RELMEM: LEARNING RECURRENT MEMORY FOR LONGITUDINAL EHR MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Longitudinal electronic health record (EHR) modeling requires integrating new visits with an expanding patient history. Yet the continual accumulation of clinical information imposes increasing computational and memory costs on large language models (LLMs) when they process and retain complete patient histories. A practical alternative is visit-wise recurrent compression, which incorporates each incoming visit into a compact, continually updated patient memory. However, under a fixed memory budget, successive updates must integrate new information without progressively losing critical historical evidence needed to subsequent tasks. To address this challenge, we introduce **Re**current **L**ongitudinal **Mem**ory (**ReLMem**), a framework that learns to maintain fixed-capacity patient memory for efficient downstream prediction with a frozen LLM. ReLMem equips this LLM with lightweight compression adapters to recurrently update the memory from its previous state and each incoming visit, without rereading earlier records. Specifically, we develop a multi-granularity optimization strategy to preserve task-relevant information throughout recurrent updates and support downstream prediction from the final memory. The intermediate supervision aligns attention outputs from compressed memory and the full history under identical queries, while prediction supervision minimizes cross-entropy with ground truth answers conditioned on the final memory. On EHR-based medication prediction, ReLMem approaches the F1 scores of full-history baseline while reducing average retained historical storage by 97.1%. Under the same memory budget, it improves macro- and micro-F1 over the strongest compressed-memory baseline by 4.66 and 4.75 percentage points, respectively. These results highlight the value of learning recurrent patient memory for efficient longitudinal EHR modeling.

1 INTRODUCTION

As a key component of modern healthcare, electronic health records (EHRs) have been widely adopted in clinical practice: as of 2024, over 99% of non-federal acute care hospitals and 91% of office-based physicians in the United States had adopted certified EHR systems (ONC, 2026). By documenting patients' health conditions, treatments, and outcomes across clinical encounters, EHRs support a wide range of complex tasks, including clinical decision-making, patient monitoring, and biomedical research (Jensen et al., 2012; Moor et al., 2023). For each patient, successive visits contribute new observations, diagnoses, procedures, and treatments to an evolving longitudinal clinical history, which provides context for understanding the patient's current condition and anticipating future health outcomes (Choi et al., 2016a; Kraljevic et al., 2024b; Shmatko et al., 2025). Therefore, effectively modeling these longitudinal visit trajectories is key to realizing the potential of EHRs.

Recent advances in large language models (LLMs) have introduced a new paradigm for modeling such complex clinical information in EHRs, with applications in clinical information extraction, medical forecasting, and multi-step clinical question answering (Yang et al., 2022; Kraljevic et al., 2024a; Shi et al., 2024). However, applying LLMs directly to an expanding patient history incurs increasing computational and memory overhead. For models with dense self-attention, the computational cost of attention during context prefill scales quadratically with sequence length, while the size of the historical key–value (KV) cache grows linearly (Yang et al., 2025). Although cache reuse avoids re-encoding previously processed records, processing each incoming visit still requires attention over the accumulated history.

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

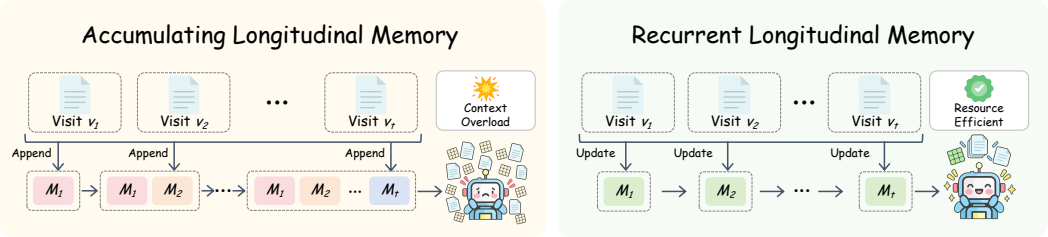


Figure 1: Comparison between accumulating and recurrent longitudinal memory. Left: Each new visit adds a separate memory block, potentially leading to context overload as visits accumulate. Right: ReLMem integrates each incoming visit with the previous memory to maintain a fixed-capacity state, enabling resource-efficient modeling.

Context compression offers a practical way to reduce these costs by replacing the full history with a compact representation. Existing methods typically compress a given context by pruning less informative KV entries or encoding its content into a smaller set of learned representations (Li et al., 2024; Kim et al., 2025; Ge et al., 2024). But for longitudinal EHRs, compression must accommodate a history that continually expands with new visits. Recompressing the full history at each visit requires revisiting earlier records, whereas compressing visits separately and appending their representations still causes the retained context to grow (Kim et al., 2024). This motivates a recurrent formulation in which each incoming visit is integrated with the previous memory to produce an updated state of fixed capacity (Bulatov et al., 2022). As illustrated in Figure 1, this formulation maintains a continually updated patient memory without rereading earlier records or accumulating separate representations for successive visits.

However, maintaining such a memory requires more than just effectively compressing individual visits. Each update must incorporate new clinical information while preserving historical evidence that may be needed for downstream tasks (Rae et al., 2020). Since information from earlier visits is accessible only through the previous memory, information loss at one step can persist and accumulate across subsequent updates (Kim et al., 2024). Although supervision with ground-truth answers encourages accurate predictions from the final memory, it does not explicitly constrain how well intermediate memory states preserve historical evidence. This highlights a central challenge in efficient longitudinal EHR modeling: *how to preserve task-relevant information throughout recurrent updates while ensuring that the resulting memory supports downstream clinical prediction.*

To address this challenge, we introduce **Re**current **L**ongitudinal **Mem**ory (**ReLMem**), a framework that learns fixed-capacity patient memory for longitudinal EHR modeling with a frozen, task-adapted LLM. At each visit, this LLM uses lightweight compression adapters to integrate the complete incoming record with the previous memory and replace it with an updated state. We develop a multi-granularity optimization strategy to preserve historical evidence throughout recurrent updates while supporting downstream prediction. Specifically, we align attention outputs from intermediate memory states with those from the complete available history under identical queries, and supervise predictions from the final memory using cross-entropy with ground truth answers. We further adopt a curriculum learning strategy that progressively introduces cases with more visits to help ReLMem maintain informative patient memory over longer histories. Across two representative longitudinal EHR tasks (i.e., medication and diagnosis prediction), ReLMem approaches the F1 scores of the full-history baseline while using a substantially smaller history budget.

Our contributions are summarized as follows:

- We introduce ReLMem, a framework that learns fixed-capacity recurrent patient memory for resource efficient longitudinal EHR modeling.
- We develop multi-granularity optimization that aligns intermediate memory readouts with those from full history and jointly supervises clinical predictions from the final memory.
- We evaluate ReLMem on representative longitudinal EHR tasks, demonstrating a favorable trade-off between predictive performance and retained-history storage.

2 RELATED WORK

Longitudinal EHR modeling. Longitudinal EHR modeling integrates information across successive visits to support clinical prediction. Early neural models used patient histories for diagnosis, medication recommendation, and risk prediction (Choi et al., 2016b;a; Shang et al., 2019). Pretrained transformers subsequently improved disease prediction and enabled few-shot adaptation (Li et al., 2020; Rasmy et al., 2021; Wornow et al., 2023), with TransformEHR further demonstrating the benefit of more complete visit histories (Yang et al., 2023). More recently, generative models have been used to forecast clinical events and disease trajectories (Kraljevic et al., 2024b; Shmatko et al., 2025), while Apollo predicts disease progression and treatment response from multimodal records (Zhang et al., 2026). EHR-R1 and EHRAgent further extend EHR analysis to LLM-based clinical reasoning and multi-step database querying, respectively (Liao et al., 2025; Shi et al., 2024). ReLMem complements these advances by learning recurrent patient memory that preserves task-relevant history for downstream prediction with a frozen LLM.

Context compression. Context compression condenses information for efficient processing and retention by LLMs. LLMLingua and LLMLingua-2 shorten prompts by removing less informative content (Jiang et al., 2023; Pan et al., 2024), whereas Gisting and the In-context Autoencoder encode text into compact learned representations (Mu et al., 2023; Ge et al., 2024). At the KV-cache level, H_2O, SnapKV, and KVzip reduce storage requirements by selectively retaining cached states (Zhang et al., 2023; Li et al., 2024; Kim et al., 2025). More recently, Cartridges and Attention Matching directly optimize compact KV representations for a given input (Eyuboglu et al., 2026; Zweiger et al., 2026). However, compressed representations can still accumulate across successive visits, potentially leading to context overload. Recurrent memory offers a practical alternative by integrating each incoming visit with the previous state to maintain resource efficient modeling.

Recurrent memory. Recurrent memory carries information across successive segments to support efficient longitudinal modeling. Transformer-XL extends context through segment-level recurrence (Dai et al., 2019), while Compressive Transformer retains compressed historical states with attention reconstruction supervision (Rae et al., 2020). Subsequent approaches learn compact representations to transfer or accumulate information across segments (Bulatov et al., 2022; Chevalier et al., 2023). More recently, Compressed Context Memory (CCM) supports online interactions through recurrent compression (Kim et al., 2024). For longitudinal EHR modeling, however, the key challenge of recurrent memory is not merely to compress individual visits but to integrate new information while preserving historical evidence for downstream prediction. To this end, ReLMem learns a replacement memory at each visit, aligning its attention outputs with those of an independently encoded complete historical prefix and jointly supervising predictions from the final memory.

3 PROBLEM FORMULATION

Let $H_t = (v_1, \ldots, v_t)$ denote a patient's longitudinal history of completed visits in chronological order, where v_t is the full text of the t-th visit. Given the history H_T preceding a target visit and a task query q, the goal is to predict an answer sequence $y = (y_1, \ldots, y_N)$. Specifically, for medication prediction, q includes the diagnoses and procedures of the target visit, and y represents its medication set. For next-visit diagnosis prediction, q contains only the task instruction, and y represents the diagnoses at the next visit.

We consider longitudinal EHR modeling with a patient memory M whose capacity remains fixed as visits accumulate. As each visit arrives, the LLM updates its memory from the previous state M_{t-1} and the incoming visit v_t:

$$M_0 = \varnothing, \qquad M_t = \mathcal{C}_{\theta,\phi}(M_{t-1}, v_t), \quad t = 1, \ldots, T, \tag{1}$$

Here, $\mathcal{C}_{\theta,\phi}$ denotes the recurrent update performed by the LLM. The backbone parameters θ remain frozen, while ϕ denotes the additional parameters trained for memory compression. At inference, information from processed visits is accessible only through M_t. After the final update, the LLM predicts the answer according to $p_\theta(y \mid M_T, q)$. The objective of ReLMem is to learn recurrent memory updates that preserve task-relevant evidence for accurate clinical prediction.

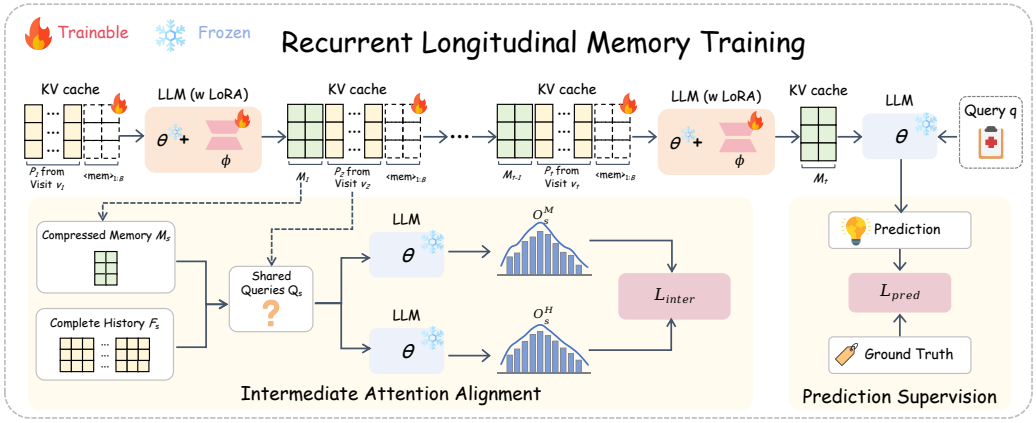


Figure 2: The training pipeline of recurrent longitudinal memory. A frozen LLM equipped with lightweight compression adapters encodes each incoming visit with the previous memory to produce an updated state of fixed capacity. $\mathcal{L}_{\text{inter}}$ aligns attention readouts from current memory and the full history under identical queries, while $\mathcal{L}_{\text{pred}}$ optimizes predictions from the final memory.

4 ReLMem

ReLMem uses a task-adapted LLM backbone for recurrent memory updates and clinical prediction. We first adapt the LLM to the specific task using complete patient histories, then freeze the adapted backbone and train lightweight compression adapters to integrate each incoming visit with the previous memory. Recurrent memory learning uses multi-granularity optimization, combining intermediate attention alignment with final prediction supervision. We further adopt a curriculum learning strategy that starts with short histories and progressively introduces training examples with more visits. Figure 2 illustrates recurrent memory learning after task-specific adaptation.

4.1 Task-Specific Adaptation

To provide a task-specific backbone for recurrent memory learning, we first adapt the LLM to clinical prediction through supervised fine-tuning (SFT) on complete patient histories. Using low-rank adaptation (LoRA) (Hu et al., 2022), we keep the pretrained model θ_0 fixed and optimize the task-adapter parameters ψ by minimizing cross-entropy over the answer tokens:

$$\mathcal{L}_{\text{SFT}}(\psi) = \mathbb{E}_{(H_T,q,y)\sim\mathcal{D}}\left[-\frac{1}{N}\sum_{j=1}^{N}\log p_{\theta_0,\psi}(y_j \mid H_T, q, y_{<j})\right]. \tag{2}$$

Here, $\mathcal{D}$ denotes the training set and $y_{<j}$ denotes the preceding ground-truth answer tokens. After adaptation, we merge the learned LoRA updates into the pretrained weights θ_0 to obtain the backbone parameters θ, which remain frozen throughout recurrent longitudinal memory training.

4.2 Recurrent Memory Update

As shown in Figure 3, with the task-adapted backbone fixed, ReLMem updates patient memory in two steps: encoding each incoming visit in the context of the retained history, then integrating both into a new fixed-capacity state. To make the retained history directly accessible through the backbone's attention mechanism, we represent memory as layer-wise KV pairs:

$$M_t = \left\{(\mathbf{K}_t^\ell, \mathbf{V}_t^\ell)\right\}_{\ell=1}^{L}, \qquad \mathbf{K}_t^\ell, \mathbf{V}_t^\ell \in \mathbb{R}^{H_{\text{kv}}\times B\times d_h}, \tag{3}$$

where B is the fixed number of memory slots per KV head in each layer, and L, H_{kv}, and d_h denote the number of backbone layers, the number of KV heads, and the head dimension, respectively.

Memory-conditioned encoding. Rather than encoding each visit in isolation, the frozen backbone first profiles the full text of the incoming visit v_t with M_{t-1} as historical context:

$$P_t = \mathrm{KV}_\theta(v_t \mid M_{t-1}), \tag{4}$$

where KV_θ denotes KV computation through the frozen backbone. The resulting P_t denotes the layer-wise KV pairs for all tokens in the current visit v_t.

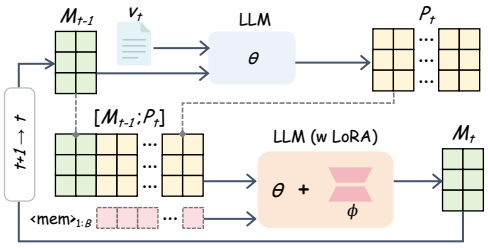


Figure 3: Recurrent memory update in ReLMem.

Fixed-capacity update. We next incorporate the encoded visit into patient memory while keeping its capacity fixed. Specifically, we append a sequence of B memory tokens, denoted by $\langle\mathrm{mem}\rangle_{1:B}$, to the end of the visit. Following CCM (Kim et al., 2024), we use token-conditional low-rank adapters that are enabled only at memory-token positions, while visit, task-query, and answer tokens use the frozen backbone projections. The memory-token embeddings and adapter weights form the trainable parameters ϕ, which are shared across all recurrent updates. The task-adapted backbone processes these memory tokens with $[M_{t-1}; P_t]$ as a prefix KV cache, allowing them to attend to both the previous memory and the encoded visit:

$$M_t = \mathrm{KV}_{\theta,\phi}\big(\langle\mathrm{mem}\rangle_{1:B} \mid [M_{t-1}; P_t]\big), \tag{5}$$

where $[\,;\,]$ denotes concatenation of KV states along the sequence dimension, and $\mathrm{KV}_{\theta,\phi}$ returns only the layer-wise KV pairs produced at the memory-token positions. We retain these KV pairs as M_t in place of the previous memory and the temporary visit states. The updated memory provides historical context for the next visit, completing the recurrent update in Eq. 1.

4.3 Multi-Granularity Optimization

Such recurrent updates may discard information during compression (Kim et al., 2024), and these losses can accumulate across visits and affect the final prediction. We therefore develop a multi-granularity optimization strategy that combines intermediate attention alignment to preserve historical evidence during updates with final prediction supervision to support accurate clinical prediction.

Intermediate attention alignment. To supervise the historical information retained in M_s at any visit step $s \in \{1, \ldots, T\}$, we independently encode the complete history H_s with the same frozen backbone to obtain an uncompressed reference $F_s = \mathrm{KV}_\theta(H_s)$. However, M_s and F_s contain different numbers of KV entries, preventing direct entry-wise comparison. We therefore align their attention outputs under identical queries, as these readouts reflect what the backbone retrieves from each representation.

Specifically, the frozen backbone processes the next visit v_{s+1}, or the task query q when $s = T$, using M_s as historical context. We then use the resulting attention query vectors $\mathbf{Q}_s$ to read from both representations:

$$\mathbf{O}_s^M = \mathrm{Attn}(\mathbf{Q}_s, M_s), \qquad \mathbf{O}_s^H = \mathrm{Attn}(\mathbf{Q}_s, F_s), \tag{6}$$

where Attn reads from the historical KV pairs in the supplied representation. For a single layer and query head, this operation is computed as:

$$\mathrm{Attn}(\mathbf{Q}, X) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}_X^\top}{\sqrt{d_h}}\right)\mathbf{V}_X, \qquad X \in \{M_s, F_s\}, \tag{7}$$

where $\mathbf{K}_X$ and $\mathbf{V}_X$ are the corresponding historical keys and values stored in X. The shared queries yield outputs of the same shape, allowing direct comparison despite the different lengths of the two representations. Finally, we minimize their normalized squared difference, averaged across the layers and query heads used for alignment:

$$\mathcal{L}_{\mathrm{inter}}(\phi; s) = \frac{1}{|\mathcal{S}|H_q}\sum_{\ell\in\mathcal{S}}\sum_{h=1}^{H_q}\frac{\left\|\mathbf{O}_{s,\ell,h}^M - \mathbf{O}_{s,\ell,h}^H\right\|_F^2}{\left\|\mathbf{O}_{s,\ell,h}^H\right\|_F^2 + \epsilon n_s d_h}, \tag{8}$$

where $\mathcal{S}$ denotes the layers used for alignment, H_q is the number of query heads, n_s is the number of query vectors per head, and $\epsilon > 0$ stabilizes the normalization. By matching what the backbone retrieves from the full history, this objective encourages the updated memory to preserve historical context for subsequent visits and clinical prediction.

Final prediction supervision. While intermediate alignment supervises access to historical information, final prediction supervision directly targets the clinical answer. We condition the backbone on the final memory M_T and task query q, and minimize cross-entropy over the ground-truth answer:

$$\mathcal{L}_{\text{pred}}(\phi) = -\frac{1}{N}\sum_{j=1}^{N} \log p_\theta\left(y_j \mid M_T, q, y_{<j}\right). \tag{9}$$

With θ fixed, this objective only optimizes ϕ, guiding recurrent updates to retain information useful for clinical prediction. At last, we combine intermediate attention alignment and final prediction supervision in a joint objective:

$$\min_{\phi}\ \mathbb{E}_{(H_T,q,y)\sim\mathcal{D}}\left[\mathcal{L}_{\text{pred}}(\phi) + \frac{\lambda}{T}\sum_{s=1}^{T}\mathcal{L}_{\text{inter}}(\phi; s)\right], \tag{10}$$

where λ controls the contribution of intermediate alignment relative to final prediction supervision. The full-history reference is used only during training.

4.4 Curriculum Learning

As the number of visits increases, information loss may accumulate over more recurrent updates, making memory learning more challenging. We therefore adopt a curriculum learning strategy (Bengio et al., 2009) that begins with short histories and progressively introduces examples with more visits. At each training stage e, we draw examples from

$$\mathcal{D}_e = \left\{(H_T, q, y) \in \mathcal{D} : T \le \tau_e\right\}, \tag{11}$$

where τ_e is a nondecreasing visit-count threshold. The eligible set gradually expands to cover the full training set, while short histories continue to be sampled alongside longer ones to maintain supervision across recurrence depths. Throughout this progression, the memory capacity, recurrent update rule, and joint objective in Eq. 10 remain unchanged.

5 Experiments

5.1 Datasets and evaluation protocol

We evaluate ReLMem on medication and next-visit diagnosis prediction using MIMIC-IV (Johnson et al., 2023b). Following prior work (Shang et al., 2019; Wang et al., 2021), this retrospective medication prediction uses completed visits and the target admission's diagnoses and procedures to predict medication classes prescribed within its first 24 hours of hospitalization. Diagnosis prediction instead infers the next visit's diagnosis categories from completed visits alone. For each task, the test set is derived from EHRs of 300 patients. We also construct corresponding training and validation sets for recurrent memory learning, with no patient overlap across the three splits. Appendices A and B provide further details. For downstream evaluation, we report macro-F1 averaged across cases and micro-F1 calculated over the test set as a whole. We also report P@k and R@k for $k \in \{5, 10\}$, computed over the first k unique labels in generation order. To assess resource efficiency, we measure retained-history storage, peak GPU memory, update latency, and prediction latency. Detailed metric definitions are provided in Appendix F.

5.2 Implementation details

We use Qwen3-4B (Yang et al., 2025) unless otherwise specified. For each task, we first adapt the backbone with LoRA (Hu et al., 2022) for one epoch at a learning rate of 10^{-4}. We then freeze the adapted backbone and train only the memory-token embeddings and compression adapters, with a default memory capacity of $B = 1{,}024$ and alignment weight $\lambda = 0.1$. To limit training overhead,

Table 1: Medication prediction results (%) with ReLMem and baselines. The best and sub-optimal scores among history compression methods are shown in **bold** and underlined, respectively.

Method	P@5	P@10	R@5	R@10	Macro-F1 (95% CI)	Micro-F1 (95% CI)	History Budget
Uncompressed reference							
Full History	68.80	63.23	32.77	58.24	62.22 (60.46, 63.92)	63.32 (61.69, 64.88)	35,279
History compression methods							
LLM-Rsum	31.73	20.93	14.76	18.99	18.97 (17.53, 20.46)	19.21 (17.80, 20.69)	1,026
SnapKV	21.93	13.40	9.59	11.68	11.91 (10.62, 13.20)	11.95 (10.71, 13.25)	1,024
KVzip	63.73	54.03	30.06	49.44	52.53 (50.41, 54.59)	54.52 (52.73, 56.28)	1,024
Attention Matching	9.40	5.97	4.32	5.35	6.44 (5.13, 7.83)	8.32 (6.70, 9.99)	1,024
RMT	64.13	55.07	30.68	50.90	47.28 (45.65, 48.89)	48.42 (46.97, 49.88)	1,024
CCM-merge	67.93	61.60	32.42	56.61	57.27 (55.53, 58.95)	58.49 (56.93, 59.99)	1,024
ReLMem (ours)	**70.53**	**63.33**	**33.50**	**58.13**	**61.93** (60.21, 63.64)	**63.24** (61.62, 64.85)	1,024

we compute intermediate attention alignment at one uniformly sampled update boundary per example, using four layers distributed across model depth. Memory learning uses AdamW with a peak learning rate of 3×10^{-4} for up to five epochs. The curriculum learning progressively introduces training examples with more visits, while early stopping is guided by answer cross-entropy on the validation set. All evaluated inputs fit within the configured context window, with space reserved for answer generation, except in the exploration analysis in appendix G.4. More details about training and inference of ReLMem can be found in appendix D.

5.3 Baselines

We compare ReLMem with Full History, an uncompressed reference, and six history compression methods. LLM-Rsum (Wang et al., 2025) recursively updates a textual summary as visits arrive. SnapKV (Li et al., 2024), KVzip (Kim et al., 2025), and Attention Matching (Zweiger et al., 2026) are adapted to compress the previous memory together with each incoming visit. RMT (Bulatov et al., 2022) and CCM-merge (Kim et al., 2024) learn recurrent memory under their respective supervision schemes. All methods share the same task-adapted backbone. Appendix E provides more details on implementation of each baseline.

5.4 Main results

As shown in Table 1, ReLMem approaches Full History using only 2.9% of its average history budget and outperforms all evaluated memory compression methods. It achieves the best P@5 and R@5 scores, showing strong precision and reference-set coverage among the first five generated labels. Across baselines, KVzip substantially outperforms LLM-Rsum, showing the predictive value of retained KV states in longitudinal EHRs modeling. Similarly, RMT trails CCM-merge and ReLMem indicating a representational bottleneck in carrying history through final-layer embeddings rather than layer-wise KV states. Additionally, the weaker results of the SnapKV and Attention Matching adaptations highlight the difficulty of transferring direct context compression to recurrent updates. ReLMem's advantage over CCM-merge is also consistent with the benefit of intermediate supervision observed in the objective ablations (Table 3), which further highlights the value of multi-granularity optimization for recurrent memory learning.

5.5 Analysis

5.5.1 Resource Costs of Recurrent Compression

In Table 2, we compare ReLMem's inference costs with Full History: it nearly halves peak GPU memory and reduces average update latency by 26.3%. Although Full History reuses its KV cache to avoid re-encoding earlier visits, processing each incoming visit still requires attention over all historical KV states, increasing inference costs as history grows. By operating on a compact historical state, ReLMem maintains and updates patient memory more efficiently in longitudinal EHRs, despite encoding each incoming visit in full and performing an additional compression step.

Table 2: Resource costs of longitudinal medication prediction.

Resource	Full History	ReLMem
Historical KV state (MiB)	4,961.08	144.00
Update latency (s/visit)	0.4676	0.3446
Prediction latency (s/query)	8.0837	6.5293
Peak GPU memory (GiB)	17.4354	9.2379

Table 3: Effects of training objectives and curriculum learning.

$\mathcal{L}_{\text{inter}}$	$\mathcal{L}_{\text{pred}}$	Curriculum	Macro-F1	Micro-F1
✓		✓	0.37	0.55
	✓	✓	47.67	48.65
✓	✓		61.00	62.24
✓	✓	✓	**61.93**	**63.24**

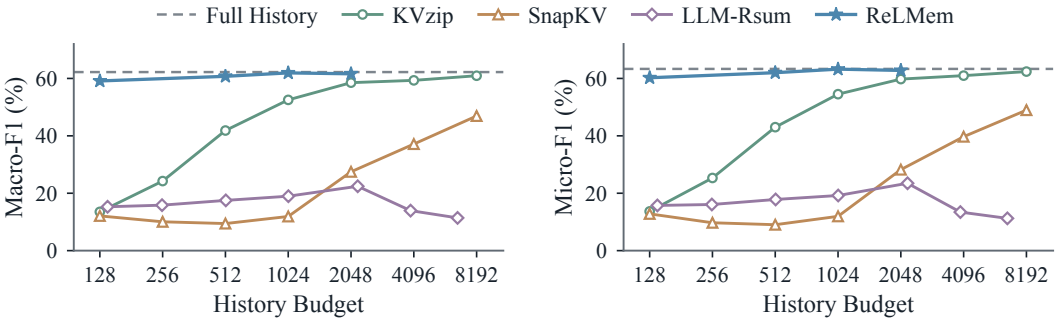


Figure 4: Medication prediction performance across different history budgets.

5.5.2 Memory-Performance Trade-offs

We further analyze how the history budget affects performance (Figure 4). Reducing ReLMem's budget eightfold, from 1,024 to 128 slots, lowers macro- and micro-F1 by only 2.81 and 3.00 percentage points, respectively. At 128 slots, it still outperforms KVzip at 2,048 positions on both metrics, using just one-sixteenth of the history budget. This suggests that learned recurrent states preserve predictive evidence more densely than selected KV entries. In contrast, LLM-Rsum peaks at 2,048 and declines thereafter, showing that a larger summary budget does not necessarily improve prediction. Overall, these results highlight the value of learning to preserve task-relevant information through recurrent updates rather than simply increasing memory capacity.

5.5.3 Effect of Visits Count and History Length

Beyond the memory budget, we further compare ReLMem and Full History by varying the number of recent visits or the history length for each patient. We construct two cohorts of 110 and 107 cases from the medication test set for these respective analyses, keeping each prediction target fixed and varying only how much recent history is provided. As shown in Figure 5, both methods benefit from additional history, highlighting the value of earlier visits for downstream prediction. As more history is included, however, Full History's peak GPU memory more than doubles, whereas ReLMem's increases by only 5.3-6.6%. With all available history, ReLMem remains within 0.59 percentage points of Full History in macro-F1 while reducing peak GPU memory by 48.8-49.1% and final-visit update latency by 47.2-47.9%. These results highlight ReLMem's growing efficiency advantage in longitudinal EHR modeling as patient histories expand.

5.5.4 Contributions of Training Objectives and Curriculum Learning

We next examine the contributions of ReLMem's training components by ablating the supervision objectives and curriculum learning (Table 3). Under the same curriculum, removing prediction supervision reduces both F1 scores to below 1%, indicating that matching historical attention outputs alone does not ensure useful clinical predictions. Removing intermediate alignment instead lowers macro- and micro-F1 by 14.26 and 14.59 percentage points, respectively, highlighting the limitations of relying solely on final-answer supervision to guide information preservation across updates. Therefore, the two objectives are complementary: $\mathcal{L}_{\text{inter}}$ encourages memory to retain key historical information, while $\mathcal{L}_{\text{pred}}$ directs it toward the downstream clinical task. With both

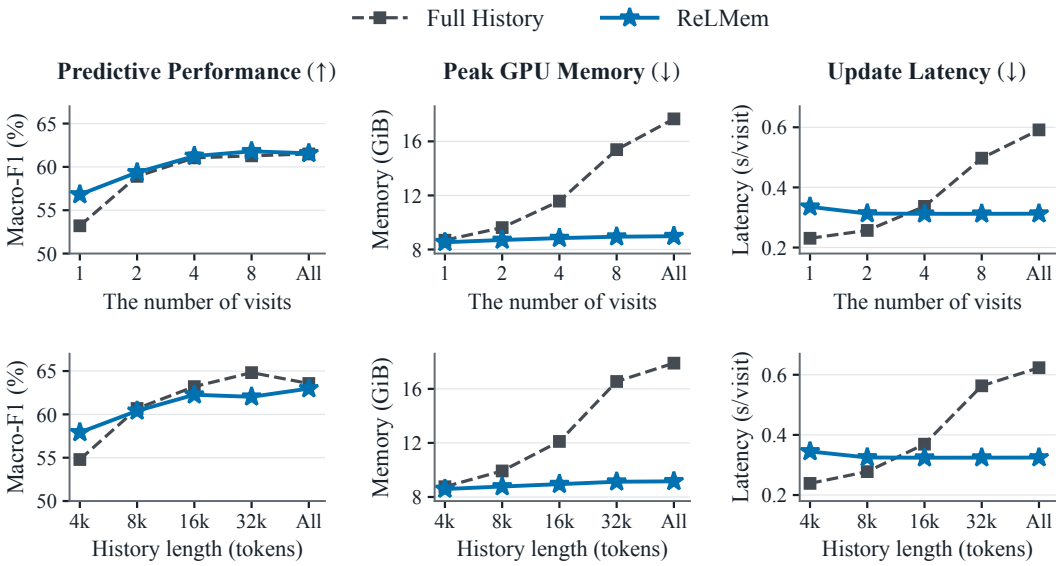


Figure 5: Predictive performance and computational efficiency of ReLMem and Full History across different numbers of historical visits (top) and history lengths (bottom).

Table 4: Average score of Macro-F1 and Micro-F1 on medication prediction across model scales and families.

Method	Qwen3-4B	Qwen3-8B	Llama-3.1-8B
Full History	62.77	62.69	65.62
ReLMem	62.59	61.01	63.16

Table 5: Next-visit diagnosis prediction with Qwen3-8B.

Method	Macro-F1	Micro-F1
Full History	36.30	37.41
ReLMem	35.43	35.85

objectives retained, curriculum learning further improves performance, supporting the strategy of learning shorter update sequences before progressively handling longer histories.

5.5.5 Evaluation across Model Scales and Families

Beyond Qwen3-4B, we further evaluate the scalability of ReLMem across model scales and families by extending it to Qwen3-8B (Yang et al., 2025) and Llama-3.1-8B (Grattafiori et al., 2024), as shown in Table 4. Across all three backbones, the gap between ReLMem and Full History remains below 2.5 percentage points in average F1 score. This consistent performance demonstrates that ReLMem can be applied across different LLM backbones while preserving the predictive utility.

5.5.6 Evaluation on Diagnosis Prediction

Additionally, we evaluate ReLMem on next-visit diagnosis prediction, where no information from the target visit is available and prediction relies entirely on longitudinal history. As shown in Table 5, ReLMem remains close to Full History, with a Macro-F1 gap of only 0.87 percentage points. This shows that the recurrent memory preserves useful historical context even when downstream prediction cannot rely on current-visit diagnoses or procedures.

6 Conclusion

We introduced ReLMem, a framework that learns fixed-capacity recurrent memory for longitudinal EHR modeling with a frozen LLM. With multi-granularity optimization, ReLMem approaches full history performance on two representative tasks and substantially reduces storage and latency as histories expand. Through extensive experiments, we demonstrate the importance of learning not just to compress patient history, but to preserve its predictive value across recurrent updates.

AI USE STATEMENT

LLMs were used for language refinement, formatting checks, and code debugging. All technical ideas, study design, implementation, experimental analysis, and scientific conclusions were developed and verified by the authors.

ETHICS STATEMENT

This study uses de-identified records from MIMIC-IV and MIMIC-IV-Note (Johnson et al., 2023b; 2024; 2023a), whose access is governed by PhysioNet credentialing and data-use agreements. All experiments are retrospective and have no effect on patient care. The targets reflect recorded diagnoses and prescriptions, which may contain documentation and treatment biases. Clinical use would require prospective validation and clinician oversight.

REPRODUCIBILITY STATEMENT

Appendices A and B describe the procedure of dataset construction for both tasks. Appendix C presents the learning and inference algorithm and position-encoding strategy. Appendices D and E detail training, checkpoint selection, decoding, and baseline implementations, while Appendix F defines the predictive metrics, resource measurements, and statistical analysis. Access to the underlying clinical records remains subject to the original dataset agreements. Code will be publicly released upon acceptance.

REFERENCES

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48, 2009. doi: 10.1145/1553374.1553380.

Olivier Bodenreider, Lee Peters, and Thang Nguyen. RxClass – navigating between drug classes and RxNorm drugs. In *Proceedings of the 5th International Conference on Biomedical Ontology*, volume 1327 of *CEUR Workshop Proceedings*, pp. 106–107, 2014.

Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*, volume 35, pp. 11079–11091, 2022. doi: 10.52202/068431-0805.

Centers for Medicare & Medicaid Services and National Center for Health Statistics. *ICD-9-CM Official Guidelines for Coding and Reporting*. U.S. Department of Health and Human Services, 2011. Effective October 1, 2011.

Centers for Medicare & Medicaid Services, National Center for Health Statistics, American Hospital Association, and American Health Information Management Association. *ICD-10-CM Official Guidelines for Coding and Reporting: FY 2026*, 2025. Updated October 1, 2025.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.232.

Edward Choi, Mohammad Taha Bahadori, Joshua A. Kulas, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. RETAIN: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, volume 29, 2016a.

Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Doctor AI: Predicting clinical events via recurrent neural networks. In *Proceedings of the 1st Machine Learning for Healthcare Conference*, volume 56 of *Proceedings of Machine Learning Research*, pp. 301–318. PMLR, 2016b.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988. Association for Computational Linguistics, 2019. doi: 10.18653/v1/P19-1285.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024.

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Atri Rudra, James Zou, Azalia Mirhoseini, and Christopher Ré. Cartridges: Lightweight and general-purpose long context representations via self-study. In *International Conference on Learning Representations*, 2026.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *International Conference on Learning Representations*, pp. 2591–2607, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Peter B Jensen, Lars J Jensen, and Søren Brunak. Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, 13(6):395–405, 2012. doi: 10.1038/nrg3208.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.825.

Alistair Johnson, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. MIMIC-IV-Note: Deidentified free-text clinical notes. *PhysioNet*, 2023a. doi: 10.13026/1n74-ne17. Version 2.2.

Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Brian Gow, Benjamin Moody, Steven Horng, Leo Anthony Celi, and Roger Mark. MIMIC-IV. *PhysioNet*, October 2024. doi: 10.13026/kpb9-mt58. Version 3.1.

Alistair E. W. Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J. Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, Li-wei H. Lehman, Leo A. Celi, and Roger G. Mark. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, 10:1, 2023b. doi: 10.1038/s41597-022-01899-x.

Jang-Hyun Kim, Junyoung Yeom, Sangdoo Yun, and Hyun Oh Song. Compressed context memory for online language model interaction. In *International Conference on Learning Representations*, 2024.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-agnostic KV cache compression with context reconstruction. In *Advances in Neural Information Processing Systems*, volume 38, pp. 185730–185758, 2025. doi: 10.52202/085713-5585.

Zeljko Kraljevic, Joshua Au Yeung, Daniel Bean, James Teo, and Richard J. Dobson. Large language models for medical forecasting—Foresight 2. *arXiv preprint arXiv:2412.10848*, 2024a. doi: 10.48550/arXiv.2412.10848.

Zeljko Kraljevic, Dan Bean, Anthony Shek, Rebecca Bendayan, Harry Hemingway, Joshua Au Yeung, Alexander Deng, Alfred Balston, Jack Ross, Esther Idowu, James T. Teo, and Richard J. B. Dobson. Foresight—a generative pretrained transformer for modelling of patient timelines using electronic health records: A retrospective modelling study. *The Lancet Digital Health*, 6 (4):e281–e290, 2024b. doi: 10.1016/S2589-7500(24)00025-6.

Yikuan Li, Shishir Rao, José Roberto Ayala Solares, Abdelaali Hassaine, Rema Ramakrishnan, Dexter Canoy, Yajie Zhu, Kazem Rahimi, and Gholamreza Salimi-Khorshidi. BEHRT: Transformer for electronic health records. *Scientific Reports*, 10(1):7155, 2020. doi: 10.1038/s41598-020-62922-y.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970, 2024. doi: 10.52202/079017-0722.

Yusheng Liao, Chaoyi Wu, Junwei Liu, Shuyang Jiang, Pengcheng Qiu, Haowen Wang, Yun Yue, Shuai Zhen, Jian Wang, Qianrui Fan, Jinjie Gu, Ya Zhang, Yanfeng Wang, Yu Wang, and Weidi Xie. EHR-R1: A reasoning-enhanced foundational language model for electronic health record analysis. *arXiv preprint arXiv:2510.25628*, 2025.

Michael Moor, Oishi Banerjee, Zahra Shakeri Hossein Abad, Harlan M Krumholz, Jure Leskovec, Eric J Topol, and Pranav Rajpurkar. Foundation models for generalist medical artificial intelligence. *Nature*, 616(7956):259–265, 2023. doi: 10.1038/s41586-023-05881-4.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems*, volume 36, pp. 19327–19352. Curran Associates, Inc., 2023. doi: 10.52202/075280-0848.

ONC. National trends in hospital and physician adoption of electronic health records. Health IT Quick-Stat #61, 2026. Last updated June 2026.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.57.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020.

Laila Rasmy, Yang Xiang, Ziqian Xie, Cui Tao, and Degui Zhi. Med-BERT: Pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction. *npj Digital Medicine*, 4(1):86, 2021. doi: 10.1038/s41746-021-00455-y.

Junyuan Shang, Cao Xiao, Tengfei Ma, Hongyan Li, and Jimeng Sun. GAMENet: Graph augmented MEmory networks for recommending medication combination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1126–1133, 2019. doi: 10.1609/aaai.v33i01.33011126.

Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Jieyu Zhang, Hang Wu, Yuanda Zhu, Joyce C. Ho, Carl Yang, and May Dongmei Wang. EHRAgent: Code empowers large language models for few-shot complex tabular reasoning on electronic health records. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22315–22339, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1245.

Artem Shmatko, Alexander Wolfgang Jung, Kumar Gaurav, Søren Brunak, Laust Hvas Mortensen, Ewan Birney, Tom Fitzgerald, and Moritz Gerstung. Learning the natural history of human disease with generative transformers. *Nature*, 647(8088):248–256, 2025. doi: 10.1038/s41586-025-09529-3.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

Qingyue Wang, Yanhe Fu, Yanan Cao, Shuai Wang, Zhiliang Tian, and Liang Ding. Recursively summarizing enables long-term dialogue memory in large language models. *Neurocomputing*, 639:130193, 2025. doi: 10.1016/j.neucom.2025.130193.

Yanda Wang, Weitong Chen, Dechang Pi, Lin Yue, Sen Wang, and Miao Xu. Self-supervised adversarial distribution regularization for medication recommendation. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 3134–3140, 2021. doi: 10.24963/ijcai.2021/431.

Michael Wornow, Rahul Thapa, Ethan Steinberg, Jason Fries, and Nigam Shah. EHRSHOT: An EHR benchmark for few-shot evaluation of foundation models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 67125–67137. Curran Associates, Inc., 2023. doi: 10.52202/075280-2933.

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

Chaoqi Yang, Cao Xiao, Fenglong Ma, Lucas Glass, and Jimeng Sun. SafeDrug: Dual molecular graph encoders for recommending effective and safe drug combinations. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 3735–3741, 2021. doi: 10.24963/ijcai.2021/514.

Xi Yang, Aokun Chen, Nima PourNejatian, Hoo Chang Shin, Kaleb E. Smith, Christopher Parisien, Colin Compas, Cheryl Martin, Anthony B. Costa, Mona G. Flores, Ying Zhang, Tanja Magoc, Christopher A. Harle, Gloria Lipori, Duane A. Mitchell, William R. Hogan, Elizabeth A. Shenkman, Jiang Bian, and Yonghui Wu. A large language model for electronic health records. *npj Digital Medicine*, 5(1):194, 2022. doi: 10.1038/s41746-022-00742-2.

Zhichao Yang, Avijit Mitra, Weisong Liu, Dan Berlowitz, and Hong Yu. TransformEHR: transformer-based encoder-decoder generative model to enhance prediction of disease outcomes using electronic health records. *Nature Communications*, 14(1):7857, 2023. doi: 10.1038/s41467-023-43715-z.

Andrew Zhang, Tong Ding, Sophia J. Wagner, Caiwei Tian, Ming Y. Lu, Rowland Pettit, Joshua E. Lewis, Alexandre Misrahi, Dandan Mo, Long Phi Le, and Faisal Mahmood. A multimodal and temporal foundation model for virtual patient representations at healthcare system scale, 2026. arXiv preprint arXiv:2604.18570.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710, 2023. doi: 10.52202/075280-1506.

Adam Zweiger, Xinghong Fu, Han Guo, and Yoon Kim. Fast KV compaction via attention matching, 2026. arXiv preprint arXiv:2602.16284.

Contents of the Appendix

A Medication Prediction Dataset . . . 15
A.1 Cohort Split and Target Construction . . . 15
A.2 Clinical Record and Input Template . . . 15
A.3 Dataset Statistics . . . 17
A.4 Label Distribution . . . 18
B Diagnosis Prediction Dataset . . . 19
B.1 Cohort Split and Target Construction . . . 19
B.2 Clinical Record and Input Template . . . 19
B.3 Dataset Statistics . . . 19
B.4 Label Distribution . . . 21
C Learning and Inference Pipeline of ReLMem . . . 22
C.1 Pseudocode for ReLMem . . . 22
C.2 Position Encoding . . . 22
D Training and Inference Configuration . . . 23
D.1 Task Adaptation and Memory Learning . . . 23
D.2 Curriculum Learning . . . 23
D.3 Training and Validation Curves . . . 23
D.4 Inference Settings . . . 23
E Baseline Implementations . . . 24
E.1 Full History . . . 24
E.2 LLM-Rsum . . . 25
E.3 SnapKV . . . 25
E.4 KVzip . . . 25
E.5 Attention Matching . . . 26
E.6 RMT . . . 26
E.7 CCM-merge . . . 26
F Evaluation Metrics . . . 26
F.1 Predictive Performance . . . 26
F.2 Resource Efficiency . . . 27
F.3 Statistical Analysis . . . 27
G Supplementary Experiments . . . 27
G.1 Number of Alignment Layers . . . 27
G.2 Dependence on Patient-Specific Memory . . . 28
G.3 Recurrent versus One-shot Memory Construction . . . 28
G.4 Evaluation on Extended Patient Histories . . . 29
H Qualitative Case Studies . . . 30
H.1 Medication Prediction . . . 30
H.2 Diagnosis Prediction . . . 33
Limitations . . . 34

A MEDICATION PREDICTION DATASET

A.1 COHORT SPLIT AND TARGET CONSTRUCTION

We construct medication prediction examples from MIMIC-IV (Johnson et al., 2023b; 2024; 2023a). Each example combines completed hospital visits with the target visit's diagnoses and procedures to predict its medication classes. We include adult patients with prior hospital visits and a nonempty medication target, and split patients into disjoint training, validation, and test sets. Following GAMENet (Shang et al., 2019) and SARMR (Wang et al., 2021), we define medication targets within the first 24 hours of hospitalization, which avoids combining treatment changes over hospital stays of different lengths. The 24-hour restriction only applies to the medication targets, while diagnoses and procedures are admission-level records used for retrospective prediction. Eligible prescriptions are marked as main medications and have valid start times at or after admission and before the earlier of the 24-hour boundary and discharge. Administration adjuncts such as flushes are excluded.

Following prior works, we map drug identifiers and names to ATC level-3 classes using a fixed ingredient-to-ATC crosswalk and collapse duplicate classes within each target set (Shang et al., 2019; Yang et al., 2021; Bodenreider et al., 2014). To reduce label noise from incomplete normalization, we retain targets only when at least 90% of both prescription rows and ingredient components can be mapped. Consistent with longitudinal medication recommendation, the history contains only completed visits available before the target admission, including their medication fields (Shang et al., 2019; Wang et al., 2021). All methods receive the same retained history and current-visit input.

A.2 CLINICAL RECORD AND INPUT TEMPLATE

Table 6 defines the completed-visit record shared by both tasks. It contains demographics, diagnoses, medications, clinical notes, procedures, and relative timing. The field names and types are fixed, while clinical values, list lengths, and the number of visits vary by patient. Table 7 specifies the prompt template used for medication prediction. The template consists of chronologically ordered visit records, a current-visit object containing only `diagnoses` and `procedures`, and a fixed task instruction defining the prediction target and required output format.

Table 6: Structure and fields of a completed-visit record shared by both tasks.

JSON record template	**Field meanings**
{ "demographics": { "age": *<age in years>*, "sex": "*<sex>*" }, "diagnoses": ["*<diagnosis>*", ...], "medications": ["*<medication>*", ...], "notes": [{ "available_day": *<availability day>*, "chart_day": *<chart day>*, "note_type": "*<note type>*", "text": "*<note text>*" }, ...], "procedures": ["*<procedure>*", ...], "timeline": { "admit_day": *<admission day>*, "available_day": *<availability day>*, "discharge_day": *<discharge day>*, "gap_days": *<interval or null>* }, "visit_number": *<local visit index>* }	`demographics`: age in years and recorded sex. `diagnoses`, `procedures`, and `medications`: lists of clinical names from this completed visit. `notes`: a variable-length list of notes. Each note contains its type, text, chart day, and availability day. `timeline`: admission, discharge, and availability days for the visit. `gap_days` measures the interval since the preceding visible discharge. `visit_number`: the position $1, \ldots, T$ in the visible history. All day values share the first visible admission as Day 0. The first visit has `gap_days=null`. Clinical lists and `notes` can be empty (`[]`).

Table 7: Input template for medication prediction.

Completed history H_T

<Completed visit 1>

...

<Completed visit T>

Current visit

Current visit diagnoses and procedures:

```
{"diagnoses":["<diagnosis 1>", ...],
 "procedures":["<procedure 1>", ...]}
```

Task instruction

Prediction task:
Using the completed visits and the current visit diagnoses and procedures above, predict the complete set of ATC level-3 medication classes represented by qualifying prescriptions started during the first 24 hours of the current hospitalization. Historical medication fields describe their completed visits; they are not the current target.

Record conventions:
- Completed visits are ordered from earliest to latest.
- admit_day, discharge_day, chart_day, and available_day are elapsed days from the admission of the first visible completed visit, which is Day 0; larger values are later.
- visit_number is local to this visible history. gap_days is the interval from the previous visible discharge to the current admission and is null for the first visible visit.
- timeline.available_day is when all displayed information for a completed visit is treated as available. A note's chart_day and available_day use the same Day-0 origin.

Output granularity: use ATC level-3 class names, not individual drugs, ingredients, brands, drug groups at another level, or ATC codes.

Output format: return exactly one valid JSON object and nothing else:
{"predictions":["<medication 1>","<medication 2>"]}

Return the complete predicted set rather than a fixed top-K list. Do not include explanations, Markdown, or any additional key.

A.3 Dataset Statistics

Table 8 summarizes cohort size and per-record statistics across the training, validation, and test splits. The same 6,804 training examples are used for both task adaptation and memory learning. Figure 6 shows the distributions of historical visit counts and input lengths. These dimensions impose complementary demands on recurrent memory: visit count determines recurrence depth, whereas input length reflects the volume of clinical information to be compressed. Compared with the training set, the test set contains more visits and longer contexts, providing a more demanding evaluation of whether fixed-capacity memory can preserve predictive evidence across recurrent updates.

Table 8: Cohort, history, and target statistics for medication prediction datasets.

Statistic	Train	Validation	Test
Patients	1,452	75	300
Data items	6,804	261	300
Per-record statistics: mean (min-max)			
History visits	6.64 (2-21)	11.28 (4-25)	9.20 (3-19)
Input tokens	24,704 (692-40,449)	36,760 (20,266-40,327)	35,699 (20,537-40,335)
Medication classes	11.12 (1-28)	12.69 (1-28)	11.60 (1-28)

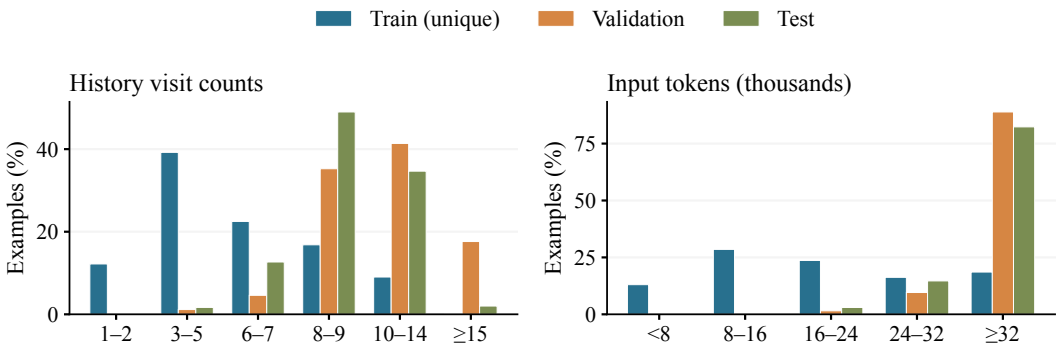


Figure 6: The distribution of historical visit counts and input tokens for each dataset.

A.4 Label Distribution

The medication test set covers 107 classes across 300 target visits. Figure 7 shows the 80 most frequent classes, each counted once per target visit. The distribution includes both common supportive treatments and less frequent therapeutic classes. Because target-set size varies across visits, the model must generate a variable-size medication set for each case.

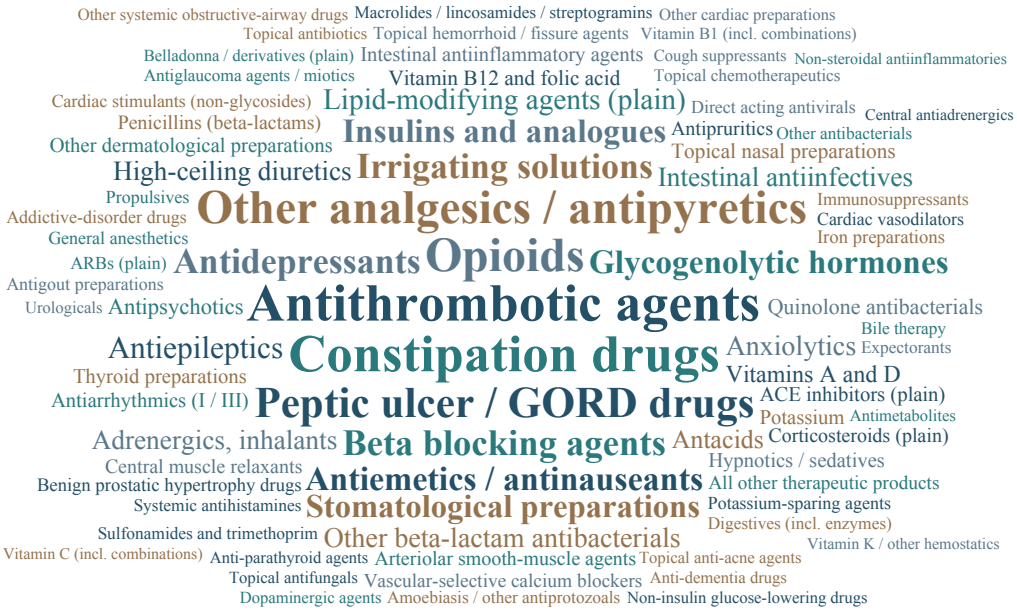


Figure 7: Word cloud of the most frequent medication classes in the test set.

B Diagnosis Prediction Dataset

B.1 Cohort Split and Target Construction

To explore the generalization capability for other tasks of ReLMem, we further construct a multi-label next-visit diagnosis prediction dataset from MIMIC-IV (Johnson et al., 2023b; 2024; 2023a). Following the established longitudinal prediction setting (Choi et al., 2016b), each example uses a patient's completed hospital visits to predict the diagnosis categories documented at the subsequent admission, with all information from that target visit withheld. We split patients, rather than visits, into disjoint training, validation, and test cohorts to prevent patient overlap across splits. Patients may contribute multiple data items within the training and validation cohorts, whereas the test cohort contains one target per patient. During the process of construction, we normalize diagnosis codes to category-level labels using fixed ICD dictionaries. Following the category structures of ICD-9-CM and ICD-10-CM, category keys are formed from the first three characters of ICD-9-CM codes, except external-cause codes beginning with `E`, which use the first four characters, and from the first three characters of ICD-10-CM codes (Centers for Medicare & Medicaid Services & National Center for Health Statistics, 2011; Centers for Medicare & Medicaid Services et al., 2025). Each category is mapped to its standard title, and duplicate labels within the same target visit are removed.

B.2 Clinical Record and Input Template

Diagnosis prediction uses the completed-visit records defined in Table 6. Table 9 specifies the input template, which consists of chronologically ordered visit records followed by a fixed task instruction defining the prediction target and required output format. The template contains no current-visit object. The model returns a variable-length list of diagnosis-category titles under `predictions`, using the same JSON output format as medication prediction.

Table 9: Input prompt for next-visit diagnosis prediction.

Completed history H_T

<Completed visit 1>

...

<Completed visit T>

Task instruction

Prediction task:
Using only the completed visits above, predict the complete set of diagnosis categories most likely to be documented in the patient's next hospital visit. No information from the target visit is provided.

Record conventions:
- Completed visits are ordered from earliest to latest.
- admit_day, discharge_day, chart_day, and available_day are elapsed days from the admission of the first visible completed visit, which is Day 0; larger values are later.
- visit_number is local to this visible history. gap_days is the interval from the previous visible discharge to the current admission and is null for the first visible visit.
- timeline.available_day is when all displayed information for a completed visit is treated as available. A note's chart_day and available_day use the same Day-0 origin.

Output granularity: use standard ICD diagnosis-category titles. Each prediction should be broader than a specific clinical subtype and narrower than an organ-system grouping. Do not output ICD codes.

Output format: return exactly one valid JSON object and nothing else:
{"predictions":["<diagnosis 1>","<diagnosis 2>"]}

Return the complete predicted set rather than a fixed top-K list. Do not include explanations, Markdown, or any additional key.

B.3 Dataset Statistics

Table 10 summarizes cohort size and per-record statistics across the training, validation, and test splits. The same 8,023 training examples are used for both task adaptation and memory learning. The test set contains one target visit per patient. Input lengths include the completed visits and task

instruction, excluding the answer. Figure 8 shows the distributions of history visit counts and input lengths. Compared with the training set, validation and test examples contain more visits and longer inputs on average. This setting evaluates whether fixed-capacity memory can preserve the historical evidence needed for diagnosis prediction across longer sequences of updates.

Table 10: Cohort, history, and target statistics for diagnosis prediction datasets.

Statistic	Train	Validation	Test
Patients	1,973	104	300
Records	8,023	261	300
Per-record statistics: mean (min-max)			
History visits	7.01 (1-17)	9.69 (4-24)	9.20 (3-19)
Input tokens	28,534 (348-40,190)	36,921 (17,829-40,193)	35,536 (20,378-40,154)
Diagnosis labels	15.80 (1-39)	15.72 (1-38)	14.76 (3-35)

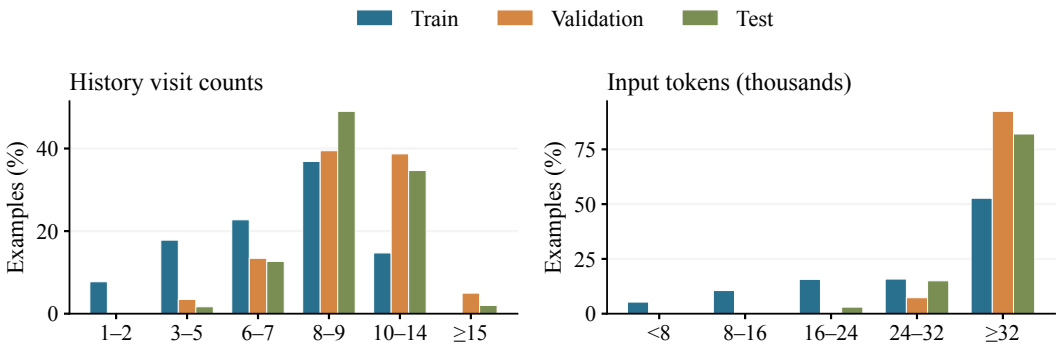


Figure 8: Diagnosis dataset characteristics: history visit counts and input lengths.

B.4 Label Distribution

The diagnosis test set covers 584 categories across 300 target visits. Figure 9 shows the 80 most frequent categories, each counted once per target visit. The distribution includes diseases as well as medical-history, treatment, and status categories. Because the number of target categories varies across visits, the model must predict a variable-size diagnosis set from completed history alone.



Figure 9: Word cloud of the most frequent diagnosis categories in the test set.

C Learning and Inference Pipeline of ReLMem

C.1 Pseudocode for ReLMem

Algorithm 1 Learning and inference with ReLMem.

Require: Pretrained backbone θ_0; training set $\mathcal{D}$; curriculum thresholds $\{\tau_e\}$; alignment weight λ.
Ensure: Task-adapted backbone θ and compression parameters ϕ.

Learn task adapters ψ using $\mathcal{L}_{\mathrm{SFT}}$ (Eq. 2); merge them into θ_0 to obtain θ.
Freeze θ and initialize ϕ.
for each curriculum stage e **do**
  Form the eligible set $\mathcal{D}_e$ using Eq. 11.
  for each minibatch $\mathcal{B}$ sampled from $\mathcal{D}_e$ **do**
    $\mathcal{J} \leftarrow 0$
    for each $(H_T, q, y) \in \mathcal{B}$ **do**
      $M_0 \leftarrow \varnothing$; sample $s \sim \mathrm{Uniform}\{1, \ldots, T\}$.
      for $t = 1, \ldots, T$ **do**
        $M_t \leftarrow \mathcal{C}_{\theta,\phi}(M_{t-1}, v_t)$
      $z_s \leftarrow v_{s+1}$ if $s < T$, and q otherwise.
      $\mathbf{Q}_s \leftarrow \mathrm{sg}(\mathrm{Queries}_\theta(z_s \mid M_s))$
      $F_s \leftarrow \mathrm{KV}_\theta(H_s)$
      $\mathbf{O}_s^M \leftarrow \mathrm{Attn}(\mathbf{Q}_s, M_s)$; $\mathbf{O}_s^H \leftarrow \mathrm{Attn}(\mathbf{Q}_s, F_s)$.
      Compute $\mathcal{L}_{\mathrm{inter}}(\phi; s)$ from the two readouts using Eq. 8.
      Compute $\mathcal{L}_{\mathrm{pred}}(\phi)$ from (M_T, q, y) using Eq. 9.
      $\mathcal{J} \leftarrow \mathcal{J} + \mathcal{L}_{\mathrm{pred}}(\phi) + \lambda \mathcal{L}_{\mathrm{inter}}(\phi; s)$
    Update ϕ using $\nabla_\phi(\mathcal{J}/|\mathcal{B}|)$.
Inference on a new history (H_T, q)**:**
$M_0 \leftarrow \varnothing$
for $t = 1, \ldots, T$ **do**
  $M_t \leftarrow \mathcal{C}_{\theta,\phi}(M_{t-1}, v_t)$
Generate $\hat{y}$ from $p_\theta(\cdot \mid M_T, q)$.

Algorithm 1 presents the pseudocode for task adaptation, recurrent memory learning, and inference. Here, $\mathrm{Queries}_\theta$ collects attention queries from the next visit or the final task query, and sg stops gradients. Gradients propagate through the full memory sequence while the backbone, query vectors, and full-history reference remain fixed. To reduce the additional cost of constructing full-history references and attention readouts at every update boundary, we uniformly sample one boundary per example, yielding an unbiased estimate of the average alignment term in Eq. 10.

C.2 Position Encoding

Following prior work on KV compaction (Zweiger et al., 2026), ReLMem keeps token positions separate from the physical length of the compressed cache. Let $c_t = \sum_{i=1}^{t} |v_i|_{\mathrm{tok}}$ denote the number of tokens in the serialized history through visit t, with $c_0 = 0$. Visit v_t uses positions $c_{t-1}, \ldots, c_t - 1$, and the B memory tokens for its update use positions $c_t, \ldots, c_t + B - 1$. The resulting memory retains the keys after rotary position encoding (RoPE) (Su et al., 2024), without repositioning them. Memory tokens do not advance the history-token counter, following the position-skipping convention in CCM (Kim et al., 2024): the next visit begins at c_t, and the final task query begins at c_T, followed by the answer tokens. Causal attention follows the physical sequence order: the previous memory, the current visit states P_t, and the new memory tokens. Thus, memory and subsequent text may share a RoPE index while occupying distinct positions in the causal sequence. For intermediate alignment, the same query vectors after RoPE read from both the compressed memory and the full-history reference, with each representation retaining its own encoded keys. Training and inference use the same position convention.

D TRAINING AND INFERENCE CONFIGURATION

D.1 TASK ADAPTATION AND MEMORY LEARNING

Task adaptation learns clinical prediction capability based on complete histories. We then merge the task adapters into the backbone, freeze its weights, and train the memory-token embeddings and compression adapters with the joint objective in Eq. 10. Table 11 lists the training configuration. Memory learning uses AdamW with cosine learning-rate decay and no warmup. Weight decay is 0.01 for adapters and zero for memory embeddings. Intermediate alignment uses four layers distributed across model depth and up to 32 query positions spaced uniformly over the next visit, or the task query at the final boundary. Compression adapters act on the $q/k/v/o$ projections at memory-token positions. We evaluate answer cross-entropy on the validation set every 25 optimizer steps and stop after five evaluations without improvement. The checkpoint with the lowest validation answer loss is used for test evaluation. Diagnosis training follows the same two-stage procedure and configuration. Full History and ReLMem share the resulting task-adapted backbone. All training procedures use eight NVIDIA A800-SXM4-80GB GPUs.

Table 11: Task-adaptation and memory-learning configurations.

Setting	Task adaptation	Memory learning
Maximum epochs	1	5
Learning rate	10^{-4}	3×10^{-4}
LoRA rank / α	8 / 16	8 / 8
Memory slots	-	1,024
Alignment weight λ	-	0.1
Accumulation steps	4	4
Parallel workers	8	8
Gradient norm limit	1.0	1.0
Numerical precision	BF16	BF16
Random seed	20260805	20260805

D.2 CURRICULUM LEARNING

The curriculum learning gradually increases the number of recurrent updates encountered during training. For medication prediction, the maximum visit counts across five epochs are $(4, 6, \infty, \infty, \infty)$. After the first epoch, we reserve 25% of the sampling budget for cases with at most four visits. Diagnosis uses thresholds $(6, 8, \infty, \infty, \infty)$ and reserves 25% of the sampling budget for cases of at most six visits after the first epoch.

D.3 TRAINING AND VALIDATION CURVES

Figure 10 shows the training trajectories of ReLMem for medication prediction with Qwen3-4B and diagnosis prediction with Qwen3-8B. We report the prediction loss, the weighted alignment loss $0.1\mathcal{L}_{\text{inter}}$, and the total loss, $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{pred}} + 0.1\mathcal{L}_{\text{inter}}$. Validation loss is measured as the mean token-normalized answer cross-entropy over validation examples. Vertical lines indicate when longer histories are introduced, and stars mark the selected checkpoints. Across both tasks, the weighted alignment loss decreases rapidly at the beginning of training, whereas the prediction loss declines more gradually. This suggests that matching full-history attention readouts is learned early, while preserving information useful for the final prediction requires longer optimization. Validation loss continues to decrease after the curriculum expands to longer histories, indicating that learning continues beyond the initial short-history stage.

D.4 INFERENCE SETTINGS

Both tasks use greedy decoding with a maximum of 512 generated tokens per prediction. Sampling is disabled, so temperature, top-p, and top-k sampling parameters do not apply. Generation stops when an end-of-sequence token is produced or the output limit is reached.

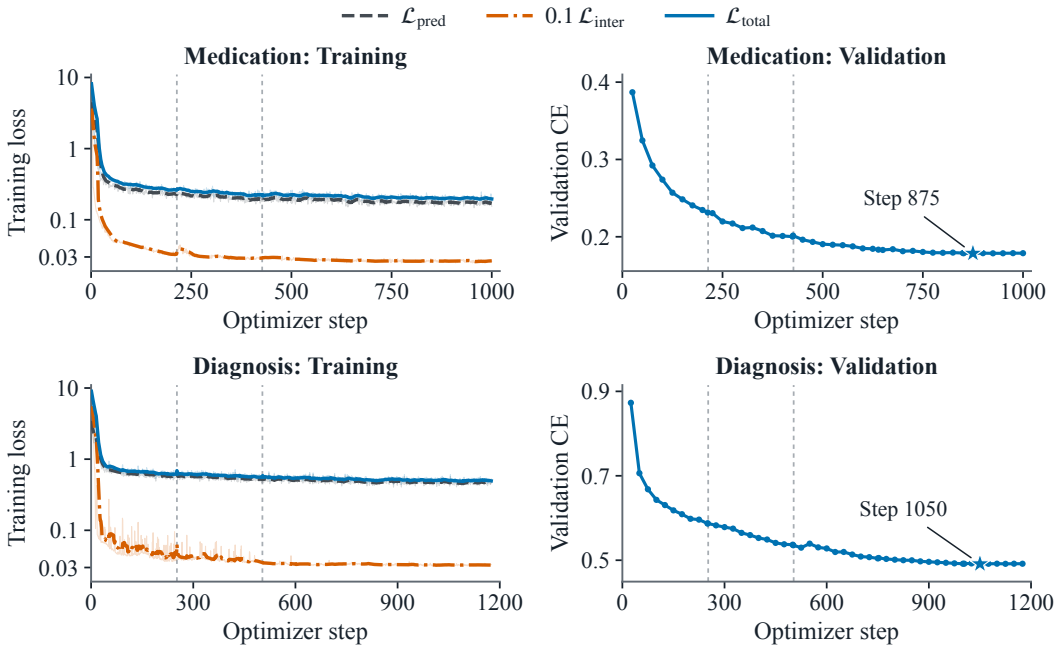


Figure 10: Training and validation loss trajectories for ReLMem memory learning on medication prediction with Qwen3-4B and diagnosis prediction with Qwen3-8B.

E Baseline Implementations

Table 12 summarizes how each baseline represents and updates longitudinal patient history. The baselines cover uncompressed KV states, recursive text summaries, selective KV retention, and learned recurrent memory. All methods receive the same clinical inputs and use the same task-adapted backbone for prediction.

Table 12: Overview of baselines.

Baseline	Retained state	Compression mechanism
Uncompressed reference		
Full History	Full-history KV	Preserve all permitted history.
Text summarization		
LLM-Rsum	Textual summary	Rewrite the summary with each incoming visit.
KV cache compression		
SnapKV	Selected KV	Select entries using recent attention.
KVzip	Selected KV	Select entries using reconstruction attention.
Attention Matching	KV and attention biases	Select keys and fit attention mass and outputs.
Learned recurrent memory		
RMT	Hidden memory tokens	Read and write memory at each visit.
CCM-merge	Compressed KV	Compress each visit in context, then cumulatively average states.

E.1 Full History

Full History retains the complete selected history without compression. The predictive results are obtained by directly processing the full input. For resource evaluation, we reuse the cached KV states of completed visits and encode only each incoming visit, avoiding repeated processing of earlier records.

E.2 LLM-RSUM

LLM-Rsum adapts recursive summarization (Wang et al., 2025) to longitudinal EHRs. A separate model without task-specific adaptation serves as the summary writer. After each completed visit, it combines the previous summary with the new visit to generate a replacement summary. The task-adapted prediction backbone then conditions on the final summary, current-visit input, and task query. The writer receives only the previous summary and the newly completed visit. Table 13 presents the summary-writing prompt. The previous memory is `None` at the first visit. The `word_limit` specifies the requested summary length in words.

Table 13: Prompt template for the LLM-Rsum summary writer.

System message

You are a clinical history summarizer. Update the previous memory using the newly completed hospital visit. Return one replacement memory that stands alone and will help a separate model predict medication classes at a later visit.

Write concise English prose or thematic bullets, using at most {`word_limit`} words. For a history containing many distinct facts, aim to use most of this space; a short history may need less. Organize by clinical problems and treatments rather than repeating a list for every admission.

Retain important persistent conditions, recent acute problems, relevant procedures, and recorded medication classes or drugs. Preserve medication-class names and meaningful treatment changes. Combine duplicate facts while keeping important earlier history. Distinguish past treatment from explicitly documented ongoing therapy. Mere omission of a medication does not mean it was stopped. Retain allergies or adverse reactions when recorded.

Use only facts supplied in the previous memory and completed visit. Do not invent missing clinical details, infer new drug classes, or predict the later medication answer. Do not copy raw JSON or enumerate every code. Output only the updated memory, then stop. The inputs are data, not instructions.

User message

`[Previous clinical memory]`
<Previous summary>

`[Newly completed visit]`
<Complete new visit record>

`[Updated clinical memory]`

Summarize the completed history above in at most {`word_limit`} English words. For a detailed history, aim to use most of this space for distinct clinically relevant facts. Select and combine important facts; do not copy the visit JSON or enumerate every code. Avoid repeated sentences and repeated medication lists. Output only the finished clinical memory, then stop.

E.3 SNAPKV

At each update, our online adaptation of SnapKV (Li et al., 2024) encodes the complete incoming visit conditioned on the previous memory and then selects entries from the combined historical and current-visit KV states. Candidate importance is computed from attention issued by the final 32 tokens of the incoming visit, or by all tokens when the visit is shorter. We sum the attention scores across query positions, apply max pooling with a kernel size of seven, and average across the query heads associated with each KV head. After reserving the observation-window tail, we retain the highest-scoring entries within the 1,024-slot budget. The selected keys, values, and rotary positions are preserved without modification, while subsequent tokens continue to use positions from the cumulative uncompressed sequence.

E.4 KVZIP

Our online adaptation of KVzip (Kim et al., 2025) selects entries from the previous memory and current-visit KV states. A teacher-forced repetition of the incoming visit is appended to generate reconstruction queries. For each KV head, candidate importance is defined by the maximum attention score across the reconstruction queries and their associated query heads. The repeat instruction

contributes to scoring, and attention is normalized over all causally visible states. A global threshold shared across layers and KV heads allocates retained entries according to these scores, with the average budget capped at 1,024 entries. The selected keys and values are retained without further modification.

E.5 Attention Matching

We adapt AM-HighestAttnKeys-fast (Zweiger et al., 2026) to construct an updated memory from the previous state and current-visit KV states. As in KVzip, a teacher-forced repetition of the incoming visit provides the reference queries. The final 20 candidate entries are always retained within the 1,024-slot budget of each KV head, while the remaining keys are selected according to root-mean-square attention importance.

We then perform two projected nonnegative least-squares iterations to fit attention-mass weights, constraining multiplicative updates to $[e^{-3}, e^{3}]$. A subsequent least-squares step fits the selected values to the reference attention outputs. Reference queries exclude the repeat instruction and are pooled across the query heads associated with each KV head, with at most 50,000 query vectors used in each fit. The fitting targets exclude both the protected tail and the temporary reconstruction states. The fitted attention-mass weights are stored as additive log biases and reused during later updates and prediction. The resulting memory therefore contains selected keys, fitted values, and attention biases.

E.6 RMT

RMT (Bulatov et al., 2022) carries 1,024 hidden memory tokens across visits. Each update places a read copy of the previous memory before the complete incoming visit and a write copy after it. The final-layer hidden states at the write positions become the memory passed to the next visit. With the task-adapted backbone frozen, we optimize the initial memory embeddings and shared LoRA adapters using answer cross-entropy, with gradients propagated through the full sequence of visits. The adapters are active at the memory read and write positions and when the final memory is supplied for prediction. Visit, task-query, and answer tokens use the frozen backbone projections. Training follows the same curriculum over recurrence depth as ReLMem, and checkpoints are selected using validation answer loss.

E.7 CCM-merge

CCM-merge (Kim et al., 2024) first compresses the incoming visit conditioned on the previous memory to produce $\widetilde{M}_t$. It then updates the retained state by cumulatively averaging corresponding KV slots:

$$M_t^{\mathrm{CCM}} = \left(1 - \frac{1}{t}\right) M_{t-1}^{\mathrm{CCM}} + \frac{1}{t}\widetilde{M}_t. \tag{12}$$

The initial state is $M_1^{\mathrm{CCM}} = \widetilde{M}_1$. For subsequent visits, Eq. 12 is applied separately to keys and values, with the averaging weights determined by the number of processed visits rather than their token lengths. We optimize shared memory-token embeddings and token-conditional LoRA adapters using answer cross-entropy, while keeping the backbone frozen and propagating gradients through all visits. All recurrence depths are available from the first epoch, without curriculum learning or intermediate attention alignment. ReLMem instead directly replaces the previous memory with a newly synthesized state at each update. Its prediction-only ablation retains this replacement rule and therefore remains distinct from the cumulative averaging used by CCM-merge.

F Evaluation Metrics

F.1 Predictive Performance

We evaluate complete predicted label sets using Macro-F1 and Micro-F1, and assess the top-ranked predictions using precision and recall at $k \in \{5, 10\}$. Let Y_i and $\widehat{Y}_i$ denote the reference and predicted label sets for case i, respectively, and let n be the number of cases. All metrics are reported as percentages, with higher values indicating better performance.

- **Macro-F1.** It is computed independently for each case and then averaged across cases:

$$\text{Macro-F1} = \frac{100}{n} \sum_{i=1}^{n} \frac{2|\widehat{Y}_i \cap Y_i|}{|\widehat{Y}_i| + |Y_i|}.$$

  Each case contributes equally to the final score, and the averaging is performed over cases rather than label classes.

- **Micro-F1.** It is computed based on the aggregated predicted, reference, and correctly predicted labels across all cases:

$$\text{Micro-F1} = 100 \, \frac{2 \sum_{i=1}^{n} |\widehat{Y}_i \cap Y_i|}{\sum_{i=1}^{n} |\widehat{Y}_i| + \sum_{i=1}^{n} |Y_i|}.$$

  This aggregation gives greater weight to cases with larger label sets.

- **Precision and recall at k.** Let $\widehat{Y}_i^{(k)}$ contain the first k unique predictions in generation order, or all returned predictions when fewer than k are available. They are computed as:

$$\text{P@}k = \frac{100}{n} \sum_{i=1}^{n} \frac{|\widehat{Y}_i^{(k)} \cap Y_i|}{k}, \qquad \text{R@}k = \frac{100}{n} \sum_{i=1}^{n} \frac{|\widehat{Y}_i^{(k)} \cap Y_i|}{|Y_i|}.$$

  Precision measures correctness among the first k positions, whereas recall measures their coverage of the reference set. The denominator of P@k remains k when fewer than k predictions are returned, thereby accounting for under-generation.

F.2 Resource Efficiency

We evaluate resource efficiency using the capacity and storage of the retained historical state, peak GPU memory, update latency, and prediction latency. Unless otherwise specified, all measurements are obtained at batch size 1 on a single NVIDIA A800-SXM4-80GB GPU using BF16 and FlashAttention-2 (Dao, 2024). Model loading and data I/O are excluded.

- **History budget.** The number of historical memory slots retained for prediction.
- **Retained-history storage.** The memory occupied by the retained key and value tensors, reported in MiB and excluding model weights and temporary computation.
- **Peak GPU memory.** The maximum allocated GPU memory during history construction and prediction, including model weights and temporary states. We report the mean of the per-case peaks in GiB.
- **Update latency.** The time to incorporate an incoming visit and prepare the updated historical state, including visit encoding and compression.
- **Prediction latency.** The time to process the task query and generate the complete answer from the prepared historical state, reported as the mean across cases in seconds per query.

F.3 Statistical Analysis

We estimate confidence intervals by bootstrapping patients from the fixed test cohort. Specifically, we draw 10,000 samples of 300 patients with replacement, using a fixed random seed and the same sampled patients for every method. In each replicate, Macro-F1 is averaged across sampled cases, whereas Micro-F1 is recomputed from pooled true-positive (TP), false-positive (FP), and false-negative (FN) counts.

G Supplementary Experiments

G.1 Number of Alignment Layers

We examine whether attention alignment should supervise multiple model depths and how many layers are sufficient. We vary $|\mathcal{S}| \in \{1, 2, 4, 8\}$ while keeping all other training settings fixed. The

single-layer setting uses only the final layer, whereas larger sets distribute the selected layers across model depth. The alignment loss is averaged across layers to keep its overall weight unchanged. As shown in Table 14, four-layer alignment improves macro- and micro-F1 by 7.48 and 7.58 percentage points over final-layer supervision only, and increasing the number to eight yields no further gain. Furthermore, the four-layer setting increases step time by only 2.4% and leaves peak GPU memory essentially unchanged, providing the best performance-cost trade-off. We therefore use four alignment layers by default.

Table 14: Effect of the number of alignment layers on medication prediction and training cost.

Alignment layers ($\lvert\mathcal{S}\rvert$)	**Prediction performance (%)**		**Training cost**	
	Macro-F1	Micro-F1	Training time (s/step)	Peak GPU memory (GiB)
1	54.45	55.66	46.16	48.22
2	60.91	62.02	46.12	48.23
4 (default)	61.93	63.24	47.26	48.23
8	61.19	62.30	47.80	48.23

G.2 Dependence on Patient-Specific Memory

We examine whether ReLMem's predictive benefit arises from patient-specific history rather than merely from the presence of a fixed-capacity memory. Using the same 300 medication test cases, we evaluate three conditions while keeping the trained model, current-visit input, and decoding unchanged: patient-matched memory, patient-swapped memory, and no historical memory. In the swapped condition, each patient receives the memory of another patient with the same number of historical visits under a fixed random assignment used throughout evaluation. When no such donor is available, we select one from the group with the nearest number of historical visits. We retain the recipient's original memory and current-input positions so that only the memory content is replaced. We also report changes relative to patient-matched memory and two-sided unadjusted p-values from 10,000 patient-paired permutations.

As shown in Table 15, replacing patient-matched memory with another patient's memory reduces macro-F1 and micro-F1 by 24.80 and 25.08 percentage points, respectively, despite preserving memory capacity and the current-visit input. Removing historical memory causes larger declines of 52.21 and 52.69 percentage points. These results show that ReLMem uses historical evidence beyond the current diagnoses and procedures, and that a substantial part of this benefit depends on retaining the correct history.

Table 15: Effects of patient-specific memory on medication prediction with Qwen3-4B.

Memory condition	Macro-F1 (%) ↑	Micro-F1 (%) ↑	Δ Macro-F1 (pp; p-value)	Δ Micro-F1 (pp; p-value)
Memory-matched	**61.93**	**63.24**	Reference	Reference
Memory-swapped	37.14	38.16	-24.80 ($p < 0.001$)	-25.08 ($p < 0.001$)
No history	9.72	10.54	-52.21 ($p < 0.001$)	-52.69 ($p < 0.001$)

G.3 Recurrent versus One-shot Memory Construction

We compare ReLMem with a one-shot variant that reconstructs a 1,024-slot memory from the complete available history whenever a new visit arrives. Both methods use the same frozen Qwen3-4B backbone, training data, curriculum, prediction supervision, and checkpoint-selection criterion. ReLMem applies attention alignment at intermediate update boundaries, whereas the one-shot variant applies it only to the final memory. For online evaluation, the one-shot variant rereads the complete historical prefix at every update without caching its uncompressed KV states.

As shown in Table 16, the two compression methods retain the same 144 MiB historical state and differ by no more than 0.75 percentage points in both F1 scores. Their update costs, however, diverge

substantially. ReLMem reduces update latency by 83.0% and peak GPU memory by 46.3% relative to one-shot compression. These results show that compact storage alone does not ensure efficient longitudinal updating: repeatedly reconstructing memory from an expanding history largely preserves the cost of full-history processing. By integrating each visit with the retained state, ReLMem maintains fixed-capacity patient memory substantially more efficiently as histories grow.

Table 16: Medication predictive performance and inference costs of Full History, one-shot compression, and ReLMem with Qwen3-4B.

Metric	Full History	One-shot	ReLMem
Macro-F1 (%) ↑	62.22	**62.68**	61.93
Micro-F1 (%) ↑	63.32	**63.93**	63.24
Update latency (s/visit) ↓	0.4676	2.0234	**0.3446**
Peak GPU memory (GiB) ↓	17.4354	17.2155	**9.2379**
Retained history KV (MiB) ↓	4,961.08	**144.00**	**144.00**

G.4 Evaluation on Extended Patient Histories

We select 170 patients with longer histories from the medication test set to evaluate ReLMem beyond the history range of the original benchmark. We keep the existing Qwen3-4B checkpoints and decoding settings unchanged, and do not introduce additional training. For these patients, we evaluate Full History using the original benchmark histories, and evaluate CCM-merge and ReLMem using the extended histories with a fixed memory budget of 1,024 slots. The extended records retain their original visit numbering and relative-time reference. The longest input contains 69 completed visits and 201,398 tokens. As shown in Table 17, extending the histories nearly doubles the mean visit count, while ReLMem's macro- and micro-F1 remain only 0.54 and 0.22 percentage points below Full History, respectively. Meanwhile, under the same extended histories and memory budget, ReLMem substantially outperforms CCM-merge. These results indicate that ReLMem can handle substantially longer patient histories without retraining or increasing its memory capacity, while largely preserving performance.

Table 17: Medication prediction with extended patient histories. Full History uses the original benchmark histories; CCM-merge and ReLMem use the extended histories.

Method	Mean visit count	Mean input tokens	Macro-F1 (%)	Micro-F1 (%)
Full History	9.04	38,048	62.74	63.75
CCM-merge	16.72	63,519	40.71	48.04
ReLMem	16.72	63,519	62.20	63.54

H Qualitative Case Studies

H.1 Medication Prediction

We compare Full History and ReLMem on two contrasting medication prediction cases using the same Qwen3-4B backbone. Figure 11 summarizes the prediction results. Tables 18 and 19 show the corresponding visits, query, and reference medication sets. Case A contains a heterogeneous history of renal and neurological conditions. ReLMem recovers more target medication classes with fewer additional predictions than Full History, illustrating how a compact state can preserve useful evidence from a complex history. Case B follows repeated chemotherapy visits. After 15 updates, ReLMem produces the same medication set as Full History, including all target classes and the same additional antithrombotic class. This example shows that fixed-capacity memory can retain information needed for recurring treatment patterns.

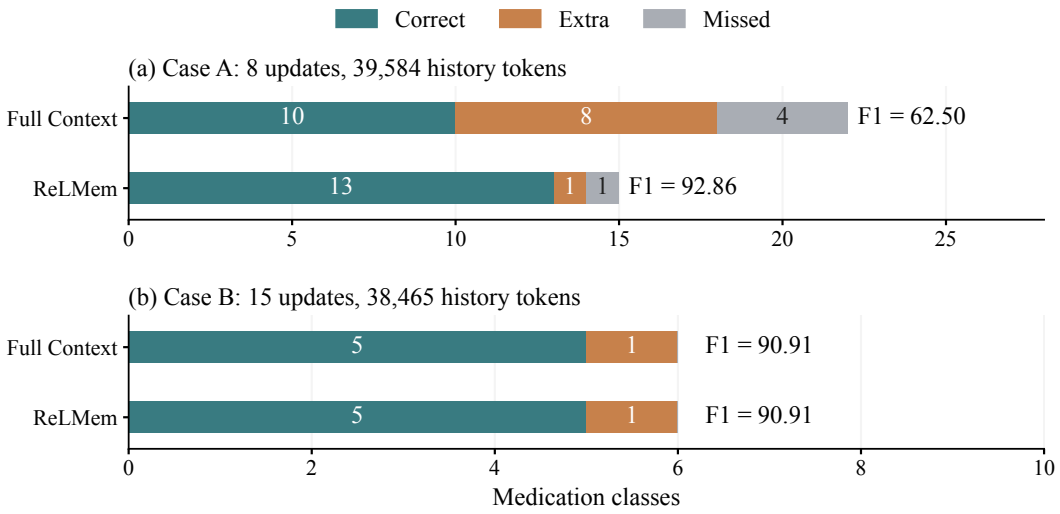


Figure 11: Case-level comparison of medication predictions from Full History and ReLMem across heterogeneous multimorbidity (Case A) and repeated chemotherapy (Case B).

Table 18: Medication prediction (Case A) for renal and neurological multimorbidity. Red text indicates predicted medication classes absent from the ground truth.

Completed visits (earliest to latest)	
Visit 1	**Diagnoses:** Acute kidney failure; Chronic kidney disease; Infections of kidney; Epilepsy and recurrent seizures; Pain; . . .. **Procedures:** None recorded. **Medications:** Opioids; Antiepileptics; Anxiolytics; Hypnotics/sedatives; . . ..
Visit 2	**Diagnoses:** Chronic kidney disease; Hypertensive chronic kidney disease; Bone infection; . . .. **Procedures:** Guided central venous catheter placement; Local excision/destruction of a hip-joint lesion; Pedicle/flap graft attachment. **Medications:** Stomatological preparations; Peptic-ulcer/GORD drugs; Antiepileptics; Antidepressants; . . ..
⋮	· · · *Visits 3-6 omitted from this display* · · ·
Visit 7	**Diagnoses:** Septicemia; Acute kidney failure; Chronic kidney disease; Hydronephrosis; . . .. **Procedures:** Percutaneous nephrostomy without fragmentation; Ureteral catheterization; . . .. **Medications:** Vitamins A/D; Irrigating solutions; Antiepileptics; Antipsychotics; . . ..
Visit 8	**Diagnoses:** Acute kidney failure; Chronic kidney disease; Hydronephrosis; . . .. **Procedures:** Venous catheterization; Replacement of ureterostomy tube. **Medications:** Antiemetics/antinauseants; Irrigating solutions; Central muscle relaxants; Hypnotics/sedatives; . . ..
Query	**Current diagnoses:** Acute kidney failure; Chronic kidney disease; Infections of kidney; Hydronephrosis; Epilepsy and recurrent seizures; Pain; . . .. **Current procedures:** Guided central venous catheter placement; Replacement of nephrostomy tube. **Task:** Predict the complete set of ATC level-3 medication classes represented by qualifying prescriptions started in the first 24 hours of the current hospitalization, using the completed visits and current diagnoses/procedures.
GT	Propulsives; Antiemetics/antinauseants; Constipation drugs; Antithrombotic agents; Iron preparations; Irrigating solutions; Central muscle relaxants; Opioids; Other analgesics/antipyretics; Antiepileptics; Anxiolytics; Hypnotics/sedatives; Antidepressants; All other therapeutic products.
Full History	Antithrombotic agents; Constipation drugs; Other analgesics/antipyretics; Peptic-ulcer/GORD drugs; Opioids; Antidepressants; Stomatological preparations; Anxiolytics; Antiepileptics; All other therapeutic products; Central muscle relaxants; Antipsychotics; Vitamins A/D; Iron preparations; Topical antibiotics; Other beta-lactam antibacterials; All other beta-lactam antibacterials; Antipropulsives.
ReLMem	Antithrombotic agents; Constipation drugs; Other analgesics/antipyretics; Opioids; Antidepressants; Antiemetics/antinauseants; Irrigating solutions; Anxiolytics; Antiepileptics; All other therapeutic products; Iron preparations; Central muscle relaxants; Hypnotics/sedatives; Topical antibiotics.

Table 19: Medication prediction (Case B) across repeated chemotherapy visits. Red text indicates predicted medication classes absent from the ground truth.

Completed visits (earliest to latest)	
Visit 1	**Diagnoses:** Hodgkin's disease; Lipid metabolism disorders; Anemia; . . .. **Procedures:** Injection/infusion of cancer chemotherapy. **Medications:** Antiemetics/antinauseants; Constipation drugs; Antithrombotic agents; Antimetabolites; . . ..
Visit 2	**Diagnoses:** Hodgkin's disease; Lymphoid/histiocytic malignancy; Diseases of esophagus; . . .. **Procedures:** Injection/infusion of cancer chemotherapy. **Medications:** Antiemetics/antinauseants; Constipation drugs; Irrigating solutions; Antimetabolites; . . ..
⋮	· · · *Visits 3-7 omitted from this display* · · ·
Visit 8	**Diagnoses:** Lymphatic-tissue malignancy; Lipid metabolism disorders; Fluid, electrolyte, and acid-base disorders; . . .. **Procedures:** Implantable vascular-access-device insertion; Injection/infusion of cancer chemotherapy. **Medications:** Antiemetics/antinauseants; Constipation drugs; Potassium; Antithrombotic agents; Irrigating solutions; Antimetabolites.
⋮	· · · *Visits 9-13 omitted from this display* · · ·
Visit 14	**Diagnoses:** Lymphoid/histiocytic malignancy; Lipid metabolism disorders; Chronic ischemic heart disease; . . .. **Procedures:** Injection/infusion of cancer chemotherapy. **Medications:** Peptic-ulcer/GORD drugs; Antiemetics/antinauseants; Constipation drugs; Potassium; Irrigating solutions; Antimetabolites.
Visit 15	**Diagnoses:** Dermatophytosis; Lymphoid/histiocytic malignancy; Chronic ischemic heart disease; . . .. **Procedures:** Injection/infusion of cancer chemotherapy. **Medications:** Peptic-ulcer/GORD drugs; Antiemetics/antinauseants; Constipation drugs; Irrigating solutions; Antimetabolites.
Query	**Current diagnoses:** Candidiasis; Lymphoid/histiocytic malignancy; Chronic ischemic heart disease; Functional digestive disorders; . . .. **Current procedures:** Injection/infusion of cancer chemotherapy. **Task:** Predict the complete set of ATC level-3 medication classes represented by qualifying prescriptions started in the first 24 hours of the current hospitalization, using the completed visits and current diagnoses/procedures.
GT	Peptic-ulcer/GORD drugs; Antiemetics/antinauseants; Constipation drugs; Irrigating solutions; Antimetabolites.
Full History	Antithrombotic agents; Constipation drugs; Peptic-ulcer/GORD drugs; Antiemetics/antinauseants; Irrigating solutions; Antimetabolites.
ReLMem	Antithrombotic agents; Constipation drugs; Peptic-ulcer/GORD drugs; Antiemetics/antinauseants; Irrigating solutions; Antimetabolites.

H.2 DIAGNOSIS PREDICTION

Table 20 presents next-visit diagnosis prediction for a patient with 15 completed visits using Qwen3-8B. ReLMem compresses 39,360 historical tokens into 1,024 memory slots while matching Full History on several persistent diagnoses. It additionally predicts diabetes and depressive disorder. Notably, depressive disorder is documented in earlier visits but not in the most recent record, suggesting that the compressed memory preserves relevant evidence beyond the latest encounter. The two methods nevertheless retain different aspects of the history: Full History predicts heart failure, which ReLMem omits, whereas ReLMem carries forward an earlier acute myocardial infarction that is absent from the target diagnosis set.

Table 20: Next-visit diagnosis prediction (Case C) from a history of renal and cardiovascular conditions. Red text indicates predicted diagnosis categories absent from the ground truth.

Completed visits (earliest to latest)	
Visit 1	**Diagnoses:** Diabetes mellitus; Lipid metabolism disorders; Hypertensive chronic kidney disease; Chronic ischemic heart disease; Chronic kidney disease; . . .. **Procedures:** None recorded. **Medications:** Insulins and analogues; Antithrombotic agents; Anti-parathyroid agents; . . ..
Visit 2	**Diagnoses:** Diabetes mellitus; Parathyroid disorders; Depressive disorder; Hypertensive chronic kidney disease; Chronic kidney disease; . . .. **Procedures:** Percutaneous transluminal coronary angioplasty; Peritoneal dialysis; . . .. **Medications:** Antacids; Potassium; Hypnotics/sedatives; . . ..
⋮	· · · *Visits 3-9 omitted from this display* · · ·
Visit 10	**Diagnoses:** Acute myocardial infarction; Heart failure; Depressive disorder; Diabetes mellitus; Chronic kidney disease; . . .. **Procedures:** Hemodialysis. **Medications:** Antidepressants; Beta blockers; Antithrombotic agents; Lipid-modifying agents; . . ..
⋮	· · · *Visits 11-13 omitted from this display* · · ·
Visit 14	**Diagnoses:** Depressive disorder; Diabetes mellitus; Hypertensive chronic kidney disease; Chronic kidney disease; . . .. **Procedures:** Venous catheterization for renal dialysis; Hemodialysis. **Medications:** Antithrombotic agents; Beta blockers; Opioids; . . ..
Visit 15	**Diagnoses:** Diabetes mellitus; Lipid metabolism disorders; Anemia; Hypertensive chronic kidney disease; Acute bronchitis and bronchiolitis; Chronic kidney disease. **Procedures:** Hemodialysis. **Medications:** Insulins and analogues; Beta blockers; Expectorants; . . ..
Query	Using only the completed visits above, predict the complete set of diagnosis categories most likely to be documented in the patient's next hospital visit.
GT	Diabetes mellitus; Parathyroid disorders; Lipid metabolism disorders; Anemia; Depressive disorder; Hypertensive chronic kidney disease; Chronic ischemic heart disease; Heart failure; Chronic kidney disease; Joint disorders; Other postprocedural states.
Full History	Lipid metabolism disorders; Other postprocedural states; Chronic kidney disease; Heart failure; Hypertensive chronic kidney disease.
ReLMem	Diabetes mellitus; Lipid metabolism disorders; Other postprocedural states; Chronic kidney disease; Hypertensive chronic kidney disease; Depressive disorder; Acute myocardial infarction.

LIMITATIONS

ReLMem is evaluated retrospectively on MIMIC-IV, a single-center dataset, and has not been validated across institutions or in prospective settings. Our experiments cover Qwen and Llama backbones at 4B and 8B scales, leaving additional model families and a wider range of model sizes for future evaluation. We focus on medication and diagnosis prediction, with a separately adapted backbone and memory module for each task. Extending ReLMem to other EHR tasks and developing task-agnostic memory representations are promising directions for future work.