DEEP FUSION MEMORY: A CHUNK-LEVEL PARAMETRIC MEMORY FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models need effective ways to acquire and use external knowledge beyond what they retain from pretraining. Continued pretraining risks catastrophic forgetting, while retrieval at inference time can add substantial overhead. External parametric memory avoids online retrieval, but constructing its training data requires extensive offline search, and output-level interpolation provides only shallow fusion. We introduce Deep Fusion Memory (DFM), a parametric memory architecture that reduces this construction workload and incorporates learned memory into intermediate hidden states. A causal memory predictor learns to generate memory from the preceding context, using retrieved chunk embeddings from the corpus as supervision. The predicted memory is shared across tokens within each chunk and fused into their hidden states through gated cross-attention. Memory prediction and fusion are trained end to end with the backbone frozen, and inference requires no explicit retrieval. Compared with existing parametric memory approaches, which often construct token-level memory targets, DFM requires $32\times$ fewer retrieval queries and $2048\times$ fewer retrieved items to construct its training data. Across six QA tasks, DFM performs better over the strongest baseline by 12.71% on Mistral-7B-v0.3 and 11.19% on Qwen3-8B-Base. Beyond QA, domain adaptation yields average gains of 6.51 and 4.38 points over Qwen3-4B-Base and Qwen3-8B-Base, respectively.

1 INTRODUCTION

Large language models (LLMs) (Yang et al., 2025; Liu et al., 2025; Team et al., 2025; Bai et al., 2026) have advanced rapidly in language understanding, generation, and reasoning. Their growing capabilities support knowledge-intensive applications, but reliable answers require facts that can be missing, poorly retained, or outdated in pretrained parameters (Lewis et al., 2020). Knowledge injection addresses this gap by incorporating information from an external corpus. A central challenge is to incorporate this knowledge effectively while preserving existing capabilities and keeping knowledge acquisition and inference costs manageable.

Existing approaches face different trade-offs in addressing this challenge. Continued pretraining (CPT) (Gururangan et al., 2020) learns from an additional corpus through language modeling, while LoRA (Hu et al., 2021) offers a parameter-efficient way to adapt the model. These approaches keep inference self-contained, but adaptation can cause catastrophic forgetting of previously acquired capabilities (Biderman et al., 2024). Non-parametric retrieval instead provides external evidence during generation. RAG (Lewis et al., 2020) conditions on retrieved passages, and RETRO (Borgeaud et al., 2022) incorporates retrieved chunks within the decoder. These methods provide access to corpus knowledge without first learning it in model parameters, but datastore search and evidence processing can add substantial inference latency (Cao et al., 2026).

Parametric memory learns from retrieved knowledge offline, enabling inference without explicit retrieval. MLP Memory (Wei et al., 2026a) and Memory Decoder (Cao et al., 2026) learn to predict retrieval-derived next-token distributions. However, this inference advantage comes with substantial offline costs. Constructing token-level training targets requires retrieval at individual token positions across the corpus, resulting in substantial data construction costs (Wei et al., 2026b). Training memory modules on these targets further incurs significant computational cost. Moreover, these methods rely on shallow fusion through output-probability interpolation, without incorporating memory into

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.

the decoder's hidden states. Intuitively, integrating memory into intermediate hidden states allows subsequent layers to further process and combine it with the evolving context. The broader challenge is to enable language models to acquire external knowledge efficiently and use it effectively.

In this work, we introduce Deep Fusion Memory (DFM), a parametric memory architecture that combines efficient knowledge acquisition with deep fusion within the decoder (Figure 1). At a high level, DFM shares memory supervision across multiple token predictions while allowing each token to use the learned memory according to its context. We jointly train a memory predictor and a memory fusion module, using retrieved target memory to supervise the predictor and the standard language modeling objective to train both modules. In this way, external knowledge is not only stored in the predictor's parameters, but also injected into intermediate hidden states through the fusion module, allowing subsequent decoder layers to utilize the memory during generation. The LLM backbone remains frozen throughout, and memory prediction is performed at the chunk level.

Our experiments demonstrate improvements in both efficiency and downstream performance. Compared with existing parametric memory approaches that construct token-level memory targets, DFM requires $32\times$ fewer retrieval queries and $2048\times$ fewer retrieved items to construct its training data. DFM achieves the highest average score across six QA tasks on both backbones, outperforming the strongest baseline by 12.71% on Mistral-7B-v0.3 and 11.19% on Qwen3-8B-Base. Beyond QA, adaptation across code, mathematics, and law yields average gains of 6.51 and 4.38 points over Qwen3-4B-Base and Qwen3-8B-Base, respectively. Ablations further highlight the importance of fusion placement and gate initialization, while showing comparatively small changes in average QA performance across the tested chunk sizes and numbers of memory slots.

Our contributions are summarized as follows:

- We introduce **DFM, a parametric memory architecture with deep fusion**, which integrates learned corpus knowledge into intermediate hidden states so that subsequent decoder layers can process memory together with the evolving context. Inference requires no explicit retrieval.
- We reduce **the overhead of supervision construction and memory training** through chunk-level supervision and end-to-end training with a frozen backbone. For training data construction, DFM requires $32\times$ fewer retrieval queries and $2048\times$ fewer retrieved items than existing parametric memory approaches based on token-level targets.
- We demonstrate **improved QA performance and domain adaptation**. Across six QA tasks, DFM performs better over the strongest baseline by 12.71% on Mistral-7B-v0.3 and 11.19% on Qwen3-8B-Base. Across code, mathematics, and law, average gains over Qwen3-4B-Base and Qwen3-8B-Base reach 6.51 and 4.38 points, respectively.

2 Related Work

Model Adaptation. Continued pretraining adapts language models to additional corpora through further language modeling (Gururangan et al., 2020), while LoRA (Hu et al., 2021) offers a parameter-efficient way to adapt the model. Such adaptation can cause catastrophic forgetting of previously acquired capabilities (Biderman et al., 2024). Deep Fusion Memory keeps the backbone frozen and learns an added memory module from retrieval-derived targets, jointly training memory prediction and fusion.

Retrieval-Augmented Language Models. Non-parametric memory exposes corpus information through several interfaces. kNN-LM retrieves context–token pairs and interpolates a neighbor-derived next-token distribution with the language model output (Khandelwal et al., 2019), whereas RAG conditions generation on retrieved passages (Lewis et al., 2020). RETRO combines chunk retrieval with cross-attention inside the Transformer (Borgeaud et al., 2022). xRAG projects a retrieved document embedding into a single token representation prepended to the query, reducing the context passed to the generator (Cheng et al., 2024). These methods retain an external retrieval dependency at inference. Deep Fusion Memory predicts continuous memory parametrically and uses it inside the decoder without online search.

Parametric Memory. MLP Memory trains an MLP to predict kNN token distributions from language model representations (Wei et al., 2026a). Memory Decoder learns a similar distributional target with a standalone decoder (Cao et al., 2026). Memory Decoder at Scale extends this approach

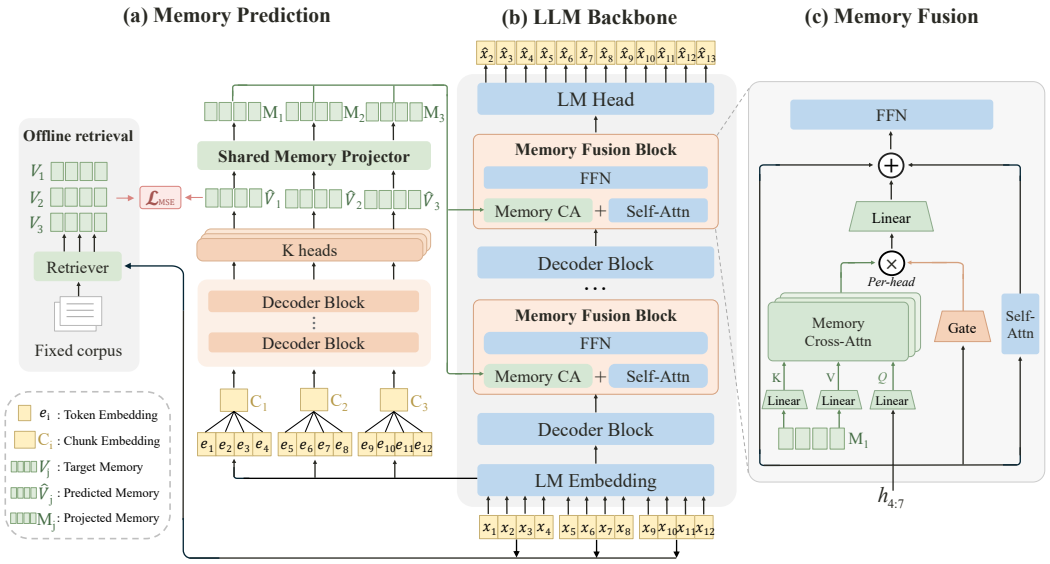

Figure 1: Overview of Deep Fusion Memory. **(a) Memory Prediction.** The memory predictor maps pooled chunk embeddings to predicted memory, supervised by offline target memory. A shared projector maps predicted memory to the decoder's hidden space. **(b) LLM Backbone.** Selected decoder layers incorporate memory for next-token prediction. **(c) Memory Fusion.** Memory predicted from the preceding context supports token predictions in the next chunk. Per-head gates control the memory cross-attention contribution alongside self-attention. Inference requires no retrieval.

to large-scale general and domain memory pretraining (Wei et al., 2026b). MemSFT specializes frozen backbones through a domain memory and a learned token-wise router over the backbone and memory output distributions (Wang et al., 2026). These approaches avoid explicit retrieval at inference, but token-level retrieval incurs substantial data construction costs, and output-probability interpolation provides only shallow fusion. Deep Fusion Memory reduces the cost of supervision construction by sharing target memory across tokens within each chunk and fuses predicted memory into hidden states through gated cross-attention.

3 Deep Fusion Memory

Deep Fusion Memory consists of a causal chunk-level memory predictor and memory cross-attention modules inserted into selected decoder layers (Figure 1). We construct target memory offline, then jointly train the added modules with language-modeling and memory-prediction objectives while keeping the backbone frozen. At inference, the predictor supplies memory directly from the input sequence.

3.1 Training data construction

We partition each training sequence into non-overlapping chunks $X_0, \ldots, X_{J-1}$ of C tokens. For each chunk X_j with $j \geq 1$, an offline retriever encodes the preceding chunk X_{j-1} and retrieves its top-K neighbors from a corpus datastore. Each retrieved chunk is represented by a fixed dense embedding produced by a frozen text encoder. Stacking these embeddings in retrieval-rank order gives the *target memory* $V_j \in \mathbb{R}^{K \times d_m}$, where d_m is the embedding dimension. This ordering aligns the target memory with the predictor's output slots.

Each training example pairs the original token sequence with its chunk-level target memory. Each target memory supervises the predictor output shared across token predictions in X_j. Predictions for the first chunk use no memory, and this chunk is excluded from memory supervision because it has no preceding-chunk query. Target memory is prepared offline and remains fixed during model training.

3.2 TRAINING

Memory Predictor. The predictor generates K continuous memory vectors for each chunk from the preceding context (Figure 1(a)). Let $e_i = E_\theta(x_i)$ denote the embedding of token x_i in the frozen backbone. For $j \geq 1$, we mean-pool the token embeddings in the preceding chunk X_{j-1} and, when needed, apply a learned linear projection to match the input width of the memory predictor. We denote the resulting chunk embedding by C_j. The predictor uses the prefix of these embeddings to produce memory for X_j:

$$\widehat{V}_j = \mathrm{Predictor}_\phi(C_{\leq j}). \tag{1}$$

The first input C_0 is a masked placeholder. The predictor combines a causal Transformer with K separate linear output heads, one per memory slot. The *predicted memory* $\widehat{V}_j \in \mathbb{R}^{K \times d_m}$ stacks the vectors from these heads in retrieval-rank order. Causal attention lets the prediction for X_j use $C_1, \ldots, C_j$, which summarize chunks $X_0, \ldots, X_{j-1}$.

Memory Fusion. Let P_ψ denote the shared memory projector in Figure 1(a). It maps predicted memory into the decoder hidden space, yielding the *projected memory* $M_j = P_\psi(\widehat{V}_j) \in \mathbb{R}^{K \times d}$, where d is the hidden width. The projector applies LayerNorm followed by two linear layers with a GELU activation independently to each slot. The resulting M_j is reused across the selected decoder layers.

Consider one selected decoder layer in Mistral or Qwen3 (Figure 1(c)), omitting the layer index for clarity. Let $z_t = \mathrm{Norm}(h_t)$ be the normalized token state used by self-attention. The schematic omits this normalization step. Let j denote the chunk containing the next token predicted at position t. Let $o_{t,a}$ denote the scaled dot-product cross-attention output of head a, with queries projected from z_t and keys and values from M_j. Each of the H heads applies a softmax over the K slots. A learned gate controls the memory contribution of each head:

$$g_t = \sigma(W_g z_t + b_g), \qquad u_t = W_O \,\mathrm{Concat}_{a=1}^{H}(g_{t,a}\, o_{t,a})\,. \tag{2}$$

The sigmoid gate $g_t \in (0,1)^H$ assigns one scalar to each token and head, shared across all channels in the head. It scales the head output before concatenation and output projection. Attention selects information from the shared memory for each token, while the gate controls how much each head contributes. For token predictions in the first chunk, the memory update is zero.

The memory branch runs in parallel with causal self-attention. Both outputs enter the same residual update:

$$\widetilde{h}_t = h_t + \mathrm{SelfAttn}(z_{\leq t}) + u_t. \tag{3}$$

The updated state $\widetilde{h}_t$ then passes through the original decoder feed-forward sublayer, including the associated normalization and residual connection. The memory predictor and memory projector are shared across fusion layers. Each fusion layer uses layer-specific memory-attention and gate projections, separate from the self-attention parameters. Other decoder layers retain their original computation.

Memory alignment follows next-token prediction. With zero-based token positions, the state at t predicts x_{t+1} and uses chunk index $j = \lfloor (t+1)/C \rfloor$. At a chunk boundary, the preceding chunk is complete before the pooled query is used. Together with the memory predictor, this ensures that memory generation uses only the input available for the next-token prediction.

Joint Training Objective. We jointly train the predictor and fusion modules end to end using next-token cross-entropy, while additionally supervising the memory predictor with a mean squared error loss between predicted and target memory:

$$\mathcal{L} = \mathcal{L}_{\mathrm{LM}} + \beta \mathcal{L}_{\mathrm{MSE}}, \qquad \mathcal{L}_{\mathrm{MSE}} = \frac{\sum_{j=1}^{J-1} \|\widehat{V}_j - V_j\|_F^2}{(J-1)Kd_m}. \tag{4}$$

The MSE excludes X_0, which has no preceding chunk. We set $\beta = 1$. The MSE is computed directly against the L2-normalized target embeddings, without renormalizing the predictions. The language model consumes predicted memory throughout training. Target memory is used only for supervision. Gradients from $\mathcal{L}_{\mathrm{LM}}$ pass through memory fusion into the predictor, while $\mathcal{L}_{\mathrm{MSE}}$ directly supervises the predicted memory slots. The predictor, query projection, memory projector, cross-attention, and gate parameters are optimized jointly. The backbone, retrieval encoder, and target memory remain fixed.

3.3 INFERENCE

At inference, the language model uses the same predictor and fusion modules without constructing target memory. Chunk queries are formed from backbone input embeddings, and the memory predictor supplies predicted memory to the selected decoder layers. No datastore search, retrieved text, or external embedding encoder is required. For QA evaluation, predicted memory is computed from the visible prompt and remains fixed during answer generation. Selected decoder layers attend to the predicted memory only at positions associated with the prompt chunks. As defined in Eq. 2, a sigmoid gate conditioned on the current token state scales each head before output projection. With layer-specific gate parameters, the contribution of the fixed memory can vary across tokens, heads, and layers.

4 EXPERIMENTS

We evaluate the effectiveness and efficiency of DFM across knowledge-intensive question answering and domain adaptation. We first describe the datasets, training configurations, and baselines (§4.1). We then evaluate DFM on six knowledge-intensive question-answering tasks (§4.2) and examine its effectiveness beyond QA through domain adaptation in Code, Math, and Law (§4.3). Finally, we compare parameter overhead, training time, and target-construction workload with token-level memory (§4.4).

4.1 EXPERIMENTAL SETUP

Datasets. For question answering, we train on English Wikipedia from December 2018, comprising approximately 4B tokens. We evaluate single-hop QA on NQ (Kwiatkowski et al., 2019) and WebQA (Berant et al., 2013), and multi-hop QA on HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). For domain adaptation, we sample subsets from Swallow-Code (Fujii et al., 2026) and Swallow-Math (Fujii et al., 2026), and use legal text from DISC-Law-SFT (Yue et al., 2023). We evaluate Code on HumanEval (Chen et al., 2021) and HumanEval+ (Liu et al., 2023), Math on GSM8K (Cobbe et al., 2021), and Law on LawBench (Fei et al., 2024). Appendices A.1 and A.2 give data-construction and evaluation details for QA and domain adaptation.

Training Details. We use Mistral-7B-v0.3 (Jiang et al., 2023) and Qwen3-8B-Base (Yang et al., 2025) for question answering, and Qwen3-4B-Base and Qwen3-8B-Base (Yang et al., 2025) for domain adaptation. DFM uses 32-token chunks and target memory formed from the top-16 chunk embeddings retrieved using Qwen3-Embedding-0.6B (Zhang et al., 2025), with gated memory fusion at layers $[9, 16, 24, 31]$. The added modules contain 0.75B parameters on both QA backbones and 0.6B on Qwen3-4B-Base. We freeze the backbone and train the added modules for one epoch with next-token cross-entropy and memory-prediction MSE, using AdamW with learning rate 1×10^{-5}. Wikipedia training uses 2,048-token sequences and a global batch size of 128 on 16 A800 80GB GPUs. For domain adaptation, we train a separate model for each domain with 262,144 tokens per optimizer update. Further architectural details and training configurations are given in Appendix A.

Baselines. We compare against three non-parametric and three parametric baselines. Among non-parametric methods, **RAG** (Lewis et al., 2020) augments the input with the top-1 passage retrieved using Qwen3-Embedding-0.6B. **xRAG** (Cheng et al., 2024) projects a retrieved document's dense embedding into a single token prepended to the question. **RETRO** (Borgeaud et al., 2022) incorporates retrieved chunks through cross-attention. Among parametric methods, **CPT** updates all backbone parameters through continued pretraining for one epoch. **LoRA** (Hu et al., 2021) trains low-rank adapters for one epoch, with its added parameter budget matched to DFM. **MLP Memory** (Wei et al., 2026a) learns to predict token-level kNN distributions from Wikipedia 2018 and interpolates its predictions with the backbone's output probabilities.

4.2 QUESTION ANSWERING PERFORMANCE

Overall Performance. Table 1 compares DFM with non-parametric and parametric methods across six QA benchmarks. DFM achieves the highest average on both backbones without online retrieval. On Mistral, it raises the six-task mean from xRAG's 24.14 to 27.21, a **12.71% relative improve-**

Table 1: Question answering performance on two single-hop benchmarks and four multi-hop benchmarks. Avg is the unweighted mean across all six tasks. Red subscripts show decreases from Base. Blue highlights DFM. Bold scores indicate the best results within each backbone.

Method	Single-hop QA		Multi-hop QA				Avg
	NQ	WebQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	
Mistral-7B-v0.3							
Base	20.30	29.28	20.96	5.87	1.43	3.91	13.63
Non-parametric methods							
RAG	24.03	27.90	29.09	**31.02**	9.87	18.56	23.41
xRAG	27.39	40.06	27.55	27.78	9.57	12.51	24.14
RETRO	20.26	29.48	21.06	5.90	1.48	4.19	13.73
Parametric methods							
CPT	11.44	32.92	12.13	14.22	4.51	5.65	13.48$_{-0.15}$
LoRA	14.24	40.21	13.43	14.98	4.90	5.08	15.47
MLP Memory	26.32	39.57	24.83	22.36	7.36	12.08	22.09
DFM	**33.57**	**42.47**	**29.53**	25.00	**10.05**	**22.65**	**27.21**
Qwen3-8B-Base							
Base	17.60	41.78	24.65	24.83	9.72	17.66	22.71
Non-parametric methods							
RAG	24.42	36.32	25.62	22.21	5.73	10.86	20.86$_{-1.85}$
xRAG	24.34	41.54	**27.74**	22.46	9.67	15.30	23.51
RETRO	17.65	42.32	24.55	24.85	9.63	17.11	22.69$_{-0.02}$
Parametric methods							
CPT	11.83	42.47	11.87	13.36	4.71	6.51	15.13$_{-7.58}$
LoRA	11.72	40.16	11.67	13.41	4.59	6.43	14.66$_{-8.04}$
MLP Memory	18.30	**43.26**	25.27	27.47	10.81	17.66	23.80
DFM	**27.46**	39.47	26.46	**32.38**	**12.89**	**20.09**	**26.46**

ment over the strongest baseline. On Qwen3-8B-Base, it improves from MLP Memory's 23.80 to 26.46, a **11.19% relative improvement**. These correspond to absolute gains of 3.07 and 2.66 points, showing comparable relative benefits across the two backbone families. Relative to Base, DFM gains 13.59 and 3.75 points, improving all six tasks on Mistral and five on Qwen3. It also outperforms CPT and LoRA on every Mistral task and five Qwen3 tasks.

Single-Hop Question Answering. On Mistral, DFM reaches 33.57 on NQ and 42.47 on WebQA, improving over Base by 13.27 and 13.19 points. These scores exceed the strongest baseline on each task by 6.18 and 2.26 points, showing gains beyond those obtained through either explicit retrieval or parameter adaptation alone. On Qwen3, NQ improves from 17.60 to 27.46, exceeding RAG by 3.04 points. WebQA instead decreases from 41.78 to 39.47, while MLP Memory achieves 43.26. The contrasting WebQA outcomes show that memory adaptation is not uniformly beneficial across backbones, even when the original weights remain frozen.

Multi-Hop Question Answering. DFM improves over Base on all four multi-hop benchmarks for both backbones. On Mistral, it leads on HotpotQA, MuSiQue, and Bamboogle, with Bamboogle increasing from 3.91 to 22.65. Its 2Wiki score rises from 5.87 to 25.00, although RAG remains stronger at 31.02. On Qwen3, DFM achieves the best results on 2Wiki, MuSiQue, and Bamboogle, including a 7.55-point gain over Base on 2Wiki. It also improves HotpotQA to 26.46, below xRAG's 27.74. These results indicate that predicted memory can support tasks requiring multiple pieces of evidence, while explicit retrieval retains an advantage on some benchmarks.

4.3 Domain Adaptation

Table 2 examines whether DFM also supports reasoning-oriented tasks and adaptation to specialized domains. Across Code, Math, and Law, DFM improves all four benchmarks over Base on

Table 2: Domain adaptation on Qwen3-4B-Base and Qwen3-8B-Base across Code, Math, and Law. Bold and underlined scores mark the best and second-best results within each backbone. Red subscripts show percentage-point decreases from Base.

Method	Code		Math	Law	Avg
	HumanEval	HumanEval+	GSM8K	LawBench	
Qwen3-4B-Base					
Base	62.04	57.32	83.40	29.96	58.18
RAG	62.80	55.49$_{-1.83}$	84.99	24.35$_{-5.61}$	56.91$_{-1.27}$
CPT	<u>66.62</u>	<u>58.54</u>	84.84	**41.66**	<u>62.92</u>
LoRA	59.45$_{-2.59}$	<u>58.54</u>	<u>85.37</u>	39.43	60.70
DFM	**69.51**	**62.80**	**86.05**	<u>40.41</u>	**64.69**
Qwen3-8B-Base					
Base	69.36	59.15	85.60	38.45	63.14
RAG	<u>70.88</u>	<u>64.02</u>	85.60	26.89$_{-11.56}$	61.85$_{-1.29}$
CPT	68.29$_{-1.07}$	59.76	<u>87.11</u>	45.09	65.06
LoRA	69.05$_{-0.31}$	**65.85**	**87.19**	<u>45.73</u>	<u>66.96</u>
DFM	**73.02**	<u>64.02</u>	86.43	**46.61**	**67.52**

both Qwen3-4B-Base and Qwen3-8B-Base. It achieves the highest average among the compared methods, reaching 64.69 on 4B and 67.52 on 8B, gains of 6.51 and 4.38 points.

Code and Mathematical Reasoning. On 4B, DFM leads on HumanEval, HumanEval+, and GSM8K, improving over Base by 7.47, 5.48, and 2.65 points, respectively. It exceeds CPT by 2.89 and 4.26 points on the two code benchmarks and LoRA by 0.68 points on GSM8K. On 8B, DFM achieves the highest HumanEval score of 73.02 and raises HumanEval+ from 59.15 to 64.02. LoRA remains stronger on HumanEval+ and GSM8K, where DFM gains 0.83 points over Base. DFM also matches or exceeds RAG on these benchmarks, tying only on 8B HumanEval+. Memory adaptation thus benefits reasoning-oriented tasks at both scales, although its advantage over other adaptation methods varies by benchmark.

Legal Domain Adaptation. Training on legal text raises LawBench from 29.96 to 40.41 on 4B and from 38.45 to 46.61 on 8B. RAG scores 24.35 and 26.89, leaving DFM ahead by 16.06 and 19.72 points, respectively. On 4B, DFM exceeds LoRA by 0.98 points but remains 1.25 points below CPT. On 8B, it achieves the best score, exceeding CPT and LoRA by 1.52 and 0.88 points. Per-task LawBench results appear in Appendix B.2, with domain-specific training configurations in Appendix A.2.

4.4 Efficiency Analysis

Figure 2 examines the resource requirements of chunk-level parametric memory relative to token-level MLP Memory. We compare added model capacity, training time on 16 A800 GPUs, and the offline retrieval workload for constructing targets from a common 4B-token corpus.

Parameter and Training Costs. As shown in Figure 2(a), DFM adds 0.75B parameters, approximately 50% fewer than the 1.5B Mistral MLP Memory configuration (Wei et al., 2026a). Trained on 16 A800 GPUs with the same corpus and number of epochs, MLP Memory and DFM take approximately 64 and 24 hours, respectively, reducing training time by approximately 62.5%. Figure 2(b) compares these measured training costs. Appendix D provides the timing details.

Target Construction. Figure 2(c) shows that, for a 4B-token corpus, token-level construction requires 4B retrieval queries, whereas 32-token chunks require 125M, a 96.88% reduction. Each query produces one target entry. With Top-1024 for MLP Memory and Top-16 for DFM, the total number of retrieved items falls from $4\text{B} \times 1024 = 4.096\text{T}$ to $125\text{M} \times 16 = 2\text{B}$, a 99.95% reduction. By reducing both the number of retrieval queries and the neighbors returned per query, chunk-level supervision substantially lowers the retrieval overhead of target construction.

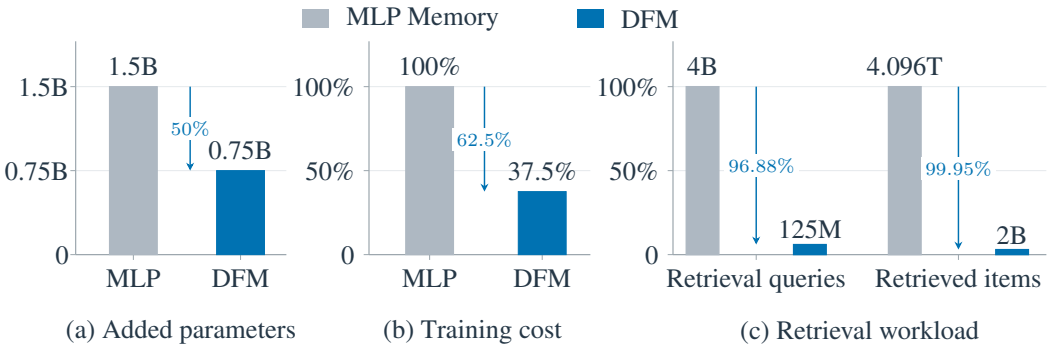

Figure 2: Resource comparison with token-level MLP Memory. (a) Added parameters. (b) Training cost: MLP Memory and DFM take approximately 64 and 24 hours, respectively, on 16 A800 GPUs, a 62.5% reduction. (c) Retrieval workload: retrieval queries and total retrieved items (queries $\times K$) for a 4B-token corpus. Reference bars in (b,c) equal 100%; arrows show percentage reductions.

5 Ablations and Analysis

We study chunk size, fusion-layer configurations, memory slots, and memory gating on Mistral-7B-v0.3 with predicted memory, varying one factor from the main configuration in Table 1. Figure 3 reports six-task QA averages. DFM is robust to the tested chunk sizes and slot counts, and to fusion configurations spanning intermediate and late layers. Appendix C gives exact settings, task-level results, and additional ablations on learning rate, projector sharing, predictor depth, and prediction loss. Appendix E presents QA and offline-retrieval case studies.

5.1 Ablation Studies

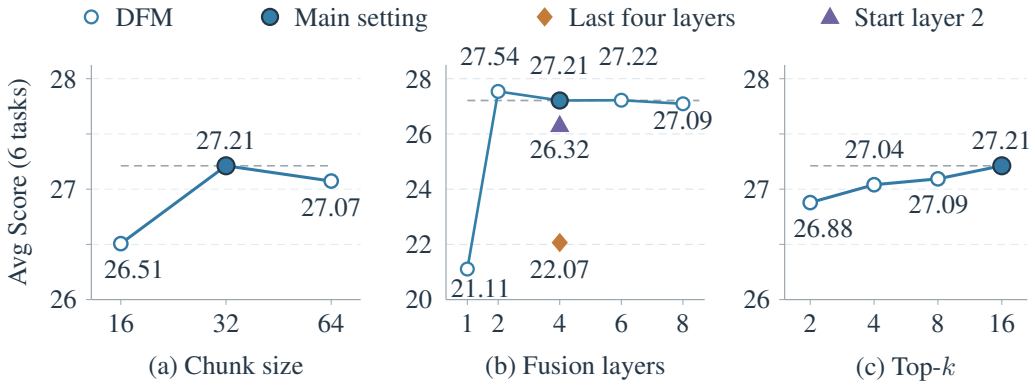

Figure 3: DFM ablations with predicted memory on Mistral-7B-v0.3. Points show single-run six-task QA means. Filled circles and dashed lines mark the main setting (chunk 32, layers $[9, 16, 24, 31]$, Top-16). In (b), the diamond and triangle use four fusion layers at $[28, 29, 30, 31]$ and $[2, 10, 21, 31]$, respectively.

Chunk Size. Figure 3(a) reports averages of 26.51, 27.21, and 27.07 for chunk sizes 16, 32, and 64, respectively. The 32-token setting performs best. Doubling the chunk size to 64 halves target-construction queries with only a 0.14-point drop. The 16-token setting doubles retrieval queries without improving the mean, so finer-grained supervision is not necessarily more effective. This supports 32-token chunks as a practical balance between QA performance and construction workload.

Fusion Depth and Placement. In Figure 3(b), two to eight fusion layers starting at layer 9 achieve 27.09–27.54, versus 21.11 with final-layer-only fusion and 22.07 with the final four layers. With

four fusion modules, distributing memory access across intermediate and late layers is more effective than concentrating it near the output. Four, six, and eight fusion layers remain within 0.13 points of one another, supporting the four-layer default without a consistent gain from greater fusion depth. The four-layer configuration $[2, 10, 21, 31]$ scores 26.32, 0.89 points below the default. These results favor intermediate-to-late fusion, as also used in RETRO (Borgeaud et al., 2022). A possible explanation is that more developed semantic representations aid memory integration, with later layers processing the fused information.

Number of Memory Slots. In Figure 3(c), K values of 2, 4, 8, and 16 yield averages of 26.88, 27.04, 27.09, and 27.21. Top-16 exceeds Top-8 by only 0.12 points: Top-8 scores higher on five tasks, while Top-16 improves Bamboogle. Even Top-2 remains within 0.33 points of Top-16, indicating that DFM maintains similar average QA performance with substantially fewer memory slots. This makes smaller K a practical option for using fewer target vectors and reducing memory-attention computation while retaining similar average performance.

5.2 Memory Gating

Table 3 examines the effect of token-wise, per-head gating on memory fusion and its sensitivity to initialization. Removing the gate reduces the six-task mean from 27.21 to 26.83, with the decrease driven mainly by Bamboogle and smaller, mixed changes on the other tasks. Initializing the gate bias at 0 yields the highest mean, 27.90, and improves five of six tasks over the no-gate variant, providing further support for gated fusion. More negative initial biases yield lower means in this sweep. We retain $b_g = -2$ as a conservative default: with zero-initialized gate weights, $\sigma(-2) \approx 0.119$ initially scales down the memory branch to limit its perturbation to the pretrained backbone, while allowing the gate to adapt during training.

Table 3: Gate initialization and removal on Mistral-7B-v0.3. Gated variants change only the initial bias b_g; $b_g = -2$ is the default. Bold marks column maxima; Avg is the six-task mean.

Gate	NQ	WebQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Avg
No gate	34.08	**42.72**	29.80	24.94	10.08	19.35	26.83
$b_g = -6$	32.34	42.18	28.54	23.55	9.03	14.94	25.10
$b_g = -4$	32.70	42.22	29.39	25.98	10.42	19.70	26.73
$b_g = -2$	33.57	42.47	29.53	25.00	10.05	**22.65**	27.21
$b_g = 0$	**34.59**	42.13	**30.38**	**29.68**	**10.94**	19.69	**27.90**

6 Conclusion

We presented Deep Fusion Memory (DFM), a parametric memory architecture that combines efficient supervision construction with deep fusion within the decoder. Sharing target memory across tokens in each chunk reduces retrieval work, while gated cross-attention allows subsequent decoder layers to process predicted memory together with the evolving context. Memory prediction and fusion are trained end to end with the backbone frozen, and inference requires no explicit retrieval. DFM improves the six-task QA average over the strongest baseline by 12.71% on Mistral-7B-v0.3 and 11.19% on Qwen3-8B-Base. Beyond QA, DFM improves performance over the base models across code, mathematics, and law. These results demonstrate that shared supervision and learned internal fusion can support efficient knowledge acquisition and effective memory use.

Reproducibility Statement

We describe our method in Section 3 and the experimental setup in Section 4. Appendix A details data preparation, training hyperparameters, and evaluation protocols. Appendices C and D provide detailed ablation configurations and efficiency accounting.

AI USE STATEMENT

We utilized ChatGPT as an editorial assistant to improve the manuscript's language quality and correct grammatical errors, with additional feedback on methodological descriptions and result interpretation. The authors led and completed all scientific work, including the research design, experimental procedures, and analytical interpretations.

REFERENCES

Lei Bai, Jiaqi Cao, Chiyu Chen, Guanzhou Chen, Kai Chen, Guangran Cheng, Erfei Cui, Xuanlang Dai, Shengyuan Ding, Shangheng Du, et al. Intern-s2-preview: Scientific agentic foundation model. *arXiv preprint arXiv:2608.13505*, 2026.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1533–1544, 2013.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, et al. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*, 2024.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pp. 2206–2240. PMLR, 2022.

Jiaqi Cao, Jiarui Wang, Rubin Wei, Qipeng Guo, Kai Chen, Bowen Zhou, and Zhouhan Lin. Memory decoder: A pretrained, plug-and-play memory for large language models. *Advances in Neural Information Processing Systems*, 38:115487–115510, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xrag: Extreme context compression for retrieval-augmented generation with one token. *Advances in Neural Information Processing Systems*, 37:109487–109516, 2024.

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

Zhiwei Fei, Xiaoyu Shen, Dawei Zhu, Fengzhe Zhou, Zhuo Han, Alan Huang, Songyang Zhang, Kai Chen, Zhixin Yin, Zongwen Shen, et al. Lawbench: Benchmarking legal knowledge of large language models. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7933–7962, 2024.

Kazuki Fujii, Yukito Tajima, Sakae Mizuki, Masaki Kawamura, Hinari Shimada, Taihei Shiotani, Koshiro Saito, Masanari Oi, Taishi Nakamura, Okamoto Okamoto, et al. Rewriting pre-training data boosts llm performance in math and code. In *International Conference on Learning Representations*, volume 2026, pp. 87643–87675, 2026.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 8342–8360, 2020.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

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. *arXiv preprint arXiv:2106.09685*, 2021.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL `https://arxiv.org/abs/2310.06825`.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572, 2023.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023.

Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Jiarui Wang, Xiang Shi, Jiaqi Cao, Rubin Wei, Xiquan Wang, Hao Sun, Jingzhi Wang, Zhiqi Yang, Qipeng Guo, Bowen Zhou, et al. Memsft: Mitigating alignment tax with an external parametric memory. *arXiv preprint arXiv:2607.25614*, 2026.

Rubin Wei, Jiaqi Cao, Jiarui Wang, Jushi Kai, Qipeng Guo, Bowen Zhou, and Zhouhan Lin. Mlp memory: A retriever-pretrained memory for large language models. In *International Conference on Learning Representations*, volume 2026, pp. 132772–132795, 2026a.

Rubin Wei, Jiaqi Cao, Jiarui Wang, Junming Zhang, Qipeng Guo, Bowen Zhou, and Zhouhan Lin. Memory decoder at scale: A pretrained, parametric long-term memory. *arXiv preprint arXiv:2607.27919*, 2026b.

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.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Shengbin Yue, Wei Chen, Siyuan Wang, Bingxuan Li, Chenchen Shen, Shujun Liu, Yuxuan Zhou, Yao Xiao, Song Yun, Xuanjing Huang, et al. Disc-lawllm: Fine-tuning large language models for intelligent legal services. *arXiv preprint arXiv:2309.11325*, 2023.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

A Experimental Details

A.1 Question Answering

Data construction. We use section and infobox text from Wikipedia December 2018. Records are grouped by normalized article titles and assigned to training, validation, and test splits in proportions of 99.8/0.1/0.1% using seed 42. We remove exact text duplicates across splits, append EOS to each record, and pack records within each split into 2,048-token sequences. Incomplete final sequences are discarded. Tokenization yields 4.70B, 4.29B, and 4.08B training tokens for Mistral, Qwen3, and OLMo3, respectively. Each sequence is divided into non-overlapping 32-token chunks. We encode training chunks with the frozen Qwen3-Embedding-0.6B encoder, using last-token pooling and L2 normalization to obtain 1,024-dimensional representations. A FAISS IVF-PQ index supports offline neighbor retrieval. Following Section 3.1, each preceding chunk retrieves target memory for the next chunk. We retain the top-16 full chunk embeddings in retrieval-rank order, including the nearest neighbor, and mask the first chunk from memory supervision. The datastore contains training chunks only.

Training. DFM freezes the backbone and jointly trains its chunk-level memory predictor, shared projector, and gated fusion modules. The predictor has 28 layers, hidden width 1,024, FFN width 3,072, 16 query heads, and eight KV heads. Its 16 output heads predict the continuous memory slots. The L2-normalized targets have coordinates in $[-1, 1]$, and the predictor clips output coordinates to this range. Memory cross-attention is inserted at zero-based layers $[9, 16, 24, 31]$, with gate weights initialized to zero and gate biases to -2. OLMo3 retains its original attention-output normalization. The objective combines next-token cross-entropy and raw memory-prediction MSE with equal weights. We train for one epoch on 16 A800 80GB GPUs, with one sequence per GPU and eight gradient-accumulation steps, giving a global batch size of 128. AdamW uses learning rate 1×10^{-5}, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, zero weight decay, and gradient clipping at 1. We use 500 warmup steps followed by cosine decay, BF16 precision, seed 42, and the final checkpoint for evaluation.

Baselines. **RAG** (Lewis et al., 2020) retrieves the top-1 document from a Wikipedia training-corpus index using Qwen3-Embedding-0.6B and adds it to the question context. **xRAG** (Cheng et al., 2024) projects a retrieved document's dense embedding into a single token prepended to the question. **RETRO** (Borgeaud et al., 2022) uses chunk-level retrieval and cross-attention to incorporate retrieved text into hidden states. **CPT** updates all backbone parameters for one epoch on Wikipedia at learning rate 2×10^{-5}. **LoRA** (Hu et al., 2021) trains low-rank adapters for one epoch at the same learning rate of 2×10^{-5}, with the added parameter budget matched to DFM. **MLP Memory** (Wei et al., 2026a) learns to predict token-level kNN distributions from Wikipedia 2018 and interpolates its predictions with the backbone probabilities. As in MLP Memory, interpolation weights are task-specific; we select the best $\lambda \in \{0, 0.05, \ldots, 1\}$ on each reported evaluation split. In the task order of Table 4, we use $\lambda = (0.55, 0.55, 0.55, 1.00, 0.90, 1.00)$ for Mistral and $(0.35, 0.45, 0.25, 0.80, 0.55, 0.00)$ for Qwen3.

Evaluation. Table 4 summarizes the six QA benchmarks. DFM uses zero-shot greedy decoding without online retrieval, gold documents, or chain-of-thought demonstrations. Questions are formatted with task-specific question–answer templates. Generation stops at a newline or the token limit, with period and comma as additional stop strings for WebQA. Evaluation batches contain one example per GPU, except WebQA, which uses four. For QA, each memory query is formed from the current visible prompt chunk rather than the preceding chunk used during training. The queries and predicted memory remain fixed during generation. For newly added chunk positions, the adapter pads queries and query-validity masks with zeros, so these positions receive no direct memory update. Generated answer chunks do not refresh the queries, and incremental predictor caching is not used in the reported implementation. F1 measures normalized token overlap and takes the maximum over accepted answers. Substring match checks whether a normalized reference answer occurs in the generated response. Scores are reported on a 0–100 scale, and Avg. is the unweighted mean across the six tasks.

Table 4: Evaluation datasets and metrics for question answering. Max. tokens is the maximum number of newly generated tokens.

Dataset	Split	Samples	Metric	Max. tokens
NQ	Test	3,610	F1	15
WebQA	Test	2,032	Substring match	100
HotpotQA	Validation	7,405	F1	15
2Wiki	Validation	12,576	F1	32
MuSiQue	Validation	2,417	F1	32
Bamboogle	Test	125	F1	32

A.2 Domain Adaptation

Data. Code and Math each use 100M training tokens sampled from Swallow-Code and Swallow-Math (Fujii et al., 2026), with document-level train/validation/test splits of 98/1/1%. Law uses 123.5M training tokens from DISC-Law-SFT (Yue et al., 2023), holding out 1% of source-train for validation. We preserve its `Reference:` field when present, together with `Input:` and `Output:`. We deduplicate documents, screen benchmark overlap through normalized exact matching, append EOS, and pack sequences within each split. Target memory is constructed as in Appendix A.1, using training-only datastores, 32-token chunks, and Top-16 retrieval.

Training. We use the default DFM configuration for Qwen3-4B-Base and Qwen3-8B-Base, with the backbone frozen and the same memory predictor, shared projector, fusion layers, and gate initialization as in the QA experiments. DFM uses learning rate 1×10^{-5}, cosine decay, zero weight decay, and 11 warmup steps for Code/Math or 14 for Law. All adapted models train for one epoch in BF16 with 262,144 tokens per update. Sequence length is 8K for Code training on Qwen3-4B-Base and 2K for all other backbone–domain pairs. CPT and LoRA use the same data and sequence length as DFM. Both use learning rate 2×10^{-5}, with 3% warmup and weight decay 0.1. LoRA uses rank 256, alpha 512, and zero dropout. We train a separate checkpoint for each domain.

RAG. RAG uses the frozen Base backbone without additional training. It retrieves the top-1 document from the corresponding domain's training corpus using Qwen3-Embedding-0.6B. Retrieved Code is prefixed as comments, while Math and Law use a background prefix. The original task prompt is preserved in full; only retrieved background is truncated when the input budget is exceeded. Few-shot examples, sample counts, generation settings, and scorers are identical to Base.

Evaluation. HumanEval uses three demonstrations and four samples per problem, while HumanEval+ uses zero demonstrations and one sample. Code reports sample-average execution correctness, estimating pass@1. GSM8K uses eight demonstrations and one sample. LawBench evaluates 500 examples for each of its 20 tasks in the zero-shot setting. All methods use a common `Input:`/`Output:` template without a chat template, greedy decoding, and the official task scorers. The LawBench score is the equal-weight macro average of the 20 task metrics. Main-table Avg. equally weights HumanEval, HumanEval+, GSM8K, and LawBench. Appendix B.2 provides the per-task LawBench results for the same models.

B Additional Experimental Results

B.1 Question Answering Performance

Table 5 extends the Wikipedia experiment to OLMo-3-1025-7B under the same six-task protocol as Table 1. DFM increases the mean from 17.99 to 25.49, a gain of 7.50 points. It improves four tasks over Base, including 22.60 points on 2Wiki and 12.12 on Bamboogle, while WebQA and HotpotQA decrease by 1.28 and 0.63 points. Among the methods in Table 5, DFM achieves the highest NQ, 2Wiki, and MuSiQue scores. Its mean exceeds the strongest baseline, RETRO at 23.63, by 1.86 points.

Table 5: Question answering performance on OLMo-3-1025-7B across two single-hop and four multi-hop benchmarks. Avg is the six-task mean. Red subscripts show decreases from Base. Blue highlights DFM. Bold and underlined scores mark the best and second-best results.

Method	Single-hop QA		Multi-hop QA				Avg
	NQ	WebQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	
OLMo-3-1025-7B							
Base	21.69	**40.31**	**27.99**	11.05	4.67	2.25	17.99
RAG	22.57	31.50	21.97	7.91	2.58	2.75	14.88$_{-3.11}$
xRAG	<u>23.64</u>	34.79	26.33	23.15	<u>11.95</u>	**16.65**	22.75
RETRO	21.61	38.09	<u>27.97</u>	<u>29.90</u>	11.61	12.57	<u>23.63</u>
LoRA	7.39	34.40	7.32	6.73	2.55	1.21	9.93$_{-8.06}$
DFM	**26.08**	<u>39.03</u>	27.36	**33.65**	**12.44**	<u>14.37</u>	**25.49**

B.2 Domain Adaptation

Table 6 maps LawBench task IDs to their names and definitions, grouped into legal knowledge memorization, understanding, and application. Tables 7 and 8 report the 20 LawBench subtasks for Qwen3-4B-Base and Qwen3-8B-Base, respectively, comparing Base, RAG, CPT, LoRA, and DFM. They use the same checkpoints and zero-shot evaluation protocol as Table 2, with RAG using the frozen Base model. The Macro row reports the equal-weight average over the 20 task metrics, corresponding to the LawBench score in the main table. Appendix A.2 gives the training and evaluation settings.

Table 6: LawBench task IDs and brief definitions, following Fei et al. (2024). The first digit identifies the legal knowledge category. These IDs are used in Tables 7 and 8.

ID	Task	Description
Legal Knowledge Memorization		
1-1	Article Recitation	Recall a specified law article.
1-2	Knowledge Question Answering	Answer a legal multiple-choice question.
Legal Knowledge Understanding		
2-1	Document Proofreading	Correct errors in a legal sentence.
2-2	Dispute Focus Identification	Identify issues disputed by the parties.
2-3	Marital Disputes Identification	Classify types of marital disputes.
2-4	Issue Topic Identification	Classify a legal inquiry by topic.
2-5	Reading Comprehension	Extract answers from judgment text.
2-6	Named-Entity Recognition	Extract legal entities of specified types.
2-7	Opinion Summarization	Summarize news on a legal matter.
2-8	Argument Mining	Select an argument opposing the plaintiff.
2-9	Event Detection	Detect events mentioned in legal text.
2-10	Trigger Word Extraction	Extract trigger words for given events.
Legal Knowledge Application		
3-1	Fact-based Article Prediction	Predict article numbers from case facts.
3-2	Scene-based Article Prediction	Generate article text for a legal scenario.
3-3	Charge Prediction	Predict charges from facts and article IDs.
3-4	Prison Term Prediction without Article Text	Use facts, charges, and article numbers.
3-5	Prison Term Prediction with Article Text	Also provide the applicable article text.
3-6	Case Analysis	Answer a multiple-choice case question.
3-7	Criminal Damages Calculation	Calculate the amount involved in a crime.
3-8	Consultation	Answer a request for legal advice.

Table 7: LawBench task results on Qwen3-4B-Base under zero-shot evaluation. DFM uses the same checkpoint as the main table. Task definitions appear in Table 6. Bold marks the highest score among the displayed methods. Macro is the equal-weight average over all 20 task metrics.

Task	Base	RAG	CPT	LoRA	DFM
1-1	18.76	11.41	**42.46**	31.70	22.23
1-2	6.20	20.00	27.00	28.20	**31.60**
2-1	16.83	20.06	26.82	**35.36**	24.13
2-2	12.20	0.60	31.40	23.80	**35.60**
2-3	32.73	16.10	32.75	30.16	**38.97**
2-4	0.00	0.20	**34.00**	33.40	30.80
2-5	6.02	3.29	38.35	41.46	**54.54**
2-6	**39.56**	34.64	10.32	5.30	27.81
2-7	38.15	14.43	**49.34**	48.68	40.85
2-8	0.00	0.00	34.20	**39.80**	36.80
2-9	35.64	14.89	**63.71**	63.32	57.32
2-10	13.25	3.03	17.35	22.07	**24.07**
3-1	50.22	44.65	**72.03**	68.87	37.32
3-2	24.92	9.20	34.54	26.30	**35.24**
3-3	46.11	34.16	**51.14**	48.00	44.96
3-4	**82.88**	80.45	76.91	59.92	74.23
3-5	**83.02**	81.31	76.72	65.50	76.12
3-6	2.00	9.20	31.20	**31.60**	31.40
3-7	72.00	**75.20**	65.80	68.00	66.20
3-8	**18.79**	14.11	17.23	17.08	17.94
Macro	29.96	24.35	**41.66**	39.43	40.41

Table 8: LawBench task results on Qwen3-8B-Base under zero-shot evaluation. DFM uses the same checkpoint as the main table. Task definitions appear in Table 6. Bold marks the highest score among the displayed methods. Macro is the equal-weight average over all 20 task metrics.

Task	Base	RAG	CPT	LoRA	DFM
1-1	16.72	8.99	**54.57**	54.15	29.71
1-2	**70.60**	17.20	32.20	30.00	53.80
2-1	20.74	21.89	**38.60**	27.72	30.85
2-2	0.20	0.40	14.60	9.60	**36.80**
2-3	16.01	16.41	46.21	40.52	**52.40**
2-4	0.00	0.00	33.80	**35.00**	34.40
2-5	**52.28**	5.95	37.58	35.82	51.40
2-6	**44.52**	40.11	2.22	36.87	33.50
2-7	29.31	15.00	47.44	**49.19**	43.26
2-8	**41.40**	1.80	33.80	37.00	37.00
2-9	26.38	15.26	**74.02**	70.16	49.72
2-10	17.01	3.96	32.16	**32.19**	25.02
3-1	53.92	54.20	78.37	**81.09**	62.09
3-2	12.86	9.66	**40.65**	39.85	38.44
3-3	**50.92**	46.93	48.75	50.77	48.05
3-4	81.95	**84.73**	81.77	81.70	81.60
3-5	81.55	**82.94**	81.92	82.63	81.47
3-6	**58.40**	22.00	33.60	31.80	52.60
3-7	74.00	**75.80**	69.20	71.40	69.40
3-8	20.16	14.63	20.42	17.09	**20.65**
Macro	38.45	26.89	45.09	45.73	**46.61**

C Ablation Studies

All ablations start from the Mistral-7B-v0.3 configuration in Table 1 and change only the factor under study. They use predicted memory, a frozen backbone, one training epoch, and the same six-task QA protocol. All results are single runs.

C.1 Chunk Size, Fusion Placement, and Memory Slots

Reference configuration. The reference for Figure 3 uses continuous predicted memory, 32-token chunks, target memory from Top-16 retrieval, a shared projector, gate bias -2, and learning rate 1×10^{-5}. Its fusion layers are $[9, 16, 24, 31]$ (zero-based), and its six task scores match Table 1 exactly. These structural sweeps retain the other named architectural settings. Appendix A.1 gives the reference training recipe.

Change	NQ	WebQA	HotpotQA	2Wiki	MuSiQue	Bamb.	Avg.
Chunk 16	-1.49	-0.84	-0.60	+0.56	+0.02	-1.88	**-0.71**
Chunk 64	-0.04	-0.25	-0.46	+1.03	-0.03	-1.08	**-0.14**
Fusion 1	-12.42	-0.69	-5.65	-4.85	-2.27	-10.75	**-6.11**
Fusion 2	+0.54	-0.05	+0.88	+3.69	+0.82	-3.91	**+0.33**
Fusion 6	+0.43	+0.15	-0.02	+0.87	+0.01	-1.38	**+0.01**
Fusion 8	+0.09	-0.59	+0.06	+0.72	+0.56	-1.56	**-0.12**
Last four	-10.63	-0.15	-5.05	-4.67	-2.54	-7.84	**-5.15**
Start layer 2	+0.14	-1.23	-0.28	+0.19	+0.41	-4.57	**-0.89**
Top-2	-0.23	-0.44	+0.29	+2.13	+0.33	-4.08	**-0.33**
Top-4	+0.40	-0.10	+0.23	+1.88	+0.66	-4.10	**-0.17**
Top-8	+0.07	+0.25	+0.52	+0.80	+0.72	-3.07	**-0.12**



Figure 4: Task-level score changes relative to the main DFM configuration on Mistral-7B-v0.3. Blue and brown indicate increases and decreases, respectively. Avg. gives the change in the unweighted six-task mean. Fusion labels denote the number of fusion layers, while Last four and Start layer 2 use $[28, 29, 30, 31]$ and $[2, 10, 21, 31]$, respectively. All differences are computed before rounding.

Chunk size. Relative to 32 tokens, the 16-token setting improves 2Wiki and MuSiQue but decreases the other four scores. The 64-token setting improves 2Wiki by 1.03 points, with decreases of 0.03–1.08 on the other tasks. For a fixed token budget, target-construction query counts scale inversely with chunk size. This establishes a workload trade-off but does not measure elapsed construction time.

Fusion depth and placement. The one-, two-, four-, six-, and eight-layer configurations use $[31]$, $[9, 31]$, $[9, 16, 24, 31]$, $[9, 13, 18, 22, 27, 31]$, and $[9, 12, 15, 18, 22, 25, 28, 31]$, respectively. This sweep changes both layer count and placement. The two-layer setting attains the highest mean (27.54), while the four-layer reference is retained across experiments for consistency. The additional four-layer condition uses $[28, 29, 30, 31]$ and scores 22.07. All six tasks decrease relative to the distributed four-layer reference, including 10.63 points on NQ and 7.84 on Bamboogle. An ad-

ditional four-layer configuration, $[2, 10, 21, 31]$, obtains 26.32, 0.89 points below the reference. It keeps the number of modules fixed but changes multiple fusion positions, so the difference cannot be attributed solely to the starting layer.

Memory slots. Figure 4 shows opposing task-level changes. Top-8 exceeds Top-16 by 0.07–0.80 points on five tasks but scores 3.07 points lower on Bamboogle, which contains 125 examples. Across Top-2 to Top-16, the six-task mean varies by only 0.33 points, indicating robustness to the tested memory-slot counts.

C.2 Learning Rate

Table 9 compares the default learning rate 1×10^{-5} with 2×10^{-5}, retaining 32-token chunks, Top-16, a shared projector, gate bias -2, and fusion layers $[9, 16, 24, 31]$. The default obtains 27.21 versus 27.10, with higher scores on NQ, WebQA, HotpotQA, and Bamboogle but lower scores on 2Wiki and MuSiQue. The 0.11-point difference indicates that average QA performance is robust to this twofold change in learning rate.

Table 9: Learning-rate ablation from the main DFM configuration. All other settings are fixed. The blue 1×10^{-5} row is the default and reproduces the DFM results in Table 1. Bold marks the higher Avg.

Learning rate	NQ	WebQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Avg
2×10^{-5}	33.47	41.93	29.18	28.60	10.43	19.00	27.10
1×10^{-5}	33.57	42.47	29.53	25.00	10.05	22.65	**27.21**

C.3 Projector Sharing

Table 10 compares the shared memory projector with an independent projector at each fusion layer. The shared design scores 27.21 versus 26.88. Per-layer projectors improve WebQA, HotpotQA, 2Wiki, and MuSiQue but reduce NQ by 0.59 points and Bamboogle by 3.95 points. Sharing avoids replicating projector parameters across fusion layers and attains a slightly higher mean here.

Table 10: Shared versus per-layer memory projectors on Mistral-7B-v0.3. All other training settings are fixed. Blue marks the default shared projector. Bold marks the higher Avg. Avg is the six-task mean.

Projector	NQ	WebQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Avg
Per-layer	32.98	43.21	29.88	26.35	10.18	18.70	26.88
Shared	33.57	42.47	29.53	25.00	10.05	22.65	**27.21**

C.4 Memory Predictor Depth

Table 11 compares 14-, 28-, and 56-layer memory predictors while retaining the other training settings. Their six-task averages are 27.31, 27.21, and 26.47, respectively. The 0.10-point difference between 14 and 28 layers shows that average QA performance is stable when predictor depth is halved. Doubling depth reduces the mean by 0.75 points, driven mainly by a 4.74-point decrease on Bamboogle; NQ, HotpotQA, 2Wiki, and MuSiQue improve slightly. We retain 28 layers as the default for consistency across the main experiments.

Table 11: Memory predictor depth on Mistral-7B-v0.3. Only the number of predictor layers changes. Blue marks the default 28-layer predictor. Bold marks the highest Avg. Avg is the six-task mean.

Predictor	NQ	WebQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Avg
14 layers	34.27	41.83	30.14	26.31	10.30	21.03	**27.31**
56 layers	33.89	42.18	29.58	25.10	10.14	17.91	26.47
28 layers	33.57	42.47	29.53	25.00	10.05	22.65	27.21

C.5 Memory Prediction Objective

Table 12 replaces raw MSE with a cosine-based memory prediction loss while retaining the other training settings. We compute one minus the cosine similarity for each predicted–target slot pair and average over all slots in the valid chunks, retaining $\beta = 1$ in Eq. 4. Raw MSE scores 27.21, compared with 26.87 for cosine loss. Cosine loss improves WebQA and 2Wiki by 0.39 and 0.48 points, respectively, but decreases the other four scores, with the largest drop on Bamboogle (2.23 points). The two objectives yield similar averages, with the default raw MSE higher by 0.34 points.

Table 12: Memory prediction objectives on Mistral-7B-v0.3. Only the memory prediction loss changes; all other training settings are fixed. Blue marks the default raw MSE objective. Bold marks the higher Avg. Avg is the six-task mean.

Objective	NQ	WebQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Avg
Cosine	33.02	42.86	29.40	25.49	10.04	20.42	26.87
Raw MSE	33.57	42.47	29.53	25.00	10.05	22.65	**27.21**

D EFFICIENCY ANALYSIS DETAILS

Table 13 lists the numerical inputs and quantities used in Figure 2. Training costs compare measured training times on 16 A800 GPUs for both methods; retrieval counts use a common 4B-token corpus.

Table 13: Numerical accounting for the comparison in Figure 2. Construction counts assume a common 4B-token corpus. Training hours are rounded measurements on 16 A800 GPUs for both methods.

Quantity	MLP Memory	DFM	Basis
Added parameters (B)	1.50	0.75	Configuration
A800 GPUs	16	16	Configuration
Measured training hours	64	24	Timing
Training GPU-hours	1,024	384	GPUs $\times$ hours
Target queries	4B	125M	Workload count
Neighbors per query	1,024	16	Configuration
Returned neighbor slots	4.096T	2B	Workload count

Reference configurations. We use the 1.5B MLP Memory configuration (Wei et al., 2026a) throughout the experiments, including the parameter and training-time comparisons. DFM adds approximately 0.75B parameters, a reduction of approximately 50%.

Measured training time. Both methods are trained on 16 A800 GPUs using the same training corpus and number of epochs. Timings cover training only, excluding target construction and evaluation. MLP Memory takes approximately 64 hours, while DFM takes approximately 24 hours, corresponding to 1,024 and 384 GPU-hours. DFM therefore uses $24/64 = 37.5\%$ of the training time, a reduction of approximately 62.5%. All reported costs use these rounded training times.

Construction counts. For $N = 4 \times 10^9$ tokens, ignoring incomplete chunks and sequence-start masking, token-level targets require N queries and chunk-level targets require $N/32$. Retrieving 1,024 versus 16 neighbors yields $1024N$ versus $16N/32$ returned slots. Their ratios are 32 for queries and 2048 for returned slots. The latter counts search outputs before MLP Memory removes the self neighbor and aggregates token probabilities. It does not count distinct tokens, search candidates examined, stored bytes, or FLOPs: DFM's continuous vector targets differ from token-level distribution targets.

E CASE STUDIES

E.1 QUESTION ANSWERING

Figure 5 compares answers from Base, MLP Memory, and DFM on six selected HotpotQA examples using Mistral-7B-v0.3. DFM answers all six correctly, while Base and MLP Memory give the same incorrect answer in each case. DFM identifies Manglietia from the stated species count and geographic origin, whereas both baselines choose Abronia. For the film comparison, DFM correctly selects *What Ever Happened to Baby Jane?*, while both baselines select *All the Marbles*. The other four cases cover a shared property of two bands, a composer's nationality, a university's association with a cartoon character, and the film underlying a Pixar short.

Cases were selected for a correct normalized exact match from DFM and zero exact match, substring match, and F1 from both baselines. MLP Memory uses $\lambda = 0.55$. These selected successes illustrate differences in final answers, without establishing their frequency or revealing the models' intermediate reasoning.

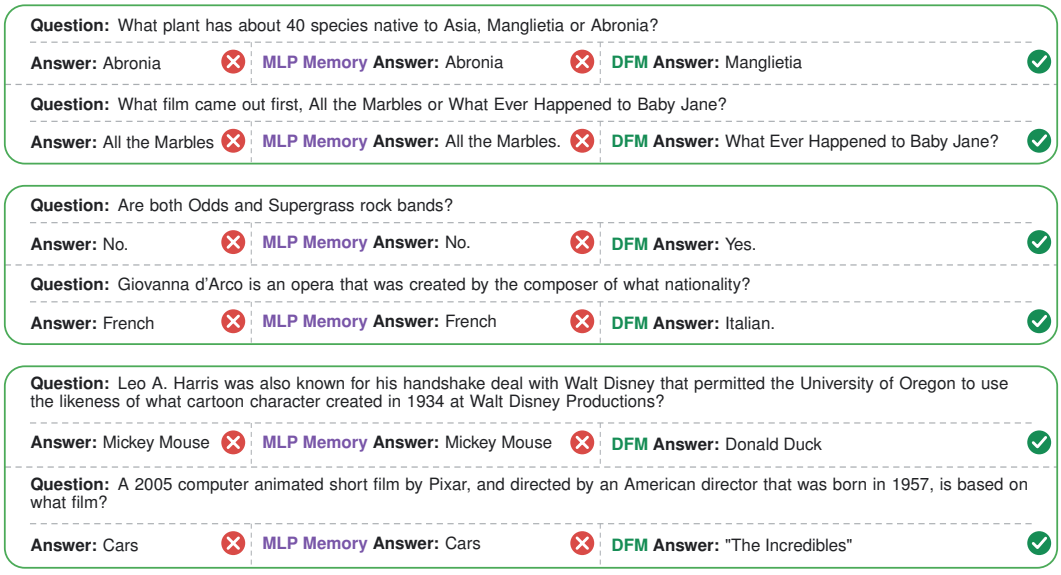

Figure 5: Six selected HotpotQA answer comparisons using Mistral-7B-v0.3. Columns show Base (left), MLP Memory (middle), and DFM (right). Check and cross symbols indicate correct and incorrect normalized exact-match answers.

E.2 Offline Retrieval Supervision

Figure 6 presents seven examples of the offline retrieval used to construct training targets. Each row pairs a preceding 32-token query chunk with retrieved chunk text and the ground-truth next chunk. For Pakenham, the retrieved text supplies the network operator, Metro Trains Melbourne. For Bhutan cricket, it supplies the 2001 ICC membership and international debut. The other cases cover a chart position, railway privatization, Welsh place names, a mountain range, and a legal proceeding. Highlighted spans identify continuation-relevant information that is missing or incomplete in the query chunk.

The selected records exclude the query chunk itself, exact query-text matches, and chunks from the same source block. Cross-block near duplicates may remain. These examples illustrate the relevance of retrieved supervision targets. They do not directly measure the fidelity of predicted memory or attribute the answer differences above to particular retrieved chunks.

Preceding Query Chunk	Retrieved Chunk Text	Ground-Truth Continuation
Retrieval case: Pakenham line ...a commuter rail passenger train service operating between Flinders Street in the Melbourne CBD and Pakenham in Melbourne's sout-	...is a commuter railway line in the eastern suburbs of Melbourne, Australia, and is part of the Melbourne rail **network operated by Metro Trains Melbourne**. The...	-heast. The service is part of the Public Transport Victoria railway **network** and is **operated by Metro Trains Melbourne**. The Pakenham line traverses mainly flat...
Retrieval case: Bhutan cricket ...cricket. The team is organised by the Bhutan Cricket Council Board, which has been a member of the International Cricket Council (ICC)...	...nations in the region. Bhutan became an affiliate member of the ICC **in 2001**. They made their **international debut** at the 2...	...**since 2001**. Bhutan made its **international debut** at the 2009 ACC Women's Twenty20 Championship in Malaysia...
Retrieval case: Music chart entry ..."Something to Remind You" did not enter the...	...Something to Remind You", despite not being a single, **peaked at No. 19 on the Bubbling Under Hot 100**,...	"Billboard" Hot 100, but **peaked at number 19 on the Bubbling Under Hot 100** Singles chart, which...
Retrieval case: JR Central station ...asō Station opened on November 25, 1922. The station was absorbed into the JR Central network upon the privatization of...	...928. The station was **absorbed into the JR Central network upon the privatization of Japanese National Railways (JNR) on April 1**...	**Japanese National Railways (JNR) on April 1, 1987**. Kawabe Kita Elementary School Hida River List of...
Retrieval case: Welsh place names ...a large portion of north-west Shropshire. In general, placenames of Welsh origin are found to the west of the border, and...	...of Brythonic origin. A final set of **Welsh place-names** are those for settlements in England which lie **close to the modern border with Wales**....	those of English origin to the east. However, many historically **Welsh names** are also found **east of the border**, particularly around Oswestry in northern...
Retrieval case: Gros Ventre Range ...The highest summit in the range is Doubletop Peak at. The Gros Ventre Range is mostly within the G-	...Ventre Range. Doubletop Peak is within the **Gros Ventre Wilderness region of Bridger-Teton National Forest**. The...	-ros **Ventre Wilderness of Bridger-Teton National Forest**. To the northwest of the range lies the valley known as Jackson Hole. Snow...
Retrieval case: Samata litigation ...The High Court dismissed the case after which Samata filed a...	...after which Samata filed a **Special Leave Petition in the Supreme Court of India**. **In July 1997** by a **three judge-bench**. In...	**Special Leave Petition in the Supreme Court of India**. A four- year battle led to a historic judgement **in July 1997** by a three...

Figure 6: Seven selected offline retrieval examples. Each row shows the preceding query chunk, retrieved chunk text, and ground-truth continuation. Brown spans mark continuation-relevant information in the retrieved text. Text is shortened for display, and split words preserve chunk boundaries.