In-Context Optimization for Retrieval-Augmented Generation: A Gradient-Descent Perspective

Mingchen Li[1,6]*, **Jiatan Huang**[2]*, **Chuxu Zhang**[2], **Liang Zhao**[3,4], **Hong Yu**[1,5,6]

[1]University of Massachusetts Amherst [2]University of Connecticut

[3]Emory University [4]Causal Dynamics Lab

[5]University of Massachusetts Lowell [6]VA Bedford Health Care

Abstract

In-context learning has recently been linked to implicit gradient descent in linear self-attention models, suggesting that context can induce a forward-pass update. Retrieval-augmented generation (RAG) also relies on context, but retrieved documents are usually treated as static evidence rather than signals for adaptation. We study RAG as an in-context optimization process. First, we show that one linear self-attention layer can implement one gradient-descent step on a unified linearized RAG objective covering both projection-based and dot-product retrieval interfaces. This gives an exact regime where retrieval-augmented prediction and in-context optimization coincide. We use this result not as a literal model of LLM computation, but as a guide for adapting the interaction between queries and retrieved evidence. We then test the boundary of this correspondence: it remains stable under controlled linear extensions, but becomes feature-distribution dependent under nonlinear architectures. Finally, we turn this view into a lightweight method for frozen RAG LLMs. The method keeps the retriever and backbone fixed, and predicts a context-conditioned update to a generator-side evidence-use interface. Across seven QA benchmarks, two retrievers, and two frozen LLM backbones, this forward-only update improves a shared-interface baseline, transfers to held-out tasks, and approaches test-time gradient adaptation at much lower per-query cost.

1 Introduction

Large language models (LLMs) have achieved strong performance across many natural-language tasks, but adapting them to knowledge outside their static pretraining corpus remains difficult. Retrieval-augmented generation (RAG) [28] addresses this limitation by conditioning a frozen LLM on documents retrieved from an external corpus. However, retrieval alone does not solve the full adaptation problem. After relevant documents are retrieved, the model must still decide how to use them for a new task, domain, or query distribution.

Existing RAG systems usually address this problem in one of three ways. The first is to keep both the retriever and generator fixed, and simply prepend retrieved documents to the input. This strategy is efficient, but it treats retrieved documents as static evidence and gives the model no mechanism to adjust how evidence should be used. The second is to fine-tune the retriever, the generator, or both. This can improve task performance, but it requires additional training and can be expensive when the task or domain changes. The third is to use in-context learning (ICL), where a few input-output examples are provided at inference time. ICL is attractive because it avoids full model retraining, but it is still unclear whether these examples merely provide extra demonstrations, or whether they can induce a more systematic update to how a RAG model uses retrieved evidence.

*indicates equal contribution | The code is available at GitHub.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

This paper asks a simple question: *can retrieved evidence and a few RAG examples act not only as context to read from, but also as a signal for adapting how the model uses evidence?* Answering this question requires connecting two views that have mostly been studied separately. On one side, recent theory shows that, under linear self-attention, in-context learning can implement gradient descent on the examples in the context [58, 2, 42]. This suggests that context can behave like a forward-pass update, rather than only as additional input text. On the other side, RAG introduces structure that is absent from standard ICL theory: a query, retrieved documents, query-evidence interactions, and a generator that must combine them to produce an answer. It remains unclear whether the gradient-descent view of ICL extends to retrieval-augmented prediction, and whether such a view can guide practical adaptation in real RAG systems.

We study RAG from this in-context optimization perspective. Our goal is not to claim that modern retrieval-augmented LLMs literally perform gradient descent during inference. Instead, we use a controlled linear setting to identify where such an update would act. In linear RAG, the relevant update acts on the interaction between the query and retrieved evidence. This gives a simple design principle for LLM-scale RAG: rather than changing which documents are retrieved, we adapt how the frozen generator uses the retrieved documents. Guided by this principle, we propose a forward-only adaptation method for frozen RAG LLMs. The method keeps both the external retriever and the LLM backbone fixed, and adapts only a lightweight generator-side evidence-use interface implemented with LoRA. At inference time, given new few-shot RAG demonstrations, the predictor produces the update in a single forward pass, enabling the generator to adjust how it uses retrieved documents without re-training on the new dataset.

We develop this idea in three steps. **First**, we prove that one linear self-attention layer can implement one gradient-descent step on a unified linearized RAG objective covering both projection-based and dot-product retrieval interfaces. **Second**, we test how far this correspondence extends beyond the exact construction. A trained self-attention layer closely matches the constructed gradient-descent predictor under controlled linear shifts, varying document counts, and stacked depths, while nonlinear architectures and real-world regression data reveal a clear dependence on feature distribution. **Third**, we use the optimization view to guide LLM-scale RAG adaptation. Across seven QA benchmarks, two retrievers, and two frozen LLM backbones, the predicted update improves a shared-interface baseline, transfers to held-out tasks, and approaches test-time gradient adaptation at much lower per-query cost. Our contributions are summarized as follows:

- **An in-context optimization view of linear RAG.** We extend the ICL-as-gradient-descent perspective from generator-only prediction to retrieval-augmented prediction. We prove that one linear self-attention layer can implement one gradient-descent step on a unified linearized RAG loss covering both linear-projection and dot-product retrieval interfaces. We also show that stacking K linear self-attention layers gives a multi-step view of in-context optimization for linear RAG.
- **A boundary analysis beyond the exact linear setting.** We test when the linear construction remains predictive and when it breaks. On synthetic linear regression tasks, a trained self-attention layer closely matches the constructed gradient-descent predictor under distribution shift, varying document counts, and stacked depths. On nonlinear architectures and four real-world regression datasets, the alignment degrades in a structured way and becomes sensitive to feature distribution.
- **Adapting evidence use without test-time backpropagation.** We use the optimization view to guide adaptation in frozen RAG LLMs. Rather than changing the external retriever, we adapt a generator-side evidence-use interface implemented with Q/K/V LoRA modules. A small context-conditioned predictor amortizes the autograd-defined K-step update to this interface. Across seven QA benchmarks, two backbones, and two retrievers, the predicted update improves a shared-interface baseline, transfers to held-out domains, and approaches test-time gradient adaptation at much lower per-query cost.

2 Related Work

Retrieval-augmented generation (RAG) conditions a language model on documents retrieved from an external corpus [28, 13, 24, 21, 6, 62]. Prior work explores retrieval, prompting, evidence fusion, memory-based knowledge access, and learning-based adaptation [50, 3, 18, 63, 35, 29]. Our focus

is complementary: rather than changing which documents are retrieved, we study how a frozen generator can adapt its use of already-retrieved evidence. We position this contribution relative to three lines of work.

In-context learning as gradient descent. A growing line of theory interprets in-context learning as implicit optimization. Under linear self-attention, a single ICL forward pass can implement one gradient-descent step, and stacked layers can implement multiple steps [58, 2, 42, 8]. Later work extends this view to preconditioned gradient descent [1], in-context algorithm selection [4], the role of depth [57, 10], and kernel-regression interpretations of attention [52, 51]. These analyses mainly study generator-only settings, often through linear regression or simplified attention. RAG introduces additional structure, including retrieved documents, query-evidence interactions, and evidence-conditioned generation. We extend the gradient-descent view to a linearized RAG setting and use it to identify where evidence-use adaptation should act.

Context-conditioned weight prediction. Another line of work learns auxiliary networks that produce model updates from a small context. HyperTuning [46] predicts soft prompts or low-rank weights from few-shot examples. HyperFlow [25] learns support-conditioned fine-tuning dynamics. MAC [54] maps documents into memory modulations. MEND [44] maps fine-tuning gradients into knowledge edits. RAG-GD follows the broad template of predicting an update from context, but differs in both the target and the adaptation site. The target is not a downstream task loss, a meta-learning objective, a memory objective, or a single editing gradient. Instead, the predictor matches an autograd-defined K-step SGD update induced by RAG-formatted demonstrations. The adapted parameters are also restricted to a generator-side evidence-use interface, while the retriever and backbone remain fixed.

Test-time adaptation. Standard adaptation either updates model parameters before deployment, as in fine-tuning and LoRA [15], or leaves the model unchanged at inference, as in pure ICL. Test-time training [53] lies between these extremes by updating parameters for each test instance, but this requires per-instance backpropagation and becomes expensive for large LLMs. Recent studies examine system-level strategies spanning ICL, fine-tuning, trainable RAG, and adaptive test-time compute [59, 45, 38, 41]. RAG-GD targets the same goal of adapting at inference time, but amortizes the update: a small predictor emits a LoRA update to the generator's evidence-use interface in one forward pass. Thus, it avoids backpropagation through the LLM at deployment while keeping both the external retriever and the frozen backbone unchanged.

3 A Linear RAG Setting Where Self-Attention Implements Gradient Descent

This section establishes the linear-regime basis for our in-context optimization view of RAG. We study a controlled setting in which retrieval-augmented prediction and gradient descent can be connected exactly. The goal is not to model modern RAG systems literally: real retrievers involve discrete document selection, and modern LLMs are deep and nonlinear. Instead, we isolate a differentiable retrieval-augmented prediction problem and show that one linear self-attention layer can realize the prediction shift produced by one gradient-descent step. The proof and explicit construction are in Appendix A, and the derivations for the retrieval variants are in Appendix B.

3.1 Self-Attention

We begin with a multi-head self-attention block parameterized by $\theta = \{P_h, W_{h,V}, W_{h,K}, W_{h,Q}\}_{h=1}^{H}$. Given tokens $\{e_1, \dots, e_N\} \subset \mathbb{R}^d$, the update for token e_j is

$$e_j \leftarrow e_j + \mathrm{SA}_\theta(j, \{e_i\}_{i=1}^N) = e_j + \sum_h P_h V_h \,\mathrm{softmax}(K_h^\top q_{h,j}), \tag{1}$$

where V_h, K_h, and $q_{h,j}$ are the value matrix, key matrix, and query vector for head h. Following [58, 57], we remove the softmax and bias terms to obtain the linear self-attention (LSA) update:

$$e_j \leftarrow e_j + \mathrm{LSA}_\theta(j, \{e_i\}_{i=1}^N) = e_j + \sum_h P_h V_h K_h^\top q_{h,j}. \tag{2}$$

3.2 A Unified Linearized RAG Predictor

We use a linearized abstraction of retrieval-augmented prediction. Rather than modeling discrete top-k selection, this abstraction captures a differentiable interface in which query features and retrieval-derived features jointly determine the prediction. Both a projection-based retrieval interface [28] and a dot-product retrieval interface [24] can be written as $y = W_1 x_1 + W_2 x_2$, where x_1 denotes the query-side feature and x_2 denotes the retrieval-derived feature. For the projection-based interface, we set $x_1 = x_q$, $x_2 = D$, and $W_2 \triangleq W_1 W_d$, where W_d projects document embeddings into the prediction space. For the dot-product interface, we set $x_1 = x_2 = x_q$ and $W_2 = W_z \left(\sum_i d_i d_i^\top\right) M^\top$, where M parameterizes query-document similarity. For tractability, we use the shared-encoder simplification $M = W_e^\top W_e$, so M is symmetric. The general DPR formulation [24] allows separate query and document encoders, with $M = W_q^\top W_d$. Full derivations are provided in Appendix B.

3.3 Optimization Objective

Given training examples $\{(x_1^i, x_2^i, y_i)\}_{i=1}^N$, we consider the squared loss

$$L(W_1, W_2) = \frac{1}{2N} \sum_{i=1}^{N} \left\| W_1 x_1^i + W_2 x_2^i - y_i \right\|^2 . \tag{3}$$

One gradient-descent step with learning rate η gives

$$\Delta W_k = -\eta \nabla_{W_k} L = -\frac{\eta}{N} \sum_{i=1}^{N} \left(W_1 x_1^i + W_2 x_2^i - y_i \right) (x_k^i)^\top, \qquad k \in \{1, 2\}. \tag{4}$$

For a query token with features (x_1, x_2), the corresponding prediction shift is $\Delta y \triangleq \Delta W_1 x_1 + \Delta W_2 x_2$. Thus, Δy is the change in prediction after updating W_k to $W_k' = W_k + \Delta W_k$.

3.4 Linear Self-Attention Reproduces One Gradient Step

Lemma 1 (Linear self-attention implements one RAG gradient step)**.** *Consider a 1-head linear self-attention layer, context tokens $e_i = (x_1^i, x_2^i, y^i)$ for $i = 1, \ldots, N$, and a query token $e_j = (x_1^j, x_2^j, y^j)$. Let ΔW_1 and ΔW_2 be the one-step gradient-descent updates in Eq. 4. There exist matrices W_K, W_Q, W_V and an output projection P such that one LSA update changes only the y-coordinate of e_j:*

$$e_j \leftarrow e_j + \left(0,\, 0,\, \Delta W_1 x_1^j + \Delta W_2 x_2^j\right). \tag{5}$$

Equivalently,

$$PVK^\top q_j = \left(0,\, 0,\, \Delta W_1 x_1^j + \Delta W_2 x_2^j\right). \tag{6}$$

Therefore, the LSA update exactly matches the prediction shift induced by one gradient-descent step on the unified linearized RAG predictor.

The construction is given in Appendix A. Intuitively, the value projection encodes the residual $W_1 x_1^i + W_2 x_2^i - y^i$. The key-query interaction computes the inner products $(x_1^i)^\top x_1^j$ and $(x_2^i)^\top x_2^j$. The output projection then writes the resulting weighted residual sum into the query token's prediction coordinate.

This construction also gives a controlled multi-step analogue. If each LSA layer represents one gradient-like update, then after K layers,

$$\hat{y}_{N+1}^{(K)} = \hat{y}_{N+1}^{(0)} + \sum_{t=0}^{K-1} \left(\Delta W_1^{(t)} x_{N+1}^1 + \Delta W_2^{(t)} x_{N+1}^2 \right), \tag{7}$$

where $\Delta W_1^{(t)}$ and $\Delta W_2^{(t)}$ are the implicit updates represented by layer t. We use this multi-step view as a linear-regime guide rather than as a literal claim about frozen LLM computation. In later sections, this view motivates a forward-only mechanism that adapts how a frozen generator uses retrieved evidence.

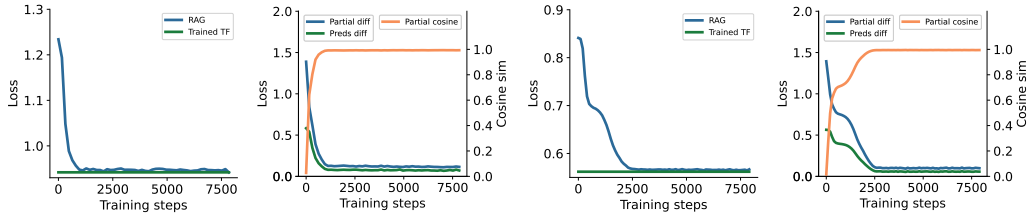


Figure 1: Single-layer LSA reproduces one gradient-descent step on the unified linearized RAG loss. The two left panels report the projection-based interface, and the two right panels report the dot-product interface. Across both variants, the trained LSA layer and the constructed gradient-descent predictor are nearly indistinguishable on held-out tasks.

4 Testing the Boundary of the Linear Correspondence

Lemma 1 gives an exact correspondence in a controlled linear setting. We now ask how far this correspondence remains predictive when the setting is varied. The experiments have two goals: first, to verify that a trained linear self-attention layer can reproduce the constructed gradient-descent predictor; second, to identify where the correspondence begins to break beyond the exact regime.

4.1 Linear-Regime Verification

Each token concatenates an input feature, a retrieval-derived feature, and a target, $e_i = (x_i, z_i, y_i)$ for $i = 1, \ldots, N$. The auxiliary slot z_i instantiates the unified RAG view. For the projection-based interface, z_i is a document-derived feature. For the dot-product interface, $z_i = x_i$, and document information is injected into the keys and values. We train an LSA layer θ to minimize expected squared error across tasks, using minibatch SGD over freshly sampled tasks. Following prior work [9, 58], each task is generated from a teacher with weights $W_\tau \sim \mathcal{N}(0, I)$. Inputs are sampled as $x_{\tau,i} \sim \mathcal{U}(-1, 1)^{n_I}$, and targets are generated by $y_{\tau,i} = W_\tau^1 x_{\tau,i}^1 + W_\tau^2 x_{\tau,i}^2$. We set $N = n_I = 10$ and sweep the document count $k \in \{2, 5, 10, 25\}$. We compare the trained layer θ^* with the constructed predictor that exactly realizes one gradient-descent step on the unified RAG loss. On $T_{\text{val}} = 10^4$ held-out tasks, we report the prediction difference $\|\hat{y}_{\theta^*} - \hat{y}_{\theta,\text{rag}}\|_2$, the cosine similarity between input sensitivities $\partial\hat{y}/\partial x_{\text{test}}$, and the corresponding sensitivity ℓ_2 difference. Details are in Appendix C. Figure 1 verifies the construction for both retrieval interfaces. The trained LSA layer closely matches the constructed predictor: the loss difference is small, the sensitivity cosine is close to 1, and the sensitivity ℓ_2 difference is negligible. This numerically confirms the algebraic correspondence in Lemma 1.

4.2 Controlled Stress Tests

We next test whether the agreement persists under controlled changes within the linear regime. We vary the document count, shift the test-input distribution, and stack LSA layers with shared parameters. When sweeping $n \in \{2, 5, 10, 25\}$ and shifting the test-input range to $\alpha \in \{0.5, 1, 1.5, 2\}$ while keeping training fixed at $\alpha = 1$, the loss difference between the trained Transformer and the gradient predictor remains small (Figure 5, Appendix C). Stacking LSA layers further supports the multi-step picture in Eq. 7. At depths 2 and 5, the loss and prediction differences remain small across document counts. The residual gap at $\text{Docs} = 25$ also shrinks as depth increases (Figure 6, Appendix C). The projection-based interface shows similar behavior (Appendix E). These results suggest that the linear correspondence is not a fragile single-step artifact, but remains stable under controlled linear extensions.

4.3 Nonlinear Stress Test

We then examine where the correspondence begins to break. We add MLP layers after the input embedding and evaluate on four real-world regression datasets: California Housing, Bike Sharing, Wine Quality, and Predict Calorie Expenditure. We focus on the dot-product interface throughout. Dataset details are in Appendix F. This experiment is diagnostic. We do not claim that normalization solves RAG adaptation. Instead, we use normalization to control the feature geometry that interacts with dot-product retrieval. We compare Z-score [5], Min–Max [5], rank-based normalization [7],

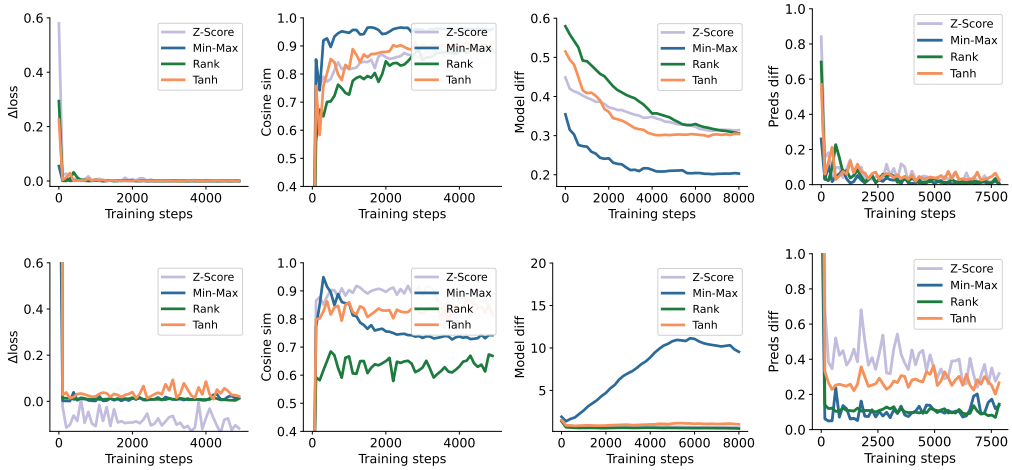


Figure 2: Effect of input normalization on the alignment between the trained nonlinear Transformer and the gradient-descent predictor under the dot-product interface. *Top row:* Bike Sharing. *Bottom row:* California Housing. Columns report loss difference, sensitivity cosine, model difference, and prediction difference. Min–Max normalization closely matches the gradient-descent predictor on Bike Sharing, where features are bounded and roughly uniform. On California Housing, where features are skewed and heavy-tailed, the alignment degrades.

and Tanh normalization. The training set is used as the retrieval corpus and is normalized with Z-score throughout. Only the input-side normalization is varied, and alignment is measured using the same metrics as in Section 4.1. Figure 2 shows two representative cases. On Bike Sharing, Min–Max normalization gives the closest agreement between the trained nonlinear Transformer and the gradient-descent predictor, likely because the features are bounded and not dominated by outliers. On California Housing, the alignment is weaker: skewed and heavy-tailed features make dot-product geometry more sensitive to outliers. The sensitivity cosine drops, the model difference grows, and prediction differences become less stable. The same pattern appears on the remaining datasets. Predict Calorie Expenditure behaves similarly to Bike Sharing, while Wine Quality behaves more like California Housing (Figure 7, Appendix D). Overall, the linear optimization view remains informative when feature geometry is stable, but becomes less predictive when retrieval-derived dot products are dominated by skewed or heavy-tailed features. This empirical boundary supports our use of the linear construction as a guide for adaptation, rather than as a literal model of LLM computation. Next, we use this view to design a forward-only update to the generator-side evidence-use interface in LLM-scale RAG.

5 LLM-Scale RAG: Amortizing the Gradient-Descent Update

We now instantiate this view as RAG-GD, a forward-only adaptation method for frozen billion-parameter RAG LLMs. In this setting, we do not assume an exact equivalence between an LLM forward pass and gradient descent. Instead, we use gradient descent as an operational target for adapting how the generator uses retrieval-conditioned information. Given a few RAG-formatted demonstrations, we use autograd during training to compute the update that gradient descent would make to the generator-side retrieval interface. We then train a lightweight predictor to approximate this update. At inference time, the predictor produces the update in a single forward pass, without further training the RAG system or backpropagating through the frozen LLM.

Concretely, we first train a base retrieval adapter W_0^{ret} on RAG-formatted examples from the NQ training split. It is a low-rank LoRA perturbation to the Q/K/V projections of every attention layer in a frozen LLM. All inputs are RAG-formatted instances $(x, \mathcal{D}_x, y)$, where y is the gold answer. In this work, we use a fixed external retriever, either BM25 or E5, to select the retrieved documents $\mathcal{D}_x$ for each question x from a fixed corpus. This adapter serves as a generator-side retrieval interface: it does not select documents, but modulates how the generator uses evidence selected by BM25 or E5. The predictor g_ϕ is then meta-trained on few-shot support contexts $C = \{(x_i, \mathcal{D}_i, y_i)\}_{i=1}^{N}$ from NQ,

TriviaQA, HotpotQA, 2WikiMultiHopQA, and MuSiQue. PopQA and Bamboogle are held out from both stages and used only for evaluation.

5.1 Supervision Target: Autograd-Defined Interface Update

The supervision target is the update that K SGD steps would produce on the generator-side retrieval interface using a support context C. Starting from $W_{\mathrm{ret}}^{(0)} = W_0^{\mathrm{ret}}$, we run

$$\Delta W_{\mathrm{GD}}^{(K)}(C) = W_{\mathrm{ret}}^{(K)} - W_0^{\mathrm{ret}}, \qquad W_{\mathrm{ret}}^{(t+1)} = W_{\mathrm{ret}}^{(t)} - \eta \nabla \mathcal{L}(W_{\mathrm{ret}}^{(t)}; C), \tag{8}$$

for $t = 0, \ldots, K-1$, where $\mathcal{L}$ is the answer-token cross-entropy conditioned on the question and retrieved documents. We compute $\Delta W_{\mathrm{GD}}^{(K)}(C)$ with autograd and detach it from the predictor optimizer. The practical method is amortized test-time adaptation through a context-conditioned LoRA predictor, rather than a direct implementation of Lemma 1. The linear analysis motivates two design decisions: *where to adapt*, since the support-induced update acts on the query–evidence interaction, and *what to predict*, since this update is a gradient-descent step in the linear setting. We therefore adapt the generator-side evidence-use interface and train the predictor to match the K-step gradient update rather than a downstream task loss.

5.2 Predictor Architecture and Matching Objective

The predictor g_ϕ is a context encoder with per-layer, per-projection update heads. Each demonstration $(x_i, \mathcal{D}_i, y_i) \in C$ is formatted by concatenating the question, retrieved documents, and gold answer. The frozen LLM with the base adapter encodes each sequence, and we use the EOS hidden state $h_i \in \mathbb{R}^{d_h}$. We aggregate demonstrations by mean pooling, $\bar{h}(C) = \frac{1}{N}\sum_{i=1}^{N} h_i$. For each layer ℓ and projection type $\pi \in \{Q, K, V\}$, the update head outputs $\widetilde{\Delta W}_{\ell,\pi} = U_{\ell,\pi} V_{\ell,\pi}^\top \in \mathbb{R}^{d\times d}$, where $U_{\ell,\pi}, V_{\ell,\pi} \in \mathbb{R}^{d\times r}$ are generated by a two-layer MLP from $\bar{h}(C)$. The rank r matches the base adapter, so $g_\phi(C)$ has the same shape as W_0^{ret}. We train g_ϕ to match the autograd-defined target per layer and projection type. Let $\Delta W_{\ell,\pi}^\star \triangleq [\Delta W_{\mathrm{GD}}^{(K)}(C)]_{\ell,\pi}$. The matching loss is

$$\mathcal{L}_{\mathrm{match}}(\phi; C) = \sum_{\ell,\pi} \left[1 - \frac{\langle \widetilde{\Delta W}_{\ell,\pi}, \Delta W_{\ell,\pi}^\star \rangle}{\|\widetilde{\Delta W}_{\ell,\pi}\|_F \|\Delta W_{\ell,\pi}^\star\|_F} + \lambda \left| \log \frac{\|\widetilde{\Delta W}_{\ell,\pi}\|_F}{\|\Delta W_{\ell,\pi}^\star\|_F} \right| \right], \tag{9}$$

where the cosine term matches direction and the log-magnitude term matches scale. We use $\lambda = 0.1$ throughout. At deployment, the predictor emits $\widetilde{\Delta W}(C)$ in one forward pass, and the frozen LLM answers with the adapted interface $W_0^{\mathrm{ret}} + \widetilde{\Delta W}(C)$.

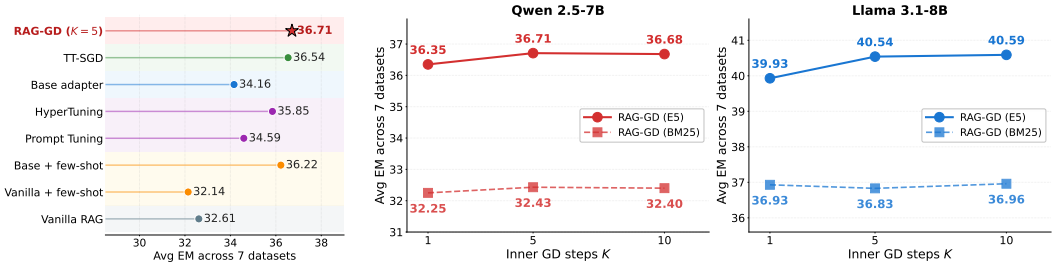
5.3 Benchmarks, Baselines, and Metrics

Benchmarks. We evaluate on seven open-domain QA benchmarks: NQ, TriviaQA, PopQA, HotpotQA, 2WikiMultiHopQA, MuSiQue, and Bamboogle. NQ, TriviaQA, and PopQA are single-hop, while the remaining four are multi-hop. We use Qwen 2.5-7B-Instruct [48] and Llama 3.1-8B-Instruct [11] as frozen backbones. Each query is augmented with the top five documents from BM25 or E5-large [60], using the same retrieval cache for all methods. The support size is $N = 3$, and we train predictors with $K \in \{1, 5, 10\}$.

Baselines and metrics. We compare RAG-GD with six baselines. **Query Only** uses no retrieved documents. **Vanilla RAG** prepends retrieved documents to the prompt. **Base adapter** applies W_0^{ret} without context-conditioned perturbation. **+ few-shot** variants concatenate support demonstrations into Vanilla RAG or Base adapter prompts. **Prompt Tuning** [27] learns a soft prefix using the same supervision pool as W_0^{ret}. **HyperTuning** [46] trains a hypernetwork to generate task-specific adaptation parameters from few-shot examples by optimizing the prediction loss of a frozen downstream model. **TT-SGD** performs K SGD steps on C at test time and serves as the non-amortized reference. RAG-GD shares W_0^{ret} with Base adapter and adds only $\widetilde{\Delta W}(C)$. Table 1 reports the headline comparison against no-perturbation baselines. Full results for Prompt Tuning, HyperTuning, TT-SGD, and + few-shot variants are in Appendix I. We report SQuAD-style [49] exact match (EM) and token-overlap F1.

Table 1: Main QA results across seven benchmarks, two frozen LLM backbones, and two retrievers. RAG-GD (K=5) applies a context-conditioned update on top of the same static retrieval adapter W_0^{ret} used by Base adapter. Bold values mark the best result per column for each backbone–retriever pair. Full context-conditioned baseline results are in Appendix I.

Method	Retriever	Single-Hop QA						Multi-Hop QA								Avg.	
		NQ		TriviaQA		PopQA		HotpotQA		2Wiki		MuSiQue		Bamboogle			
		EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Qwen 2.5-7B																	
Query Only	–	15.95	24.28	43.33	49.51	16.02	19.76	18.40	25.39	23.91	28.12	3.80	10.57	11.20	18.02	18.94	25.09
Vanilla RAG	BM25	27.61	36.66	58.24	65.77	28.84	33.18	31.28	41.25	27.87	33.24	5.87	13.05	10.40	21.16	27.16	34.90
	E5	39.16	50.03	62.99	70.80	44.03	50.21	32.45	42.21	25.48	31.43	5.79	12.77	18.40	26.74	32.61	40.60
Base adapter	BM25	32.57	41.45	60.11	67.93	31.55	35.63	32.31	43.45	28.22	34.13	6.41	15.46	16.80	26.01	29.71	37.72
	E5	41.77	51.22	63.31	71.62	47.05	51.82	33.78	44.64	27.89	33.97	6.95	15.76	18.40	28.78	34.16	42.54
RAG-GD (K=5)	BM25	**34.46**	**43.54**	**63.27**	**70.69**	**33.22**	**37.69**	**35.54**	**47.14**	**28.86**	**34.48**	**9.26**	**19.73**	**22.40**	**32.85**	**32.43**	**40.87**
	E5	**42.91**	**52.71**	**65.98**	**73.60**	**48.12**	**52.61**	**35.54**	**47.00**	**29.67**	**35.47**	**9.14**	**19.26**	**25.60**	**35.12**	**36.71**	**45.11**
Llama 3.1-8B																	
Query Only	–	22.46	32.51	52.67	59.79	20.63	25.09	18.31	25.71	26.39	31.04	3.81	9.51	6.40	12.88	21.52	28.08
Vanilla RAG	BM25	31.41	40.95	60.43	68.35	31.08	35.43	31.92	42.46	26.07	31.82	5.75	12.44	14.40	22.93	28.72	36.34
	E5	40.72	52.23	64.42	72.58	45.85	51.48	32.78	43.19	23.44	29.46	6.04	12.25	24.80	32.12	34.01	41.90
Base adapter	BM25	38.47	49.42	62.66	72.46	37.80	**42.29**	37.35	50.41	33.66	39.55	11.46	21.98	**29.60**	**41.29**	35.86	45.34
	E5	43.46	54.60	63.90	74.05	51.69	55.74	37.31	49.90	33.45	39.45	11.83	22.07	30.40	40.48	38.86	48.04
RAG-GD (K=5)	BM25	**40.22**	**50.01**	**66.13**	**74.28**	**37.81**	42.19	**38.99**	**51.14**	**34.15**	**40.04**	**12.54**	**22.61**	28.00	39.61	**36.83**	**45.70**
	E5	**45.68**	**55.84**	**67.66**	**76.07**	**52.31**	**56.48**	**39.01**	**50.94**	**33.94**	**39.83**	**13.20**	**23.47**	**32.00**	**42.08**	**40.54**	**49.24**



(a) Method-family comparison.

(b) Robustness to inner GD depth K.

Figure 3: **The gradient-derived update improves context-conditioned adaptation and is relatively insensitive to K. (a)** Average EM across seven QA benchmarks on Qwen 2.5-7B with E5 retrieval. RAG-GD matches the TT-SGD reference using one predictor forward pass. **(b)** Average EM across $K \in \{1, 5, 10\}$ on both backbones and retrievers. Solid lines use E5, and dashed lines use BM25.

5.4 Results

Table 1 reports the headline comparison between RAG-GD (K=5) and the no-perturbation baselines. Figures 3 and 4 compare against additional context-conditioned methods, including Prompt Tuning, HyperTuning, TT-SGD, and + few-shot variants. Full per-method and per-benchmark results are in Appendix I, and Algorithm 1 gives the deployment procedure.

The predicted update improves the base retrieval adapter. Across all backbone and retriever configurations, RAG-GD improves average EM and F1 over Base adapter. This comparison is controlled: both methods share the external retriever, retrieval cache, frozen backbone, and base adapter W_0^{ret}. The only difference is the predicted perturbation $\widetilde{\Delta W}(C)$, which isolates the effect of adapting the generator-side retrieval interface from the support context.

The learned update transfers to held-out tasks. PopQA and Bamboogle are held out from training for both W_0^{ret} and g_ϕ. On PopQA, RAG-GD improves EM over Base adapter in every backbone and retriever setting, with F1 close or improved in most cases. On Bamboogle, gains are strongest on Qwen, while Llama with BM25 stays close to Base adapter. The transfer is consistent but not uniform, suggesting that g_ϕ learns a reusable update rule for generator-side evidence use rather than only fitting the meta-training tasks.

Gradient-update supervision matters. Figure 3a compares RAG-GD with context-conditioned baselines on Qwen 2.5-7B with E5 retrieval. HyperTuning uses the same predictor architecture but

trains through downstream task loss rather than matching the autograd-defined update. RAG-GD improves average EM and F1 over HyperTuning and Prompt Tuning, indicating that the gradient-derived target contributes beyond the predictor architecture.

Performance is largely insensitive to K. Figure 3b sweeps the inner-loop depth $K \in \{1, 5, 10\}$ across both backbones and retrievers. A single amortized step already recovers most of the gain, while additional steps bring only small and configuration-dependent changes. Thus, the K=5 setting in Table 1 is representative. Full per-K results are in Appendix I.

Robustness to retrieval distractors. We evaluate Qwen 2.5-7B-Instruct with E5 retrieval by replacing $r \in \{0, 1, 3, 5\}$ of the top-five documents for each query with the top-ranked document retrieved for another query from the same benchmark. The document count and context budget are held fixed. Replacement positions are sampled uniformly with a per-query seed, giving all methods identical corrupted contexts; support demonstrations remain clean.

Table 2: Retrieval-distractor robustness over seven benchmarks. We report average EM; parentheses show retention relative to clean performance. Bold marks the highest EM at each r.

Method	$r = 0$ (clean)	$r = 1$	$r = 3$	$r = 5$
Vanilla RAG	32.61	30.57 (93.7%)	26.40 (81.0%)	19.12 (58.6%)
Base adapter	34.16	32.01 (93.7%)	27.52 (80.6%)	21.07 (61.7%)
RAG-GD (K=5)	**36.71**	**34.09** (92.9%)	**30.38** (82.8%)	**23.88** (65.1%)

RAG-GD achieves the highest EM at every tested level (Table 2). Its retention is slightly lower at $r = 1$ but higher at $r \in \{3, 5\}$. At $r = 5$, it exceeds Base adapter by 2.81 EM points, retaining 65.1% of its clean performance versus 61.7% for Base adapter. Robustness to passages explicitly supporting incorrect answers requires separate evaluation.

Amortization approaches test-time adaptation at lower cost. Figure 4 shows the EM-cost tradeoff for Qwen 2.5-7B with E5 retrieval. TT-SGD performs inner-loop backpropagation through the 7B LLM at test time, while RAG-GD moves this computation into training and uses only one forward pass through g_ϕ at inference. As a result, RAG-GD reaches a similar average EM and F1 operating point at substantially lower per-query cost.

5.5 Discussion

These results complete the theory-to-practice arc. The linear regime gives an exact gradient-descent correspondence, while the nonlinear experiments show where this correspondence becomes feature-dependent. At LLM scale, we do not claim that frozen RAG LLMs implement the linear equivalence internally. Instead, the autograd-defined update provides a practical target for context-conditioned adaptation. Because RAG-GD and Base adapter share the same retriever, retrieval cache, backbone, and W_0^{ret}, the gains isolate the contribution of the predicted update to the generator-side retrieval interface. The held-out transfer and cost-performance tradeoff suggest that gradient-supervised retrieval-interface adaptation is a promising forward-only alternative to test-time backpropagation for RAG.

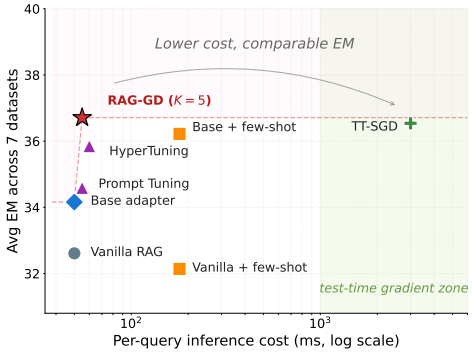


Figure 4: **EM-cost tradeoff** on Qwen 2.5-7B with E5 retrieval. Per-query cost is shown on a log scale. The shaded region marks methods that run inner GD at test time.

6 Conclusion

We studied retrieval-augmented generation through an in-context optimization lens. In a controlled linear setting, we showed that one linear self-attention layer can implement one gradient-descent step on a unified linearized RAG loss covering projection-based and dot-product retrieval interfaces.

Empirical tests verified this construction in the exact regime and revealed a structured boundary under nonlinear architectures and real regression data, where alignment becomes sensitive to feature distribution. At LLM scale, we turned this view into a forward-only adaptation method by using the autograd-defined K-step update to a generator-side Q/K/V LoRA interface as supervision for a lightweight predictor. Across seven QA benchmarks, two retrievers, and two frozen backbones, the predicted update improved a shared-adapter baseline, transferred to held-out domains, and was largely insensitive to K. Overall, these results suggest that retrieved evidence can be treated not only as external context, but also as a signal for context-induced adaptation in RAG.

7 Limitations

Our linear construction is an analytical starting point rather than a literal account of modern RAG LLMs, and the nonlinear experiments show that the correspondence depends on architecture and feature distribution. At LLM scale, we instantiate the view with a generator-side Q/K/V LoRA interface while keeping the retriever and backbone fixed. Open questions remain about the inner-loop optimizer, update parameterization, predictor capacity, scaling to larger backbones, and robustness across broader retrieval settings. Future work should study whether larger support sets can be aggregated into parameter updates to reduce the prompt-length overhead of concatenated ICL, while accounting for support-encoding cost and information loss in mean pooling. Further directions include combining predicted updates with selectively retained demonstrations and using uncertainty-aware gating to suppress unreliable updates under noisy retrieval.

8 Acknowledgments

This work was supported in part by the National Institutes of Health (NIH; grants 5R01DA056470 and 5R01AG080670) and the U.S. Department of Veterans Affairs Health Systems Research (HSR; awards 5I01HX003711 and 1I01HX003969). The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH or the U.S. Department of Veterans Affairs.

References

[1] Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. In *NeurIPS*, 2023.

[2] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *ICLR*, 2023.

[3] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *ICLR*, 2024.

[4] Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. In *NeurIPS*, 2023.

[5] C.M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[6] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, et al. Improving language models by retrieving from trillions of tokens. In *ICML*, 2022.

[7] William Jay Conover. *Practical nonparametric statistics*. Wiley, 1999.

[8] Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers. In *ACL*, 2023.

[9] Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In *NeurIPS*, 2022.

[10] Khashayar Gatmiry, Nikunj Saunshi, Sashank J. Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can looped transformers learn to implement multi-step gradient descent for in-context learning? In *ICML*, 2024.

[11] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, et al. The llama 3 herd of models. *CoRR*, 2024.

[12] Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. Deeprag: Thinking to retrieve step by step for large language models. In *International Conference on Learning Representations*, volume 2026, pages 117056–117071, 2026.

[13] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. In *ICML*, 2020.

[14] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *COLING*, 2020.

[15] 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 *ICLR*, 2022.

[16] Jiatan Huang, Mingchen Li, Ziming Li, Sunjae Kwon, Hong Yu, and Chuxu Zhang. Counterfactual graph for multi-agent llm calibration. *arXiv preprint arXiv:2605.30653*, 2026.

[17] Jiatan Huang, Mingchen Li, Zonghai Yao, Dawei Li, Yuxin Zhang, Zhichao Yang, Yongkang Xiao, Feiyun Ouyang, Xiaohan Li, Shuo Han, et al. Ritek: A dataset for large language models complex reasoning over textual knowledge graphs in medicine. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 35985–36001, 2026.

[18] Jiatan Huang, Mingchen Li, Zonghai Yao, Dawei Li, Yuxin Zhang, Zhichao Yang, Yongkang Xiao, Feiyun Ouyang, Xiaohan Li, Shuo Han, and Hong Yu. Ritek: A dataset for large language models complex reasoning over textual knowledge graphs in medicine. In *ACL*, 2026.

[19] Jiatan Huang, Zheyuan Zhang, Tianyi Ma, Mingchen Li, Yaning Zheng, Yanfang Ye, and Chuxu Zhang. Glen-bench: A graph-language based benchmark for nutritional health. *arXiv preprint arXiv:2601.18106*, 2026.

[20] Jiatan Huang, Zheyuan Zhang, Kaiwen Shi, Yanfang Ye, and Chuxu Zhang. Evolver-outer: Co-evolving routing and prompt for multi-agent question answering. *arXiv preprint arXiv:2604.05149*, 2026.

[21] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*, 2021.

[22] Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. In *WWW*, 2025.

[23] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, 2017.

[24] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP*, 2020.

[25] Donggyun Kim, Chanwoo Kim, and Seunghoon Hong. Hyperflow: Gradient-free emulation of few-shot fine-tuning. *CoRR*, 2025.

[26] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. In *TACL*, 2019.

[27] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 2021.

[28] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*, 2020.

[29] Mingchen Li, Jiatan Huang, Zonghai Yao, and Hong yu. Efficient and effective internal memory retrieval for llm-based healthcare prediction. In *ACL*, 2026.

[30] Mingchen Li, Jiatan Huang, Zonghai Yao, and Hong Yu. Efficient and effective internal memory retrieval for llm-based healthcare prediction. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 35895–35906, 2026.

[31] Mingchen Li, Jiatan Huang, and Hong Yu. Rivo: Lightweight adaptive rerouting for deep research agents. *arXiv preprint arXiv:2610.15910*, 2026.

[32] Mingchen Li and Lifu Huang. Understand the dynamic world: An end-to-end knowledge informed framework for open domain entity state tracking. In *SIGIR*, 2023.

[33] Mingchen Li, Halil Kilicoglu, Hua Xu, and Rui Zhang. Biomedrag: A retrieval augmented large language model for biomedicine. *JBI*, 2025.

[34] Mingchen Li, Chen Ling, Rui Zhang, and Liang Zhao. A condensed transition graph framework for zero-shot link prediction with large language models. In *ICDM*, 2024.

[35] Mingchen Li, Hansi Zeng, Zhuo Qian, Jiatan Huang, Sunjae Kwon, Hamed Zamani, and Hong Yu. Rice-po: Turning retrieval interactions into credit signals for reasoning agents. *arXiv preprint arXiv:2605.26352*, 2026.

[36] Mingchen Li, Hansi Zeng, Zhuo Qian, Jiatan Huang, Sunjae Kwon, Hamed Zamani, and Hong Yu. Rice-po: Turning retrieval interactions into credit signals for reasoning agents. *arXiv preprint arXiv:2605.26352*, 2026.

[37] Mingchen Li, Zaifu Zhan, Jiatan Huang, Jeremy Yeung, Kai Ding, Anne Blaes, Steven Johnson, Hongfang Liu, Hua Xu, and Rui Zhang. Cancerllm: a large language model in cancer domain. *NPJ Digital Medicine*, 9(1):266, 2026.

[38] Mingchen Li, Zaifu Zhan, Han Yang, Yongkang Xiao, Jiatan Huang, and Rui Zhang. Benchmarking retrieval-augmented large language models in biomedical nlp: Application, robustness, and self-awareness. *Science Advances*, 2025.

[39] Mingchen Li, Zaifu Zhan, Han Yang, Yongkang Xiao, Huixue Zhou, Jiatan Huang, and Rui Zhang. Benchmarking retrieval-augmented large language models in biomedical nlp: Application, robustness, and self-awareness. *Science Advances*, 11(47):eadr1443, 2025.

[40] Mingchen Li, Huixue Zhou, Han Yang, and Rui Zhang. Rt: a retrieving and chain-of-thought framework for few-shot medical named entity recognition. *Journal of the American Medical Informatics Association*, 31(9):1929–1938, 2024.

[41] Ziming Li, Jiatan Huang, Xiaoguang Guo, Guiling Wang, and Chuxu Zhang. Same signal, opposite meaning: Direction-informed adaptive learning for llm agents. *arXiv preprint arXiv:2605.06908*, 2026.

[42] Arvind Mahankali, Tatsunori B. Hashimoto, and Tengyu Ma. One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. *CoRR*, 2023.

[43] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *ACL*, 2023.

[44] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *ICLR*, 2022.

[45] Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. In *ACL*, 2023.

[46] Jason Phang, Yi Mao, Pengcheng He, and Weizhu Chen. Hypertuning: Toward adapting large language models without back-propagation. In *ICML*, 2023.

[47] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *EMNLP*, 2023.

[48] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, et al. Qwen2.5 technical report. *CoRR*, 2024.

[49] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016.

[50] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. In *TACL*, 2023.

[51] Ruifeng Ren and Yong Liu. Towards understanding how transformers learn in-context through a representation learning lens. In *NeurIPS*, 2024.

[52] Zhaiming Shen, Alexander Hsu, Rongjie Lai, and Wenjing Liao. Understanding in-context learning on structured manifolds: Bridging attention to kernel methods. In *ICLR*, 2026.

[53] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

[54] Jihoon Tack, Jaehyung Kim, Eric Mitchell, Jinwoo Shin, Yee Whye Teh, and Jonathan Richard Schwarz. Online adaptation of language models with a memory of amortized contexts. In *NeurIPS*, 2024.

[55] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. In *TACL*, 2022.

[56] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 2008.

[57] Max Vladymyrov, Johannes von Oswald, Mark Sandler, and Rong Ge. Linear transformers are versatile in-context learners. In *ICML*, 2024.

[58] Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *ICML*, 2023.

[59] Fan Wang, Chuan Lin, Yang Cao, and Yu Kang. Benchmarking general-purpose in-context learning. *CoRR*, 2024.

[60] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *CoRR*, 2024.

[61] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, 2018.

[62] Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Hao Chen, Yilin Xiao, Chuang Zhou, Junnan Dong, Yi Chang, and Xiao Huang. A survey of graph retrieval-augmented generation for customized large language models. *CoRR*, 2025.

[63] Yedi Zhang, Aaditya K. Singh, Peter E. Latham, and Andrew Saxe. Training dynamics of in-context learning in linear attention. In *ICML*, 2025.

Broader Impacts

The method we propose lowers the cost of adapting a deployed large language model to a new task or domain. At inference, the LLM still runs a forward pass to generate the answer, but no backward pass through the LLM is required: adapting to the context costs only a single small forward pass through a context-conditional weight predictor. If adopted at scale, this could reduce the energy footprint of test-time adaptation pipelines for retrieval-augmented systems. The flip side is that lower adaptation cost could also accelerate the deployment of models in domains for which the underlying LLM has not been carefully evaluated, including settings where retrieval can amplify biases in the corpus. We encourage practitioners adopting this style of adaptation to retain the same evaluation discipline that would apply to a fine-tuned model. All datasets used in our evaluation are publicly available QA and regression benchmarks that do not contain personally identifiable or sensitive information, and our work does not raise additional ethical concerns beyond those discussed above.

Reproducibility Statement

All datasets used in our main evaluation are publicly available QA benchmarks (NQ, TriviaQA, PopQA, HotpotQA, 2WikiMultiHopQA, MuSiQue, Bamboogle). The supplementary regression datasets discussed in the appendix are also public. Preprocessing steps, dataset splits, and the training pools used for the reference retrieval adapter W_0^{ret} and the predictor g_ϕ are documented in the appendix.

Our implementation uses PyTorch with the Hugging Face Transformers library on top of frozen Qwen 2.5-7B-Instruct [48] and Llama 3.1-8B-Instruct [11] backbones. Hyperparameters for the reference retrieval adapter W_0^{ret}, the predictor g_ϕ, and the inner-loop SGD target (η, K, number of demonstrations N) are listed in Appendix G. All experiments were conducted on NVIDIA A100 GPUs.

A Proof of Lemma 1

Statement. Given a 1-head linear-attention layer and tokens $e_j = (x_1^j,\, x_2^j,\, y^j)$ for $j = 1, \ldots, N$, we construct key, query, and value matrices W_K, W_Q, W_V and a projection P such that one linear self-attention update on each e_j matches one gradient-descent step on the unified RAG loss of Section 3. The update modifies only the y-coordinate of each token:

$$e_j \leftarrow e_j + \big(0,\ 0,\ \Delta W_1 x_1^j + \Delta W_2 x_2^j\big) \ =\ e_j + P\,V\,K^\top q_j, \tag{1}$$

where ΔW_1 and ΔW_2 are the gradient-step updates of Eq. 4 in the main text.

Setup. We are given N context tokens, each of the form $e^i = (x_1^i,\, x_2^i,\, y^i)$ corresponding to one training pair, plus a query token $e_{N+1} = (x_1^{N+1},\, x_2^{N+1},\, 0)$ at position $N{+}1$. The model is asked to predict the updated y-value at position $N{+}1$.

Step 1: expand the post-update prediction. Writing the post-update prediction y' as the original prediction plus the contributions of ΔW_1 and ΔW_2,

$$\begin{aligned} y' &= W_1' x_1 + W_2' x_2 \\ &= (W_1 + \Delta W_1)\, x_1 + (W_2 + \Delta W_2)\, x_2 \\ &= W_1 x_1 + W_2 x_2 \ +\ \Delta W_1 x_1 + \Delta W_2 x_2. \end{aligned} \tag{2}$$

Step 2: gradient step on the unified RAG loss. Under the squared loss $L(W_1, W_2) = \frac{1}{2N}\sum_{i=1}^N \|W_1 x_1^i + W_2 x_2^i - y_i\|^2$, one gradient step with learning rate η yields

$$\Delta W_1 = -\eta\, \nabla_{W_1} L = -\frac{\eta}{N} \sum_{i=1}^N \big(W_1 x_1^i + W_2 x_2^i - y_i\big)\,(x_1^i)^\top, \tag{3}$$

$$\Delta W_2 = -\eta\, \nabla_{W_2} L = -\frac{\eta}{N} \sum_{i=1}^N \big(W_1 x_1^i + W_2 x_2^i - y_i\big)\,(x_2^i)^\top, \tag{4}$$

$$\Delta y = \Delta W_1 x_1 + \Delta W_2 x_2. \tag{5}$$

Step 3: rewrite Δy as a sum of outer-product contractions. Substituting Eqs. 3–4 into Eq. 5 and evaluating at the query token j,

$$\Delta y \;=\; -\frac{\eta}{N}\sum_{i=1}^{N}\left(W_1x_1^i + W_2x_2^i - y_i\right)(x_1^i)^\top x_1^j \;-\; \frac{\eta}{N}\sum_{i=1}^{N}\left(W_1x_1^i + W_2x_2^i - y_i\right)(x_2^i)^\top x_2^j. \quad (6)$$

Equivalently, the update applied to the token at position j is

$$\begin{pmatrix} x_1^j \\ x_2^j \\ y^j \end{pmatrix} \leftarrow \begin{pmatrix} x_1^j \\ x_2^j \\ y^j \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \Delta y \end{pmatrix}, \qquad \text{with} \qquad \begin{pmatrix} 0 \\ 0 \\ \Delta y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \Delta W_1 x_1 + \Delta W_2 x_2 \end{pmatrix}. \quad (7)$$

Step 4: cast the update as a linear self-attention output. Using the identity $a\,b^\top c = (a \otimes b^\top)\,c$ and grouping terms, Eq. 6 can be written as

$$\begin{pmatrix} 0 \\ 0 \\ \Delta y \end{pmatrix} \;=\; -\frac{\eta}{N}\sum_{i=1}^{N} \underbrace{\begin{pmatrix} 0 \\ 0 \\ W_1x_1^i + W_2x_2^i - y^i \end{pmatrix}}_{\text{value vector } v_i} \otimes \underbrace{\begin{pmatrix} x_1^i & x_2^i & 0 \end{pmatrix}}_{\text{key vector } k_i^\top} \underbrace{\begin{pmatrix} x_1^j \\ x_2^j \\ 0 \end{pmatrix}}_{\text{query vector } q_j}. \quad (8)$$

Each factor in Eq. 8 can be obtained by applying a fixed linear projection to the token $e^i = (x_1^i, x_2^i, y^i)$ or e^j:

$$v_i = \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ W_1 & W_2 & -I_y \end{pmatrix}}_{W_V} e^i, \qquad k_i = \underbrace{\begin{pmatrix} I_x & 0 & 0 \\ 0 & I_x & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{W_K} e^i, \qquad q_j = \underbrace{\begin{pmatrix} I_x & 0 & 0 \\ 0 & I_x & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{W_Q} e^j. \quad (9)$$

Step 5: explicit construction. Combining Eqs. 8 and 9, the gradient step of Eqs. 3–4 is realized as a linear self-attention update with the closed-form projections

$$\begin{pmatrix} x_1^j \\ x_2^j \\ y^j \end{pmatrix} \leftarrow \begin{pmatrix} x_1^j \\ x_2^j \\ y^j \end{pmatrix} - \tfrac{\eta}{N}\textstyle\sum_{i=1}^{N} \left(\underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ W_1 & W_2 & -I_y \end{pmatrix}}_{W_V} \begin{pmatrix} x_1^i \\ x_2^i \\ y^i \end{pmatrix} \right) \otimes \left(\underbrace{\begin{pmatrix} I_x & 0 & 0 \\ 0 & I_x & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{W_K} \begin{pmatrix} x_1^i \\ x_2^i \\ y^i \end{pmatrix} \right)^\top \left(\underbrace{\begin{pmatrix} I_x & 0 & 0 \\ 0 & I_x & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{W_Q} \begin{pmatrix} x_1^j \\ x_2^j \\ y^j \end{pmatrix} \right) \quad (10)$$

The projection P is taken to be the identity on the y-coordinate, so that the value contribution lands in the y-slot of e^j. Comparing the right-hand side of Eq. 10 to the gradient step in Eqs. 3–4, the two sides agree term by term. One linear self-attention update therefore reproduces one gradient-descent step on the unified RAG loss, as claimed. □

B Linear RAG: Derivation of the Unified Retriever Formulation

Main function.

$$y = (W_q, W_z)\begin{bmatrix} x_q \\ \sum_{i=1}^{n}(W_e x_q)^\top (W_e d_i)\, d_i \end{bmatrix} = W_q x_q + W_z \sum_{i=1}^{n}(W_e x_q)^\top (W_e d_i)\, d_i. \quad (11)$$

Define M and rewrite the similarity. We adopt the shared-encoder simplification, where the same linear encoder W_e maps both queries and documents into the retrieval space. The general DPR formulation [24] permits separate W_q and W_d, in which case the analysis below carries through with $M = W_q^\top W_d$ but M need not be symmetric. Defining

$$M \triangleq W_e^\top W_e \quad \Rightarrow \quad (W_e x_q)^\top (W_e d_i) = x_q^\top W_e^\top W_e d_i = x_q^\top M d_i. \quad (12)$$

Hence,

$$y = W_q x_q + W_z \sum_{i=1}^{n}(x_q^\top M d_i)\, d_i. \quad (13)$$

Converting "scalar $\times$ vector" into "matrix $\times$ vector." Note that $x_q^\top M d_i$ is a scalar, and the following identity holds:

$$(x_q^\top M d_i)\, d_i = d_i (d_i^\top M^\top x_q) = (d_i d_i^\top) M^\top x_q. \tag{14}$$

Therefore,

$$\sum_{i=1}^{n} (x_q^\top M d_i)\, d_i = \sum_{i=1}^{n} (d_i d_i^\top) M^\top x_q = \Big(\sum_{i=1}^{n} d_i d_i^\top \Big) M^\top x_q. \tag{15}$$

Define the document second-moment matrix D.

$$D \triangleq \sum_{i=1}^{n} d_i d_i^\top \quad \Rightarrow \quad \sum_{i=1}^{n} (x_q^\top M d_i)\, d_i = D M^\top x_q. \tag{16}$$

Substituting back into y.

$$y = W_q x_q + W_z D M^\top x_q. \tag{17}$$

Then $M = W_e^\top W_e$ is symmetric, i.e., $M^\top = M$. Thus the expression simplifies to

$$y = W_q x_q + W_z D M x_q. \tag{18}$$

The right-hand side is grouped into an equivalent linear mapping:

$$y = (W_q + W_z D M)\, x_q. \tag{19}$$

C Linear-Regime Equivalence: Full Setup and Additional Figures

This appendix supplements Section 4.1 of the main paper with a full description of the synthetic-regression setup and the additional robustness and stacked-layer figures referenced there.

C.1 Setup Details

Tokens. Each token concatenates an input vector, a retrieval-derived feature, and a target,

$$e_i = (x_i,\, z_i,\, y_i), \qquad i = 1, \ldots, N, \tag{20}$$

where N is the number of in-context examples for a single task τ. The auxiliary slot z_i instantiates the unified RAG view of Section 3. Under the linear-projection retriever, z_i is a document-derived feature; under the dot-product retriever, $z_i = x_i$ and the document information is injected through the keys and values rather than through the token (see "dot-product injection" below).

Pre-training objective. We train an LSA layer parameterized by θ to minimize the expected squared prediction error across tasks:

$$\mathcal{L}(\theta) = \frac{1}{B} \sum_{\tau=1}^{B} \big\| \hat{y}_\theta\big(\{e_{\tau,i}\}_{i=1}^{N},\, e_{\tau,N+1}\big) - y_{\tau,N+1} \big\|^2, \tag{21}$$

where the query token at position $N+1$ is $e_{\tau,N+1} = (x_{\text{test}}, z_{\text{test}}, 0)$ and $y_{\tau,N+1}$ is its target. The objective is optimized with minibatch SGD over a fresh batch of tasks at each iteration. We denote the parameters at convergence by θ^*.

Synthetic data. Following [9, 58], we generate each task τ from a teacher with weights $W_\tau \sim \mathcal{N}(0, I)$. Inputs are drawn from $x_{\tau,i} \sim \mathcal{U}(-1, 1)^{n_I}$ and targets are constructed as $y_{\tau,i} = W_\tau^1 x_{\tau,i}^1 + W_\tau^2 x_{\tau,i}^2$. We set $N = n_I = 10$ with output dimension 1, and we sweep the document count $k \in \{2, 5, 10, 25\}$.

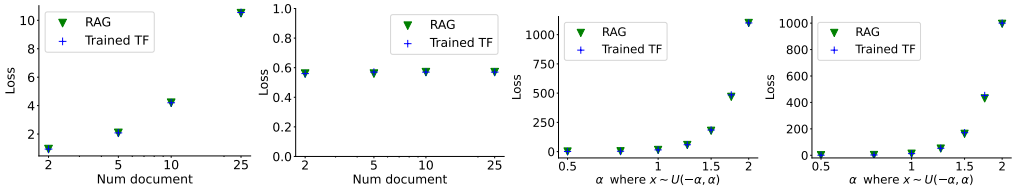


Figure 5: Robustness of the single-layer agreement of Section 4.1. *Left two:* loss as a function of document count for the linear-projection (left) and dot-product (center-left) retrievers. *Right two:* loss under input distribution shift, with test inputs drawn from $\mathcal{U}(-\alpha,\alpha)^{n_I}$ for varying α, for the linear-projection (center-right) and dot-product (right) retrievers. The trained Transformer, the constructed gradient-descent predictor, and their interpolation track each other closely in all settings.

Dot-product injection. Under the linear-projection retriever the document is included in the token, $e_i = (x_i, \mathcal{D}, y_i)$, and the LSA layer can learn to select relevant documents during pre-training. Under the dot-product retriever the document is not concatenated into the token; instead, document information is injected directly into the key and value matrices,

$$K = \begin{bmatrix} K_{\text{ctx}} \\ h_d \end{bmatrix}, \qquad V = \begin{bmatrix} V_{\text{ctx}} \\ h_d \end{bmatrix}, \tag{22}$$

where $K_{\text{ctx}}, V_{\text{ctx}}$ are the contextual key/value rows and $h_d = f(\mathcal{D}) \in \mathbb{R}^{B\times \dim(x)}$ is a fixed projection of the document set $\mathcal{D} = \{d_1, \ldots, d_n\}$ into the input dimension.

Constructed reference predictor. The trained LSA layer (parameters θ^*) is compared against a *constructed* predictor that realizes one gradient-descent step on the unified RAG loss exactly. Following the construction of Eq. 10, we set the value, key, and query projections so that one LSA update reproduces the gradient step. For the linear-projection retriever, W_1 and W_2 inside W_V are initialized to zero, following [58]. For the dot-product retriever, $W_2 = W_z\big(\sum_i d_i d_i^\top\big)M^\top$, with $W_z \in \mathbb{R}^{d_y\times d_d}$ and $M \in \mathbb{R}^{d_d\times d_d}$ sampled independently from $\mathcal{N}(0,\sigma^2)$, and document features $C \sim \mathcal{U}(-\frac{1}{2},\frac{1}{2})^{k\times d_d}$. The inner-loop learning rate η for the constructed predictor is chosen by line search to minimize the constructed model's loss over 10^4 training tasks. We write the resulting prediction $\hat{y}_{\theta,\text{rag}}(x_{\text{test}})$.

Evaluation metrics. On $T_{\text{val}} = 10^4$ held-out validation tasks, following [58], we report the mean of three quantities between the trained and constructed predictors: (i) the prediction difference $\big\|\hat{y}_{\theta^*}(x_{\tau,\text{test}}) - \hat{y}_{\theta,\text{rag}}(x_{\tau,\text{test}})\big\|_2$; (ii) the cosine similarity between the input-sensitivities $\partial\hat{y}_{\theta,\text{rag}}/\partial x_{\text{test}}$ and $\partial\hat{y}_{\theta^*}/\partial x_{\text{test}}$; and (iii) the corresponding ℓ_2 sensitivity difference.

C.2 Robustness to Distribution Shift and Document Count

To probe whether the trained LSA layer captures a generalizable update rule rather than memorizing the training distribution, we vary two factors at test time. First, we sweep the document count $n \in \{2, 5, 10, 25\}$ and recompute the comparison; second, we sample test inputs from $\mathcal{U}(-\alpha,\alpha)^{n_I}$ with $\alpha \in \{0.5, 1, 1.5, 2\}$ while training is fixed at $\alpha = 1$. Figure 5 reports the resulting loss curves. With the linear-projection retriever, the absolute loss rises with document count (the projected document features carry more variance) but the LSA layer follows the gradient predictor in lockstep. With the dot-product retriever, where the document information enters through the second-moment matrix $\sum_i d_i d_i^\top$, the loss is largely insensitive to document count. The dot-product variant is also computationally cheaper, since no per-document projection is required.

C.3 Stacked-Layer Agreement under the Dot-Product Retriever

Figure 6 reports the dot-product variant at depths 2 and 5. The loss differences between the trained Transformer and the constructed predictor remain small across document counts, and the prediction differences converge to similar values. The number of retrieved documents has a depth-dependent effect on the residual: at depth 2, the prediction difference is smaller for Docs $= 2$ than for Docs $= 25$, but this gap narrows at depth 5. The corresponding analysis for the linear-projection retriever is reported in Appendix E.

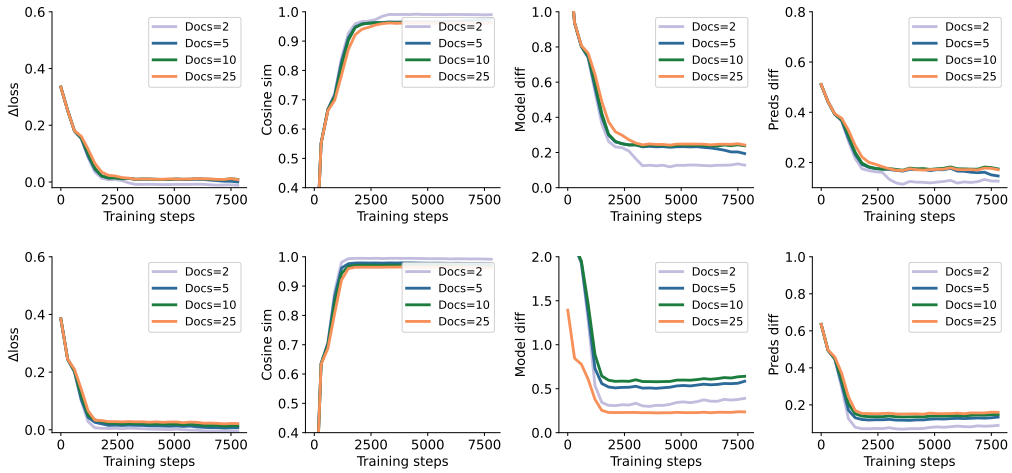


Figure 6: Stacked-layer agreement under the dot-product retriever (Section 4.2). *Top row:* 2-layer model. *Bottom row:* 5-layer model. Columns: (a) loss difference between trained Transformer and constructed gradient-descent predictor, (b) sensitivity cosine, (c) model difference, (d) prediction difference. Agreement remains close at both depths; the small residual gap at Docs $= 25$ in the 2-layer setting shrinks as depth increases to 5.

D Normalization Analysis: Per-Dataset Extended Results

This appendix supplements Section 4.3 of the main paper. Section 4.3 reports the headline result on Bike Sharing and California Housing; here we cover the remaining two datasets, Predict Calorie Expenditure and Wine Quality. Datasets and normalization methods are as defined in Section 4.3 and Appendix F. The training set is used as the retrieval corpus and is normalized with Z-score throughout; only the input-side normalization is varied, between Z-score [5], Min–Max [5], rank-based [7], and Tanh [56].

On Predict Calorie Expenditure, the trained Transformer continues to track the gradient-descent predictor closely, mirroring the alignment seen on Bike Sharing in Section 4.3. Wine Quality is the harder case. Under Min–Max normalization, a few outliers dominate the scaling and compress most of the samples near zero. Two effects follow. The sensitivity cosine drops because the sensitivity vectors diverge from those of the gradient-descent predictor. The prediction difference also fluctuates more strongly, indicating instability in the alignment between RAG and ICL dynamics under heavy-tailed feature distributions. This is consistent with the California Housing pattern in Section 4.3: when retrieval-derived dot products are dominated by skewed features, the linear correspondence becomes less predictive.

E Stacked-Layer Agreement under the Projection-Based Retriever

This appendix complements the dot-product analysis of Appendix C with the projection-based retriever. Under this interface, the retrieved documents are concatenated with the input tokens, $e_i = (x_i, \mathcal{D}, y_i)$, so each per-document feature is processed jointly with the query through every stacked layer. As the document count grows, the variance of the per-token features grows with it, which amplifies the discrepancy between the trained Transformer and the constructed gradient-descent predictor at any fixed depth. Stacking layers reduces this discrepancy: at depth 5, the loss and prediction differences are uniformly smaller across document counts than at depth 2, and the sensitivity cosine is closer to 1.

Whereas the dot-product sweep in Appendix C reports results at 2, 5, 10, and 25 documents, the projection-based sweep is restricted to 2, 5, 10, and 15. The compute cost of stacking concatenated-document tokens grows substantially with retrieval size, and the growing-residual trend is already clear at 15 documents, so the 25-document run is omitted.

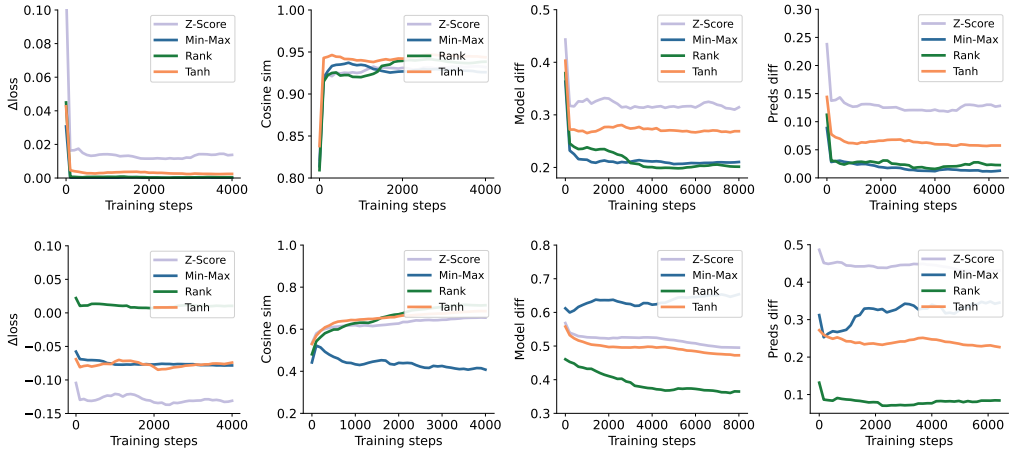


Figure 7: Per-dataset normalization results. *Top row:* Predict Calorie Expenditure. *Bottom row:* Wine Quality. From left to right, the columns report loss difference with the trained Transformer, sensitivity cosine, model difference, and prediction difference. The four normalization schemes (Z-score, Min–Max, rank-based, Tanh) are overlaid within each panel.

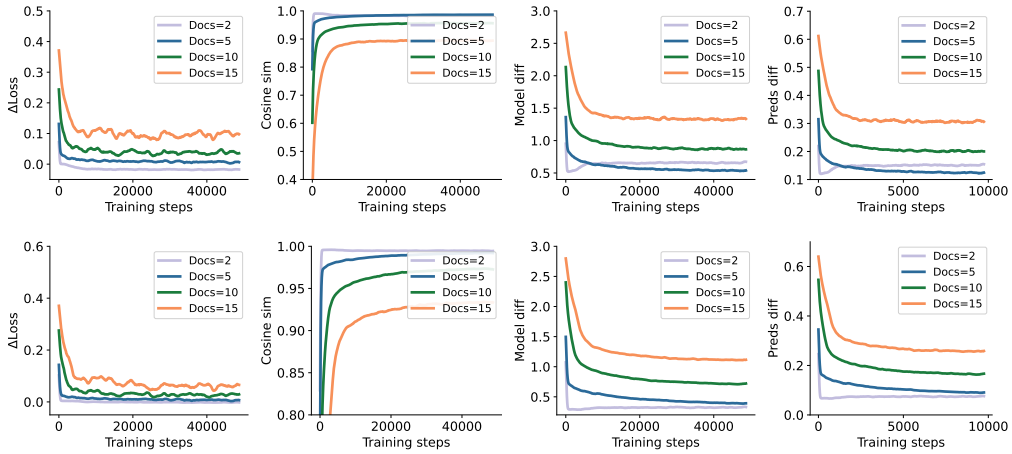


Figure 8: Stacked-layer agreement under the projection-based retriever. *Top row:* 2-layer model. *Bottom row:* 5-layer model. Columns: (a) loss difference between trained Transformer and constructed gradient-descent predictor, (b) sensitivity cosine, (c) model difference, (d) prediction difference.

F Dataset Details

QA benchmarks (main evaluation). The seven question-answering benchmarks used in the main evaluation are all publicly available:

- **Natural Questions (NQ)** [26]: open-domain factoid QA over Wikipedia.
- **TriviaQA** [23]: large-scale trivia question answering with evidence documents.
- **PopQA** [43]: popularity-stratified entity-centric QA (held out from training).
- **HotpotQA** [61]: multi-hop QA with comparison and bridge questions.
- **2WikiMultiHopQA** [14]: multi-hop questions grounded in Wikipedia article pairs.
- **MuSiQue** [55]: compositional multi-hop QA constructed by composing single-hop pairs.
- **Bamboogle** [47]: small multi-hop benchmark on long-tail entities (held out from training).

For each benchmark we use the standard FlashRAG [22] corpus and retrieval splits. NQ is used to train the source retrieval adapter W_0^{ret}. NQ, TriviaQA, HotpotQA, 2WikiMultiHopQA, and MuSiQue

are used to meta-train the predictor g_ϕ. PopQA and Bamboogle are held out from both adapter training and predictor meta-training, and are used only for evaluation.

Synthetic and tabular regression datasets (linear-attention sanity check and normalization analysis).

- **California Housing**: Given eight features, `['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup', 'Latitude', 'Longitude']`, the task is to predict `MedHouseVal`. The dataset is split into 16,640 training samples and 2,000 test samples.
- **Bike Sharing**: Using the features `['season', 'yr', 'mnth', 'hr', 'holiday', 'weekday', 'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed', 'casual', 'registered']`, the task is to predict `count`. The dataset contains 15,641 training samples and 1,738 test samples.
- **Wine Quality**: Given eleven physicochemical features, `[fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol]`, the task is to predict the wine `quality` (a sensory score ranging from 0 to 10). The dataset is split into 4,408 training samples and 490 test samples.
- **Predict Calorie Expenditure**: Using the features `[Gender, Age, Height, Weight, Duration, Heart_Rate, Body_Temp]`, the task is to predict the number of `Calories` expended. The dataset is split into 13,500 training samples and 1,540 test samples.

G Implementation Details

Source retrieval adapter W_0^{ret}. We implement the generator-side retrieval interface with LoRA modules on the $\{q, k, v\}$ projections of every Transformer block. The LoRA rank is 16, with α=32 and dropout 0. The LLM backbone is kept frozen and loaded in 4-bit NF4 precision. We train W_0^{ret} with AdamW using learning rate 10^{-4}, weight decay 0.01, gradient clipping 1.0, and gradient accumulation 4 for 3,000 steps on RAG-formatted examples from the NQ training split. Each example is paired with the top-K_{ret}=5 retrieved documents from the fixed external retriever. For each retriever setting, we use the corresponding fixed retrieval cache and keep this cache unchanged across methods. This adapter serves only as a generator-side evidence-use interface: it does not select documents, but modulates how the frozen generator uses the retrieved evidence.

Predictor g_ϕ. The predictor maps a support context $C = \{(x_i, D_i, y_i)\}_{i=1}^N$ with N=3 demonstrations to a context-conditioned update for the generator-side retrieval interface. Each demonstration is formatted by concatenating the question, retrieved documents, and gold answer, and is encoded by the frozen LLM equipped with W_0^{ret}. We take the EOS hidden state h_i of each demonstration and aggregate the support context by mean pooling,

$$\bar{h}(C) = \frac{1}{N} \sum_{i=1}^{N} h_i.$$

The pooled representation is passed to a two-layer MLP encoder with hidden dimension 256 and output dimension 64, followed by per-layer and per-projection update heads for the $\{q, k, v\}$ LoRA modules. The update heads output low-rank perturbations with the same LoRA rank as W_0^{ret}, so that $g_\phi(C)$ has the same parameter shape as the base retrieval adapter. We train g_ϕ with AdamW using learning rate 5×10^{-4}, weight decay 0.01, gradient clipping 1.0, and gradient accumulation 4 for 3,000 steps. The predictor is meta-trained on support contexts sampled from NQ, TriviaQA, HotpotQA, 2WikiMultiHopQA, and MuSiQue. PopQA and Bamboogle are excluded from both adapter training and predictor meta-training, and are used only for held-out evaluation.

Matching objective. The predictor is trained to match the autograd-defined inner-GD target for each layer and projection type. The matching loss uses a cosine term for update direction and a log-magnitude term for update scale, as defined in Section 5.2. We set the magnitude weight to λ=0.1 throughout. No downstream answer loss is applied when training g_ϕ for the main RAG-GD results.

Inner GD target. For each support context, the supervision target is computed by running K steps of SGD on the $N{=}3$ demonstrations, starting from W_0^{ret}. The inner-loop learning rate is $\eta{=}10^{-2}$, and we evaluate $K \in \{1, 5, 10\}$. Gradients are taken only with respect to the LoRA parameters of the generator-side retrieval interface; the external retriever and the LLM backbone remain fixed.

Compute. All experiments are run on NVIDIA A100 80GB GPUs. A single training run for W_0^{ret} takes approximately 50–60 minutes. Training g_ϕ takes approximately 1.5 hours for $K{=}1$, 4 hours for $K{=}5$, and 7 hours for $K{=}10$. The full result table requires approximately 50 A100-hours.

H Algorithm and Inference Procedure for RAG-GD

Algorithm 1 summarizes the deployment-time procedure of RAG-GD: build a RAG-formatted support context, predict the retrieval-interface update with a single forward pass through g_ϕ, and generate with the perturbed interface. No backward pass through the LLM is required at deployment.

Algorithm 1: RAG-GD: Forward-Only Retrieval-Interface Adaptation

Input: External retriever R, frozen LLM f, source adapter W_0^{ret}, predictor g_ϕ, support size N, query q
Output: Generated answer $\hat{y}_q$

// Phase 1: Build RAG-formatted support context
$C \leftarrow \emptyset$
for $i = 1, \ldots, N$ **do**
  Sample support pair (x_i, y_i) from the task support pool
  $\mathcal{D}_i \leftarrow R(x_i)$ `// retrieve top-`k` documents`
  $C \leftarrow C \cup \{(x_i, \mathcal{D}_i, y_i)\}$
// Phase 2: Predict retrieval-interface update
for $(x_i, \mathcal{D}_i, y_i) \in C$ **do**
  $h_i \leftarrow f_{W_0^{\text{ret}}}(x_i, \mathcal{D}_i, y_i)_{\text{EOS}}$
$\bar{h}(C) \leftarrow \frac{1}{N} \sum_{i=1}^{N} h_i$
$\widetilde{\Delta W}(C) \leftarrow g_\phi(\bar{h}(C))$
// Phase 3: Generate with adapted interface
$\mathcal{D}_q \leftarrow R(q)$
$\hat{y}_q \leftarrow f_{W_0^{\text{ret}} + \widetilde{\Delta W}(C)}(q, \mathcal{D}_q)$
return $\hat{y}_q$

I Full Per-Method Comparison on QA Benchmarks

We report the per-benchmark results in two tables and one figure. Table 3 reports the methods that we ran on *both* Qwen 2.5-7B-Instruct and Llama 3.1-8B-Instruct: Query Only, Vanilla RAG, Base adapter, and RAG-GD at $K \in \{1, 5, 10\}$. Table 4 reports the additional context-conditioned baselines on Qwen: Vanilla RAG + few-shot, Base adapter + few-shot, Prompt Tuning, HyperTuning, and TT-SGD at $K{=}5$. Figure 9 reports four of these baselines on Llama with E5 retrieval, alongside RAG-GD ($K{=}5$). Together they complement the slim main-text Table 1 and the per-family aggregates in Figures 3 and 4. **HyperTuning** [46] uses the same predictor architecture as RAG-GD but supervises against a downstream task loss instead of the SGD-update target, so the contrast between the HyperTuning rows in Table 4 and the RAG-GD ($K{=}5$) rows for Qwen in Table 3 isolates the choice of supervision signal. **TT-SGD** performs $K{=}5$ inner gradient-descent steps at test time and serves as a reference for what test-time gradient adaptation would achieve at the same backbone and retriever. **+ few-shot** variants concatenate the support demonstrations into the prompt under the corresponding base configuration.

We evaluate four additional baselines on Llama-3.1-8B-Instruct with E5 retrieval, following the same protocol as the Qwen experiments (Figure 9). RAG-GD outperforms HyperTuning on all seven benchmarks, achieving an average EM of 40.54 compared with 38.18 for HyperTuning. It also

Table 3: Methods run on both backbones. Per-benchmark exact match (EM) and F1 for Query Only, Vanilla RAG, Base adapter, and RAG-GD at $K \in \{1, 5, 10\}$.

Method	Retriever	Single-Hop QA						Multi-Hop QA								Avg.	
		NQ		TriviaQA		PopQA		HotpotQA		2Wiki		MuSiQue		Bamboogle			
		EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Qwen 2.5-7B																	
Query Only	–	15.95	24.28	43.33	49.51	16.02	19.76	18.40	25.39	23.91	28.12	3.80	10.57	11.20	18.02	18.94	25.09
Vanilla RAG	BM25	27.61	36.66	58.24	65.77	28.84	33.18	31.28	41.25	27.87	33.24	5.87	13.05	10.40	21.16	27.16	34.90
	E5	39.16	50.03	62.99	70.80	44.03	50.21	32.45	42.21	25.48	31.43	5.79	12.77	18.40	26.74	32.61	40.60
Base adapter	BM25	32.57	41.45	60.11	67.93	31.55	35.63	32.31	43.45	28.22	34.13	6.41	15.46	16.80	26.01	29.71	37.72
	E5	41.77	51.22	63.31	71.62	47.05	51.82	33.78	44.64	27.89	33.97	6.95	15.76	18.40	28.78	34.16	42.54
RAG-GD (K=1)	BM25	33.88	43.20	63.02	70.66	33.25	37.73	35.25	46.79	28.84	34.59	9.14	19.62	22.40	33.06	32.25	40.81
	E5	42.49	52.32	65.61	73.51	48.41	52.93	35.34	46.94	29.66	35.50	8.93	19.17	24.00	34.18	36.35	44.94
RAG-GD (K=5)	BM25	34.46	43.54	63.27	70.69	33.22	37.69	35.54	47.14	28.86	34.48	9.26	19.73	22.40	32.85	32.43	40.87
	E5	42.91	52.71	65.98	73.60	48.12	52.61	35.54	47.00	29.67	35.47	9.14	19.26	25.60	35.12	36.71	45.11
RAG-GD (K=10)	BM25	34.57	43.72	63.26	70.57	33.11	37.57	35.45	47.07	28.64	34.32	9.35	19.58	22.40	31.92	32.40	40.68
	E5	42.46	52.14	65.93	73.49	47.96	52.43	35.69	47.05	29.43	35.14	8.90	19.09	26.40	34.67	36.68	44.86
Llama 3.1-8B																	
Query Only	–	22.46	32.51	52.67	59.79	20.63	25.09	18.31	25.71	26.39	31.04	3.81	9.51	6.40	12.88	21.52	28.08
Vanilla RAG	BM25	31.41	40.95	60.43	68.35	31.08	35.43	31.92	42.46	26.07	31.82	5.75	12.44	14.40	22.93	28.72	36.34
	E5	40.72	52.23	64.42	72.58	45.85	51.48	32.78	43.19	23.44	29.46	6.04	12.25	24.80	32.12	34.01	41.90
Base adapter	BM25	38.47	49.42	62.66	72.46	37.80	42.29	37.35	50.41	33.66	39.55	11.46	21.98	29.60	41.29	35.86	45.34
	E5	43.46	54.60	63.90	74.05	51.69	55.74	37.31	49.90	33.45	39.45	11.83	22.07	30.40	40.48	38.86	48.04
RAG-GD (K=1)	BM25	39.58	49.60	66.13	74.13	37.89	42.38	38.53	50.67	33.72	39.61	12.25	22.48	30.40	41.69	36.93	45.79
	E5	45.10	55.19	67.63	75.98	52.38	56.61	38.27	50.09	33.44	39.47	12.32	22.52	30.40	40.72	39.93	48.65
RAG-GD (K=5)	BM25	40.22	50.01	66.13	74.28	37.81	42.19	38.99	51.14	34.15	40.04	12.54	22.61	28.00	39.61	36.83	45.70
	E5	45.68	55.84	67.66	76.07	52.31	56.48	39.01	50.94	33.94	39.83	13.20	23.47	32.00	42.08	40.54	49.24
RAG-GD (K=10)	BM25	40.28	50.29	65.93	74.07	37.73	42.10	39.10	51.20	34.26	40.08	12.62	22.86	28.80	39.92	36.96	45.79
	E5	45.48	55.51	67.71	76.03	52.38	56.49	39.10	50.96	34.03	39.79	13.41	23.41	32.00	42.83	40.59	49.29

Table 4: Additional context-conditioned baselines on Qwen 2.5-7B. For comparison anchors (Qwen Base adapter and RAG-GD at the same retriever and benchmark), see Table 3.

Method	Retriever	Single-Hop QA						Multi-Hop QA								Avg.	
		NQ		TriviaQA		PopQA		HotpotQA		2Wiki		MuSiQue		Bamboogle			
		EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Vanilla RAG + few-shot	BM25	27.72	36.80	58.41	65.98	28.80	33.18	31.73	41.43	26.27	32.05	5.88	13.58	11.20	20.27	27.14	34.76
	E5	38.78	49.75	63.14	70.94	43.82	50.00	32.28	42.05	24.20	30.80	5.99	13.21	16.80	25.42	32.14	40.31
Base adapter + few-shot	BM25	34.07	43.18	62.26	69.97	32.94	37.65	34.65	46.18	28.91	34.56	9.06	19.66	19.20	30.19	31.58	40.20
	E5	41.52	51.25	65.03	73.11	47.85	52.38	34.51	46.08	29.81	35.70	8.44	18.67	26.40	36.08	36.22	44.75
Prompt Tuning	BM25	28.94	40.81	58.47	69.11	32.42	37.14	32.43	44.98	32.18	38.54	7.36	17.79	22.40	33.15	30.60	40.22
	E5	35.92	48.16	61.09	71.73	47.10	51.96	32.72	45.14	31.60	38.04	7.28	17.22	26.40	32.98	34.59	43.60
HyperTuning	BM25	33.62	43.21	61.38	69.90	33.71	37.97	34.55	46.34	31.87	37.81	7.52	17.60	19.20	31.69	31.69	40.65
	E5	40.72	50.51	64.19	72.57	48.88	53.25	34.76	46.16	31.47	37.44	6.95	16.74	24.00	32.83	35.85	44.21
TT-SGD (K=5)	BM25	34.37	43.32	63.32	70.75	32.94	37.63	35.16	47.13	28.76	34.53	8.77	19.51	23.20	33.26	32.36	40.88
	E5	42.52	52.14	65.50	73.45	47.95	52.51	35.62	47.43	29.67	35.59	8.93	19.13	25.60	34.88	36.54	45.02

exceeds Base adapter + few-shot, the strongest of these four baselines, by 0.62 EM points on average, consistent with the 0.49-point improvement on Qwen.

J Additional Evaluation on Qwen3-14B

We additionally evaluate Qwen3-14B with E5 retrieval under the same protocol as the Qwen 2.5-7B experiments on five benchmarks: NQ, TriviaQA, PopQA, HotpotQA, and Bamboogle. Table 5 reports EM and F1, with averages computed over these five benchmarks.

RAG-GD (K=5) outperforms Base adapter + few-shot on all five benchmarks in both EM and F1, improving average performance by 1.45 EM and 1.13 F1 points. On the same five-benchmark subset, the corresponding EM margin on Qwen 2.5-7B is 0.57 points.

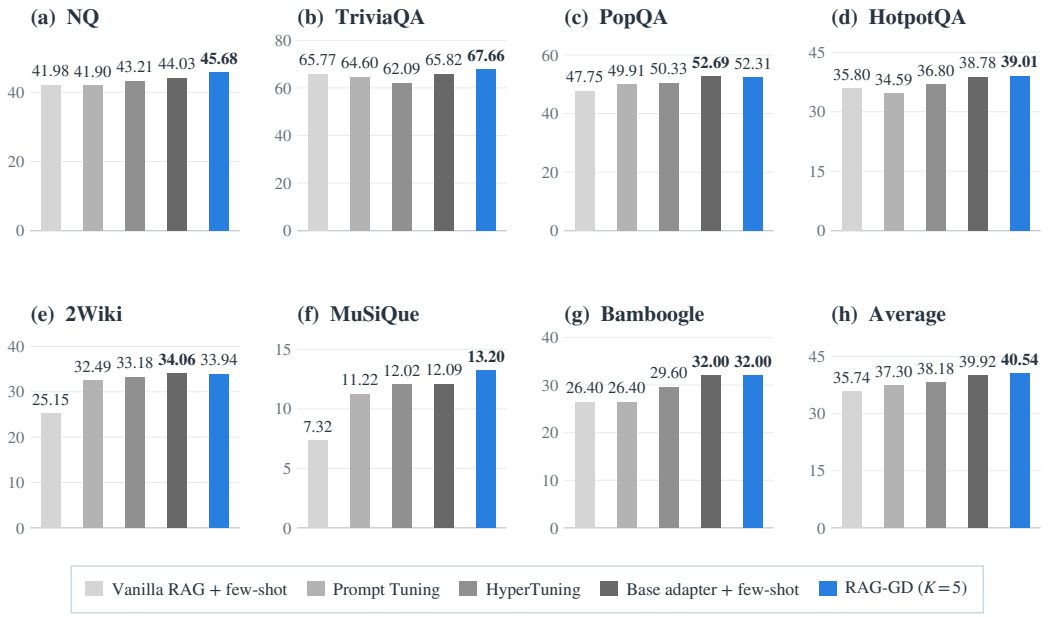


Figure 9: Context-conditioned baseline comparison on Llama 3.1-8B-Instruct with E5 retrieval. Panels (a)–(g) show exact match (EM) on the seven benchmarks; (h) shows their unweighted average. Values above bars report EM, with the best score in each panel shown in bold, including ties. Each panel uses its own vertical scale starting at zero.

Table 5: Additional RAG-GD evaluation on Qwen3-14B with E5 retrieval. Avg. is the unweighted mean over the five evaluated benchmarks. Bold values mark the best result per column.

Method	Single-Hop QA						Multi-Hop QA				Avg.	
	NQ		TriviaQA		PopQA		HotpotQA		Bamboogle			
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Vanilla RAG	39.03	51.81	63.59	71.48	39.79	51.98	37.81	49.14	23.20	31.05	40.68	51.09
Vanilla RAG + few-shot	41.31	52.60	65.41	72.18	42.51	53.41	38.10	49.48	22.40	32.21	41.95	51.98
Base adapter	44.30	54.84	64.92	73.53	49.38	57.16	41.09	53.50	28.80	44.17	45.70	56.64
Base adapter + few-shot	44.62	55.35	65.11	73.51	50.39	57.77	40.58	52.80	29.60	44.61	46.06	56.81
RAG-GD ($K=5$)	**45.24**	**55.62**	**66.42**	**74.94**	**51.80**	**57.83**	**42.11**	**54.66**	**32.00**	**46.65**	**47.51**	**57.94**

On Qwen3-14B, few-shot prompting raises Vanilla RAG's average EM from 40.68 to 41.95, but raises Base adapter's average EM only from 45.70 to 46.06. On Qwen 2.5-7B, the corresponding changes on the same subset are -0.44 and $+2.20$ EM points, respectively. These results show that the benefit of RAG-GD persists on Qwen3-14B beyond few-shot prompting.

K Reference-Based Semantic Evaluation

To complement EM and F1, we evaluate the Qwen 2.5-7B-Instruct predictions with E5 retrieval using Qwen2.5-32B-Instruct as a reference-based judge with deterministic decoding. Given the question, gold answer, and prediction, the judge assesses whether the prediction is semantically equivalent to the reference. LLM-judge accuracy is the fraction of predictions judged equivalent, macro-averaged across the same seven benchmarks.

As shown in Table 6, LLM-judge scores are 8.64–9.72 percentage points higher than EM for all evaluated methods. The method ranking is unchanged across the three metrics, with RAG-GD achieving the highest score under each metric.

Table 6: EM, F1, and reference-based LLM-judge accuracy for Qwen 2.5-7B-Instruct with E5 retrieval. All scores are percentages macro-averaged over the same seven benchmarks. Bold values mark the best result per column.

Method	EM	F1	LLM-judge accuracy
Vanilla RAG	32.61	40.60	41.25
Base adapter	34.16	42.54	43.88
Base adapter + few-shot	36.22	44.75	45.38
HyperTuning	35.85	44.21	44.91
RAG-GD (K=5)	**36.71**	**45.11**	**46.04**

Examples of predictions accepted by the judge but rejected by EM include "International Whaling Commission" for "The International Whaling Commission or IWC", "documentaries" for "documentary", and "health points" for "hit points or health points". These examples illustrate mismatches involving aliases, inflections, and reference formatting.

L Additional Related Work

Retrieval and Knowledge Integration. Retrieval-augmented generation (RAG) incorporates external knowledge into language models to support knowledge-intensive tasks [28]. Its applications have expanded to specialized domains and diverse prediction tasks. RT [40] combines retrieved demonstrations with chain-of-thought reasoning for few-shot medical named entity recognition. BiomedRAG [33] integrates biomedical evidence into language models for healthcare applications, while CancerLLM [37] uses retrieval-based example selection for cancer phenotyping and diagnosis generation. Prior studies have also examined the robustness and self-awareness of retrieval-augmented models under noisy and heterogeneous evidence [39]. Beyond external knowledge sources, K2K [30] enables internal key-based knowledge access through activation-guided probing and cross-attentive reranking, providing an alternative mechanism for incorporating task-relevant knowledge.

Structured Knowledge Retrieval and Reasoning. Beyond unstructured text, structured knowledge provides relational evidence that can support knowledge-intensive retrieval and downstream reasoning. Kiest [32] retrieves relevant entities and attributes from knowledge graphs to facilitate entity state tracking, while CTLP [34] captures multi-hop relational paths through condensed transition graphs for zero-shot link prediction. RiTeK [17] introduces a benchmark for complex reasoning over textual knowledge graphs and evaluates retrieval methods for identifying relational evidence. GLEN-Bench [19] further investigates graph-language reasoning over structured knowledge. Related efforts explore the use of graph structures to support downstream reasoning processes. EvolveRouter [20] leverages graph-based relationships for adaptive agent routing and prompt optimization, while CAGE-CAL [16] models inter-agent dependencies through counterfactual graphs to improve reasoning reliability. Together, these studies highlight the complementary roles of structured evidence retrieval and relational reasoning in knowledge-intensive tasks.

Interactive Retrieval and Learning. Recent research has moved beyond fixed retrieval pipelines toward systems that dynamically adjust evidence acquisition and reasoning behavior. DeepRAG [12] introduces sequential retrieval decisions that determine when additional information should be acquired. RIVO [31] improves long-horizon research through lightweight, state-conditioned interventions that redirect agent trajectories without retraining the underlying model. RICE-PO [36] investigates policy optimization for interactive retrieval agents by deriving localized credit signals from executable retrieval actions and counterfactual interactions. These methods address different aspects of adaptive retrieval, including search decisions, trajectory control, and learning from retrieval feedback.

While prior studies investigate how knowledge is retrieved, represented, or incorporated into language models, a related question is whether retrieved context can also serve as a signal for adapting the generator itself. In this work, we study this question from an in-context optimization perspective.

Rather than modifying the retrieval process or updating the entire language model, we examine how contextual examples can induce changes in the interaction between queries and retrieved evidence. We establish a gradient-descent correspondence in a linearized RAG setting and use this perspective to develop a forward-only, context-conditioned adaptation mechanism for frozen RAG LLMs.

M Use of Large Language Models

Large language models (LLMs) were used to assist with language polishing and minor grammatical editing of this manuscript. Qwen2.5-32B-Instruct was also used for the reference-based semantic evaluation described in Appendix K.