AdapShot: Adaptive Many-Shot In-Context Learning with Semantic-Aware KV Cache Reuse

Jie Ou[1], **Jinyu Guo**[1*], **Shiyao Guo**[1], **Yuang Li**[1], **Ruiqi Wu**[1], **Zhaokun Wang**[1], **Wenyi Li**[1], **Wenhong Tian**[1*]

[1] School of Information and Software Engineering,
University of Electronic Science and Technology of China

Abstract

Many-Shot In-Context Learning (ICL) has emerged as a promising paradigm, leveraging extensive examples to unlock the reasoning potential of Large Language Models (LLMs). However, existing methods typically rely on a predetermined, fixed number of shots. This static approach often fails to adapt to the varying difficulty of different queries, leading to either insufficient context or interference from noise. Furthermore, the prohibitive computational and memory costs of long contexts severely limit Many-Shot's feasibility. To address the above limitations, we propose **AdapShot**, which dynamically optimizes shot counts and leverages KV cache reuse for efficient inference. Specifically, we design a probe-based evaluation mechanism that utilizes output entropy to determine the optimal number of shots. To bypass the redundant prefilling computation during both the probing and inference phases, we incorporate a semantics-aware KV cache reuse strategy. Within this reuse strategy, to address positional encoding incompatibilities, we introduce a decoupling and re-encoding method that enables the flexible reordering of cached key-value pairs. Extensive experiments demonstrate that AdapShot achieves an average performance gain of ∼10% and a 4.64× speedup compared to state-of-the-art DBSA.

1 Introduction

In recent years, Large Language Models (LLMs) have achieved milestone breakthroughs in natural language processing. From GPT-3 (Brown et al., 2020) to the LLaMA (Touvron et al., 2023) series, the reasoning capabilities of large language models are continuously being explored and enhanced (Zhang et al., 2026a,b). And LLMs have demonstrated remarkable In-Context Learning (ICL) capabilities, enabling models to adapt to downstream tasks without parameter updates.

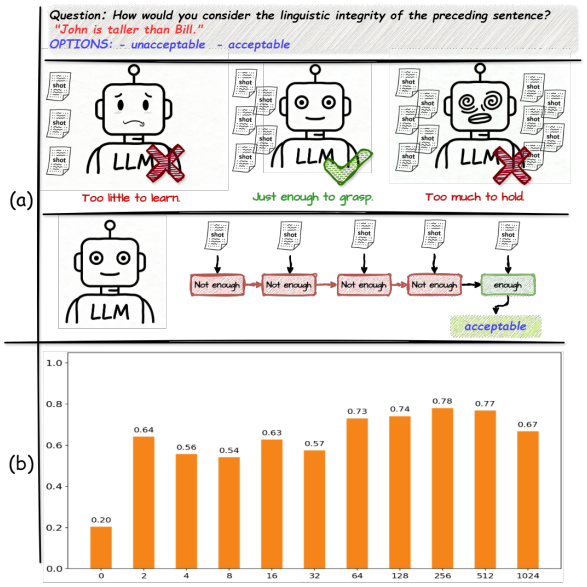

Figure 1: (a) Comparison of Few-Shot, adaptive Many-Shot, and Many-Shot approaches. (b) The EM accuracy on the QNLI dataset under different shot settings.

While traditional Few-Shot ICL typically relies on fewer than 10 examples, the latest research paradigm is gradually evolving toward **Many-Shot ICL** (Agarwal et al., 2024; Li et al., 2023; Bertsch et al., 2025). Studies show that extending the context window to include hundreds or even thousands of examples can significantly unlock the model's potential in complex reasoning and knowledge-intensive tasks. As illustrated in Figure1(b), model accuracy overall improves substantially as the number of examples increases. This scaled ICL paradigm powerfully demonstrates the strong capability of LLMs to perform pattern recognition and reasoning through large-scale contexts.

Many-shot ICL has demonstrated remarkable performance, even rivaling fine-tuning methods in certain scenarios (Yin et al., 2024; Agarwal et al., 2024). Recent studies (Bhope et al., 2025; He et al., 2024) have further enhanced its effectiveness by exploring and optimizing the ordering strategies of

*Corresponding author Emails: guojinyu@uestc.edu.cn, tian_wenhong@uestc.edu.cn

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 42946–42960
July 2-7, 2026

shots within prompts. However, existing research typically employs a predetermined fixed number of shots rather than dynamically adjusting based on actual task requirements. As illustrated in Figure 1(a), this fixed strategy exhibits significant limitations. When the preset number of shots is insufficient (degrading to a few-shot regime), it leads to inadequate learning of complex tasks by Large Language Models (LLMs). Conversely, when the preset number is excessive, it may exceed the LLM's comprehension capacity or introduce irrelevant noise, thereby interfering with accurate task understanding. As shown in Figure 1(b), more shots do not always yield better results, indicating that different queries require varying numbers of shots rather than following a more-is-better principle.

On the other hand, regardless of the number of examples used, many-shot ICL faces severe efficiency challenges when fully adopting the large-scale in-context learning paradigm. Furthermore, due to the O(n^2) time complexity of the self-attention mechanism in the Transformer architecture (Vaswani et al., 2017), inference latency increases dramatically as context length grows. This poses significant challenges to GPU memory capacity in practical deployment. Consequently, the efficiency bottleneck of many-shot ICL has become a critical issue that must be addressed.

To address the above challenges, we propose **AdapShot**, an **ada**ptive many-**shot** ICL approach, which dynamically allocates context budget based on the difficulty of each input query. In further, to alleviate the increasingly severe efficiency issues in many-shot ICL, we introduce KV cache reuse into this domain and propose a semantics-aware KV cache construction and position re-encoding method to break the barrier of context sharing across samples. Specifically, we first design a probe-based dynamic evaluation mechanism. This mechanism inserts probes between context windows and calculates output entropy to quantify model confidence, thereby determining the minimum required context budget. To deal with the efficiency challenges posed by long contexts in many-shot ICL, we leverage an offline global KV cache that eliminates prefilling overhead during probe evaluation and enables effective KV cache reuse. Within this module, to flexibly reorder samples without compromising the correctness of position information, we design a position decoupling and re-encoding mechanism. It leverages the mathematical properties of Rotary Position Embedding (RoPE) to dynamically map retrieved KV pairs to new logical positions through low-cost vector rotation operations. This design allows us to freely reorder and concatenate KV blocks according to semantic importance without expensive model forward passes.

Extensive experiments validate AdapShot's superiority, showing it outperforms DBSA with an average $\sim$10% performance gain and a 4.64$\times$ speedup. These results confirm that AdapShot effectively optimizes the trade-off between accuracy and efficiency in many-shot ICL. The main contributions of this paper are summarized as follows:

- We propose a probe-based dynamic budget allocation mechanism that evaluates task difficulty to optimize shot usage and reduce redundant computation. To the best of our knowledge, this is the first work to explore adaptive many-shot ICL based on the probe.
- We introduce KV cache reuse into many-shot ICL scenarios to address the increasingly severe efficiency challenges. Furthermore, we propose a position decoupling and re-encoding strategy that leverages the rotational properties of RoPE to enable low-cost, lossless, and flexible reordering of shots.
- Through comprehensive evaluation on various mainstream LLMs and multiple benchmark datasets, experimental results show that our method significantly reduces inference latency while maintaining LLM reasoning quality.

2 Related Work

2.1 Many-Shot In-Context Learning

As large language models scale to support extended context windows, in-context learning has evolved from few-shot to many-shot paradigms. Agarwal et al. (2024) pioneered systematic investigation of many-shot ICL, demonstrating substantial performance improvements when incorporating hundreds of demonstration examples. They further explored "Reinforced ICL" and "Unsupervised ICL" to mitigate dependency on annotated data. Bertsch et al. (2025) revealed continued performance gains with increasing examples, particularly for tasks with large label spaces, while also exhibiting reduced sensitivity to input permutations.

Existing research has explored many-shot ICL optimization across several dimensions. For a

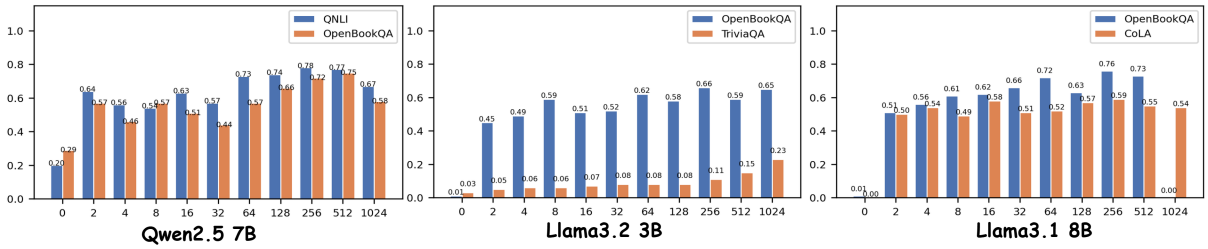

Figure 2: Many-Shot ICL performance of different models across multiple datasets.

demonstration organization, He et al. (2024) proposed HIDO to address order instability. For selection strategies, Golchin et al. (2025) reduced computational overhead through similarity-based retrieval with cached demonstrations, while Wan et al. (2025) introduced BRIDGE to optimize influential example selection. For scaling approaches, Gu et al. (2025) and Chen et al. (2025b) developed IterPSD and MAPLE using pseudo-labeling techniques. Zou et al. (2025) categorized ICL into Similar Sample Learning (SSL) and All Sample Learning (ASL), revealing that models handle 64k tokens in SSL but some degrade at 16k tokens in ASL. Applications have extended to specialized domains such as molecular inverse design (Moayedpour et al., 2024).

Despite these advances, current methods still rely on empirically fixed demonstration quantities, face order-sensitivity constraints that disrupt KV cache sharing, and encounter severe efficiency bottlenecks. While Xiao et al. (2025) enhanced efficiency via dynamic block-sparse attention, validation remains confined to simple NLU tasks. Our AdapShot addresses these challenges by dynamically adjusting the demonstration scale based on query difficulty.

2.2 Efficient LLM Inference Technology

Sparse attention methods like block-sparse mechanisms Child et al. (2019); Zaheer et al. (2020); Wang et al. (2024); Acharya et al. (2024) and hierarchical structures Yang et al. (2016) reduce computational complexity but typically require retraining. Ratner et al. (2023) proposed Parallel Context Windows that avoids retraining through block-sparse attention, though it remains limited to context-constrained models.

KV cache compression techniques include token eviction strategies, where Xiao et al. (2023) discovered "attention sink" phenomena leading to StreamingLLM that preserves only initial and recent KV pairs. Subsequent works Li et al. (2024); Zhang et al. (2023); Liu et al. (2025) improved selective retention approaches. However, eviction strategies face challenges in many-shot ICL scenarios where different queries require attention to different example subsets. Alternative compression methods like quantization and low-rank approximation Liu et al. (2024); Zhang et al. (2024) reduce memory usage while preserving all tokens. The efficiency of LLMs has also been extensively studied in multimedia and computer vision (Zheng et al. (2025); Cao et al. (2026); Zheng et al. (2026)).

Our work integrates inference acceleration techniques into the many-shot ICL domain, enabling practical deployment of demonstration-rich learning that was previously computationally prohibitive.

3 Preliminary Study and Motivation

In Many-Shot ICL, we maintain a repository $\mathcal{S} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ of N input-output pairs. For a query q, we select k examples (typically hundreds to thousands) from $\mathcal{S}$ to form context C. The LLM then processes LLM$(I \oplus C \oplus q)$ to generate the output, where I is the instruction and $\oplus$ denotes concatenation.

Observation 1: The relationship between the number of examples and performance is non-monotonic. As shown in Figure2, as the number of examples increases from 0 to 1024, model performance exhibits complex variation patterns. For instance, Llama-3.1-8B on OpenBookQA shows accuracy dropping from 76% with 256 examples to 73% with 512 examples, and crashes due to memory limitations at 1024 examples. Additionally, Qwen2.5-7B on QNLI reaches peak performance of 78% at 256 examples, followed by a gradual decline. This performance degradation phenomenon indicates that excessive examples may severely interfere with the model's reasoning process.

Observation 2: Different models exhibit significant variations in perceiving task complex-

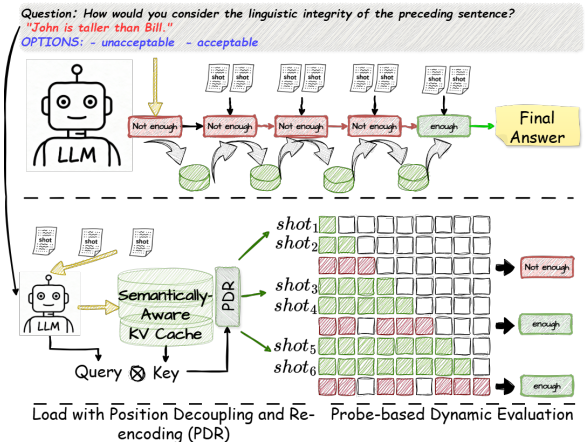

Figure 3: The pipeline of AdapShot.

ity. Notably, "task difficulty" is not an objective universal standard but is highly dependent on the capability boundaries of specific models. For example, Llama-3.2-3B achieves only 23% accuracy on TriviaQA even with 1024 examples, indicating this task is extremely challenging for 3B-scale models. However, on another knowledge-intensive task, OpenBookQA, the same model achieves approximately 65% performance with 64-256 examples. Furthermore, the "optimal number of examples" varies dramatically across models: Qwen2.5-7B requires 256-512 examples to reach optimal performance on most tasks, while Llama-3.1-8B's performance on CoLA is relatively insensitive to example count, consistently fluctuating between 50-60%.

Based on these observations, we argue that the current "one-size-fits-all" static example configuration strategy is fundamentally flawed. Each "model-task-query" triplet requires a unique optimal example configuration. This adaptive approach avoids both computational waste and performance degradation from example overload.

4 Method

AdapShot provides an adaptive inference framework that dynamically adjusts context scale based on the difficulty of input queries, enabling flexible and efficient allocation of computational resources. As illustrated in Figure 3, AdapShot first constructs a semantically-aware global KV cache pool during the offline phase. Subsequently, during the inference phase, it employs a probe-based dynamic evaluation mechanism that estimates model confidence using minimal single-token generation. This allows adaptive determination of the minimum effective example set. For activated examples, a position re-encoding mechanism is utilized to perform concatenation and position correction solely at the KV level, avoiding redundant prefilling computation.

4.1 Probe-based Dynamic Evaluation

Many-Shot ICL deployment often falls into the misconception that "more examples are always better," leading to significant memory burden and inference latency, and even performance degradation. To address this, we propose a **probe-based dynamic evaluation mechanism** to assess the model's confidence on the current query and dynamically determine whether to continue expanding the number of examples.

Before formal probe evaluation, the system first performs semantic relevance ranking on the global example pool $\mathcal{S} = \{\text{shot}_1, \ldots, \text{shot}_N\}$ based on query q. Following the sorting method in Section 4.2, the obtained $\mathcal{S}_{\text{sorted}} = [\text{shot}^{(1)}, \ldots, \text{shot}^{(N)}]$ ensures that the most relevant examples are prioritized for subsequent procedures.

Based on the sorted results, the probe mechanism adopts an iterative process to dynamically determine the final context scale. Assuming a step size of n examples per iteration, the k-th iteration ($k \geq 1$) activates the top $(k \times n)$ most relevant examples and constructs a probe context:

$$\text{C}_k^{\text{probe}} = I \oplus \mathcal{S}_{\text{sorted}}[:k \times n] \oplus q \oplus \text{P}_{\text{probe}}, \quad (1)$$

where I represents an optional instruction prefix or task description, $\oplus$ denotes string concatenation, and P_{probe} is a probe prompt (e.g., "Based on the above information, are you confident enough to answer?"). The model then generates only a single token (e.g., "Yes" or "No") conditioned on $\text{C}_k^{\text{probe}}$, and computes the entropy of its output distribution $\{p(\text{Yes}), p(\text{No})\}$:

$$H_k = -\sum_{c \in \{\text{Yes, No}\}} p\big(c \mid \text{C}_k^{\text{probe}}\big) \log p\big(c \mid \text{C}_k^{\text{probe}}\big). \quad (2)$$

If $H_k \leq \tau$ (where τ is a preset threshold), the model is considered to have sufficient confidence for the task. The iteration stops, and the currently activated example set is used for subsequent formal inference. If $H_k > \tau$, we set $k \leftarrow k + 1$ and proceed to the next probe iteration.

To mitigate the increased time overhead from multiple probe cycles, we leverage tree-structured attention for single-round parallel multi-probe verification. Given the accelerator's inherent parallelism, this approach does not introduce significant

computational overhead but rather reduces the number of iteration rounds.

For candidate example counts $\{n_1, n_2, \ldots, n_m\}$, we construct the following parallel input sequence: $[\text{shot}_1, \ldots, \text{shot}_{n_1}, \text{probe}_1, \text{shot}_{n_1+1}, \ldots, \text{shot}_{n_2}, \text{probe}_2, \ldots]$ Additionally, we construct a tree-structured attention mask M such that each probe probe_i only attends to its corresponding first n_i examples:

$$M_{ij} = \begin{cases} 1 & \text{if } j \leq n_k \text{ and } i = \text{probe}_k \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

This allows each probe probe_i to independently generate confidence assessments based on different amounts of context in a single forward pass, while the overall computational complexity is comparable to processing the longest sequence. Subsequently, the minimum number of examples satisfying the confidence threshold is selected as the optimal configuration.

4.2 Semantically-Aware KV Cache

Many-Shot ICL requires prefilling hundreds or even thousands of examples during inference. The Key-Value (KV) representations of these examples introduce memory redundancy and additional inference latency. To substantially reduce this computational overhead without compromising model performance, AdapShot proposes a semantically-aware hierarchical KV cache for maximum cross-sample reuse.

During the offline precomputation phase, AdapShot performs one-time offline computation on the global example pool $\mathcal{S}$ to obtain the KV representations for each example shot_i across all layers $\ell \in [1, L]$ and attention heads $h \in [1, H]$:

$$\begin{aligned} \left(\mathcal{K}_{\ell,h}^{(i)}, \mathcal{V}_{\ell,h}^{(i)}\right) &= \text{Prefill}\big(\text{shot}_i\big), \\ \forall\, i \in [1, N],\ \ell \in [1, L],\ h \in [1, H], & \end{aligned} \quad (4)$$

where L denotes the number of model layers, H represents the number of attention heads per layer, and $\mathcal{K}_{\ell,h}^{(i)}$ and $\mathcal{V}_{\ell,h}^{(i)}$ are the Key and Value tensors for example shot_i at layer ℓ and head h, respectively. These KV vectors are organized and stored in a hierarchical structure within the global cache pool.

During online inference, for a new query q, AdapShot first encodes q to obtain Query vectors across all layers and heads: $\mathbf{Q}_{\ell,h}^{(q)} \in \mathbb{R}^{T_q \times d}$:

$$\mathbf{Q}_{\ell,h}^{(q)} = \text{Encode}_{\ell,h}(q), \quad (5)$$

where T_q denotes the number of tokens in the query sequence and d is the hidden dimension.

Next, the system computes semantic relevance between the query vectors and the Key vectors of all examples in the global pool. Specifically, for example shot_i, we calculate the attention scores between the query and the example at layer ℓ and head h:

$$\mathbf{A}_{\ell,h}^{(i)} = \text{softmax}\left(\frac{\mathbf{Q}_{\ell,h}^{(q)} \cdot (\mathcal{K}_{\ell,h}^{(i)})^T}{\sqrt{d}}\right), \quad (6)$$

where $\mathbf{A}_{\ell,h}^{(i)} \in \mathbb{R}^{T_q \times T_i}$ represents the token-level attention matrix between the query sequence and example i, with T_i being the number of tokens in example i. We average the attention scores across all tokens to obtain the relevance score $s^{(i)}$ for example i. We rank all examples by their relevance scores $\{s^{(1)}, s^{(2)}, \ldots, s^{(N)}\}$ in descending order and selects the top k examples as the active set $\mathcal{S}_{\text{active}}$. This attention-based retrieval identifies the most relevant examples for each query without additional encoding models, while reusing the corresponding KV cache.

4.3 Position Decoupling and Re-encoding

During offline construction, each example shot_i is prefilled independently with position encodings starting from 0. However, when examples are reordered during online inference based on relevance scores, directly concatenating cached KV pairs causes position conflicts.

Consider two examples shot_A and shot_B with lengths T_A and T_B. During offline prefilling, their position indices are $[0, 1, \ldots, T_A - 1]$ and $[0, 1, \ldots, T_B - 1]$ respectively. When concatenating shot_B after shot_A, the expected positions should be $[0, \ldots, T_A - 1, T_A, \ldots, T_A + T_B - 1]$, but shot_B's cached Keys still encode positions starting from 0, resulting in a position offset $\Delta = T_A$.

To address this, we leverage the rotational composability of RoPE. The RoPE encoding formula is:

$$\begin{aligned} \text{RoPE}(\mathbf{x}, p) = \mathbf{x} &\odot \cos(p\boldsymbol{\theta}) \\ &+ \text{rotate\_half}(\mathbf{x}) \odot \sin(p\boldsymbol{\theta}), \end{aligned} \quad (7)$$

where $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_{d/2-1}]$ with $\theta_i = 10000^{-2i/d}$. The key property of RoPE is that a vector with position p_1 can be transformed to position p_2 through an additional rotation of $\Delta = p_2 - p_1$.

Table 1: Performance comparison (Exact Match) of AdapShot and baselines across LLaMA and Qwen architectures. Best results are bolded. O.O.M. denotes Out of Memory. We extended DBSA† to these datasets.

Method	LLaMA-3.2 (3B)			Qwen2.5-7B						
	CoLA	QNLI	PIQA	CoLA	QNLI	PIQA	SQuAD v2	SVAMP	GSM8K	MathQA
Zero-shot	0.421	0.508	0.107	0.169	0.203	0.298	0.095	0.136	0.456	0.012
Few-shot (8)	0.449	0.517	0.193	0.484	0.541	0.318	0.089	0.107	0.305	0.350
Many-shot (256)	0.528	0.557	0.345	0.488	0.780	0.560	0.309	0.146	0.450	0.483
Many-shot (512)	0.540	0.579	**0.473**	0.542	0.768	0.522	0.236	0.456	0.581	0.232
Many-shot (1024)	0.533	0.526	O.O.M.	0.550	0.667	O.O.M.	O.O.M.	O.O.M.	O.O.M.	O.O.M.
DBSA†	0.649	**0.619**	0.368	0.657	0.807	0.404	0.404	0.737	0.343	0.323
AdapShot (Ours)	**0.777**	0.512	0.469	**0.699**	**0.825**	**0.599**	**0.465**	**0.790**	**0.649**	**0.524**

Table 2: Inference speedup of AdapShot relative to Many-shot baselines and DBSA on Qwen2.5-7B. Speedup is defined as the ratio of the baseline's average latency to AdapShot's average latency. Higher is better.

Dataset	vs. 64-shot	vs. 128-shot	vs. 256-shot	vs. 512-shot	vs. DBSA
CoLA	2.18×	3.03×	3.41×	3.23×	4.62×
QNLI	3.42×	2.85×	2.98×	4.63×	5.63×
PIQA	2.08×	2.15×	2.63×	3.54×	4.27×
SQuAD v2	1.72×	1.65×	1.77×	4.10×	2.64×
SVAMP	2.09×	2.93×	3.49×	4.53×	4.72×
GSM8K	3.21×	3.41×	3.59×	7.59×	4.27×
MathQA	2.56×	3.58×	4.77×	9.12×	6.33×
Average	**2.47**×	**2.80**×	**3.23**×	**5.25**×	**4.64**×

Thus, for a cached Key vector $\mathcal{K}_{\ell,h}^{(i)}$ at original position p_{old} that needs to be at position p_{new}, we compute the corrected Key as:

$$\begin{aligned}\mathcal{K}_{\ell,h,\text{new}}^{(i)} = \mathcal{K}_{\ell,h}^{(i)} \odot \cos(\Delta\boldsymbol{\theta}) \\ + \text{rotate\_half}(\mathcal{K}_{\ell,h}^{(i)}) \odot \sin(\Delta\boldsymbol{\theta})\end{aligned} \tag{8}$$

This mechanism enables efficient position re-encoding without recomputing Keys, completely decoupling the physical storage order from logical positions during inference. This allows AdapShot to reuse cached KV pairs flexibly while maintaining correct attention computation after semantic reordering.

5 Experiments

We employ the CoT Collection dataset (Kim et al., 2023) for comprehensive evaluation, selecting 7 representative tasks across diverse domains and reasoning types: natural language understanding (CoLA, QNLI, PIQA), question answering (SQuAD v2), and mathematical reasoning (SVAMP, GSM8K, MathQA). Details are provided in Appendix B.

5.1 Baselines

We compare AdapShot against multiple baselines for comprehensive evaluation. For performance, we test: Zero-shot (no examples), Few-shot (4-8 examples), Many-shot (64, 128, 256, 512, 1024 examples), and DBSA (Xiao et al., 2025) (dynamic block sparse attention with cached example groups). For efficiency, we measure speedup (ratio of baseline to our method's inference time) against Many-shot variants and DBSA.

5.2 Experimental Setup

All experiments were conducted on a Huawei Ascend 910B2 NPU cluster. The computational resources included 8 Ascend 910B2 NPUs, each equipped with 65,536MB of memory. The CPU was a HUAWEI Kunpeng 920 5250, with 48 cores per socket, and the system was configured with 1.5TB of memory. The software environment was based on the Huawei Cloud EulerOS 2.0 operating system. For more details in Appendix A.

5.3 Performance Analysis

Table 1 presents the performance comparison between AdapShot and baseline methods across LLaMA and Qwen architectures. On the more ca-

pable Qwen2.5-7B model, AdapShot demonstrates superior cross-task generalization, outperforming all baselines, including the state-of-the-art DBSA method, across every evaluated dataset. Notably, AdapShot achieves substantial gains in complex reasoning and comprehension tasks, securing 0.790 on SVAMP (vs. 0.737 for DBSA) and 0.465 on SQuAD v2 (vs. 0.404 for DBSA). On LLaMA-3.2 (3B), our method attains an exceptional score of 0.777 on CoLA, representing a 19.7% improvement over the best DBSA†. These results validate that AdapShot effectively surpasses current SOTA approaches by adaptively tailoring retrieval strategies to varying model capabilities.

Furthermore, the results expose the inherent limitations of fixed-length strategies, where increasing context length does not guarantee better performance. As observed in the Qwen2.5-7B results on QNLI, extending the context from 256 to 1024 shots causes performance to degrade from 0.780 to 0.667, indicating that excessive examples introduce detrimental noise. Additionally, the 1024-shot setting frequently triggers Out-Of-Memory failures. In contrast, AdapShot mitigates these issues by dynamically identifying the optimal context budget for each query. It avoids the noise accumulation seen in over-extended contexts while preventing memory overflows, demonstrating a robust balance between computational efficiency and task accuracy that rigid baselines fail to achieve.

5.4 Efficiency Evaluation

Table 2 presents the inference speedup of AdapShot relative to fixed Many-shot baselines and the DBSA method on Qwen2.5-7B. The results demonstrate that AdapShot achieves substantial improvements in computational efficiency across tasks of varying complexity, with the advantage becoming increasingly pronounced as the context scale expands. In reasoning-heavy benchmarks, AdapShot attains peak speedups of 9.12× on MathQA and 7.59× on GSM8K compared to the 512-shot baseline. Even against DBSA, which employs dynamic block-sparse attention, AdapShot maintains a robust average speedup of 4.64×, reaching up to 6.33× on MathQA. These findings confirm that AdapShot effectively minimizes redundant computation, offering superior scalability in large-context scenarios.

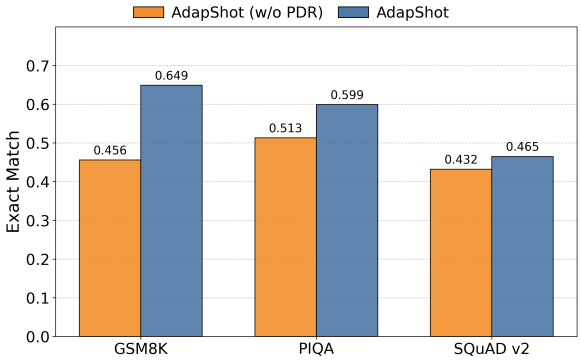

Figure 4: Ablation study on Position Decoupling and Re-encoding (PDR) module across different datasets and methods.

5.5 Ablation Studies

Effectiveness of PDR: As illustrated in Figure 4, removing the PDR module results in substantial performance degradation, with accuracy dropping by 29.7% on GSM8K (0.649→0.456), 14.4% on PIQA, and 7.1% on SQuAD v2. These results underscore the necessity of PDR in correcting position encoding misalignments caused by dynamic example reordering, ensuring the attention mechanism functions correctly during KV cache reuse.

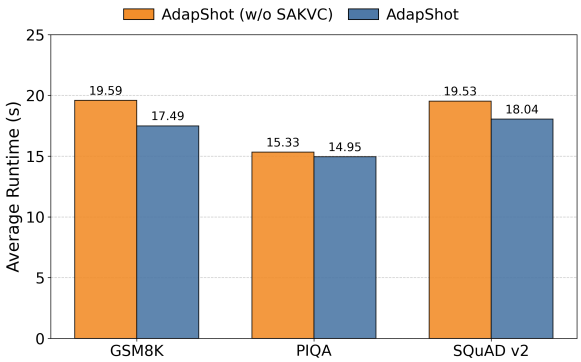

Figure 5: Runtime comparison between AdapShot with and without Semantically-Aware KV Cache (SAKVC) across different datasets.

Efficiency Gains from SAKVC: Figure 5 highlights the latency benefits of our semantically-aware kv cache strategy. By bypassing redundant prefilling for activated examples, SAKVC consistently reduces runtime, achieving reductions of 10.7% on GSM8K (19.59s→17.49s), 7.6% on SQuAD v2, and 2.5% on PIQA. This confirms that our SAKVC effectively alleviates the prefilling bottleneck inherent in Many-Shot ICL.

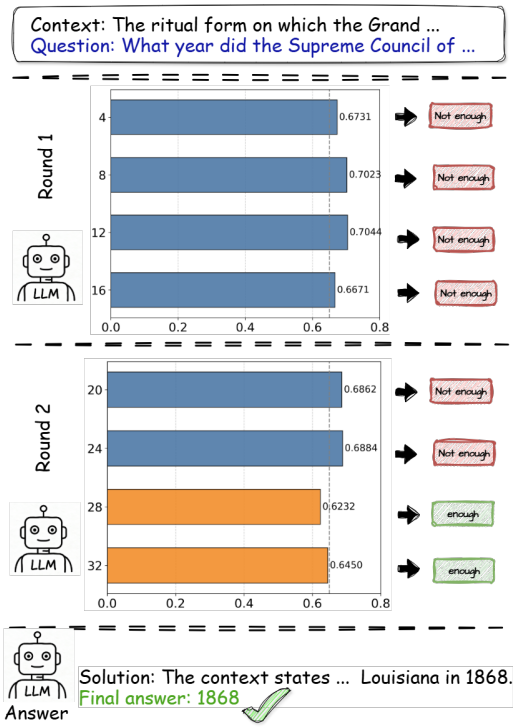

Figure 6: Visualization of AdapShot's dynamic shot selection process on a SQuAD reading comprehension task.

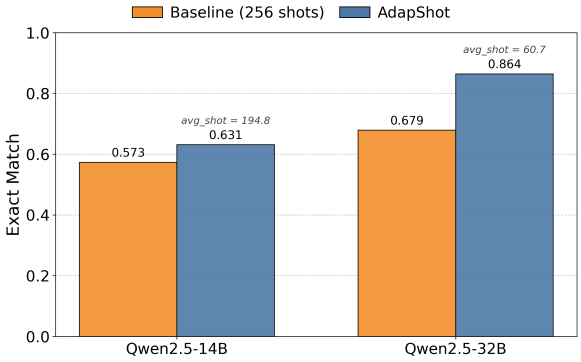

Figure 7: Scalability analysis of AdapShot.

5.6 Extended Analysis

Case Study: We visualize AdapShot's dynamic shot selection process and its effectiveness in activating knowledge using Qwen2.5-7B on a SQuAD reading comprehension task that asks: *What year did the Supreme Council of the Ancient and Accepted Scottish Rite of Louisiana appear in the jurisdiction of the Grand Lodge of Louisiana?* As shown in Figure 6, initially, four parallel probes with 4, 8, 12, and 16 shots produced entropy values of 0.6731, 0.7023, 0.7044, and 0.6671, respectively, all exceeding the 0.65 threshold and signaling insufficient context. AdapShot then conducted a second round with 20, 24, 28, and 32 shots, achieving entropy values of 0.6862, 0.6884, 0.6232, and 0.6450. With both 28 and 32 shots falling below the threshold, AdapShot efficiently selected 28 shots as the optimal configuration. The model subsequently employed chain-of-thought reasoning to correctly extract *1868* from the context.

Scaling with LLM Parameters: We evaluated scalability on Qwen2.5-14B and 32B using the CoLA dataset. As shown in Figure 7, AdapShot's advantage amplifies with model scale. Notably, on the 32B model, AdapShot achieves 86.41% Exact Match using only ∼60 shots, whereas the fixed 256-shot baseline degrades significantly to 67.91%. This sharp divergence confirms that larger models are highly sensitive to noise from irrelevant contexts. AdapShot effectively mitigates the heightened sensitivity of larger models to context noise by activating internal knowledge with a minimal, optimal set of demonstrations. For a detailed analysis, please refer to Appendix C.1.

Table 3: Compare with BM25 based Many-Shot ICL.

Method	GSM8K	PIQA	SQuAD v2
BM25 + 256-shot	0.456	0.379	0.418
BM25 + 512-shot	0.301	0.388	0.126
AdapShot (Ours)	**0.649**	**0.599**	**0.465**

Comparison with BM25-Baseline: Table 3 validates the robustness of AdapShot by comparing it against baselines utilizing BM25 for example retrieval. AdapShot outperforms the best BM25 configuration by significant margins (+0.193 on GSM8K, +0.211 on PIQA, and +0.047 on SQuAD v2). These results underscore that fixed-shot retrieval strategies are suboptimal compared to AdapShot, proving that the adaptive determination of shot count is a crucial factor.

Our AdapShot is not sensitive to specific probe threshold values (τ) and maintains consistent effectiveness. For detailed data, please refer to Appendix C.2.

Table 4: Performance of **openPangu-Embedded-1B-V1.1** under different in-context learning shot settings.

Method (Shots)	ARC-Easy	CoLA	PIQA	QNLI
Baseline (0-shot)	0.687	0.502	0.221	0.679
Baseline (8-shot)	0.679	0.512	0.246	0.699
Baseline (128-shot)	0.182	0.413	0.172	0.546
AdapShot (Ours)	0.704	0.650	0.271	0.694

Performance Comparison on OpenPangu: To further evaluate the generalizability of the proposed AdapShot method, we conduct additional experiments on the **openPangu-Embedded-1B-V1.1** (Chen et al., 2025a). Table 4 details the performance comparison including ARC-Easy, CoLA, PIQA, and QNLI. AdapShot outperforms the fixed-shot baselines on most tasks and achieves the better performance, without requiring manual shot selection. For a detailed analysis, please refer to Appendix C.3.

Table 5: Inference speedup of AdapShot on **openPangu-Embedded-1B-V1.1**.

Dataset	vs. 8-shot	vs. 64-shot	vs. 128-shot
ARC-Easy	1.47×	2.03×	3.50×
GSM8K	3.88×	5.07×	6.73×
PIQA	1.71×	1.56×	2.13×
QNLI	1.23×	1.57×	2.78×

Table 5 illustrates the computational efficiency of AdapShot compared with the many-shot baseline on the openPangu architecture. Overall, our method delivers speedup across all evaluated scenarios, confirming that AdapShot provides both superior predictive performance and significant latency reductions. For a detailed analysis, please refer to Appendix C.3.

6 Conclusion

This paper proposes AdapShot, which utilizes a probing-based mechanism to assess query difficulty and combines semantic-aware KV cache reuse with position-decoupled re-encoding techniques to dynamically match the optimal number of shots for each query. This method effectively eliminates repetitive context prefilling computations, significantly reducing redundant overhead while improving operational efficiency. Experiments demonstrate that AdapShot reduces latency while boosting performance. In the future, we will explore implicit shot generation techniques to reduce reliance on real data.

7 Limitations

Although AdapShot demonstrates superior performance in improving inference efficiency and dynamically adapting the number of shots, this study still presents certain limitations. Specifically, similar to traditional many-shot ICL methods, AdapShot relies on retrieving or constructing real data samples to serve as context demonstrations. This process inevitably incurs additional data storage overhead and requires significant effort in sample curation. Therefore, we believe a promising direction for future work is to explore the synthesis of "implicit shots" to replace the current paradigm of explicitly constructing prompts with real samples, thereby further reducing reliance on real-world data and enhancing generalizability.

8 Acknowledgments

This work is supported by the National Key R&D Program of China (No. 2026YFE0199800), the Chengdu Science and Technology Bureau Project (No. 2024-YF09-00041-SN), the National Natural Science Foundation of China Project with ID W2433163, the Sichuan Science and Technology Program (Grant No. 2026NSFSC1474), the Postdoctoral Fellowship Program (Grade C) of the China Postdoctoral Science Foundation (Grant No. GZC20251053) and the UESTC Kunpeng & Ascend Center of Cultivation (Project ID: H04W241592).

References

Shantanu Acharya, Fei Jia, and Boris Ginsburg. 2024. Star attention: Efficient llm inference over long sequences. In *Forty-second International Conference on Machine Learning*.

Rishabh Agarwal, Avi Singh, Lei Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, and 1 others. 2024. Many-shot in-context learning. *Advances in Neural Information Processing Systems*, 37:76930–76966.

Amanda Bertsch, Maor Ivgi, Emily Xiao, Uri Alon, Jonathan Berant, Matthew R Gormley, and Graham Neubig. 2025. In-context learning with long-context models: An in-depth exploration. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 12119–12149.

Rahul Atul Bhope, Praveen Venkateswaran, KR Jayaram, Vatche Isahagian, Vinod Muthusamy, and Nalini Venkatasubramanian. 2025. Optiseq: Ordering examples on-the-fly for in-context learning. *arXiv preprint arXiv:2501.15030*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sihan Cao, Jianwei Zhang, Pengcheng Zheng, Jiaxin Yan, Caiyan Qin, Yalan Ye, Wei Dong, Peng Wang, Yang Yang, and Chaoning Zhang. 2026. Language-guided token compression with reinforcement learning in large vision-language models. *arXiv preprint arXiv:2603.13394*.

Hanting Chen, Yasheng Wang, Kai Han, Dong Li, Lin Li, Zhenni Bi, Jinpeng Li, Haoyu Wang, Fei Mi, Mingjian Zhu, and 1 others. 2025a. Pangu embedded: An efficient dual-system llm reasoner with metacognition. *arXiv preprint arXiv:2505.22375*.

Zihan Chen, Song Wang, Zhen Tan, Jundong Li, and Cong Shen. 2025b. Maple: Many-shot adaptive pseudo-labeling for in-context learning. In *Forty-second International Conference on Machine Learning*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Shahriar Golchin, Yanfei Chen, Rujun Han, Manan Gandhi, Tianli Yu, Swaroop Mishra, Mihai Surdeanu, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. 2025. Towards compute-optimal many-shot in-context learning. In *Second Conference on Language Modeling*.

Zhengyao Gu, Henry Peng Zou, Aiwei Liu, Yankai Chen, Weizhi Zhang, and Philip S Yu. 2025. Scaling laws for many-shot in-context learning with self-generated annotations. In *ICML 2025 Workshop on Long-Context Foundation Models*.

Yinhan He, Wendy Zheng, Song Wang, Zaiyi Zheng, Yushun Dong, Yaochen Zhu, and Jundong Li. 2024. Hierarchical demonstration order optimization for many-shot in-context learning.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. *Preprint*, arXiv:2310.05736.

Seungone Kim, Se June Joo, Doyoung Kim, Joel Jang, Seonghyeon Ye, Jamin Shin, and Minjoon Seo. 2023. The cot collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning. *arXiv preprint arXiv:2305.14045*.

Mukai Li, Shansan Gong, Jiangtao Feng, Yiheng Xu, Jun Zhang, Zhiyong Wu, and Lingpeng Kong. 2023. In-context learning with many demonstration examples. *arXiv preprint arXiv:2302.04931*.

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

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. 2024. Minicache: Kv cache compression in depth dimension for large language models. *Advances in Neural Information Processing Systems*, 37:139997–140031.

Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Yue Liu, Bo Li, Xuming Hu, and Xiaowen Chu. 2025. Chunkkv: Semantic-preserving kv cache compression for efficient long-context llm inference. *arXiv preprint arXiv:2502.00299*.

Saeed Moayedpour, Alejandro Corrochano-Navarro, Faryad Sahneh, Alexander Koetter, Jiří Vyměťal, Lorenzo Kogler Anele, Pablo Mas, Yasser Jangjoo, Sizhen Li, Michael Bailey, and 1 others. 2024. Many-shot in-context learning for molecular inverse design. In *ICML 2024 AI for Science Workshop*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. Parallel context windows for large language models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 6383–6402.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xingchen Wan, Han Zhou, Ruoxi Sun, and Sercan O Arik. 2025. From few to many: Self-improving many-shot reasoners through iterative optimization and generation. In *The Thirteenth International Conference on Learning Representations*.

Haonan Wang, Qian Liu, Chao Du, Tongyao Zhu, Cunxiao Du, Kenji Kawaguchi, and Tianyu Pang. 2024. When precision meets position: Bfloat16 breaks down rope in long-context training. *Transactions on Machine Learning Research*.

Emily Xiao, Chin-Jou Li, Yilin Zhang, Graham Neubig, and Amanda Bertsch. 2025. Efficient many-shot in-context learning with dynamic block-sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31946–31958, Vienna, Austria. Association for Computational Linguistics.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Qingyu Yin, Xuzheng He, Chak Tou Leong, Fan Wang, Yanzhao Yan, Xiaoyu Shen, and Qiang Zhang. 2024. Deeper insights without updates: The power of in-context learning over fine-tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4138–4151.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and 1 others. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Jiaquan Zhang, Qigan Sun, Chaoning Zhang, Xudong Wang, Zhenzhen Huang, Yitian Zhou, Pengcheng Zheng, Chi-lok Andy Tai, Sung-Ho Bae, Zeyu Ma, and 1 others. 2026a. Tda-rc: Task-driven alignment for knowledge-based reasoning chains in large language models. *arXiv preprint arXiv:2604.04942*.

Jiaquan Zhang, Chaoning Zhang, Shuxu Chen, Xudong Wang, Zhenzhen Huang, Pengcheng Zheng, Shuai Yuan, Sheng Zheng, Qigan Sun, Jie Zou, and 1 others. 2026b. Learning global hypothesis space for enhancing synergistic reasoning chain. *arXiv preprint arXiv:2602.09794*.

Yanqi Zhang, Yuwei Hu, Runyuan Zhao, John Lui, and Haibo Chen. 2024. Unifying kv cache compression for large language models with leankv. *arXiv preprint arXiv:2412.03131*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

Pengcheng Zheng, Xiaorong Pu, Kecheng Chen, Jiaxin Huang, Meng Yang, Bai Feng, Yazhou Ren, Jianan Jiang, Chaoning Zhang, Yang Yang, and 1 others. 2025. Joint lossless compression and steganography for medical images via large language models. *arXiv preprint arXiv:2508.01782*.

Pengcheng Zheng, Chaoning Zhang, Jiarong Mo, Guohui Li, Jiaquan Zhang, Jiahao Zhang, Sihan Cao, Sheng Zheng, Caiyan Qin, Guoqing Wang, and 1 others. 2026. Llava-fa: Learning fourier approximation for compressing large multimodal models. *arXiv preprint arXiv:2602.00135*.

Kaijian Zou, Muhammad Khalifa, and Lu Wang. 2025. On many-shot in-context learning for long-context evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25605–25639.

A Implementation Details

All algorithms were implemented using the PyTorch framework (Paszke et al., 2019). To facilitate hardware acceleration on NPUs, we utilized the torch_npu extension package. The specific software environment was configured with CANN 8.1.rc1, PyTorch 2.5.1, and torch_npu 2.5.1.post1. All Large Language Models (LLMs) evaluated in this study were sourced from the Hugging Face repository[1].

Regarding the configuration of our Semantically-Aware KV Cache, we set the cache capacity to accommodate a maximum of 1,024 shots. To illustrate the memory overhead, for the Qwen2.5-7B model on the GSM8K dataset, this configuration requires approximately 25GB of memory for KV cache storage. We adjusted the hardware allocation based on model scale: models with 7B parameters or fewer were deployed on a single NPU, while 14B and 32B models were distributed across 2 and 4 NPUs, respectively, using tensor parallelism.

We sampled 1,000 instances to form a test set for evaluating both generation performance and computational efficiency. In our efficiency analysis, the total end-to-end latency is defined to encompass the entire pipeline: retrieval, ranking, the probing mechanism, and the final model inference.

B Dataset Specifications

CoT Collection Dataset The CoT Collection is a large-scale instruction-tuning dataset that augments the existing Flan Collection with an additional 1.84 million rationales across 1,060 tasks (Kim et al., 2023). The dataset is specifically designed to induce Chain-of-Thought (CoT) capabilities into language models.

These tasks collectively constitute a comprehensive evaluation framework that can test the effectiveness and generalization capability of the AdapShot method from multiple dimensions, covering a wide range of task types from basic language understanding to complex mathematical reasoning. The

[1] https://huggingface.co/

diversity in task complexity, domain knowledge requirements, and reasoning patterns ensures robust evaluation of few-shot learning capabilities across different cognitive and linguistic competencies.

B.1 Natural Language Understanding Tasks

CoLA (Corpus of Linguistic Acceptability) CoLA serves as a benchmark for evaluating the ability of artificial neural networks to judge the grammatical correctness of sentences. It contains English sentences from published linguistics literature that were manually labeled either as grammatical or ungrammatical.

QNLI (Question-answering Natural Language Inference) The QNLI dataset is a Natural Language Inference dataset automatically derived from the Stanford Question Answering Dataset v1.1 (SQuAD). SQuAD v1.1 consists of question-paragraph pairs, where one of the sentences in the paragraph contains the answer to the corresponding question. The dataset was converted into sentence pair classification by forming pairs between each question and each sentence in the corresponding context. The task is to determine whether the context sentence contains the answer to the question.

PIQA (Physical Interaction: Question Answering) PIQA introduces the task of physical commonsense reasoning and a corresponding benchmark dataset. The dataset contains several high-quality physics problems, with the goal to construct a resource that requires concrete physical reasoning. PIQA focuses on everyday situations with a preference for atypical solutions, inspired by instructables.com, which provides users with instructions on how to build, craft, bake, or manipulate objects using everyday materials.

B.2 Reading Comprehension Tasks

SQuAD v2 (Stanford Question Answering Dataset v2.0) Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage. SQuAD2.0 combines the questions in SQuAD1.1 with others unanswerable questions written adversarially by crowdworkers to look similar to answerable ones. To do well on SQuAD 2.0, systems must not only answer questions when possible, but also determine when no answer is supported by the paragraph and abstain from answering.

OpenBookQA OpenBookQA is an open-domain question-answering dataset that requires multi-step reasoning, where models must combine external scientific knowledge to answer commonsense-based questions. The dataset is designed to evaluate a model's ability to apply elementary scientific knowledge to novel situations.

B.3 Mathematical Reasoning Tasks

SVAMP (Simple Variants of Arithmetic Math Word Problems) SVAMP is a dataset of simple variants of arithmetic math word problems, specifically designed to test model robustness to structural variations in mathematical problem statements. The dataset systematically creates variations of existing math problems to evaluate whether models truly understand mathematical reasoning or rely on superficial pattern matching.

GSM8K (Grade School Math 8K) GSM8K consists of high-quality grade school math problems created by human problem writers. These problems take between 2 and 8 steps to solve, and solutions primarily involve performing a sequence of elementary calculations using basic arithmetic operations (+, -, /, *) to reach the final answer. GSM8K is a dataset of high-quality linguistically diverse grade school math word problems created to support the task of question answering on basic mathematical problems that require multi-step reasoning. A bright middle school student should be able to solve every problem.

MathQA MathQA is a mathematical question-answering dataset that encompasses various mathematical concepts and reasoning steps, requiring models to possess knowledge across multiple mathematical domains, including algebra, geometry, and probability. The dataset challenges models to apply mathematical reasoning in diverse problem-solving contexts.

B.4 Prompt Structure

We use a unified chat-style prompting framework consisting of a system message and a user message. Across tasks, we keep the same chat structure and demonstration concatenation strategy, while adapting the task instruction and answer format to the target benchmark. For mathematical reasoning

tasks, the system message requests step-by-step reasoning and enforces a fixed answer marker "#### <number>", where the final numerical answer must appear only after the marker. The user message concatenates k demonstrations. Each demonstration follows the format "Problem: ... Solution: ..." and, for mathematical reasoning tasks, ends with "#### <number>" as the final answer. We then append the test instance as "Problem: ... Solution:", after which the model completes the solution.

Chat Template Formatting For instruction-tuned models, we apply the chat template. We add explicit role delimiters. Examples include <|im_start|>system and <|im_start|>user. This matches the expected input format. It also improves reproducibility across backbones.

Prompt Example Figure 8 shows an illustrative GSM8K-style mathematical reasoning example after applying the chat template. It includes the system instruction, k demonstrations, and the final test query. For other benchmarks, we use the same chat structure but adapt the task instruction and answer format accordingly.

B.5 Hyperparameters for generation

We specify a small set of hyperparameters to control the in-context shot construction and the generation budget. For the DBSA configuration, block_size controls how many shots are grouped into one block, and we set it to 64. N_selected_blocks controls how many blocks are selected for each test instance, and we set it to 8. Pool_size controls the total size of the candidate shot pool, and we set it to 1024.

We use an entropy threshold to control entropy-based decisions in AdapShot, and we set a different threshold for each dataset. The threshold is 0.50 for cola, 0.53 for svamp, 0.68 for piqa, 0.65 for squad_v2, 0.62 for mathqa, 0.58 for gsm8k, and 0.70 for qnli.

For batched probing, we use a milestone schedule to increase the probing budget progressively. Tree_milestone_init is set to 4, which sets the initial milestone. Tree_milestone_step is set to 4, which sets the increment between milestones. Tree_max_milestones is set to 128, which caps the total number of milestones.

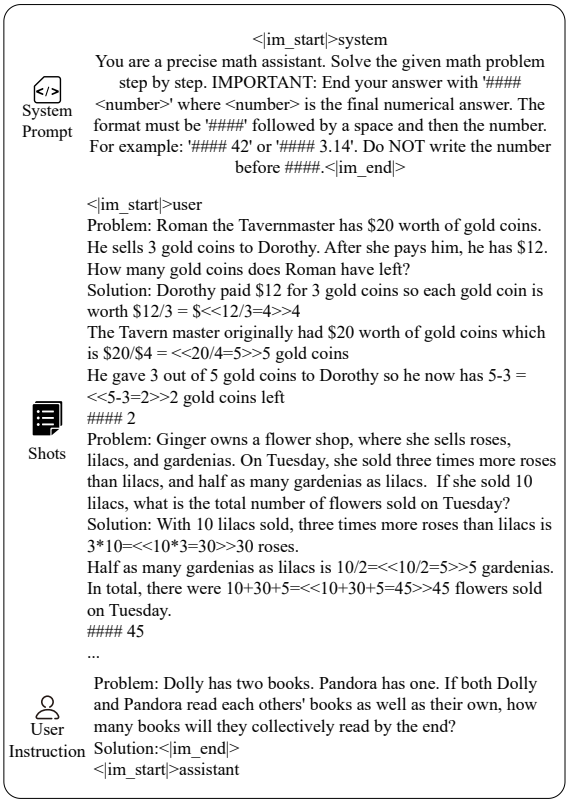

Figure 8: An illustrative GSM8K-style CoT prompt used in our experiments, shown after applying the chat template.

C More Analysis Details

C.1 Scaling with LLM Parameters:

To evaluate the scalability of AdapShot with respect to increased model capacity, we conducted experiments on Qwen2.5-14B and Qwen2.5-32B using the CoLA dataset. As shown in Figure 9, AdapShot's performance advantage amplifies significantly with model scale. On Qwen2.5-14B, AdapShot achieves 63.14% accuracy with an average shot count of 194.83, surpassing the fixed 256-shot baseline accuracy of 57.33%. When scaled to Qwen2.5-32B, AdapShot achieves 86.41% accuracy while reducing the average shot count to 60.74. The fixed 256-shot baseline reaches 67.91% accuracy on the 32B model but remains substantially lower than AdapShot. These results demonstrate that AdapShot's dynamic budget allocation achieves superior scalability by maximizing accuracy and computational efficiency simultaneously as model capacity grows, effectively activating internal knowledge through an optimal set of demonstrations.

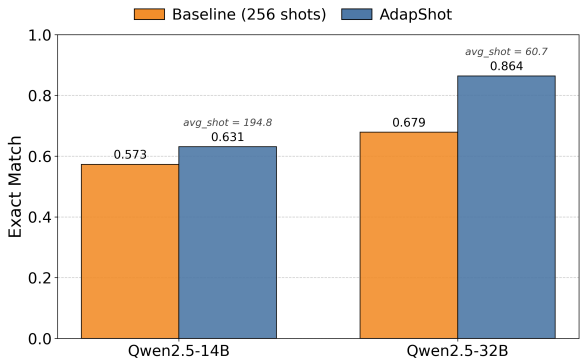

Figure 9: Comparison of accuracy and average context length (shot count) between AdapShot and the fixed Many-shot baseline across different model scales on the CoLA dataset. AdapShot demonstrates superior scalability, achieving higher accuracy with significantly fewer examples as model capacity increases.

C.2 Sensitivity Analysis of Entropy Threshold

Table 6: Sensitivity analysis of the entropy threshold τ on QNLI and GSM8K using Qwen2.5-7B. Bold values indicate the optimal performance points used in our main experiments.

QNLI (NLU)		GSM8K (Math)	
Threshold (τ)	**EM**	**Threshold (τ)**	**EM**
0.60	0.807	0.48	0.614
0.65	0.815	0.53	0.640
0.70	**0.825**	**0.58**	**0.649**
0.75	0.812	0.63	0.632
0.80	0.771	0.68	0.623

The entropy threshold τ (Eq. 2) serves as a critical hyperparameter governing the termination condition for our probe-based dynamic evaluation. To investigate the robustness of AdapShot against variations in this parameter, we conducted a sensitivity analysis on two distinct task types: NLU (QNLI) and Mathematical Reasoning (GSM8K).

Parameter Robustness and Stability: As demonstrated in Table 6, AdapShot exhibits significant stability across a broad range of threshold values. For the QNLI dataset, the accuracy fluctuation remains within a narrow margin ($< 1.8\%$) for $\tau \in [0.60, 0.75]$. Similarly, on the more complex GSM8K task, the model maintains high performance ($> 62\%$) across the interval $\tau \in [0.53, 0.68]$. This plateau of stability suggests that AdapShot is not overly sensitive to precise hyperparameter tuning. While the optimal τ shifts slightly depending on task difficulty (e.g., lower entropy/higher confidence is preferred for math reasoning), there exists a wide operational window where the model effectively balances noise suppression with sufficient information retrieval. Consequently, AdapShot can be reliably deployed without the need for exhaustive grid searches for every new dataset.

C.3 Performance Comparison on OpenPangu

Table 7: Performance of **openPangu-Embedded-1B-V1.1** under different in-context learning shot settings. Rows denote methods with corresponding shot numbers, and columns denote task benchmarks.

Method (Shots)	**ARC-Easy**	**CoLA**	**PIQA**	**QNLI**
Baseline (0-shot)	0.687	0.502	0.221	0.679
Baseline (8-shot)	0.679	0.512	0.246	0.699
Baseline (128-shot)	0.182	0.413	0.172	0.546
Baseline (256-shot)	0.039	0.034	0.103	0.147
AdapShot (Ours)	0.704	0.650	0.271	0.694

To further evaluate the generalizability of the proposed AdapShot method, we conduct additional experiments on the **openPangu-Embedded-1B-V1.1** model (Chen et al., 2025a). Table 7 presents the performance comparison across four benchmarks, including ARC-Easy, CoLA, PIQA, and QNLI. The results show that standard in-context learning baselines are sensitive to the number of demonstration examples. In particular, performance tends to degrade as the shot count increases, which may be attributed to context window limitations and the dilution of task-relevant information. In contrast, AdapShot consistently outperforms the fixed-shot baselines on most tasks and achieves the highest overall accuracy, without requiring manual shot selection.

Furthermore, Table 8 reports the computational efficiency of AdapShot relative to the many-shot baselines on the openPangu architecture. AdapShot yields notable inference speedups, ranging from 1.23 $\times$ to over 8 $\times$ depending on the dataset and baseline configuration. The efficiency gain becomes more evident as the baseline shot count grows; for instance, AdapShot achieves an average speedup of 6.04 $\times$ compared with 256-shot baselines. Overall, our method achieves an average speedup of 3.17 $\times$across all evaluated scenarios, indicating that AdapShot can improve both predictive performance and inference efficiency.

Table 8: Inference speedup of AdapShot relative to Many-shot baselines on openPangu-1B. Speedup is defined as the ratio of the baseline's average latency to AdapShot's average latency. Higher is better.

Dataset	vs. 8-shot	vs. 16-shot	vs. 32-shot	vs. 64-shot	vs. 128-shot	vs. 256-shot	Average
ARC-Easy	1.47×	1.38×	1.61×	2.03×	3.50×	3.75×	2.29×
GSM8K	3.88×	4.40×	4.52×	5.07×	6.73×	7.44×	5.34×
PIQA	1.71×	1.61×	1.60×	1.56×	2.13×	4.66×	2.21×
QNLI	1.23×	1.63×	1.47×	1.57×	2.78×	8.31×	2.83×
Average	**2.07**×	**2.26**×	**2.30**×	**2.56**×	**3.78**×	**6.04**×	**3.17**×

D Comparison with Context Compression Methods

Table 9: Comparison between LLMLingua-based context compression and AdapShot on QNLI and SQuAD v2. "Rate" denotes the compression rate.

Method	Rate	QNLI	SQuAD v2
Many-shot (256)	—	0.780	0.309
Many-shot (256)	0.25	0.675	0.074
Many-shot (256)	0.50	0.586	0.059
Many-shot (256)	0.75	0.473	0.074
Many-shot (512)	—	0.768	0.236
Many-shot (512)	0.25	0.631	0.123
Many-shot (512)	0.50	0.507	0.148
Many-shot (512)	0.75	0.429	0.315
AdapShot (Ours)	—	**0.825**	**0.465**

As shown in Table 9, applying LLMLingua (Jiang et al., 2023) to many-shot ICL tends to degrade performance across both shot counts and compression rates, though marginal improvements are occasionally observed in certain settings. We attribute this to LLMLingua's token-level compression disrupting the reasoning structure within demonstrations. In contrast, AdapShot operates at the shot level, preserving the completeness of each demonstration while adaptively reducing redundancy.

Table 10: Comparison between BM25 and BGE (dense retrieval).

Method	Retriever	QNLI	PIQA	SQuAD v2
Many-shot (256)	BM25	0.667	0.379	0.418
Many-shot (256)	BGE	0.684	0.404	0.439
Many-shot (512)	BM25	0.719	0.388	0.126
Many-shot (512)	BGE	0.719	0.439	0.298
AdapShot (Ours)	SAKVC	**0.825**	**0.599**	**0.465**

E Comparison with Dense Retrieval Methods

As shown in Table 10, replacing BM25 with BGE yields moderate improvements, while AdapShot generally outperforms both retrieval strategies. This suggests that a stronger retriever alone does not resolve the fundamental question of *how many shots to use*, which is precisely the core contribution of AdapShot.