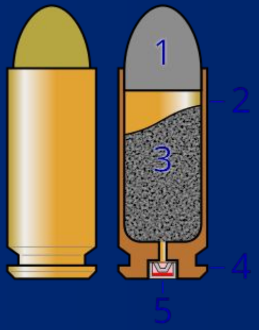
Can KV-cache serve as compact memory module when reasoning over long or unbounded contexts?

Ziyang Huang, Yingfei Xu

JOHNS HOPKINS UNIVERSITY



Cartridges: Lightweight and general-purpose long context representations via self-study

Sabri Eyuboglu, Ryan Saul Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Ruoyu Liu, Atri Rudra, James Y. Zou, Azalia Mirhoseini, Christopher Re

Long-context ICL is accurate but expensive: KV $\propto$ input length.

But can we use a swappable KV cache module and reuse it?

Cartridges

Turning long corpora into tiny, reusable KV prefixes for fast, high-quality answers.

Performance

ICL-like quality with dramatically less memory and higher throughput when many queries target the same corpus.

Motivation

Problem Setting

Document Corpus

Queries

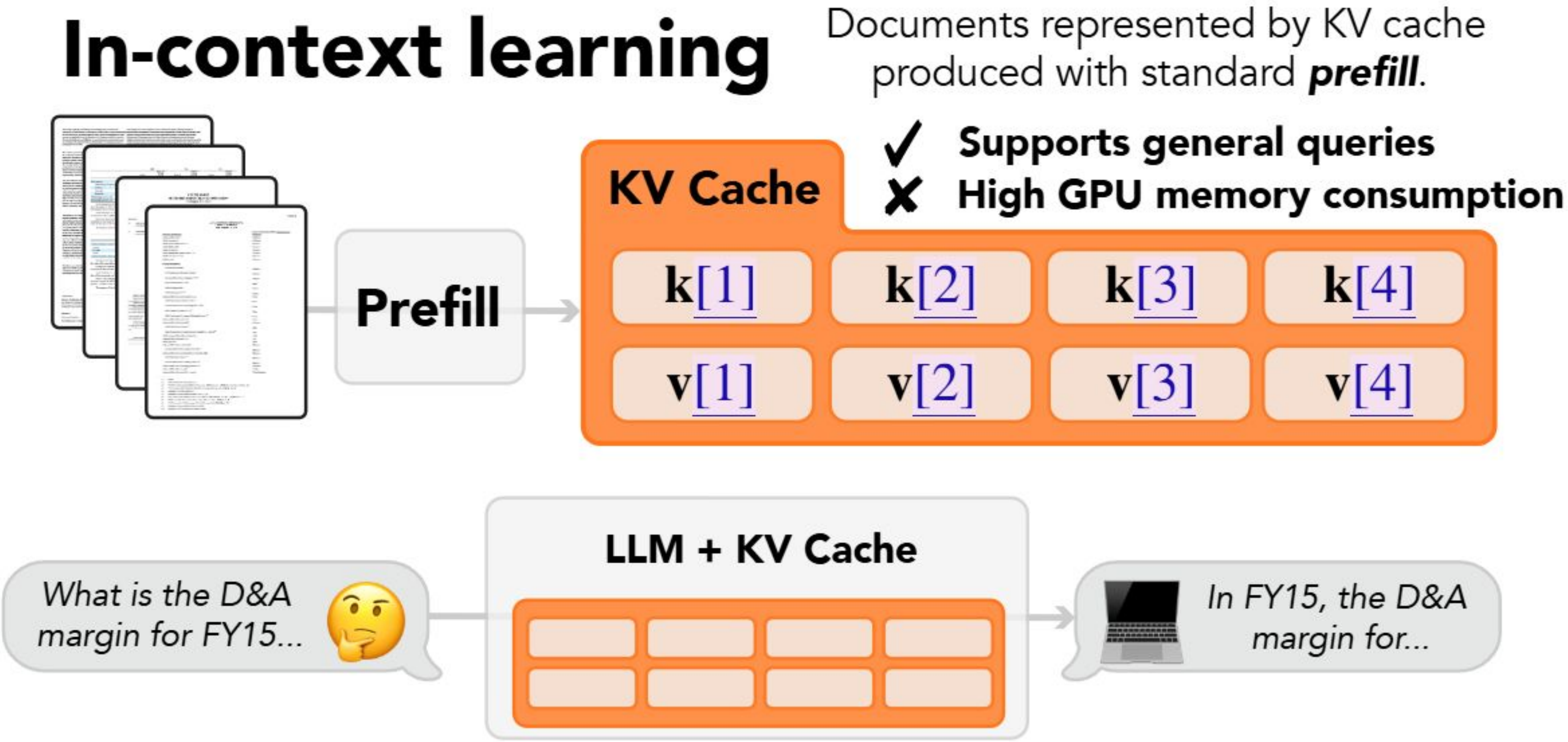
Please, summarize the documents...

Write a rock song about the docs...

What is the D&A margin for FY15...

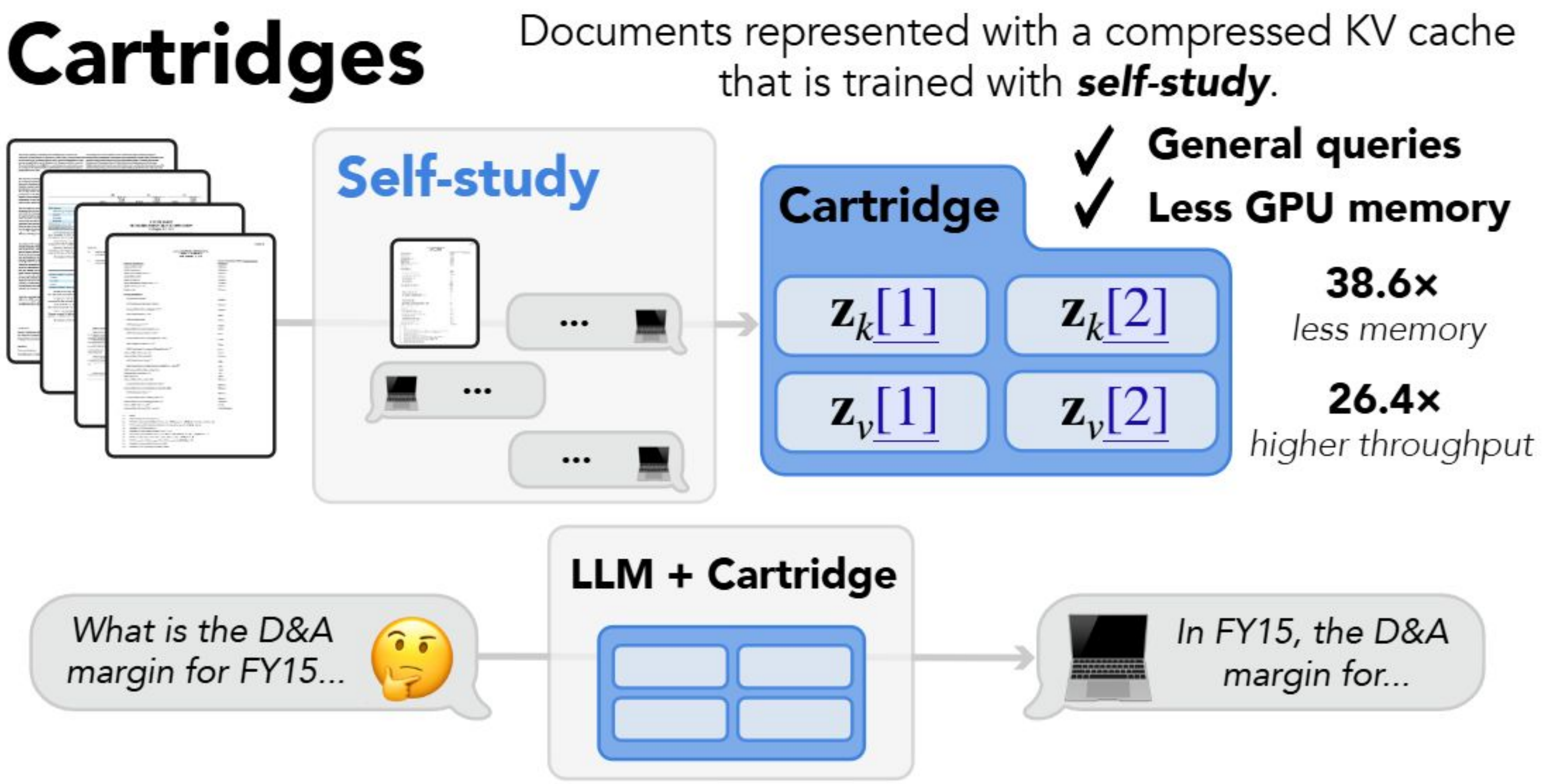
Users send **many** *messages grounded in a* **single** *large corpus of text.*

Motivation



In-context learning
Documents represented by KV cache produced with standard *prefill*.
✓ Supports general queries
✗ High GPU memory consumption
Prefill
KV Cache
k[1] k[2] k[3] k[4]
v[1] v[2] v[3] v[4]
What is the D&A margin for FY15...
LLM + KV Cache
In FY15, the D&A margin for...

Motivation


Cartridges
Documents represented with a compressed KV cache that is trained with ***self-study***.
Self-study
...
...
...
Cartridge
$\mathbf{z}_k[1]$
$\mathbf{z}_k[2]$
$\mathbf{z}_v[1]$
$\mathbf{z}_v[2]$
✓ General queries
✓ Less GPU memory
38.6× less memory
26.4× higher throughput
LLM + Cartridge
What is the D&A margin for FY15...
In FY15, the D&A margin for...

What is a Cartridge

- A learned KV prefix of length p (e.g., 128–8192 tokens) attached at inference time.
- No base-model weight updates; just load/unload per corpus.
- A composable module; concatenate multiple cartridges (e.g., Policy ⊕ API Docs ⊕ Codebase) for cross-source reasoning.

ICL KV Cache

$$\underbrace{(\mathbf{k}[1], \mathbf{v}[1]), \ldots, (\mathbf{k}[n_{\mathcal{C}}], \mathbf{v}[n_{\mathcal{C}}])}_{\text{KV pairs for } \mathcal{C}}, \underbrace{(\mathbf{k}[n_{\mathcal{C}}+1], \mathbf{v}[n_{\mathcal{C}}+1]) \ldots}_{\text{KV pairs for } q}$$

CARTRIDGE KV Cache

$$\underbrace{(\mathbf{z}_{\mathrm{k}}[1], \mathbf{z}_{\mathrm{v}}[1]), \ldots, (\mathbf{z}_{\mathrm{k}}[p], \mathbf{z}_{\mathrm{v}}[p])}_{\text{Trainable KV pairs in } Z}, \underbrace{(\mathbf{k}[1], \mathbf{v}[1]) \ldots}_{\text{KV pairs for } q}$$

Analogy: Like prefix-tuning, but trained to imitate ICL on that corpus.

Self-study Training

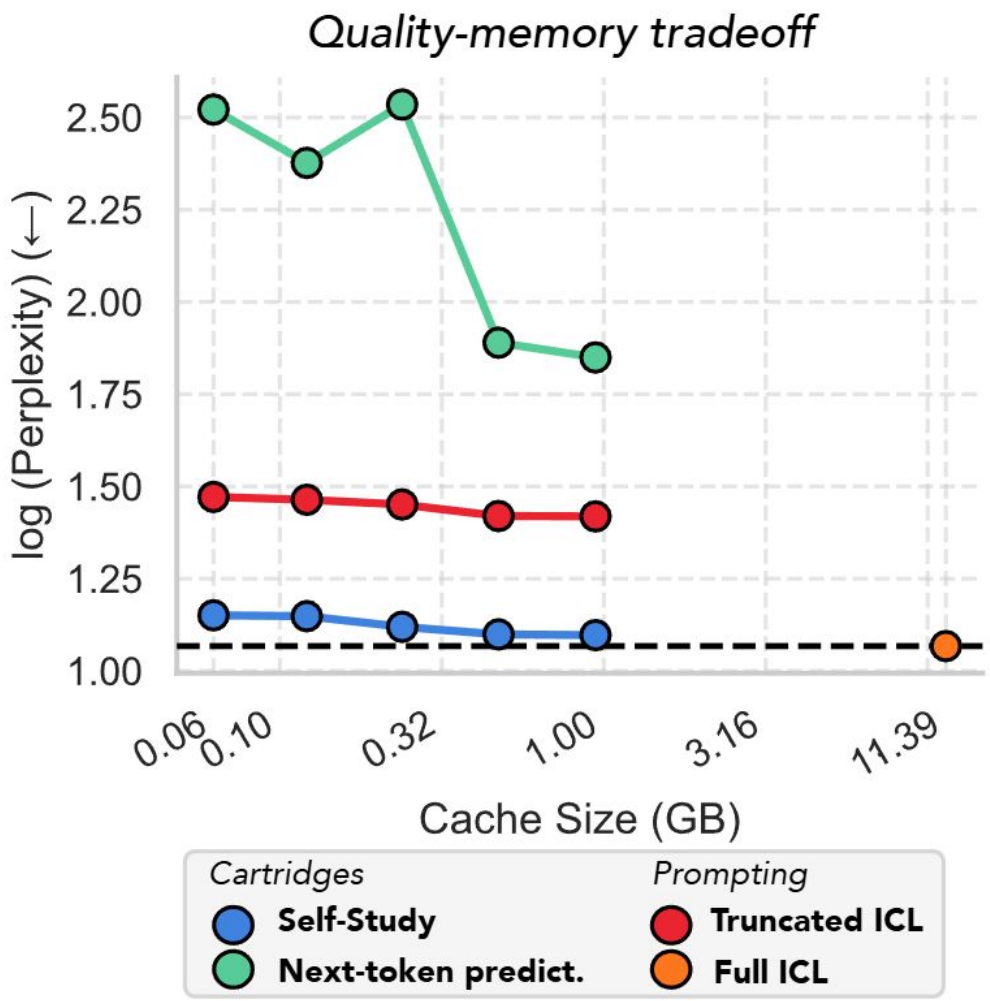
- Synthesize dialogs about the corpus: For each chunk $\tilde{c}$, auto-generate Q↔A / instruction traces.

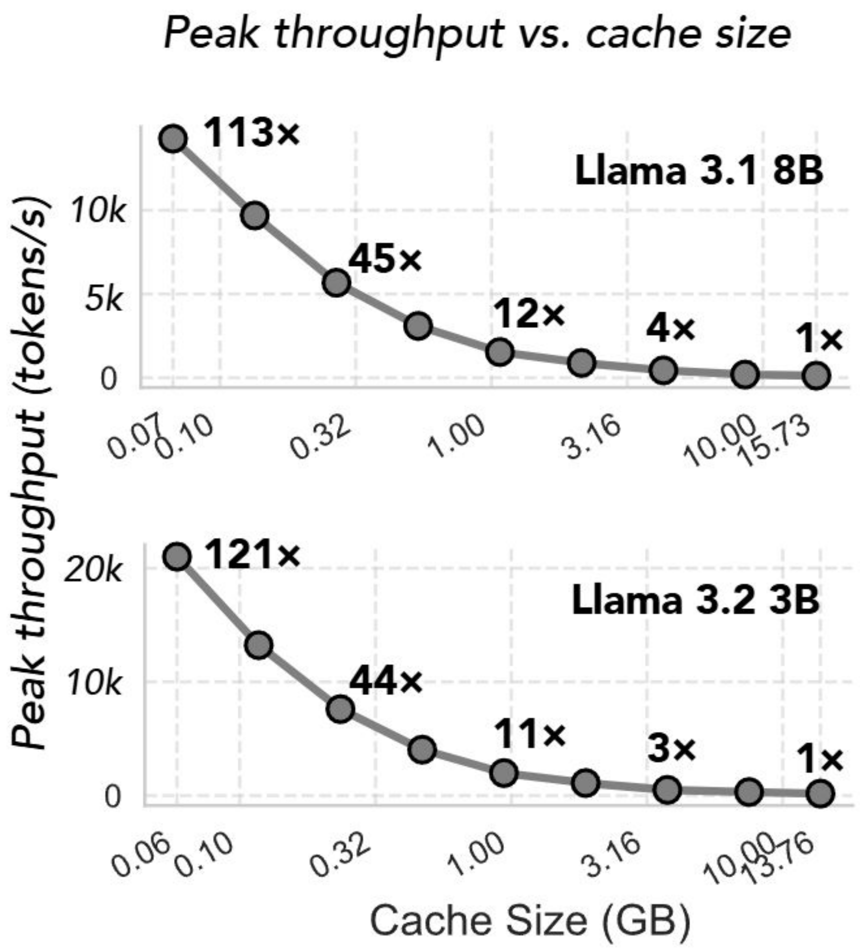
- **T**eacher = base LLM with $\tilde{c}$ in context; **S**tudent = same LLM + trainable cartridge Z without $\tilde{c}$.

- Objective: minimize step-wise KL(p_t || p_s) over next-token distributions.

Intuition: Student learns to act as if the corpus were present.

Performance: throughput and cache size

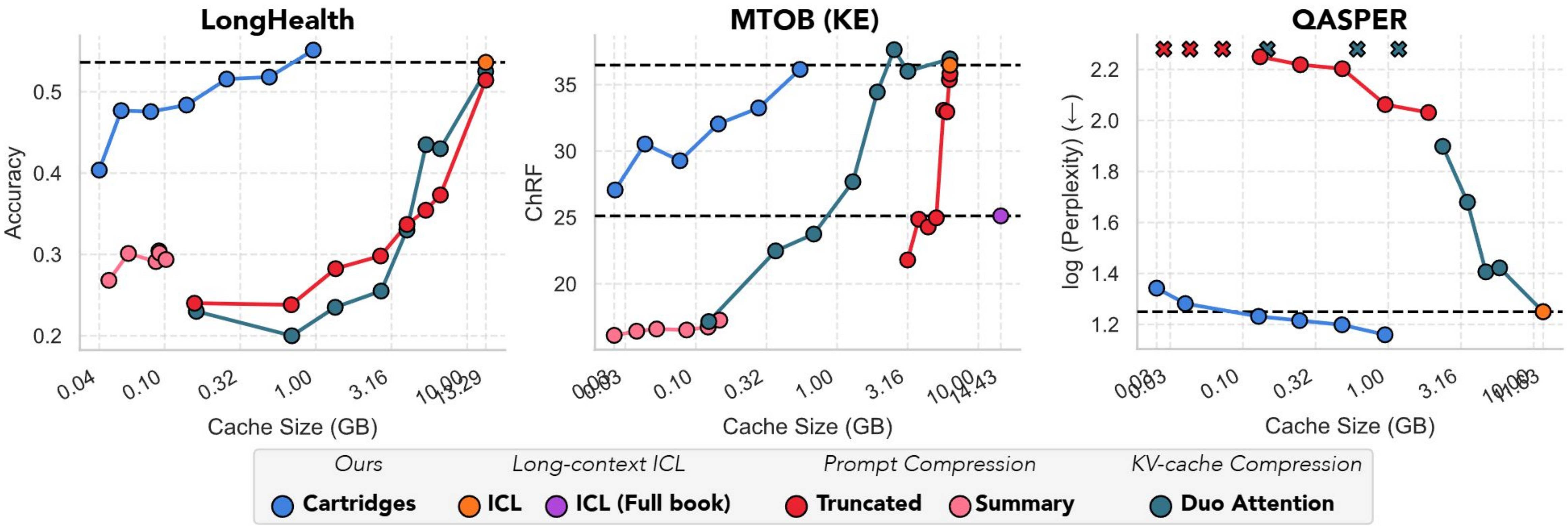




↓38.6x memory consumption and ↑26.4x peak throughput across different tasks

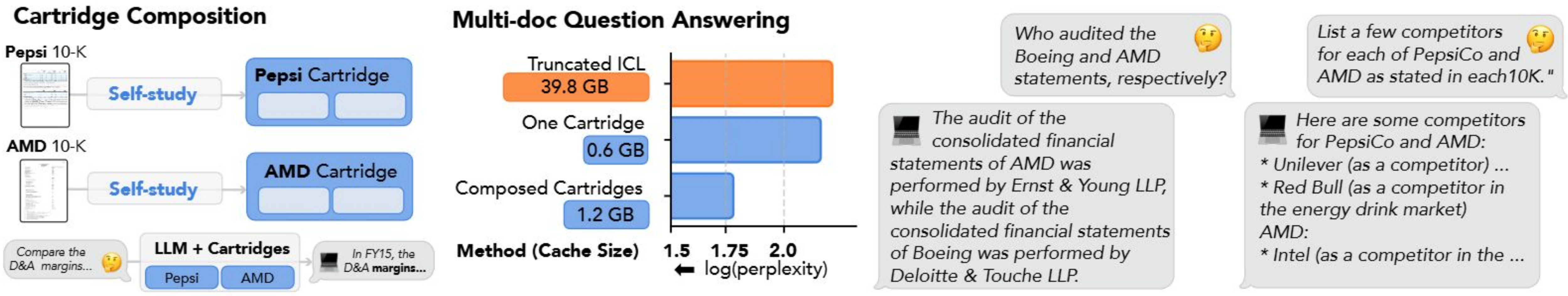
Performance: Cartridges vs baselines

Cartridges matches ICL quality with lower memory costs.



Surprising notes

- Context stretch: Effective context extended (e.g., 128k → ~484k on MTOB)
- Composition: Multiple cartridges combine without re-training.



Cartridges vs RAG? LoRA?

RAG Good for flexible and live updates, but retrieval and long prompts are still costly.

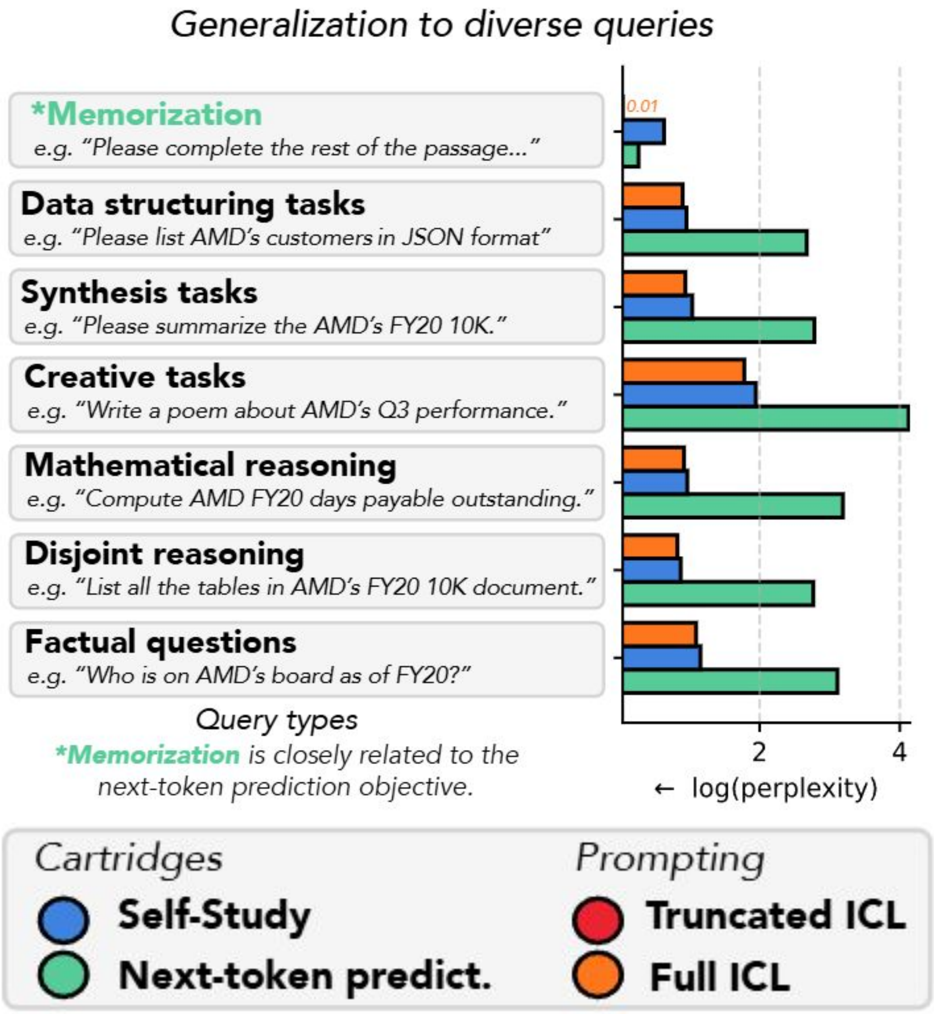
-> Cartridges are optimized on stable, repeatedly-queried corpora.

LoRA Much more expressive for training, but also to serve on infrastructures.

-> Cartridges are prefix-only; also outperperforms LoRA.

Other discussions

- A Cartridge can be served efficiently with minimal changes to existing LLM inference servers (e.g. SGLang)
- Limitations:
  - Upfront compute: Strong performance but training must be amortized
  - Coverage: training rely on synthetic dialogs; bad dialogs cause blind spots
  - Timeliness: Corpus changes require re-training or incremental fine-tuning



Takeaways

Cartridges = compact, reusable memory of a corpus with ICL-like behavior.

Use Self-study (synthetic dialogs + context distillation) for training.

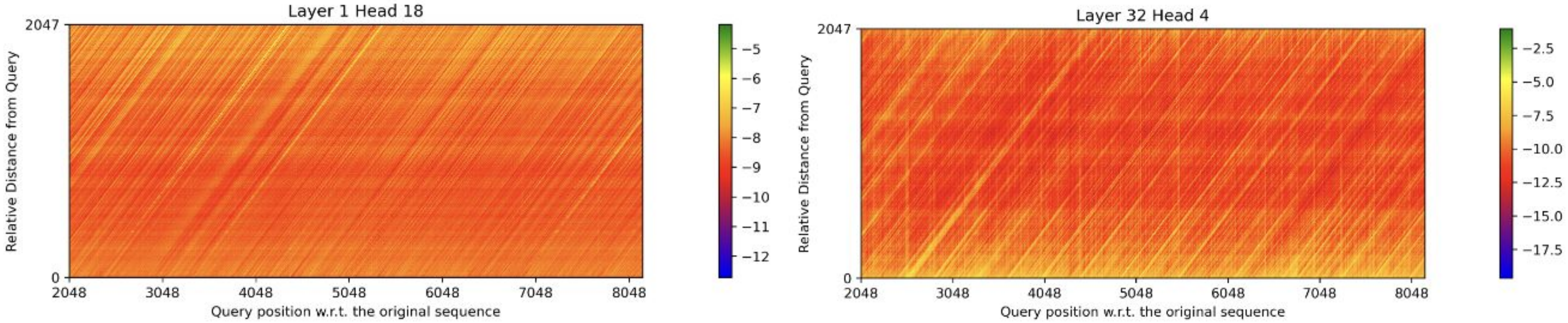
For repeated queries, strong performance in terms of larger throughput and reduced memory

BumbleBee: Dynamic KV-Cache Streaming Submodular Summarization for Infinite-Context Transformers

Lilly Kumari, Shengjie Wang, Tianyi Zhou, Nikhil Sarda, Anthony Rowe, Jeff Bilmes

Motivation

strongly attends to a small subset of tokens



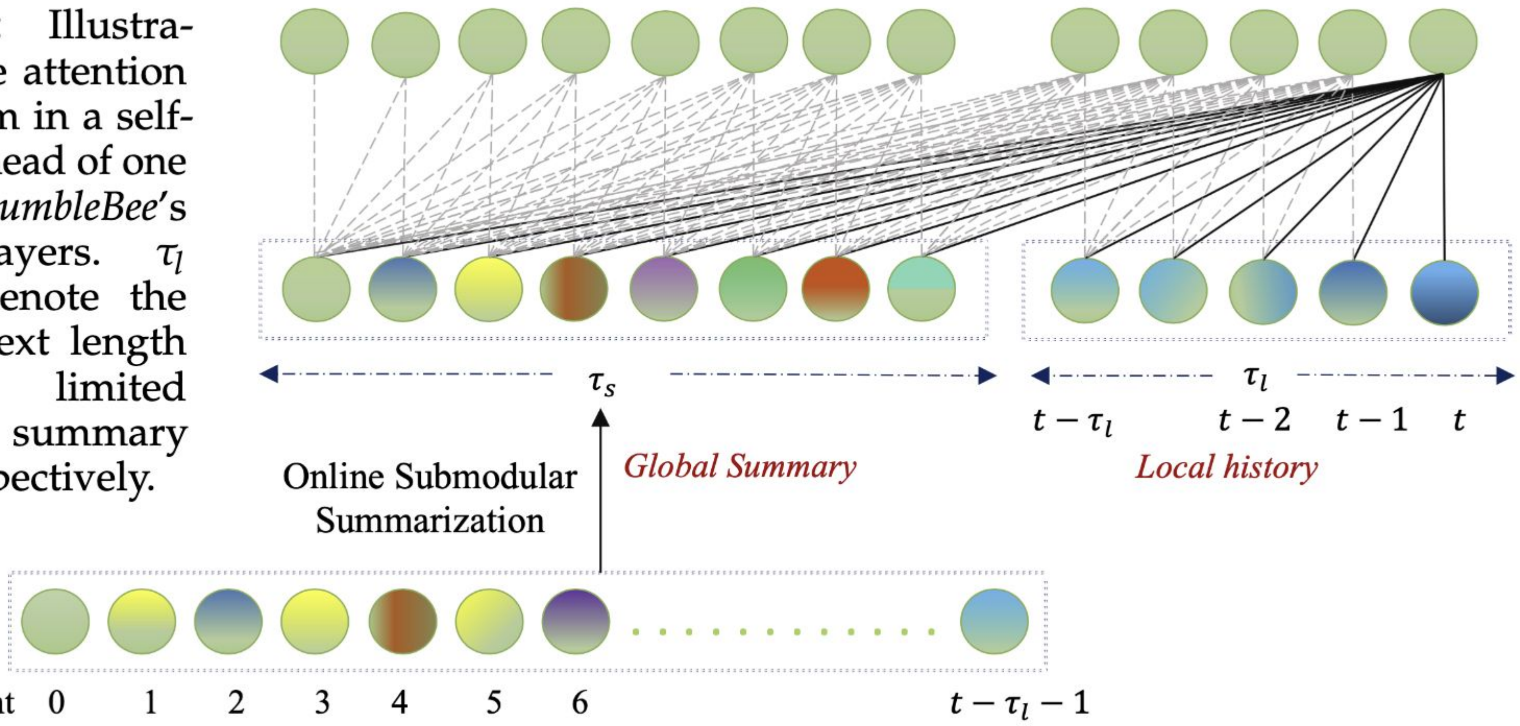
(a) Test sample 1 (b) Test sample 2

Figure 2: Attention maps for two different WikiText-103 articles using LLaMA-7B model.

Core Idea

- Existing methods use **modular scores**, where each KV state is evaluated independently
- Frame the KV cache selection as **subset selection** problem with submodular objective function

Figure 1: Illustration of the attention mechanism in a self-attention head of one of the *BumbleBee*'s decoder layers. τ_l and τ_s denote the local context length and the limited global summary length respectively.

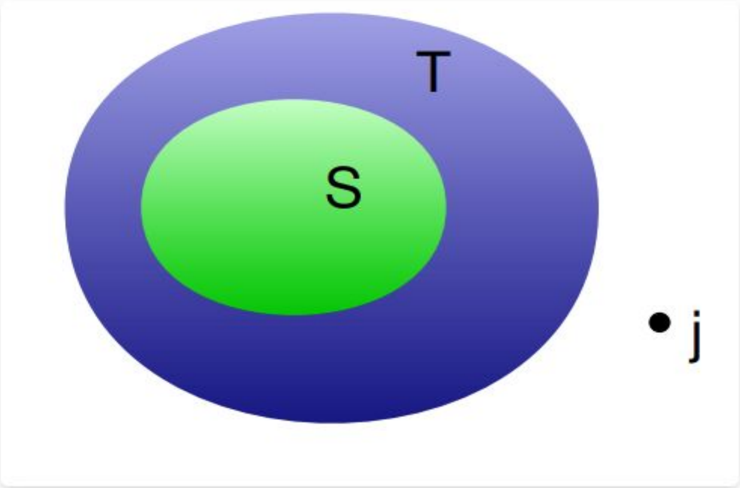


Submodular

A submodular function $f: 2^V \rightarrow R$ defined on ground set V, it has **diminishing return property**:

$$\text{if } S \subset T, j \notin T, f(S \cup \{j\}) - f(S) \geq f(T \cup \{j\}) - f(T)$$

Greedy algorithm for submodular maximizationa has theoretical (1 – 1/e) guarantee from the optimal solution



Submodularity

facility location (FL) function
how well a subset can represent the whole set

$$f_{\mathrm{FL}}(A) = \sum_{v \in V} \max_{v' \in A} \mathrm{sim}(v, v'). \qquad (2)$$

feature-based function
how much total importance is captured

$$c(A) = \sum_{u \in U} \phi_u\Big(\sum_{v \in A} m_u(v)\Big). \qquad (3)$$

How to select a subset?

Diversity

Importance

Submodularity

For Bumblebee

- $c(\cdot)$: $|U| = 1$ and weight $m_u(k_i) = a_n^i$, accumulated attention
- $f_{FL}(\cdot)$: similarity matrix with pairwise cosine, followed by ReLU

Mixture function: trade-off

normalized with $f_{\mathrm{FL}}(\emptyset) = 0$ and $f_{\mathrm{FL}}(V) = 1$ (and the same for $c(\cdot)$)

$$g_\lambda(A) = \lambda f_{\mathrm{FL}}(A) + (1 - \lambda) c(A)$$

Offline algorithm

Algorithm 1 Offline Submodular KV cache Summarization during Prefill/Encoding Phase

1: **Input:** Submodular functions capturing diversity f_{FL} in the key embeddings space and importance c via attention frequency for layer l and attention head h; mixture function $g_\lambda(\cdot) = \lambda f_{\text{FL}}(\cdot) + (1-\lambda)c(\cdot)$; a set of n KV attention states $K_n = \{(k_i)\}_{i=1}^n, V_n = \{(v_i)\}_{i=1}^n$ corresponding to the n prompt tokens; budget τ_s.

2: **Output:** A final summary S_n such that $S_n \subseteq \{(k_i, v_i)\}_{i=1}^n$ and $|S_n| \leq \tau_s$.

3: **Initialize:** $S_n = \emptyset$; compute accumulated attention score vectors a_n for each key $k \in \{k_i\}_{i=1}^n$. a_n^i denotes accumulated attention scores attributed to key k_i across all n query tokens.

4: **for** $j = 1$ to τ_s **do**

5: $\quad k_{\text{imp}} \leftarrow \text{argmax}_{e \in K_n \setminus S_n} g_\lambda(S_n \cup e) - g(S_n)$

6: $\quad S_n \leftarrow S_n \cup \{(k_{\text{imp}}, v_{\text{imp}})\}$ where v_{imp} is the value embedding associated with k_{imp}.

7: **end for**

Online algorithm

fill until budget, else greedy choose worst one to evict

for each new token compute costs

$\mathcal{O}(\tau_s \times d + \tau_s^2)$

Algorithm 2 *BumbleBee*: Streaming Submodular KV cache Summarization for Transformers

Input: Submodular functions for diversity f_{FL} in the key embeddings space and importance c w.r.t. attention frequency resp. for layer l and attention head h; mixture function $g_\lambda(\cdot) = \lambda f_{\text{FL}}(\cdot) + (1-\lambda)c(\cdot)$; stream of QKV attention states $\{(q_i, k_i, v_i)\}_{i=1}^n$; budget τ_s.
Output: A running summary S_t of for every time step t such that $S_t \subseteq \{(k_i, v_i)\}_{i=1}^t$.
Initialize: $S_0 = \emptyset$, $a_0 = \emptyset$ where $a_t \in \mathbf{R}^{|S_t|}$ denotes the accumulated attention scores corresponding to keys present in S_t across t time steps.
for $t = 1, \ldots, n$ **do**
    Update a_t for each $k \in S_{t-1}$ by adding $a(q_t, k, S_{t-1} \cup k_t)$
    if $t < \tau_s$ **then**
        $S_t \leftarrow S_{t-1} \cup \{(k_t, v_t)\}$
        Append $a(q_t, k_t, S_t)$ to a_t s.t. $|a_t| = |S_t|$
    else
        Let $S_t' = S_{t-1} \cup \{(k_t, v_t)\}$; $k_{\text{discard}} \leftarrow \text{argmin}_{k_i \in S_t'} g_\lambda(k_i | S_t' \setminus k_i)$
        $S_t \leftarrow S_t' \setminus \{(k_{\text{discard}}, v_{\text{discard}})\}$
        if $k_{\text{discard}} \neq k_t$ **then**
            Evict a_t^j (the accumulated attention score for the discarded key k_{discard}) from a_t.
            Append $a(q_t, k_t, S_t)$ to a_t
        end if
    end if
end for

Experiments

- **Datasets from benchmark:** lm-eval-harness, HELM, LongBench
- **Models**: LLaMA 7B and 13B, LLaMA2 7B and 13B, Llama-2-Chat 7B and LongChat-32k 7B
- **Baselines**: All, Local, Random+Local, Attention sinks+Local, H2+local
- *Submarine* software system for submodular computation

Experiments

llm-eval-harness benchmark, **0.1x** the input length as budget

(♥) log-based $\phi(x) = \log(1+x)$

(♦) power-based $\phi(x) = g^{-1}(x)$ where $g(y) = \alpha y^{1/\alpha} + \beta y$

Model	Methods	OpenBookQA	COPA	RTE	MathQA	PiQA	Winogrande
LLaMA-13B	**All**	47.4	85	73.28	31.86	80.36	75.69
	Local	28.4	64	53.43	23.25	58.32	49.88
	Random + Local	27.6	58	54.63	21.76	54.13	50.64
	Attn Sinks + Local	44.4	80	67.51	29.78	79.22	70.48
	H2 + Local	44.2	83	64.98	29.71	**79.49**	70.32
	BumbleBee ♥	**47.6**	**85**	**71.48**	**31.02**	79.38	71.98
	BumbleBee ♦	46.6	83	67.15	30.82	**79.49**	**73.01**
LLaMA-7B	**All**	44.6	81	68.95	29.85	80.03	71.51
	Local	28.4	56	50.90	23.02	58.27	51.38
	Random + Local	28.0	63	51.26	21.76	53.94	49.30
	Attn Sinks + Local	41.6	**82**	58.12	27.40	78.07	67.80
	H2 + Local	41.4	78	63.54	27.50	77.31	65.82
	BumbleBee ♥	**43.2**	79	**68.95**	27.74	78.24	**68.75**
	BumbleBee ♦	**43.2**	79	63.90	**28.51**	**78.56**	68.19

Experiments

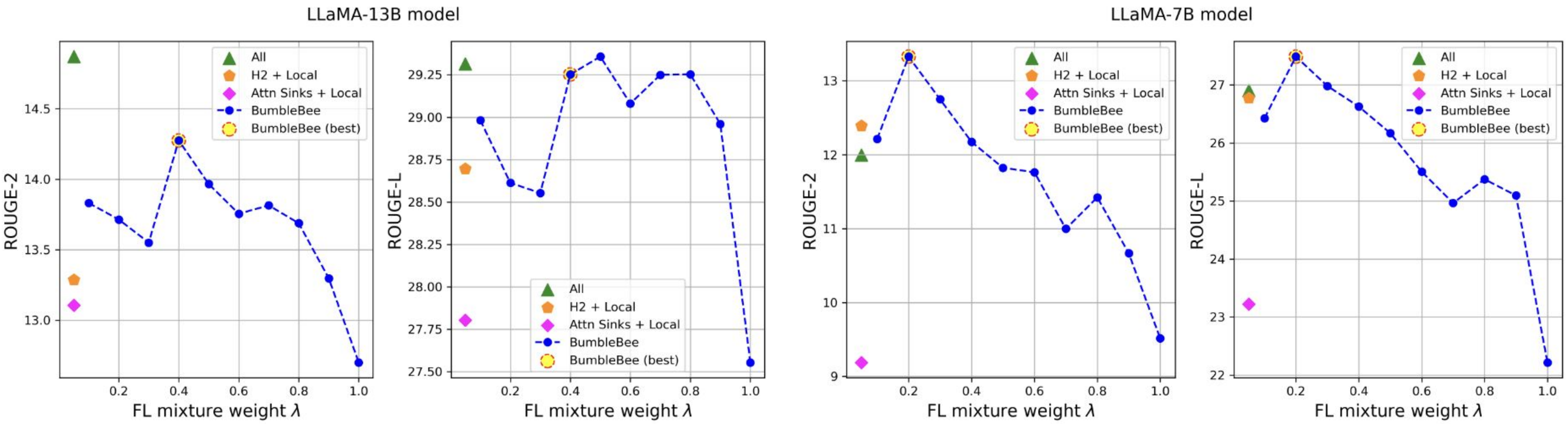
LongBench benchmark, λ=0.3

Model	Method	Qasper	MultiFieldQA-en	HotpotQA	2WikiMQA	QMSum	TREC
LLaMA-7B-chat 4k	**All***	19.20	36.80	25.40	32.80	20.80	61.5
	All (self)	21.60	36.76	27.55	31.58	20.78	64.0
	Attn Sinks + Local	14.74	22.93	22.08	29.73	19.25	56.0
	H2 (20%)	**19.82**	26.60	26.28	25.69	**21.45**	60.0
	BumbleBee (20%) ♥	19.37	27.73	26.14	27.67	20.68	**61.5**
	BumbleBee (20%) ♦	19.59	**28.60**	**28.99**	**30.19**	21.05	59.0
LongChat-7B 32k	**H2** (SW, 20%)	21.64	30.72	14.07	15.10	18.11	40.5
	BumbleBee (SW, 20%) ♦	**23.27**	**33.16**	**22.52**	**17.58**	**20.27**	**44.5**

Experiments

XSUM dataset, few-shot summarization task

λ=0.2 still better than H2+local (which equals to λ=0 and φ is identity function)



(a) LLaMA-13B

(b) LLaMA-7B

Experiments

Context reduction ratio	Original Context Length	
	16k	**100k**
1:1	59.30 ± 0.39	OOM
5:1	47.49 ± 4.16	71.50 ± 0.10
10:1	39.74 ± 1.31	48.16 ± 0.09

Table 6: Decoding speed (in ms/token) for two KV cache reduction ratios (5:1 and 10:1) and the baseline KV cache method using the entire context (1:1) across all heads. All experiments are performed on an A100 80GB GPU using the LongChat-7B-32k with a batch size of 1.

Takeaways

- introduce diversity into selection aside from only importance(H2O)
- reframe eviction as subset selection problem, and submodularity guarantees that a simple Greedy Algorithm achieves a near-optimal solution

Discussion:

- rely on submodular optimization tool(not open-source)
- redundancy/diversity matters
  - R-KV: heuristic ranking $Z = \lambda \cdot \mathrm{Imp} - (1-\lambda) \cdot \mathrm{Redu}$
  - OmniKV: inter-layer redundancy

Thank You!

jhu.edu