 180 minutes. Then adding $t = 24$ minutes, total is $180 + 24 = 204$ minutes. So yes, that's correct. Alternatively, maybe the problem expects the answer in hours? But no, the question says "number of minutes", so 204 minutes is correct. Therefore, I think the answer is 204. </think></dim_start></dim_end>

So, we need to convert the total time into minutes. 3 hours is 180 minutes, plus 24 minutes is 204 minutes. Therefore, the answer is 204 minutes. Wait, but let me check again. Let me verify the steps again to make sure. First, solving for s : we had $4s + 8s - 45 = 0$. Then discriminant was $784 = \sqrt{784} = 28$. So $s = (s + 25)/8 = 20/8 = 2.5$. Correct. Then t was found as 24 minutes. Then, when speed is $s + 1/2 = 2.5 + 0.5 = 3$ km/h. Walking time is $9/3 = 3$ hours. Then total time is 3 hours + 24 minutes. 3 hours is 180 minutes, so total is $180 + 24 = 204$ minutes. So answer is 204. But let me check if there's another way to approach this, or if I made any miscalculations. Alternatively, maybe I can check the equations again. Let me check equation 2 with $s = 2.5$ and $t = 24$. Equation 2: $9(s + 2) + 160 = 9(2.5 + 2) + 160 = (14.5 + 0.4 = 2 + 0.4 = 2.4$ hours. Which is 14.4 hours + 24 minutes. Correct. So that checks out. So, the answer is 204 minutes. Therefore, the final answer is 204. But let me think again. The problem says "the number of minutes the walk takes her, including the 1 minutes spent in the coffee shop". So, the total time is walking time plus 1 minutes. So, when she walks at 3 km/h, the walking time is $9/3 = 3$ hours, which is 180 minutes. Then adding $t = 24$ minutes, total is $180 + 24 = 204$ minutes. So yes, that's correct. Alternatively, maybe the problem expects the answer in hours? But no, the question says "number of minutes", so 204 minutes is correct. Therefore, I think the answer is 204. </think></dim_start></dim_end>

To solve the problem, we are given two scenarios involving Aya's walking and coffee shop time, and we are to determine the total time (in minutes) she spends when walking at a speed of $s + Vrac(12)/5$ km/h, including the fixed \$15 minutes spent in the coffee shop. --- **Step 1:** Define Variables Let: s is s be Aya's walking speed in km/h. t is t be the time spent in the coffee shop in minutes. From the problem: *First scenario*: Walking at $s + 1/2$ km/h, total time is 4 hours (including \$15 minutes in the coffee shop). $\$Vrac(3)(s) + Vrac(160) = 4$ *quad text{Equation 1}* $\$ -$ *Second scenario*: Walking at $s + 2$ km/h, total time is 2 hours and 24 minutes (i.e., 2.4 hours). $\$Vrac(9)(s + 2) + Vrac(160) = 2.4$ *quad text{Equation 2}* $\$ -$ **Step 2:** Eliminate $s + 15$ by Subtracting Equations Subtract Equation 2 from Equation 1: $\$ - Vrac(9)(s + 2) + Vrac(160) - Vrac(9)(s + 2) - 160 = 1.6$ *quad text{Equation 2} - Equation 1* $\$ -$ **Step 3:** Solve for s : $s + 15 = 1.6$ *quad text{Equation 1}* $\$ -$ **Step 4:** Find $s + 15$ by Substituting $s = 2.5$ into Equation 1: $\$ - Vrac(9)(2.5) + Vrac(160) = 4 - 2.4$ *quad text{Equation 1}* $\$ -$ **Step 5:** Solve for t : $t = 204 - 180 = 24$ minutes. So answer is 204. But let me check again. The problem says "the number of minutes the walk takes her, including the 1 minutes spent in the coffee shop". So, the total time is walking time plus 1 minutes. So, when she walks at 3 km/h, the walking time is $9/3 = 3$ hours, which is 180 minutes. Then adding $t = 24$ minutes, total is $180 + 24 = 204$ minutes. So yes, that's correct. Alternatively, maybe the problem expects the answer in hours? But no, the question says "number of minutes", so 204 minutes is correct. Therefore, I think the answer is 204. </think></dim_start></dim_end>

To solve the problem, we are given two scenarios involving Aya's walking and coffee shop time, and we are to determine the total time (in minutes) she spends when walking at a speed of $s + Vrac(12)/5$ km/h, including the fixed \$15 minutes spent in the coffee shop. --- **Step 1:** Define Variables Let: s is s be Aya's walking speed in km/h. t is t be the time spent in the coffee shop in minutes. From the problem: *First scenario*: Walking at $s + 1/2$ km/h, total time is 4 hours (including \$15 minutes in the coffee shop). $\$Vrac(3)(s) + Vrac(160) = 4$ *quad text{Equation 1}* $\$ -$ *Second scenario*: Walking at $s + 2$ km/h, total time is 2 hours and 24 minutes (i.e., 2.4 hours). $\$Vrac(9)(s + 2) + Vrac(160) = 2.4$ *quad text{Equation 2}* $\$ -$ **Step 2:** Eliminate $s + 15$ by Subtracting Equations Subtract Equation 2 from Equation 1: $\$ - Vrac(9)(s + 2) + Vrac(160) - Vrac(9)(s + 2) - 160 = 1.6$ *quad text{Equation 2} - Equation 1* $\$ -$ **Step 3:** Solve for s : $s + 15 = 1.6$ *quad text{Equation 1}* $\$ -$ **Step 4:** Find $s + 15$ by Substituting $s = 2.5$ into Equation 1: $\$ - Vrac(9)(2.5) + Vrac(160) = 4 - 2.4$ *quad text{Equation 1}* $\$ -$ **Step 5:** Solve for t : $t = 204 - 180 = 24$ minutes. So answer is 204. But let me check again. The problem says "the number of minutes the walk takes her, including the 1 minutes spent in the coffee shop". So, the total time is walking time plus 1 minutes. So, when she walks at 3 km/h, the walking time is $9/3 = 3$ hours, which is 180 minutes. Then adding $t = 24$ minutes, total is $180 + 24 = 204$ minutes. So yes, that's correct. Alternatively, maybe the problem expects the answer in hours? But no, the question says "number of minutes", so 204 minutes is correct. Therefore, I think the answer is 204. </think></dim_start></dim_end>

To solve the problem, we are given two scenarios involving Aya's walking and coffee shop time, and we are to determine the total time (in minutes) she spends when walking at a speed of $s + Vrac(12)/5$ km/h, including the fixed \$15 minutes spent in the coffee shop. --- **Step 1:** Define Variables Let: s is s be Aya's walking speed in km/h. t is t be the time spent in the coffee shop in minutes. From the problem: *First scenario*: Walking at $s + 1/2$ km/h, total time is 4 hours (including \$15 minutes in the coffee shop). $\$Vrac(3)(s) + Vrac(160) = 4$ *quad text{Equation 1}* $\$ -$ *Second scenario*: Walking at $s + 2$ km/h, total time is 2 hours and 24 minutes (i.e., 2.4 hours). $\$Vrac(9)(s + 2) + Vrac(160) = 2.4$ *quad text{Equation 2}* $\$ -$ **Step 2:** Eliminate $s + 15$ by Subtracting Equations Subtract Equation 2 from Equation 1: $\$ - Vrac(9)(s + 2) + Vrac(160) - Vrac(9)(s + 2) - 160 = 1.6$ *quad text{Equation 2} - Equation 1* $\$ -$ **Step 3:** Solve for s : $s + 15 = 1.6$ *quad text{Equation 1}* $\$ -$ **Step 4:** Find $s + 15$ by Substituting $s = 2.5$ into Equation 1: $\$ - Vrac(9)(2.5) + Vrac(160) = 4 - 2.4$ *quad text{Equation 1}* $\$ -$ **Step 5:** Solve for t : $t = 204 - 180 = 24$ minutes. So answer is 204. But let me check again. The problem says "the number of minutes the walk takes her, including the 1 minutes spent in the coffee shop". So, the total time is walking time plus 1 minutes. So, when she walks at 3 km/h, the walking time is $9/3 = 3$ hours, which is 180 minutes. Then adding $t = 24$ minutes, total is $180 + 24 = 204$ minutes. So yes, that's correct. Alternatively, maybe the problem expects the answer in hours? But no, the question says "number of minutes", so 204 minutes is correct. Therefore, I think the answer is 204. </think></dim_start></dim_end>

To solve the problem, we are given two scenarios involving Aya's walking and coffee shop time, and we are to determine the total time (in minutes) she spends when walking at a speed of $s + Vrac(12)/5$ km/h, including the fixed \$15 minutes spent in the coffee shop. --- **Step 1:** Define Variables Let: s is s be Aya's walking speed in km/h. t is t be the time spent in the coffee shop in minutes. From the problem: *First scenario*: Walking at $s + 1/2$ km/h, total time is 4 hours (including \$15 minutes in the coffee shop). $\$Vrac(3)(s) + Vrac(160) = 4$ *quad text{Equation 1}* $\$ -$ *Second scenario*: Walking at $s + 2$ km/h, total time is 2 hours and 24 minutes (i.e., 2.4 hours). $\$Vrac(9)(s + 2) + Vrac(160) = 2.4$ *quad text{Equation 2}* $\$ -$ **Step 2:** Eliminate $s + 15$ by Subtracting Equations Subtract Equation 2 from Equation 1: $\$ - Vrac(9)(s + 2) + Vrac(160) - Vrac(9)(s + 2) - 160 = 1.6$ *quad text{Equation 2} - Equation 1* $\$ -$ **Step 3:** Solve for s : $s + 15 = 1.6$ *quad text{Equation 1}* $\$ -$ **Step 4:** Find $s + 15$ by Substituting $s = 2.5$ into Equation 1: $\$ - Vrac(9)(2.5) + Vrac(160) = 4 - 2.4$ *quad text{Equation 1}* $\$ -$ **Step 5:** Solve for t : $t = 204 - 180 = 24$ minutes. So answer is 204. But let me check again. The problem says "the number of minutes the walk takes her, including the 1 minutes spent in the coffee shop". So, the total time is walking time plus 1 minutes. So, when she walks at 3 km/h, the walking time is $9/3 = 3$ hours, which is 180 minutes. Then adding $t = 24$ minutes, total is $180 + 24 = 204$ minutes. So yes, that's correct. Alternatively, maybe the problem expects the answer in hours? But no, the question says "number of minutes", so 204 minutes is correct. Therefore, I think the answer is 204. </think></dim_start></dim_end>

To solve the problem, we are given two scenarios involving Aya's walking and coffee shop time, and we are to determine the total time (in minutes) she spends when walking at a speed of $s + Vrac(12)/5$ km/h, including the fixed \$15 minutes spent in the coffee shop. --- **Step 1:** Define Variables Let: s is s be Aya's walking speed in km/h. t is t be the time spent in the coffee shop in minutes. From the problem: *First scenario*: Walking at $s + 1/2$ km/h, total time is 4 hours (including \$15 minutes in the coffee shop). $\$Vrac(3)(s) + Vrac(160) = 4$ *quad text{Equation 1}* $\$ -$ *Second scenario*: Walking at $s + 2$ km/h, total time is 2 hours and 24 minutes (i.e., 2.4 hours). $\$Vrac(9)(s + 2) + Vrac(160) = 2.4$ *quad text{Equation 2}* $\$ -$ **Step 2:** Eliminate $s + 15$ by Subtracting Equations Subtract Equation 2 from Equation 1: $\$ - Vrac(9)(s + 2) + Vrac(160) - Vrac(9)(s + 2) - 160 = 1.6$ *quad text{Equation 2} - Equation 1* $\$ -$ **Step 3:** Solve for s : $s + 15 = 1.6$ *quad text{Equation 1}* $\$ -$ **Step 4:** Find $s + 15$ by Substituting $s = 2.5$ into Equation 1: $\$ - Vrac(9)(2.5) + Vrac(160) = 4 - 2.4$ *quad text{Equation 1}* $\$ -$ **Step 5:** Solve for t : $t = 204 - 180 = 24$ minutes. So answer is 204. But let me check again. The problem says "the number of minutes the walk takes her, including the 1 minutes spent in the coffee shop". So, the total time is walking time plus 1 minutes. So, when she walks at 3 km/h, the walking time is $9/3 = 3$ hours, which is 180 minutes. Then adding $t = 24$ minutes, total is $180 + 24 = 204$ minutes. So yes, that's correct. Alternatively, maybe the problem expects the answer in hours? But no, the question says "number of minutes", so 204 minutes is correct. Therefore, I think the answer is 204. </think></dim_start></dim_end>

To solve the problem, we are given two scenarios involving Aya's walking and coffee shop time, and we are to determine the total time (in minutes) she spends when walking at a speed of $s + Vrac(12)/5$ km/h, including the fixed \$15 minutes spent in the coffee shop. --- **Step 1:** Define Variables Let: s is s be Aya's walking speed in km/h. t is t be the time spent in the coffee shop in minutes. From the problem: *First scenario*: Walking at $s + 1/2$ km/h, total time is 4 hours (including \$15 minutes in the coffee shop). $\$Vrac(3)(s) + Vrac(160) = 4$ *quad text{Equation 1}* $\$ -$ *Second scenario*: Walking at $s + 2$ km/h, total time is 2 hours and 24 minutes (i.e., 2.4 hours). $\$Vrac(9)(s + 2) + Vrac(160) = 2.4$ *quad text{Equation 2}* $\$ -$ **Step 2:** Eliminate $s + 15$ by Subtracting Equations Subtract Equation 2 from Equation 1: $\$ - Vrac(9)(s + 2) + Vrac(160) - Vrac(9)(s + 2) - 160 = 1.6$ *quad text{Equation 2} - Equation 1* $\$ -$ **Step 3:** Solve for s : $s + 15 = 1.6$ *quad text{Equation 1}* $\$ -$ **Step 4:** Find $s + 15$ by Substituting $s = 2.5$ into Equation 1: $\$ - Vrac(9)(2.5) + Vrac(160) = 4 - 2.4$ *quad text{Equation 1}* $\$ -$ **Step 5:** Solve for t : $t = 204 - 180 = 24$ minutes. So answer is 204. But let me check again. The problem says "the number of minutes the walk takes her, including the 1 minutes spent in the coffee shop". So, the total time is walking time plus 1 minutes. So, when she walks at 3 km/h, the walking time is $9/3 = 3$ hours, which is 180 minutes. Then adding $t = 24$ minutes, total is $180 + 24 = 204$ minutes. So yes, that's correct. Alternatively, maybe the problem expects the answer in hours? But no, the question says "number of minutes", so 204 minutes is correct. Therefore, I think the answer is 204. </think></dim_start></dim_end>

To solve the problem, we are given two scenarios involving Aya's walking and coffee shop time, and we are to determine the total time (in minutes) she spends when walking at a speed of $s + Vrac(12)/5$ km/h, including the fixed \$15 minutes spent in the coffee shop. --- **Step 1:** Define Variables Let: s is s be Aya's walking speed in km/h. t is t be the time spent in the coffee shop in minutes. From the problem: *First scenario*: Walking at $s + 1/2$ km/h, total time is 4 hours (including \$15 minutes in the coffee shop). $\$Vrac(3)(s) + Vrac(160) = 4$ *quad text{Equation 1}* $\$ -$ *Second scenario*: Walking at $s + 2$ km/h, total time is 2 hours and 24 minutes (i.e., 2.4 hours). $\$Vrac(9)(s + 2) + Vrac(160) = 2.4$ *quad text{Equation 2}* $\$ -$ **Step 2:** Eliminate $s + 15$ by Subtracting Equations Subtract Equation 2 from Equation 1: $\$ - Vrac(9)(s + 2) + Vrac(160) - Vrac(9)(s + 2) - 160 = 1.6$ *quad text{Equation 2} - Equation 1* $\$ -$ **Step 3:** Solve for s : $s + 15 = 1.6$ *quad text{Equation 1}* $\$ -$ **Step 4:** Find $s + 15$ by Substituting $s = 2.5$ into Equation 1: $\$ - Vrac(9)(2.$

1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533

Figure 22: Visualization of tokens retained in the *layer 30 head 4* of the KV cache after generation, where the KV budget is 256. The model is Qwen3-4B. Bold blue indicates tokens retained in the KV cache, where gray indicates evicted tokens. **Observations:** This head mostly keeps period tokens. This suggests that this head may implicitly perform gist tokens that summarize information from the previous sentence. This contrasts with recent trends that advocate for saving a chunk of tokens (Yuan et al., 2025; Gao et al., 2025). Our results suggest that it can be more budget-efficient to save individual tokens because they already capture contextual information.

lin_start1>say You are given a math problem. Problem: **Every morning Aya goes for a 90-minute-long walk and stops at a coffee shop afterwards. When she walks at a constant speed of 5 km/h, she takes 4 hours, including 8 minutes spent in the coffee shop. When she walks at a constant speed of 2 km/h, she takes 4 hours, including 8 minutes spent in the coffee shop. Find the number of minutes the walk takes her, including the 8 minutes spent in the coffee shop.**

You need to solve the problem step by step. First, you need to provide the **chain-of-thought**, then provide the **final answer**. Provide the final answer in the form of **Final answer :boxed{<min>}**

Final answer :boxed{24}

lin_start1>say You are given a math problem. Problem: **Every morning Aya goes for a 90-minute-long walk and stops at a coffee shop afterwards. When she walks at a constant speed of 5 km/h, she takes 4 hours, including 8 minutes spent in the coffee shop. When she walks at a constant speed of 2 km/h, she takes 4 hours, including 8 minutes spent in the coffee shop. Suppose Aya walks at s km/h for t hours. Find the number of minutes the walk takes her, including the 8 minutes spent in the coffee shop.**

You need to solve the problem step by step. First, you need to provide the **chain-of-thought**, then provide the **final answer**. Provide the final answer in the form of **Final answer :boxed{<min>}**

Final answer :boxed{24}

lin_start1>say You are given a math problem. Problem: **Aya walks 9 kilometers every morning, stops at a coffee shop, and we need to find out how long her walk takes when she walks at a speed of s km/h, including the time she spends in the coffee shop.** First, let me parse the problem again. There are two **scenarios** given. In the first scenario, when she walks at a constant speed of s km/h, the total time for the walk (including time in the coffee shop) is 4 hours. In the second scenario, when she walks at $s = 2$ km/h, the total time is 2 hours and 24 minutes, again including 8 minutes in the coffee shop. We need to find the total time when she walks at $s = 2$ km/h. Wait, so the problem mentions that in both cases, the time spent in the coffee shop is 1 minute. So, the difference between the total time and the walking time is 1 minutes. Therefore, maybe we can set up equations based on the walking time and the total time. Let me think. Let me denote: Let me define variables. Let me let t be the time spent in the coffee shop in minutes. But maybe I need to convert everything to hours or minutes? Let me see. First scenario: walking speed = s km/h. The total time is 4 hours. So, the time spent walking is total time minus 1 minutes. Wait, but t is in minutes. So, maybe I need to convert 1 minutes to hours. Let me think. Alternatively, maybe I can write equations for the walking time. Let me think. The time taken to walk 9 km at speed s is $9/s$ hours. Then, the total time including the coffee shop is $9/s + 1/60$ hours. Wait, but the problem says that when she walks at s km/h, she walks 9 km in 4 hours, including 8 minutes in the coffee shop. So, that total time is 4 hours. Therefore **Time walking + time in coffee shop = total time**. So, including walking is $9/s$ hours. Time in coffee shop is 1 minutes, which is $1/60$ hours. Therefore, $9/s + 1/60 = 4$. **Equation 1**. Similarly, in the second scenario, when she walks at $s = 2$ km/h, the time walking is $9/2 + 2/60$ hours. Then, the total time is 2 hours and 24 minutes. Let me convert 2 hours and 24 minutes to 2 hours + 24 minutes. 24 minutes = $24/60$ hours, so total time is $9/2 + 24/60$ hours. Therefore, $9/2 + 24/60 = 2 + 2/60$. **Equation 2**. Now we have two equations with one variable: s . Let's use **substitution**. We can solve for t , and then use those to find the total time. Let me solve for s in **Equation 1** and then substitute it in **Equation 2**. Let me write the equations again: **Equation 1** $9/s + 1/60 = 4$. **Equation 2** $9/(2s) + 24/60 = 2 + 2/60$. **Step 1** Let me subtract **Equation 2** from **Equation 1**. The $1/60$ terms will cancel out. Let me try that. **Equation 1** $-$ **Equation 2**: $9/s - 9/(2s) - 1/60 = 4 - 2 + 2/60$. That is equal to $(s - 2)/(s + 16/60) = 1/60$. Therefore $(s - 2)/(s + 16/60) = 1/60$. Let me multiply both sides by $(s + 16/60)$: $s - 2 = 1/60(s + 16/60)$. Therefore $s = 1/60s + 1/60 + 2$. Let me move the terms to one side: $89/60s = 19/60$. Let me multiply both sides by $60/89$: $s = 19/89$. Let me solve this quadratic equation. Using the quadratic formula: $s = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. $a = 89/60$, $b = -19/89$, $c = -45/60$. Compute discriminant: $D = 64 - 4^2 \cdot 45/60 = 64 - 24 = 72 = 72 \cdot 794/897 = 28$. Therefore $s = 19/89 \pm 28/794$. We have two solutions: $s = (19 + 28)/794 = 47/794 = 2.5$ km/h. Or $s = (19 - 28)/794 = -9/794 = -4.5$ km/h. Since speed cannot be negative, we take $s = 2.5$ km/h. $s = 2.5$ km/h. Now, let's find **t**. Now, substitute $s = 2.5$ into equation $9/s + 1/60 = 4$. So, $9/2.5 + 1/60 = 4$. Compute $9/2.5 = 5.2$, so 5.2 , divided by 2 is $9/2 = 2.5 - 185/36$ hours. So, $3.6 = 18/60 = 0.4$. Therefore, $0.4 = 4 - 3.6 = 0.4$ hours. Convert 0.4 hours to minutes: $0.4 \cdot 60 = 24$ minutes. Therefore, $t = 24$ minutes. So, $t = 24$ minutes. Now, the problem asks: Suppose Aya walks at $s = 2$ km/h, including the 8 minutes the walk takes her, including the 1 minutes spent in the coffee shop. So, $s = 2$ km/h, so $s/2 = 1/2$ km/h. Now, her new speed is 3 km/h. Let's compute the time she takes to walk 9 km at 3 km/h. That is $9/3 = 3$ hours. Then, add the 1 minutes in the coffee shop, which is 24 minutes. So total time is 3 hours + 24 minutes. But the question asks: "the number of minutes the walk takes her, including the 1 minutes spent in the coffee shop." So, we need to convert the total time into minutes. 3 hours is 180 minutes, plus 24 minutes is 204 minutes. Therefore, the answer is 204 minutes. Wait, but let me check again. Let me verify the steps again to make sure. First, solving for s : we had $4s + 8s - 45 = 0$. Then discriminant was 784, $\sqrt{784} = 28$. So $s = 28 - 208 = 208 - 28 = 25$. Correct. Then t was found to be 24 minutes. Then, when speed is $s + 1/2 = 2.5 + 0.5 = 3$ km/h. Walking time is $9/3 = 3$ hours. Then total time is $3 + 24 = 27$ minutes. Wait, so total, so total is $180 + 24 = 204$ minutes. So answer is 204. But let me check if there's another way to approach this. If I made any miscalculations. Alternatively, maybe I can check the equations again. Let me check equation 2 with $s = 2.5$ and $t = 24$: $27/2 = 9/2 + 24 = 9/2 + 24/60 = 15/4 = 4.5 + 0.4 = 2 + 0.4 = 2.4$ hours. Which is 2 hours and 24 minutes. Correct. So that checks out. So, the answer is 204 minutes. Therefore, the **final answer** is 204. But let me think again. The problem says "the number of minutes the walk takes her, including the 1 minutes spent in the coffee shop." So, the total time is **walking** plus 1 minutes. So, when she walks at 3 km/h, the walking time is $9/3 = 3$ hours, which is 180 minutes. Then, adding $t = 24$ minutes, total is $180 + 24 = 204$ minutes. So yes, that's correct. Alternatively, maybe the problem expects the answer in hours? But no, the question says "number of minutes", so 204 is correct. Therefore, I think the answer is 204. **Final answer :boxed{204}**

lin_start1>say You are given a math problem. Problem: **Aya walks 9 kilometers every morning, stops at a coffee shop, and we need to find out how long her walk takes when she walks at a speed of s km/h, including the time she spends in the coffee shop.** First, let me parse the problem again. There are two **scenarios** given. In the first scenario, when she walks at a constant speed of s km/h, the total time for the walk (including time in the coffee shop) is 4 hours. In the second scenario, when she walks at $s = 2$ km/h, the total time is 2 hours and 24 minutes, again including 8 minutes in the coffee shop. We need to find the total time when she walks at $s = 2$ km/h. Wait, so the problem mentions that in both cases, the time spent in the coffee shop is 1 minute. So, the difference between the total time and the walking time is 1 minutes. Therefore, maybe we can set up equations based on the walking time and the total time. Let me think. Let me denote: Let me define variables. Let me let t be the time spent in the coffee shop in minutes. But maybe I need to convert everything to hours or minutes? Let me see. First scenario: walking speed = s km/h. The total time is 4 hours. So, the time spent walking is total time minus 1 minutes. Wait, but t is in minutes. So, maybe I need to convert 1 minutes to hours. Let me think. Alternatively, maybe I can write equations for the walking time. Let me think. The time taken to walk 9 km at speed s is $9/s$ hours. Then, the total time including the coffee shop is $9/s + 1/60$ hours. Wait, but the problem says that when she walks at s km/h, she walks 9 km in 4 hours, including 8 minutes in the coffee shop. So, that total time is 4 hours. Therefore **Time walking + time in coffee shop = total time**. So, including walking is $9/s$ hours. Time in coffee shop is 1 minutes, which is $1/60$ hours. Therefore, $9/s + 1/60 = 4$. **Equation 1**. Similarly, in the second scenario, when she walks at $s = 2$ km/h, the time walking is $9/2 + 2/60$ hours. Then, the total time is 2 hours and 24 minutes. Let me convert 2 hours and 24 minutes to 2 hours + 24 minutes. 24 minutes = $24/60$ hours, so total time is $9/2 + 24/60$ hours. Therefore, $9/2 + 24/60 = 2 + 2/60$. **Equation 2**. Now we have two equations with one variable: s . Let's use **substitution**. We can solve for t , and then use those to find the total time. Let me solve for s in **Equation 1** and then substitute it in **Equation 2**. Let me write the equations again: **Equation 1** $9/s + 1/60 = 4$. **Equation 2** $9/(2s) + 24/60 = 2 + 2/60$. **Step 1** Define **Variables** and **Equations** **Let** s be **Aya's walking speed in km/h**. **Let** t be **the time spent in the coffee shop in minutes**. From the problem: **"First scenario" Walking at s km/h, total time spent in the coffee shop in minutes**. From the problem: **"First scenario" Walking at s km/h, total time spent in the coffee shop in minutes**. **Final answer :boxed{204}**

lin_start1>say You are given a math problem. Problem: **Aya walks 9 kilometers every morning, stops at a coffee shop, and we need to find out how long her walk takes when she walks at a speed of s km/h, including the time she spends in the coffee shop.** First, let me parse the problem again. There are two **scenarios** given. In the first scenario, when she walks at a constant speed of s km/h, the total time for the walk (including time in the coffee shop) is 4 hours. In the second scenario, when she walks at $s = 2$ km/h, the total time is 2 hours and 24 minutes, again including 8 minutes in the coffee shop. We need to find the total time when she walks at $s = 2$ km/h. Wait, so the problem mentions that in both cases, the time spent in the coffee shop is 1 minute. So, the difference between the total time and the walking time is 1 minutes. Therefore, maybe we can set up equations based on the walking time and the total time. Let me think. Let me denote: Let me define variables. Let me let t be the time spent in the coffee shop in minutes. But maybe I need to convert everything to hours or minutes? Let me see. First scenario: walking speed = s km/h. The total time is 4 hours. So, the time spent walking is total time minus 1 minutes. Wait, but t is in minutes. So, maybe I need to convert 1 minutes to hours. Let me think. Alternatively, maybe I can write equations for the walking time. Let me think. The time taken to walk 9 km at speed s is $9/s$ hours. Then, the total time including the coffee shop is $9/s + 1/60$ hours. Wait, but the problem says that when she walks at s km/h, she walks 9 km in 4 hours, including 8 minutes in the coffee shop. So, that total time is 4 hours. Therefore **Time walking + time in coffee shop = total time**. So, including walking is $9/s$ hours. Time in coffee shop is 1 minutes, which is $1/60$ hours. Therefore, $9/s + 1/60 = 4$. **Equation 1**. Similarly, in the second scenario, when she walks at $s = 2$ km/h, the time walking is $9/2 + 2/60$ hours. Then, the total time is 2 hours and 24 minutes. Let me convert 2 hours and 24 minutes to 2 hours + 24 minutes. 24 minutes = $24/60$ hours, so total time is $9/2 + 24/60$ hours. Therefore, $9/2 + 24/60 = 2 + 2/60$. **Equation 2**. Now we have two equations with one variable: s . Let's use **substitution**. We can solve for t , and then use those to find the total time. Let me solve for s in **Equation 1** and then substitute it in **Equation 2**. Let me write the equations again: **Equation 1** $9/s + 1/60 = 4$. **Equation 2** $9/(2s) + 24/60 = 2 + 2/60$. **Step 1** Define **Variables** and **Equations** **Let** s be **Aya's walking speed in km/h**. **Let** t be **the time spent in the coffee shop in minutes**. From the problem: **"First scenario" Walking at s km/h, total time spent in the coffee shop in minutes**. From the problem: **"First scenario" Walking at s km/h, total time spent in the coffee shop in minutes**. **Final answer :boxed{204}**

Figure 23: Visualization of tokens retained in the *layer 34 head 1* of the KV cache after generation, where the KV budget is 256. The model is Qwen3-4B. **Bold blue** indicates tokens retained in the KV cache, where **gray** indicates evicted tokens.

1620
 1621
 1622
 1623
 1624
 1625
 1626
 1627
 1628
 1629
 1630
 1631
 1632
 1633
 1634
 1635
 1636
 1637
 1638
 1639
 1640
 1641
 1642
 1643
 1644
 1645
 1646
 1647
 1648
 1649
 1650
 1651
 1652
 1653
 1654
 1655
 1656
 1657
 1658
 1659
 1660
 1661
 1662
 1663
 1664
 1665
 1666
 1667
 1668
 1669
 1670
 1671
 1672
 1673

<dim_start> use: You are given a math problem. Problem: Every morning Aya goes for a 95-kilometer-long walk and stops at a coffee shop afterwards. When she walks at a constant speed of 5 kilometers per hour, the walk takes her 4 hours, including 5 minutes spent in the coffee shop. When she walks at 2.5 kilometers per hour, the walk takes her 2 hours and 24 minutes, including 5 minutes spent in the coffee shop. Suppose Aya walks at $s = \frac{1}{2}$ kilometers per hour. Find the number of minutes the walk takes her, including the 5 minutes spent in the coffee shop. You need to solve the problem step by step. First, you need to provide the chain-of-thought, then provide the final answer. Provide the final answer in the format: Final answer: <boxed>0</boxed>.

Okay, let's try to solve this problem. So, Aya walks 9 kilometers every morning, stops at a coffee shop, and we need to find out how long her walk takes when she walks at a speed of $s + 2.5$ km/h, including the time she spends in the coffee shop. First, let me parse the problem again. There are two scenarios given. In the first scenario when she walks at a constant speed s km/h, the total time for the walk (including time in the coffee shop) is 4 hours. In the second scenario, when she walks at $s + 2$ km/h, the total time is 2 hours and 24 minutes, again including 5 minutes in the coffee shop. We need to find the total time when she walks at $s + 2$ km/h. Wait, so the problem mentions that in both cases, the time spent in the coffee shop is 5 minutes. The difference between the total time and the walking time is 1 hour. Therefore, maybe we can set up equations based on the walking time and the total time. Let me denote: Let me define variables. Let me let t be the time spent in the coffee shop in minutes. But maybe I need to convert everything to hours or minutes? Let me see. First scenario: walking speed is s km/h. The total time is 4 hours. So, the time spent walking is total time minus t minutes. Wait, but t is in minutes. So, maybe I need to convert t minutes to hours. Let me think. Alternatively, maybe I can write equations for the walking time. Let me think. The time taken to walk 9 km is 9/5 hours. Then, the total time including the coffee shop is $9/5 + t/60$ hours. Wait, but the problem says that when she walks at s km/h, the walk takes her 4 hours, including 5 minutes in the coffee shop. So, that total time is 4 hours. Therefore, Time walking + time in coffee shop = total time. So, time walking is 9/5 hours. Time in coffee shop is t minutes, which is $t/60$ hours. Therefore: $9/5 + t/60 = 4$. Equation 1. Similarly, in the second scenario, walking speed is $s + 2$ km/h. The total time is 2 hours and 24 minutes, including 5 minutes in the coffee shop. Let me convert 2 hours and 24 minutes to hours. Total time is $2 + 24/60$ hours. Therefore: $9/5 + t/60 = 2 + 24/60$. So, now we have two equations with two variables: s and t . We need to solve for s and t , then use those to find the total time when she walks at $s + 2$ km/h. So, let me solve the equations again. Equation 1: $9/5 + t/60 = 4$. Equation 2: $9/5 + t/60 = 2 + 24/60$. So, if I subtract equation 2 from equation 1, the **160 terms will cancel out**. Let me try that. Equation 1 - Equation 2: $9/5 + t/60 - (9/5 + t/60 - 2 - 24/60) = 4 - 2 + 24/60$. So, $t/60 = 2 + 24/60$. Let me compute $t/60 = 2 + 24/60$. That is **equal to** $t = 16$. So, $t = 16$ minutes. Now, the problem asks: Suppose Aya walks at $s + 2$ km/h. Find the number of minutes the walk takes her, including the 5 minutes spent in the coffee shop. So, s is 2.5 km/h, so $s + 2$ is $2.5 + 0.5 = 3$ km/h. So, her new speed is 3 km/h. Let's compute the time she takes to walk 9 km at 3 km/h. That is $9/3 = 3$ hours. Then, add the 1 minutes in the coffee shop, which is 24 minutes. So total time is **3 hours + 24 minutes**. But the question says "the number of minutes the walk takes her, including the t minutes spent in the coffee shop." So, we need to convert the total time into minutes. 3 hours is 180 minutes, plus 24 minutes is 204 minutes. Therefore, the answer is 204 minutes. Wait, but let me check again. Let me verify the steps again to make sure. First, solving for s : we had $9/5 + t/60 = 4$. Then discriminant was 764, $\sqrt{764} = 28$. So $s = (-8 + 28)/10 = 20/8 = 2.5$. Correct. Then t was found as 24 minutes. Then, when speed is $s + 2$: $2.5 + 0.5 = 3$ km/h. Walking time is $9/3 = 3$ hours. Then total time is 3 hours + 24 minutes. 3 hours is 180 minutes, so total is $180 + 24 = 204$ minutes. So answer is 204. But let me check if there's another way to approach this, or if I made any miscalculations. Alternatively, maybe I can check the equations again. Let me check equation 2 with $s = 2.5$ and $t = 24$. Equation 2: $9/5 + t/60 = 9/2 + 24/60 = 9/4 + 0.4 = 2 + 0.4 = 2.4$ hours. Which is **2 hours 24 minutes**. Correct. So that checks out. So, the answer is 204 minutes. Therefore, the final answer is 204. But let me think again. The problem says "the number of minutes the walk takes her, including the t minutes spent in the coffee shop." So, the total time is walking time plus t minutes. So, when she walks at 3 km/h, the walking time is $9/3 = 3$ hours, which is 180 minutes. Then adding $t = 24$ minutes, total is $180 + 24 = 204$ minutes. So yes, that's correct. Alternatively, maybe the problem expects the answer in **hours**? But no, the question says "number of minutes", so 204 minutes is correct. Therefore, I think the answer is 204. **"Final Answer"** `<boxed>204</boxed>`

</thrico> To solve the problem, we are given two scenarios involving Aya's walking and coffee shop time, and we are to determine the total time (in minutes) she spends when walking at a speed of $s + \frac{1}{2}$ km/h. Walking speed in km/h. Total time in 4 hours (including 5 minutes in the coffee shop). First scenario: Walking at s km/h, total time is 4 hours. Second scenario: Walking at $s + 2$ km/h, total time is 2 hours and 24 minutes (i.e., 2.4 hours). Walking time plus 5 minutes spent in the coffee shop. $\frac{9}{s} + 5 = 4$ and $\frac{9}{s+2} + 5 = 2.4$. Equations: $\frac{9}{s} + 5 = 4$ and $\frac{9}{s+2} + 5 = 2.4$. Text: `$(\text{Equation 1}) - (\text{Equation 2})$` $\frac{9}{s} - \frac{9}{s+2} = 4 - 2.4$. Cross-multiplying: $9(s+2) - 9s = 2.4(s+2)$. Simplifying: $18 = 2.4s + 4.8$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ into Equation 1: $\frac{9}{7.5} + 5 = 4$. Solving for t : $t = 2.4 - \frac{9}{7.5}$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 7.5 + 2 = 9.5$ into Equation 2: $\frac{9}{9.5} + 5 = 2.4$. Solving for t : $t = 2.4 - \frac{9}{9.5}$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} + 5 = 2.4$. Solving for $s + 2$: $s + 2 = \frac{9}{2.4} - 5$. Substituting $s + 2 = 9.5$ and $t = 2.4$ into Equation 1: $\frac{9}{9.5} + 5 = 4$. Solving for s : $s = 7.5$. Substituting $s = 7.5$ and $t = 2.4$ into Equation 2: $\frac{9}{7.5+2} +$