

Long-Context Demonstration Selection Using State Space Models

Ziniu Zhang

Northeastern University
zhang.zini@northeastern.edu

Zhenshuo Zhang

Northeastern University
zhang.zhens@northeastern.edu

Ruoxuan Xiong

Emory University
ruoxuan.xiong@emory.edu

Gene Cooperman

Northeastern University
gene@ccs.neu.edu

Hongyang R. Zhang

Northeastern University
ho.zhang@northeastern.edu

Abstract

We study the problem of demonstration selection, which involves selecting a subset of examples for pre-pending to a query to a language model. This problem is closely related to in-context learning and model inference. Since the inference cost of a transformer model scales quadratically with sequence length, the selection problem becomes especially challenging in a long-context scenario. In this paper, we tackle this problem by building on state space models (SSMs), which has linear inference time given the input embeddings. Our approach works in two stages. First, we learn a small set of SSMs through a distillation procedure — given a (trained) transformer model, we partition all the layers into consecutive (layer) groups — for each group, we learn one SSM to mimic the input-output behavior, thereby capturing structural similarity in adjacent layers. Second, we map the SSM outputs to a small set of tokens, and apply them for downstream demonstration selection. We run extensive experiments in both synthetic and real-world datasets to validate this approach. We show that the SSMs trained from distillation incur a relative approximation error of less than 0.7%. As for downstream demonstration selection, we find that on several text classification and reasoning tasks, our approach reduces FLOPs by $14.2\times$ and improves accuracy by 6.48% relative to baseline selection methods.

1 Introduction

Language models and autonomous agents increasingly rely on long-context prompting to adapt to new tasks and environments at inference time (Liu et al., 2023). A commonly used approach is to condition the model on demonstrations drawn from historical trajectories (Ou et al., 2024), past interactions (Wang et al., 2025), or accumulated memory states (Ouyang et al., 2026). Specifically, given a query, the goal is to select a subset of k demonstrations from a large candidate pool of size n that im-

proves downstream prediction quality. This problem is known as *demonstration selection*, which has become increasingly important in in-context learning and agent workflows (Min et al., 2022; Zhang et al., 2025b).

A major challenge in selecting demonstrations is that the usefulness of one demonstration often depends on which other demonstrations appear with it in the prompt. As a result, demonstration selection is not purely a retrieval problem: evaluating candidate subsets often requires repeatedly running the language model on many different prompts (Zhang et al., 2025b). For transformer models, the cost of processing long prompts grows quadratically with the total context length T , where T scales with both the number and length of demonstrations. This creates a major computational bottleneck for long-context demonstration selection, since exhaustively evaluating many candidate subsets quickly becomes infeasible as k increases.

This challenge connects demonstration selection to a broader problem of efficient long-context inference. Existing approaches largely fall into three categories. First, state space models such as Mamba (Gu and Dao, 2024) and related hybrid architectures (Ren et al., 2025; Oncescu et al., 2025) achieve efficient sequence modeling, but typically require retraining the underlying language model. Second, training-free methods accelerate inference by restricting attention to local windows or selected tokens (Xiao et al., 2024, 2025a). Third, existing methods (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024) still require processing the entire original prompt, which is computationally expensive for repeatedly evaluating multiple candidate subsets.

In this paper, we build on state space models to scale up demonstration selection. The key idea is to avoid repeatedly processing long demonstration prompts with full transformer attention. Instead, we train a group of state space models to

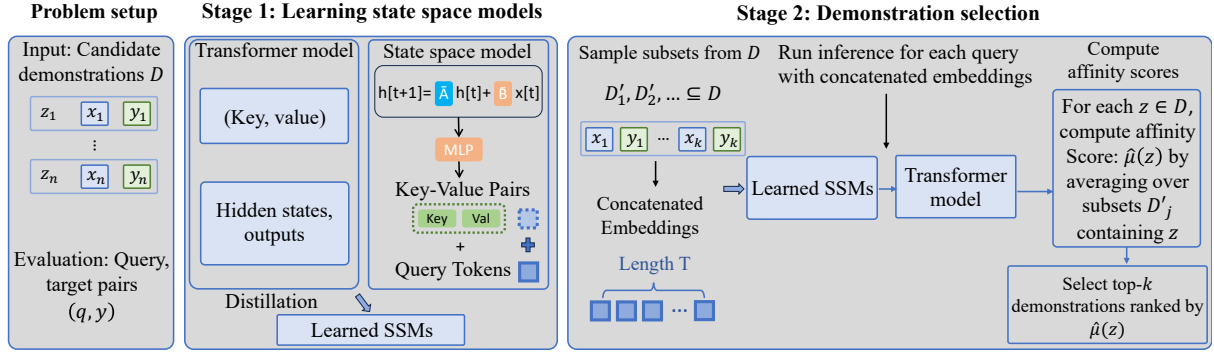


Figure 1: Overview of our approach: Given a set of candidate demonstrations and query-target pairs from an unknown distribution, our approach first trains the SSMs through key, value and output distillation from a fixed transformer, producing the trained parameter of the $\bar{B}$ matrix in SSM and parameter of MLP. Then, we use the embeddings for downstream demonstration selection. This works by sampling M candidate subsets, concatenates each subset, and evaluates the embeddings on validation queries. It then aggregates the subset losses into per-demonstration affinity scores and selects the top- k demonstrations.

map each candidate subset’s concatenated demonstrations into a small set of tokens. These tokens are computed once per subset and reused across all queries evaluated with that subset. This substantially reduces the cost of evaluating many candidate demonstration subsets. Our algorithm consists of three main components.

First, we develop a state space model based procedure, which process tokens sequentially in linear time. A direct approach using a single SSM to approximate all transformer layers simultaneously would require learning an extremely high-dimensional mapping. To address this challenge, we partition the transformer layers into disjoint groups of consecutive layers and assign one SSM to each group. Each SSM scans the embeddings of the long demonstration prompt into a compact hidden representation, which is then mapped into a small set of key-value states. The SSMs process in parallel. The output key-value states are then combined together as the tokens that transformers can process directly. These tokens serve as a substitute for the original long prompt during inference.

Second, building on this procedure, we design an affinity-based demonstration selection algorithm. This algorithm measures how useful a demonstration is when combined with other demonstrations in the same prompt. To do this, we repeatedly sample candidate subsets and evaluate their predictive performance using the representation produced by our approach. We then estimate the contribution of each demonstration based on how subsets containing that demonstration perform.

Third, we provide empirical analyses of the pro-

posed framework. We find that our approach preserves the predictive distribution of the original model with relative error below 0.6%. Across classification and reasoning benchmarks, our approach reduces FLOPs by up to $14.2\times$ and improves downstream accuracy by 6.48% over leading baselines.

In summary, we introduce a framework for scalable demonstration selection in long-context settings, with three main contributions. First, we build SSMs that map long demonstration prompts into a small set of reusable tokens. Second, we develop an affinity-based subset selection algorithm that leverages this representation to evaluate many candidate demonstration subsets. Third, we systematically validate the proposed framework through both theoretical analysis and empirical experiments on synthetic, classification, and reasoning benchmarks. The code to reproduce our experiments is at <https://github.com/VirtuosoResearch/Long-context-demonstration-selection>.

2 Preliminaries

We study demonstration selection in constructing a prompt. Let D denote the set of candidate demonstrations of size n . Let $(q, y) \sim \mathcal{P}$ denote a query-target pair drawn from an unknown distribution. We select a subset $D' \subseteq D$ containing k demonstrations with a fixed ordering for every query. Given a model f_W and a loss function $\ell : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ supported on an input domain $\mathcal{X}$ and output domain $\mathcal{Y}$, the *demonstration (subset) selection problem* is defined as the following minimization:

$$\min_{D' \subseteq D: |D'|=k} \mathbb{E}_{(q,y) \sim \mathcal{P}} [\ell(f_W(D', q), y)].$$

For the selected demonstrations D' , we concatenated them as the input demonstration prompt. Let $X = [x_1, \dots, x_T]^\top \in \mathbb{R}^{T \times H}$ denote the token embeddings of the prompt, where T is the total number of tokens and H is the embedding dimension. Since T increases with both the number and length of demonstrations, long-context prompting can produce very large input sequences. In a transformer model, the computational cost of processing the prompt grows quadratically with T because self-attention computes pairwise interactions between all tokens. As a result, evaluating the inference outcome for a large T is computationally expensive.

A natural approach for processing long prompts is to use state space models (SSMs), which summarize the input sequence through recurrent updates. Given the prompt embeddings X , a discrete SSM updates its state at each token position t as

$$h_t = \bar{A}h_{t-1} + \bar{B}x_t, \quad (1)$$

where $h_t \in \mathbb{R}^N$ is the N -dimensional state, $\bar{A} \in \mathbb{R}^{N \times N}$ is the state-transition matrix, and $\bar{B} \in \mathbb{R}^{N \times H}$ maps the token embedding x_t into the state space. The SSM processes each token once and has linear complexity in the sequence length T (Gu et al., 2022).

Previous work has shown that the HiPPO-LegS matrix provides a structured construction of the state-transition matrix by maintaining an online polynomial projection of the input history (Gu et al., 2020; Gu and Dao, 2024). The state approximates the coefficients of past inputs under a scaled Legendre basis. Lower-order coefficients capture the main structure of the sequence history, and higher-order coefficients represent finer variations. Thus, a finite-dimensional HiPPO state provides a compact summary of a long input sequence.

In the in-context learning setting, we hypothesize that task-relevant information in a large demonstration set can be preserved by low-order Legendre basis components of the HiPPO matrix. The SSM maintains these components while scanning the complete input once without discarding tokens. Thus, it can replace a transformer model’s full-prefix encoding and reduce the inference complexity from $O(T^2)$ to $O(T)$.

Motivated by these existing results, a natural question is whether one could adapt SSMs to tackle the demonstration subset selection problem.

- First, how can we design the SSM for the in-context learning setting?

- Second, how can we utilize the SSM for demonstration selection?

In the next section, we design algorithms to answer each of the above questions, respectively.

3 Our Approach

We build SSMs for efficient long-context inference. First, we present the architecture of the SSMs with theoretical and empirical evidence supporting our hypothesis. We also introduce a distillation procedure for training the SSMs. Then, we propose a demonstration selection method on the embeddings and use controlled synthetic experiments to evaluate its affinity estimates.

3.1 Learning the state space models

We use SSMs with the HiPPO-LegS matrix to map long demonstration prompts while preserving the dominant low-order structure of the sequence. In a transformer, the prompt is first processed to construct a layer-wise key-value (KV) cache, where each transformer layer stores key and value representations for all prompt tokens. These cached representations are later used during decoding.

Instead of computing the exact KV cache for the full prompt, we use SSMs to process it. The SSM sequentially scans the prompt and summarizes the long sequence into a compact hidden representation. We partition the m layers of transformer model f into g disjoint groups of consecutive layers, to yield a lower-dimensional target for each group. Each group is assigned an independent SSM that predicts the KV states only for its corresponding group of layers.

Specifically, for each layer group G_i for $i = 1, \dots, g$, the SSM processes the prompt embeddings X and produces a final hidden state $h_T^{(i)} \in \mathbb{R}^N$. A multilayer perceptron $f_W^{(i)}$ then projects $h_T^{(i)}$ into a separate pair of key-value states for every transformer layer in the group:

$$\left\{ (\tilde{K}^{(l)}, \tilde{V}^{(l)}) \right\}_{l \in G_i} = f_W^{(i)} \left(h_T^{(i)} \right).$$

For each transformer layer l , the original KV states have shape $T \times n_{kv} \times d$, while the projected states have shape $n_v \times n_{kv} \times d$, where n_{kv} is the number of key-value heads and d is the dimension of each head. The KV states generated from the outputs of all g SSMs are aligned at the same n_v token positions and assembled across layer groups to form the layer-wise KV cache of these n_v tokens. The

projection therefore preserves the key-value head and head-dimension axes while mapping T tokens to n_v tokens.

To analyze the proposed architecture, we first run the transformer on the complete input prompt using a standard full-attention forward pass and record the resulting key-value states at every layer as the teacher KV cache. We represent the exact teacher KV cache of each layer group G_i by a matrix $C^{(i)}$. This matrix stacks the key and value states from all layers and attention heads in the group, and retains the T token positions as its rows.

Proposition 3.1. *Let Π_N denote the projection along the token dimension onto the first N discrete HiPPO-LegS basis functions. The matrix $\Pi_N C^{(i)}$ is the part of the grouped cache represented, and $C^{(i)} - \Pi_N C^{(i)}$ is the information omitted by this representation.*

For each layer group G_i , let $c_r^{(i)}$ denote the vector of the r -th normalized discrete HiPPO-LegS coefficients of the teacher cache $C^{(i)}$. For any $N < T$ and $s > 0$,

$$\sum_{i=1}^g \left\| C^{(i)} - \Pi_N C^{(i)} \right\|_F^2 \leq \sum_{i=1}^g \sum_{r=1}^T \frac{r^{2s} \left\| c_r^{(i)} \right\|_2^2}{(N+1)^{2s}}.$$

This shows that the error stems from Legendre components omitted by the first N modes. When the coefficient energy is concentrated at low frequencies, this tail is small. Increasing the SSM state dimension N retains more components and further reduces the approximation error. The proof is provided in Appendix A.1.

We empirically verify this low-order dominance on Qwen-7B-Instruct and Llama-3-8B-Instruct models with $G=4$ layer groups and $k=50$ demonstrations. The rapid coefficient decay in Figure 2 indicates that a finite-dimensional HiPPO state can retain most of the information in the KV cache. Based on this observation, the SSMs process the entire T -token input using recurrent scans with $O(T)$ cost for a fixed number of groups. The transformer then processes the resulting n_v tokens with a cost of $O(n_v^2)$. Since n_v is fixed and $n_v \ll T$, the total prefix-processing complexity is $O(T + n_v^2) = O(T)$. In the next part, we introduce a two-stage distillation method to train the SSMs to produce KV states that preserve the original transformer function.

Key, value, and output distillation. To train the SSMs, we design a distillation procedure. First, we

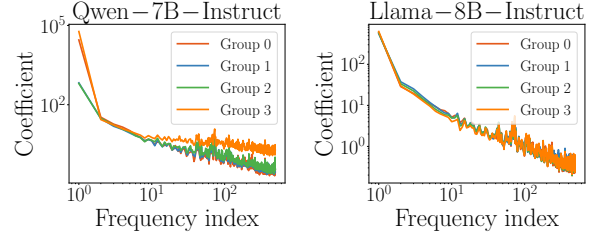


Figure 2: Mean squared discrete Legendre coefficients of KV caches on Qwen-7B-Instruct and Llama-3-8B-Instruct with $G=4$ layer groups and $k=50$ demonstrations. The coefficients decay rapidly across groups, and the KV cache is dominated by low-order HiPPO-LegS components.

align the generated tokens with a valid KV space. Given the m -layer model f_W , we extract its exact teacher KV cache $\{K^{(l)}, V^{(l)}\}_{l=1}^m$. To resolve the length mismatch, we apply pooling to the T teacher states, producing n_v pooled targets $\text{Pool}(K)$ and $\text{Pool}(V)$. For the i -th layer group, we keep the HiPPO-LegS transition matrix fixed and train only the SSM input projection $\bar{B}^{(i)}$ and the parameters of the KV projection MLP $f^{(i)}$. We denote all trainable parameters by $\Theta = \{\bar{B}^{(i)}, W^{(i)}\}_{i=1}^g$. The objective is:

$$\frac{1}{2m} \sum_{l=1}^m \sum_{i=1}^{n_v} \left(\left(1 - \cos \left(\tilde{K}_i^{(l)}, \text{Pool}(K^{(l)})_i \right) \right) + \left(1 - \cos \left(\tilde{V}_i^{(l)}, \text{Pool}(V^{(l)})_i \right) \right) \right).$$

Next, we train the SSM using joint supervision of outputs and intermediate states to preserve predictive accuracy and positional structure. For a query q , let $\Pr[y | \tilde{K}, \tilde{V}, q]$ and $\Pr[y | X, q]$ denote the output distributions of the LLM conditioned on the prefixes, respectively. We additionally align the corresponding query hidden states $\psi_1^{(l)}$ and $\psi_2^{(l)}$ across all layers. The entire loss objective, denoted by $\hat{L}(\Theta)$, is:

$$\lambda_1 D_{\text{KL}}(P_1 \| P_2) + \frac{\lambda_2}{m} \sum_{l=1}^m \left(1 - \cos \left(\psi_1^{(l)}, \psi_2^{(l)} \right) \right).$$

where λ_1 and λ_2 adjust the relative weight of the respective terms. The entire procedure is summarized in Algorithm 1.

To validate the effectiveness of the above distillation procedure, we evaluate Algorithm 1 on multiple tasks using k demonstrations from 50 to 75. As shown in Table 1, the relative error between the logits from the full prefix p and tokens $\hat{p}$ remains strictly below 0.7% across all settings. This

Table 1: Relative error of the predictive output between the full demonstration prefix p and the tokens $\hat{p}$. We vary the number of demonstrations on multiple tasks. Lower values indicate higher fidelity to the original model output. We run three times to compute the mean and standard deviations.

Model	# Demonstrations	SST-2	Poem Sentiment	Coin Flip	Modular Addition
Qwen-1.5B	$k = 50$	$2.5_{\pm 0.1} \times 10^{-3}$	$1.3_{\pm 0.1} \times 10^{-3}$	$1.9_{\pm 0.5} \times 10^{-3}$	$1.3_{\pm 0.1} \times 10^{-3}$
	$k = 75$	$2.8_{\pm 0.2} \times 10^{-3}$	$1.9_{\pm 0.2} \times 10^{-3}$	$2.1_{\pm 0.2} \times 10^{-3}$	$1.8_{\pm 0.4} \times 10^{-3}$
Qwen-3B	$k = 50$	$6.1_{\pm 0.2} \times 10^{-3}$	$2.3_{\pm 0.3} \times 10^{-3}$	$2.1_{\pm 0.8} \times 10^{-3}$	$2.5_{\pm 0.3} \times 10^{-3}$
	$k = 75$	$6.3_{\pm 0.2} \times 10^{-3}$	$2.9_{\pm 0.2} \times 10^{-3}$	$4.1_{\pm 1.1} \times 10^{-3}$	$2.1_{\pm 0.7} \times 10^{-3}$
Llama-8B	$k = 50$	$2.3_{\pm 0.1} \times 10^{-3}$	$2.7_{\pm 0.1} \times 10^{-3}$	$1.5_{\pm 0.5} \times 10^{-3}$	$3.8_{\pm 1.0} \times 10^{-3}$
	$k = 75$	$2.5_{\pm 0.2} \times 10^{-3}$	$2.9_{\pm 0.2} \times 10^{-3}$	$1.9_{\pm 0.2} \times 10^{-3}$	$4.8_{\pm 0.9} \times 10^{-3}$

Algorithm 1 Learning SSMs via Key, Value, and Output Distillation

Input: Embedding and query pairs $\{X, q\}$, parameter initialization $\Theta = \{\bar{B}^{(i)}, W^{(i)}\}_{i=1}^g$

Require: Groups $\{G_i\}_{i=1}^g$, g SSMs each with a fixed $\bar{A}$ and variable weight matrix $\bar{B}^{(i)}$ and MLPs $\{f_W^{(i)}\}_{i=1}^g$, an m -layer transformer, n_v , parameters λ_1, λ_2 , and learning rate η

Output: Trained parameters $\hat{\Theta}$

```

1: for each  $(X, q)$  do
2:   Extract  $\{K^{(l)}, V^{(l)}\}_{l=1}^m$  from transformer
3:    $T \leftarrow \text{Length of } X$ 
4:   for  $i \in \{1, \dots, g\}$  do
5:      $\{(\tilde{K}^{(l)}, \tilde{V}^{(l)})\}_{l \in G_i} \leftarrow f_W^{(i)}(h_T^{(i)}(X))$ 
6:   end for
7:   for  $i \in \{1, \dots, n_v\}, l \in \{1, \dots, m\}$  do
8:      $\kappa_1^{(i,l)} \leftarrow 1 - \cos(\tilde{K}_i^{(l)}, \text{Pool}(K^{(l)})_i)$ 
9:      $\kappa_2^{(i,l)} \leftarrow 1 - \cos(\tilde{V}_i^{(l)}, \text{Pool}(V^{(l)})_i)$ 
10:  end for
11:   $\Theta \leftarrow \Theta - \eta \nabla_{\Theta} \left( \frac{1}{2m} \sum_{l=1}^m \sum_{i=1}^{n_v} (\kappa_1^{(i,l)} + \kappa_2^{(i,l)}) \right)$  // Key-value distillation
12: end for
13: for each  $(X, q)$  do
14:   for  $i \in \{1, \dots, g\}$  do
15:      $\{(\tilde{K}^{(l)}, \tilde{V}^{(l)})\}_{l \in G_i} \leftarrow f_W^{(i)}(h_T^{(i)}(X))$ 
16:   end for
17:    $(P_1, \{\psi_1^{(l)}\}) \leftarrow f((\tilde{K}, \tilde{V}), q)$ 
18:    $(P_2, \{\psi_2^{(l)}\}) \leftarrow f(X, q)$ 
19:    $L \leftarrow \lambda_1 D_{\text{KL}}(P_1 \| P_2) + \frac{\lambda_2}{m} \sum_{l=1}^m (1 - \cos(\psi_1^{(l)}, \psi_2^{(l)}))$ 
20:    $\Theta \leftarrow \Theta - \eta \nabla_{\Theta} L$  // Output distillation
21: end for
22: return  $\hat{\Theta}$ 

```

confirms that Algorithm 1 robustly preserves the original predictive behavior. We defer the omitted result where $k = 25$ to Table 7 in Appendix B.3.

3.2 Demonstration Selection Using SSM Outputs

We now utilize Algorithm 1 for demonstration selection. We first introduce a randomized estimator that quantifies the marginal utility of each candidate via subset sampling. We define the affinity score, $\mu(z)$, of a demonstration example $z \in \mathcal{D}$ as

$$-\mathbb{E}_{D' \subseteq D, z \in D'} [\mathbb{E}_{(q,y) \sim \mathcal{P}} [\ell(f_W(D', q), y)]].$$

Thus, $\mu(z)$ is the negative expected inference loss averaged over subsets of size k that contain z .

To estimate $\mu(z)$, we uniformly sample M subsets $\{D'_1, \dots, D'_M\}$ from D sized k . Then, we estimate the affinity, denoted by $\hat{\mu}(z)$, as

$$-\frac{\sum_{i=1}^M \mathbf{1}_{z \in D'_i} \sum_{(q,y) \in \mathcal{V}} \ell(f_W(D'_i, q), y)}{\sum_{i=1}^M \mathbf{1}_{z \in D'_i}}.$$

Proposition A.1 shows that $M = O(n/(k\epsilon^2))$ sampled subsets control the uniform estimation error by ϵ . We rank the candidates by $\hat{\mu}(z)$ and select the k demonstrations with the highest scores.

Taken together, we summarize the entire procedure in Algorithm 2.

A case study of linear functions. To evaluate our algorithm, we adopt the synthetic linear regression setting from Garg et al. (2022) using a two-layer transformer. Both demonstrations and queries are generated via $y_i = \langle \beta^{(c)}, x_i \rangle + \epsilon_i$, where each input $x_i \in \mathbb{R}^{20}$ is independently sampled from the standard Gaussian distribution $\mathcal{N}(0, \text{Id}_{20})$. To simulate multiple queries clustering around shared underlying tasks, each coefficient vector $\beta^{(c)}$ is drawn randomly from a set of mutually orthogonal anchors, ensuring strict task decoupling. Finally, we inject independent Gaussian noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ to prevent the model from trivially converging to an exact solution after observing 20 examples.

Algorithm 2 Demonstration Selection Using SSM Outputs

Input: Candidate set D , validation set $\mathcal{V}$, subset size k , number of sampled subsets M , parameter initialization Θ

Require: Transformer model f , groups $\{G_i\}_{i=1}^g$, HiPPO-LegS matrix $\bar{A}$

Output: k selected demonstrations

```

1:  $\hat{\Theta} \leftarrow$  Algorithm 1
2: for  $j \in \{1, \dots, M\}$  do
3:   Sample  $D'_j \subseteq D$  of size  $k$ 
4:    $X \leftarrow$  Embeddings of  $D'_j$ 
5:    $T \leftarrow$  Length of  $X$ 
6:   for  $i \in \{1, \dots, g\}$  do
7:      $\{(\tilde{K}^{(i)}, \tilde{V}^{(i)})\}_{l \in G_i} \leftarrow f_W^{(i)}(h_T(X))$ 
8:   end for
9:    $\tilde{C} \leftarrow \{(\tilde{K}^{(i)}, \tilde{V}^{(i)})\}_{l=1}^m$ 
10:   $\hat{L}_j \leftarrow \frac{1}{|\mathcal{V}|} \sum_{(q,y) \in \mathcal{V}} \ell(f(\tilde{C}, q), y)$ 
11: end for
12: for  $z \in D$  do
13:   $\hat{\mu}(z) \leftarrow -\frac{\sum_{j=1}^M \mathbf{1}_{\{z \in D'_j\}} \hat{L}_j}{\sum_{j=1}^M \mathbf{1}_{\{z \in D'_j\}}}$ 
14: end for
15: return Top- $k$  demonstrations according to  $\hat{\mu}(z)$  for all  $z \in D$ 

```

We evaluate the predictive performance of our proposed algorithm against two baseline strategies: random demonstration selection (random- k) and embedding similarity-based selection (top- k). As shown in Figure 3a, our method achieves the lowest estimation error in the linear setting. This predictive advantage robustly extends to the non-linear setting, which is a two-layer ReLU network (Figure 3b). Note that the model is trained with 40 data points as demonstrations, thus the error increases after the number of samples reaches 40.

By leveraging the SSMs during subset evaluation, our approach reduces FLOP overhead by $15.7\times$ compared to naive inference (Figure 3), ensuring high scalability for massive contexts.

4 Experiments

We evaluate our method by addressing the following three questions. First, how accurately and efficiently does Algorithm 1 approximate full-context inference compared with existing methods? Second, how effective is Algorithm 2 in terms of the accuracy and computational cost of downstream

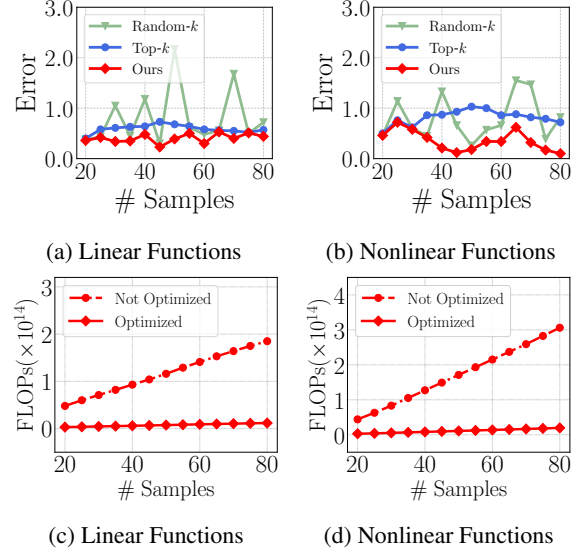


Figure 3: Evaluation of our approach for linear and nonlinear functions. The top panel provides comparison with random- k and top- k selection methods. The lower panel illustrates the computational costs relative to full inference (denoted as not optimized).

demonstration selection? Third, how well does the learned SSM scale to longer contexts and transfer across different domains?

Through extensive experiments in both text and reasoning datasets, we provide positive answers to all three questions above. Finally, we conduct detailed ablation studies to validate the design of our approach.

4.1 Experiment Setup

Datasets. We evaluate our approach across a diverse set of downstream tasks with different output formats. SST-2 and Poem Sentiment require categorical sentiment labels. Coin Flip and Edge Existence require binary decisions. Modular Addition requires numerical answers.

We further evaluate the accuracy of Algorithm 1 on MMLU and GSM8K. MMLU covers knowledge-intensive multiple-choice questions across a broad range of academic and professional subjects, and GSM8K requires the model to generate multi-step reasoning processes and final numerical answers. Thus, our evaluation covers both short-output classification tasks and longer-form chain-of-thought reasoning tasks.

Baselines. Regarding efficient inference methods, we consider full inference (dense), streaming LLM (Xiao et al., 2024), LM-infinite (Han et al., 2024), Gist tokens (Mu et al., 2023), in-context

Table 2: We report the relative error, computational cost (FLOPs), and memory usage on MMLU, Addition datasets, and GSM8K. The relative error is computed against actual inference results.

Method	Window Length	Error	MMLU		Modular Addition			Error	GSM8K	
			FLOPs	Memory	Error	FLOPs	Memory		FLOPs	Memory
Dense	-	-	$6.53e^{15}$	17.31G	-	$1.05e^{15}$	13.65G	-	$2.59e^{15}$	15.75G
StreamingLLM	64	1.1%	$7.75e^{13}$	15.90G	9.2%	$1.90e^{13}$	12.95G	5.6%	$1.26e^{14}$	14.06G
LM-Infinite	64	1.2%	$7.75e^{13}$	15.91G	10.9%	$1.91e^{13}$	12.97G	5.1%	$1.27e^{14}$	14.06G
Gist Tokens	64	7.2%	$1.66e^{15}$	14.45G	2.9%	$7.50e^{14}$	13.72G	40.4%	$2.37e^{14}$	14.43G
ICAE	64	16.5%	$3.79e^{14}$	17.02G	1.4%	$1.36e^{14}$	14.27G	0.5%	$3.33e^{14}$	14.87G
DuoAttention	64	2.7%	$8.03e^{13}$	16.24G	7.7%	$1.93e^{13}$	13.25G	4.9%	$1.31e^{14}$	14.25G
I2CL	-	14.3%	$2.62e^{14}$	15.89G	0.7%	$8.29e^{14}$	13.14G	11.7%	$8.56e^{14}$	13.86G
BSA	~ 400	3.1%	$7.86e^{13}$	16.53G	1.8%	$2.31e^{13}$	13.36G	2.1%	$1.51e^{14}$	14.31G
Algorithm 1	16	0.2%	$3.49e^{13}$	16.76G	0.2%	$2.12e^{13}$	13.27G	0.1%	$8.67e^{13}$	14.44G

Table 3: We report the test accuracy (%) and the computational cost (FLOPs) across five different kinds of datasets. We compare our approach with several existing demonstration selection methods, using 50 demonstrations. We run each experiment with three random seeds to report the standard deviations.

Category Dataset Metric	Sentiment Analysis				Math Reasoning				Graph Reasoning	
	SST-2		Poem Sentiment		Modular Addition		Coin Flip		Edge Existence	
	Acc.	FLOPs	Acc.	FLOPs	Acc.	FLOPs	Acc.	FLOPs	Acc.	FLOPs
Random- k	$74.7_{\pm 4.5}$	$1.92e^{14}$	$57.3_{\pm 4.5}$	$4.17e^{14}$	$53.2_{\pm 1.7}$	$1.68e^{14}$	$55.3_{\pm 7.5}$	$2.34e^{14}$	$48.9_{\pm 3.6}$	$6.55e^{14}$
BM25	$77.8_{\pm 1.9}$	$1.92e^{14}$	$55.7_{\pm 2.8}$	$4.17e^{14}$	$58.3_{\pm 2.5}$	$1.68e^{14}$	$42.3_{\pm 1.1}$	$2.34e^{14}$	$51.2_{\pm 0.6}$	$6.55e^{14}$
Top- k	$88.9_{\pm 1.2}$	$1.92e^{14}$	$63.8_{\pm 1.5}$	$4.17e^{14}$	$60.3_{\pm 1.1}$	$1.68e^{14}$	$45.5_{\pm 1.4}$	$2.34e^{14}$	$58.3_{\pm 1.1}$	$6.55e^{14}$
Top- k + Alg. 1	$88.7_{\pm 0.8}$	$9.14e^{12}$	$63.4_{\pm 1.0}$	$2.17e^{13}$	$60.3_{\pm 1.1}$	$7.51e^{12}$	$45.4_{\pm 0.8}$	$9.56e^{12}$	$58.0_{\pm 1.3}$	$2.72e^{13}$
GRADCE	$83.5_{\pm 0.7}$	$6.71e^{13}$	$70.3_{\pm 0.9}$	$1.47e^{14}$	$64.5_{\pm 2.1}$	$5.80e^{13}$	$68.2_{\pm 0.8}$	$7.80e^{13}$	$65.4_{\pm 1.3}$	$2.61e^{14}$
BRIDGE	$89.9_{\pm 0.3}$	$5.73e^{15}$	$73.3_{\pm 0.8}$	$7.12e^{15}$	$70.8_{\pm 1.3}$	$7.01e^{15}$	$82.3_{\pm 0.6}$	$1.00e^{16}$	$76.3_{\pm 0.4}$	$2.84e^{16}$
BRIDGE + Alg. 1	$89.9_{\pm 0.5}$	$4.93e^{14}$	$73.0_{\pm 0.9}$	$5.70e^{14}$	$70.1_{\pm 0.5}$	$5.49e^{14}$	$82.2_{\pm 0.8}$	$7.22e^{14}$	$75.8_{\pm 0.8}$	$2.08e^{15}$
Algorithm 2	$95.9_{\pm 0.7}$	$4.52e^{14}$	$78.6_{\pm 1.1}$	$5.23e^{14}$	$81.8_{\pm 1.2}$	$5.08e^{14}$	$85.3_{\pm 1.2}$	$6.61e^{14}$	$83.6_{\pm 0.6}$	$1.81e^{15}$

autoencoder (ICAE) (Ge et al., 2024), block-sparse attention (BSA) (Xiao et al., 2025a), and DuoAttention (Xiao et al., 2025b) as baselines.

Then, we consider baselines that use different measures to rank the demonstrations. These include random selection (Random- k), selection based on probabilistic relevance rankings (BM25), embedding similarities (top- k), Bayesian refinement, and iterative demonstration generation, for example (BRIDGE) (Wan et al., 2025), cross entropy-based selection with gradient estimation (GRADCE) (Zhang et al., 2025b). We defer a detailed description of datasets, models, and baselines to Appendix B.1.

Finally, all other implementation details are deferred to Appendix B.

4.2 Experiment Results

Accuracy and efficiency of Algorithm 1. We evaluate our approach using the Qwen-3B model with 50 demonstrations for MMLU and Modular Addition and 20 for GSM8K. We compute the relative error of the output logits, the computational cost (measured in the number of FLOPs), and the peak memory usage for each baseline.

As shown in Table 2, Algorithm 1 reduces the prefill computation by approximately two orders of magnitude compared to Dense inference. By compressing the global prefix into 16 tokens, our method maintains a relative output error below 0.2% across all tasks. Compared to the baseline BSA, our method yields an average error reduction of 2.17% and a 35.5% decrease in FLOPs, with minimal memory overhead. We defer the results on the remaining datasets and those using Qwen-1.5B to Table 10 and Table 11.

Accuracy and efficiency of Algorithm 2. We integrate our approach into existing selection frameworks to assess its compatibility and evaluate across five datasets. We evaluate the accuracy and FLOPs of the whole pipeline in Algorithm 2.

Using the Qwen-3B model with 50 demonstrations, Table 3 shows that our approach accelerates existing pipelines, with an average accuracy drop of 0.22% across other baselines. Compared to the baseline, BRIDGE, our approach yields an average accuracy improvement of 6.48% while requiring much fewer FLOPs. We analyze the efficiency of Algorithm 2 in Appendix B.3.

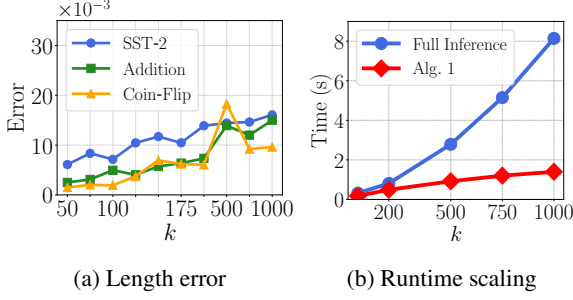


Figure 4: Figure 4a reports the relative output error as the number of demonstrations k increases. Figure 4b reports the wall-clock inference time as k increases.

Length and task generalization. To evaluate length generalization, we test the module, trained solely on 50 demonstrations, on extended contexts containing up to 1000 demonstrations. As shown in Figure 4a, our method maintains stable performance without additional fine-tuning. When increasing the number of demonstrations to 1000, the relative output error remains below 1.8% across all datasets.

We further compare the wall-clock time for processing a single query with full inference and our approach. As shown in Table 4 on the left, full inference retains the quadratic growth as the context becomes longer, while our approach scales approximately linearly.

Table 4: Left: cross-domain transfer error between training and test tasks. Right: test loss under different HiPPO state dimensions and numbers of layer groups.

Test	Train				HiPPO Dim.	Groups				
	SST2	MMLU	Add	GSM		1	2	4	8	16
SST2	0.3%	0.3%	0.4%	0.4%	64	0.46	0.43	0.40	0.37	0.34
MMLU	0.2%	0.2%	0.3%	0.3%	128	0.35	0.33	0.32	0.32	0.31
Add	0.3%	0.2%	0.2%	1.0%	256	0.32	0.31	0.31	0.31	0.30
GSM	0.4%	0.3%	1.0%	0.1%	512	0.30	0.30	0.29	0.29	0.29
					1024	0.30	0.30	0.29	0.28	0.28

Finally, we evaluate cross-task transfer across SST-2, MMLU, Modular Addition, and GSM8K. When adapting the compression module to a new task, we initialize it using the SSM module trained on the source task and apply only the second-stage fine-tuning objective to the target task.

Table 4 on the right reports the relative error for each source-target pair, where rows indicate the source training tasks and columns indicate the target evaluation tasks. Adaptation takes 65% fewer GPU hours than full training, and the relative error is lower than 1.0% for all task pairs.

4.3 Ablation Studies

We conduct ablation studies to examine the main components of our framework.

Number of groups. First, we study the effects of the HiPPO state dimension and the number of SSM layer groups on SST-2. As shown in Table 4, increasing either value reduces the test loss, but the improvement gradually becomes smaller. In particular, the effect of increasing the HiPPO state dimension largely saturates at 512.

Number of tokens. Then, we study the effect of the number of tokens, n_v , using Qwen-3B with 50 demonstrations. We vary n_v from 2 to 32 and report the output mean squared error (MSE) and the accuracy drop relative to full-context inference. As shown in Table 5, increasing n_v generally reduces both metrics. The results become stable at $n_v = 16$. Thus, we use $n_v = 16$ in the main experiments.

Empirical sample size scaling. Finally, we study the empirical sample complexity of Algorithm 2. First, we study the empirical sample complexity of the randomized ensemble method using linear functions (Garg et al., 2022). Following the setting of Garg et al. (2022), we construct a mixed candidate pool containing demonstrations from two distributions, $\mathcal{D}_1$ and $\mathcal{D}_2$, while sampling all validation queries from $\mathcal{D}_1$. We vary both the number of demonstrations from $\mathcal{D}_1$ in each subset and the number of subsets sampled by the ensemble. The results show that each subset requires more than 20 demonstrations from $\mathcal{D}_1$ to achieve near-zero validation loss. Moreover, sampling more than 50 subsets produces a clear separation between the average affinity scores assigned to demonstrations from $\mathcal{D}_1$ and $\mathcal{D}_2$. Table 6 in Appendix B.3 reports the full results.

5 Related Work

Long-context model inference. Work on reducing transformer prefill cost can be grouped into three lines. First, architectural methods use state space models (SSMs) (Gu et al., 2020, 2022) and hybrid Mamba-Transformer designs (Gu and Dao, 2024; Ren et al., 2025; Oncescu et al., 2025) to achieve linear-time long-sequence modeling. For long-convolution sequence models, Flash Inference (Oncescu et al., 2025) uses tiled FFT computations based on relaxed polynomial interpolation to perform exact autoregressive inference. Note that

Table 5: Ablation study of the number of tokens using Qwen-3B with 50 demonstrations. We report the output MSE and the accuracy drop relative to full-context inference on SST-2, Poem Sentiment, and GSM8K.

Dataset	Metric	$n_v = 2$	$n_v = 4$	$n_v = 16$	$n_v = 32$
SST-2	MSE	0.027	0.015	0.006	0.003
	Accuracy drop	0.07	0.05	0.01	0.01
Poem Sentiment	MSE	0.010	0.006	0.002	0.002
	Accuracy drop	0.06	0.05	0.04	0.04
GSM8K	MSE	0.002	0.001	0.002	0.001
	Accuracy drop	0.02	0.02	0.01	0.01

these approaches target alternative architectures and cannot be directly applied to transformer-based generative models.

Second, StreamingLLM (Xiao et al., 2024) and LM-Infinite (Han et al., 2024), rely on attention sinks and sliding windows. Third, memory-token methods map long contexts into learned slots (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024); however, these are still produced by running a transformer over the original prefix.

Data selection and prompt optimization. In-context learning relies heavily on the quality and structure of the provided demonstrations (Min et al., 2022). Traditional selection relies on static embedding retrieval, which ignores the model’s actual output. To capture these combinatorial effects, recent methods incorporate real inference feedback. For instance, BRIDGE (Wan et al., 2025) uses iterative generation and refinement, while GRADCE (Zhang et al., 2025b) employs gradient-based utility estimation, which requires massive full-model evaluations across numerous candidate subsets.

Another related line of work involves surrogate modeling (Li et al., 2023), data/task attribution (Zhang et al., 2025a, 2026), and influence estimation. For example, Li et al. (2023) use surrogate modeling to efficiently identify negative transfer among data, and KernelSM (Zhang et al., 2026) uses RBF kernel surrogate modeling to estimate affinity. Building on this, Grad-TAG (Li et al., 2024a) estimates pairwise and higher-order task affinities by linearizing model outputs around a shared initialization and using low-dimensional projected gradients to predict the loss of different task subsets. GradEx (Li et al., 2024b) applies this efficient linearization and task affinity score directly to the final subset’s fine-tuned test loss. Next, EnsembleLoRA (Li et al., 2025b) uses the affinity-based method to group related datasets that yield positive gains. Finally, the work on Auto-

BRANE (Li et al., 2025a) estimates layer-wise task affinities from gradients and uses convex relaxation to recursively partition tasks, resulting in a branching network that reduces negative interference and cuts structure search in multitask algorithmic reasoning.

Beyond gradient-based estimation and heuristic scoring, selecting efficient candidate data points connects to the broader literature on experimental design and automated trajectory optimization from historical data (Xiong et al., 2024a,b; Xiong, 2025; Zhang et al., 2025a). It would be interesting to connect ideas from this paper to automated trajectory optimization from historical data.

6 Conclusion

We address the computational bottleneck of demonstration selection with SSMs that reduce both pre-fill and decoding overhead. Empirically, our approach cuts inference computation by two orders of magnitude while keeping relative error below 0.2%, reduces output error over the best compression baseline by 2.17%, and maintains under 1.8% error at 1000 demonstrations. Building on this compressed module, we further propose a randomized estimator for large-scale subset evaluation, achieving a 6.48% average accuracy gain over selection baselines with $14.2\times$ fewer FLOPs. Our work contributed to the literature on prompt optimization by designing an efficient approach utilizing state space models through model distillation.

Limitations

While our method accelerates the input processing phase and reduces FLOPs during inference, training the SSMs module still requires an upfront cost. This introduces a trade-off between distillation and inference costs.

Potential Risks

Our method reduces the cost of repeated long-context inference, which may lower the barrier for large-scale automated model probing or prompt optimization. If applied without safeguards, it could support undesirable uses such as high-volume generation or privacy-sensitive analysis over large prompt collections. Since our method distills a transformer model, it may also inherit biases from the base model and the demonstration. These risks can be reduced by using authorized datasets, applying standard safety filters, and auditing selected demonstrations for sensitive or biased content.

Disclosure of LLM Usage

In preparing this manuscript, LLMs were used for grammar checking and language polishing. No AI tools were involved in generating scientific ideas, theoretical derivations, or experimental designs. All core concepts and scientific contributions are entirely original to the human authors, who assume full responsibility for the integrity and originality of the paper’s content.

Acknowledgment

Thanks to the anonymous reviewers and the area chair for their constructive feedback. The work of Ziniu Zhang and Zhenshuo Zhang is partially funded by NSF award IIS-2412008, a startup fund from Northeastern University, and Khoury PhD fellowships.

References

- Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3829–3846.
- Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. 2022. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:30583–30598.
- Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context autoencoder for context compression in a large language model. In *International Conference on Learning Representations (ICLR)*.
- Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling (COLM)*.
- Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Re. 2020. Hippo: Recurrent memory with optimal polynomial projections. *Advances in Neural Information Processing Systems (NeurIPS)*, 33.
- Albert Gu, Karan Goel, and Christopher Re. 2022. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations (ICLR)*.
- Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 3991–4008.
- Dongyue Li, Huy L Nguyen, and Hongyang R. Zhang. 2023. Identification of negative transfers in multitask learning using surrogate models. *Transactions on Machine Learning Research (TMLR)*.
- Dongyue Li, Aneesh Sharma, and Hongyang R. Zhang. 2024a. Scalable multitask learning using gradient-based estimation of task affinity. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1542–1553.
- Dongyue Li, Zhenshuo Zhang, Minxuan Duan, Edgar Dobriban, and Hongyang R. Zhang. 2025a. Efficiently learning branching networks for multitask algorithmic reasoning. *arXiv preprint arXiv:2512.01113*.
- Dongyue Li, Ziniu Zhang, Lu Wang, and Hongyang R. Zhang. 2024b. Scalable fine-tuning from multiple data sources: A first-order approximation approach. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5608–5623.
- Dongyue Li, Ziniu Zhang, Lu Wang, and Hongyang R. Zhang. 2025b. Efficient ensemble for fine-tuning language models on multiple datasets. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 25347–25364.
- Zhuowei Li, Zihao Xu, Ligong Han, Yunhe Gao, Song Wen, Di Liu, Hao Wang, and Dimitris Metaxas. 2025c. Implicit in-context learning. In *International Conference on Learning Representations (ICLR)*, volume 2025, pages 101644–101666.
- Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35.
- Sewon Min, Xinxu Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 11048–11064.

- Jesse Mu, Xiang Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:19327–19352.
- Costin-Andrei Oncescu, Sanket Purandare, Stratos Idreos, and Sham M Kakade. 2025. Flash inference: Near linear time inference for long convolution sequence models and beyond. In *International Conference on Learning Representations (ICLR)*.
- Tianyue Ou, Frank F Xu, Aman Madaan, Jiarui Liu, Robert Lo, Abishek Sridhar, Sudipta Sengupta, Dan Roth, Graham Neubig, and Shuyan Zhou. 2024. Synatra: Turning indirect knowledge into direct demonstrations for digital agents at scale. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:91618–91652.
- Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. 2026. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations (ICLR)*.
- Liliang Ren, Yang Liu, Yadong Lu, Chen Liang, and Weizhu Chen. 2025. Samba: Simple hybrid state space models for efficient unlimited context language modeling. In *International Conference on Learning Representations (ICLR)*.
- Xingchen Wan, Han Zhou, Ruoxi Sun, and Serkan O Arik. 2025. From few to many: Self-improving many-shot reasoners through iterative optimization and generation. In *International Conference on Learning Representations (ICLR)*.
- Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2025. Agent workflow memory. In *International Conference on Machine Learning (ICML)*, pages 63897–63911. PMLR.
- Emily Xiao, Chin-Jou Li, Yilin Zhang, Graham Neubig, and Amanda Bertsch. 2025a. Efficient many-shot in-context learning with dynamic block-sparse attention. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 31946–31958.
- Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. 2025b. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. In *International Conference on Learning Representations (ICLR)*.
- Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations (ICLR)*.
- Ruoxuan Xiong. 2025. Automated experimental design with optimization from historical data simulations. Available at SSRN 5126080.
- Ruoxuan Xiong, Susan Athey, Mohsen Bayati, and Guido Imbens. 2024a. Optimal experimental design for staggered rollouts. *Management Science*, 70(8):5317–5336.
- Ruoxuan Xiong, Alex Chin, and Sean J Taylor. 2024b. Data-driven switchback experiments: Theoretical tradeoffs and empirical bayes designs. *arXiv preprint arXiv:2406.06768*.
- Zhenshuo Zhang, Minxuan Duan, Youran Ye, and Hongyang R. Zhang. 2025a. Scalable multi-objective and meta reinforcement learning via gradient estimation. *arXiv preprint arXiv:2511.12779*.
- Zhenshuo Zhang, Minxuan Duan, and Hongyang Zhang. 2026. Efficient estimation of kernel surrogate models for task attribution. In *International Conference on Learning Representations (ICLR)*, volume 2026, pages 61822–61848.
- Ziniu Zhang, Zhenshuo Zhang, Dongyue Li, Lu Wang, Jennifer Dy, and Hongyang R. Zhang. 2025b. Linear-time demonstration selection for in-context learning via gradient estimation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 16470–16488.

A Theoretical Analysis

A.1 Error of Grouped HiPPO Projection

For completeness, the HiPPO-LegS transition matrix used by each layer group is defined as

$$\bar{A}_{p,q} = \begin{cases} -\sqrt{(2p+1)(2q+1)}, & \text{if } p > q, \\ -(p+1), & \text{if } p = q, \\ 0, & \text{if } p < q, \end{cases} \quad (2)$$

Proof. Let $\{\phi_r\}_{r=1}^T$ denote the normalized discrete HiPPO-LegS basis on the T token positions, ordered by degree. These basis vectors are orthonormal, so

$$\phi_r^\top \phi_{r'} = \mathbb{1}\{r = r'\}.$$

For each layer group G_i , write

$$C^{(i)} \in \mathbb{R}^{T \times D_i},$$

where the rows correspond to token positions and the columns contain the key and value states from all layers and key-value heads in the group. The vector of the r -th HiPPO-LegS coefficients is

$$c_r^{(i)} := (C^{(i)})^\top \phi_r \in \mathbb{R}^{D_i}.$$

Therefore, the grouped teacher cache admits the expansion

$$C^{(i)} = \sum_{r=1}^T \phi_r (c_r^{(i)})^\top.$$

Since Π_N projects along the token dimension onto the first N basis functions,

$$\Pi_N C^{(i)} = \sum_{r=1}^N \phi_r (c_r^{(i)})^\top.$$

The omitted part of the cache is thus

$$C^{(i)} - \Pi_N C^{(i)} = \sum_{r=N+1}^T \phi_r (c_r^{(i)})^\top.$$

By orthonormality of the basis,

$$\|C^{(i)} - \Pi_N C^{(i)}\|_F^2 = \sum_{r=N+1}^T \|c_r^{(i)}\|_2^2. \quad (3)$$

For every $r \geq N+1$ and $s > 0$,

$$1 \leq \frac{r^{2s}}{(N+1)^{2s}}.$$

Applying this inequality to each term in (3) gives

$$\begin{aligned} & \|C^{(i)} - \Pi_N C^{(i)}\|_F^2 \\ & \leq \frac{1}{(N+1)^{2s}} \sum_{r=N+1}^T r^{2s} \|c_r^{(i)}\|_2^2 \\ & \leq \frac{1}{(N+1)^{2s}} \sum_{r=1}^T r^{2s} \|c_r^{(i)}\|_2^2. \end{aligned}$$

Summing this bound over the g layer groups yields

$$\sum_{i=1}^g \|C^{(i)} - \Pi_N C^{(i)}\|_F^2 \leq \sum_{i=1}^g \sum_{r=1}^T \frac{r^{2s} \|c_r^{(i)}\|_2^2}{(N+1)^{2s}}.$$

which proves the proposition. $\square$

A.2 Demonstration Sampling Complexity

We analyze the number of candidate subsets required to estimate the affinity scores in Algorithm 2. We treat the validation set $\mathcal{V}$ as fixed and account for two sources of error: random subset sampling and grouped KV-cache compression.

Let $D = \{z_1, \dots, z_n\}$ be the candidate demonstration set, and define

$$\mathcal{U}_k(D) := \{D' \subseteq D : |D'| = k\}.$$

For any $D' \in \mathcal{U}_k(D)$, define its exact validation loss as

$$L(D') := \frac{1}{|\mathcal{V}|} \sum_{(q,y) \in \mathcal{V}} \ell(f_W(D', q), y). \quad (4)$$

Let $\tilde{L}(D')$ denote the corresponding loss when the demonstration prefix is replaced by the virtual KV states produced by $\{\psi^{(i)}\}_{i=1}^g$.

For a demonstration $z_a \in D$, define its affinity as

$$\mu(z_a) := -\mathbb{E}_{D' \sim \text{Unif}(\mathcal{U}_k(D))} [L(D') \mid z_a \in D']. \quad (5)$$

Algorithm 2 independently samples

$$D'_1, \dots, D'_M \sim \text{Unif}(\mathcal{U}_k(D)).$$

For each z_a , define

$$\mathcal{I}_a := \{r \in [M] : z_a \in D'_r\}, \quad M_a := |\mathcal{I}_a|.$$

The affinity estimate returned by the algorithm is

$$\hat{\mu}(z_a) := -\frac{1}{M_a} \sum_{r \in \mathcal{I}_a} \tilde{L}(D'_r). \quad (6)$$

We also define the auxiliary estimate based on exact inference:

$$\mu_M(z_a) := -\frac{1}{M_a} \sum_{r \in \mathcal{I}_a} L(D'_r). \quad (7)$$

For each subset D' , let $C^{(i)}(D')$ be the exact teacher KV cache of layer group G_i , and let $c_u^{(i)}(D')$ be its u -th normalized discrete HiPPO-LegS coefficient vector. Define

$$\mathcal{R}_s(D') := \sum_{i=1}^g \sum_{u=1}^{T(D')} u^{2s} \left\| c_u^{(i)}(D') \right\|_2^2,$$

and assume that

$$\mathcal{R}_s(D') \leq \mathcal{R}_{s,\max} \quad \text{for every } D' \in \mathcal{U}_k(D). \quad (8)$$

Define the ideal grouped projection error as

$$\Delta_N(D') := \sqrt{\sum_{i=1}^g \left\| C^{(i)}(D') - \Pi_N C^{(i)}(D') \right\|_F^2}. \quad (9)$$

Assume that

$$0 \leq L(D') \leq B \quad \text{for every } D' \in \mathcal{U}_k(D).$$

We further assume that the learned compressor satisfies

$$\left| \tilde{L}(D') - L(D') \right| \leq K (\Delta_N(D') + \eta), \quad (10)$$

where η accounts for the additional error introduced when $\{\psi^{(i)}\}_{i=1}^g$ map the HiPPO states to virtual-token KV states.

Proposition A.1. Fix $\epsilon > 0$ and $\delta \in (0, 1)$. Suppose

$$M \geq \max \left\{ \frac{8n}{k} \log \frac{2n}{\delta}, \frac{nB^2}{k\epsilon^2} \log \frac{4n}{\delta} \right\}. \quad (11)$$

Then, with probability at least $1 - \delta$, the following statements hold simultaneously for every $z_a \in D$:

$$M_a \geq \frac{Mk}{2n}, \quad (12)$$

and

$$|\hat{\mu}(z_a) - \mu(z_a)| \leq \epsilon + K \left(\frac{\sqrt{\mathcal{R}_{s,\max}}}{(N+1)^s} + \eta \right). \quad (13)$$

Proof. We separately bound the compression error and the subset-sampling error.

Step 1: Bound the cache approximation error.

By Proposition 3.1, for every $D' \in \mathcal{U}_k(D)$,

$$\begin{aligned} \Delta_N(D')^2 &= \sum_{i=1}^g \left\| C^{(i)}(D') - \Pi_N C^{(i)}(D') \right\|_F^2 \\ &\leq \frac{1}{(N+1)^{2s}} \sum_{i=1}^g \sum_{u=1}^{T(D')} u^{2s} \left\| c_u^{(i)}(D') \right\|_2^2 \\ &= \frac{\mathcal{R}_s(D')}{(N+1)^{2s}} \\ &\leq \frac{\mathcal{R}_{s,\max}}{(N+1)^{2s}}. \end{aligned}$$

Therefore,

$$\Delta_N(D') \leq \frac{\sqrt{\mathcal{R}_{s,\max}}}{(N+1)^s}. \quad (14)$$

Combining (10) and (14) gives

$$\left| \tilde{L}(D') - L(D') \right| \leq K \left(\frac{\sqrt{\mathcal{R}_{s,\max}}}{(N+1)^s} + \eta \right). \quad (15)$$

Hence, whenever $M_a > 0$,

$$\begin{aligned} |\hat{\mu}(z_a) - \mu_M(z_a)| &\leq \frac{1}{M_a} \sum_{r \in \mathcal{I}_a} \left| \tilde{L}(D'_r) - L(D'_r) \right| \\ &\leq K \left(\frac{\sqrt{\mathcal{R}_{s,\max}}}{(N+1)^s} + \eta \right). \end{aligned} \quad (16)$$

Step 2: Bound the random subset-sampling error.

A uniformly sampled size- k subset contains any fixed demonstration z_a with probability k/n . Therefore,

$$M_a \sim \text{Binomial} \left(M, \frac{k}{n} \right), \quad \mathbb{E}[M_a] = \frac{Mk}{n}.$$

The multiplicative Chernoff bound gives

$$\mathbb{P} \left(M_a < \frac{Mk}{2n} \right) \leq \exp \left(-\frac{Mk}{8n} \right).$$

Define the coverage event

$$\mathcal{E}_{\text{cov}} := \left\{ M_a \geq \frac{Mk}{2n} \text{ for every } a \in [n] \right\}.$$

Applying a union bound over all n demonstrations gives

$$\mathbb{P}(\mathcal{E}_{\text{cov}}^c) \leq n \exp \left(-\frac{Mk}{8n} \right) \leq \frac{\delta}{2}. \quad (17)$$

Conditional on $\mathcal{I}_a$, the subsets $\{D'_r : r \in \mathcal{I}_a\}$ are independent and uniformly distributed over the size- k subsets containing z_a . Since $L(D') \in [0, B]$, Hoeffding's inequality gives, on $\mathcal{E}_{\text{cov}}$,

$$\begin{aligned} \mathbb{P}\left(|\mu_M(z_a) - \mu(z_a)| > \epsilon \mid \mathcal{I}_a\right) \\ \leq 2 \exp\left(-\frac{2M_a\epsilon^2}{B^2}\right) \\ \leq 2 \exp\left(-\frac{Mk\epsilon^2}{nB^2}\right). \end{aligned}$$

A union bound over all n demonstrations yields

$$\begin{aligned} \mathbb{P}\left(\exists a \in [n] : |\mu_M(z_a) - \mu(z_a)| > \epsilon, \mathcal{E}_{\text{cov}}\right) \\ \leq 2n \exp\left(-\frac{Mk\epsilon^2}{nB^2}\right) \\ \leq \frac{\delta}{2}. \end{aligned} \quad (18)$$

Combining (17) and (18), we obtain, with probability at least $1 - \delta$,

$$M_a \geq \frac{Mk}{2n}$$

and

$$|\mu_M(z_a) - \mu(z_a)| \leq \epsilon$$

simultaneously for every $a \in [n]$.

Step 3: Combine the two errors. On this event, $M_a > 0$ for every $a \in [n]$. By the triangle inequality,

$$\begin{aligned} |\hat{\mu}(z_a) - \mu(z_a)| \\ \leq |\hat{\mu}(z_a) - \mu_M(z_a)| + |\mu_M(z_a) - \mu(z_a)| \\ \leq K \left(\frac{\sqrt{\mathcal{R}_{s,\max}}}{(N+1)^s} + \eta \right) + \epsilon. \end{aligned}$$

This proves the proposition. $\square$

Proposition A.1 shows that, up to logarithmic factors, the number of sampled subsets required for uniform affinity estimation scales as n/k . A larger subset size k increases the probability that each sampled subset contains any fixed demonstration, thereby increasing the effective sample size used to estimate its affinity. The total error consists of the random subset-sampling error and the grouped KV-cache compression error.

B Omitted Experiments

B.1 Implementation Setup

Datasets. We evaluate our method on datasets covering several task types, including language understanding, algorithmic reasoning, arithmetic reasoning, and graph reasoning. The number of demonstrations within each task is 1500.

The **SST-2** dataset is a binary sentiment classification benchmark from **GLUE**, consisting of movie reviews labeled as positive or negative.

The **Poem Sentiment** dataset contains sentiment labels for lines of poetry, with three classes: positive, neutral, and negative. This dataset is used to evaluate sentiment understanding in literary text.

Modular Addition is an arithmetic reasoning task in which the model is given two integers $a, b \in \{0, \dots, P-1\}$ for a prime P and is asked to predict their sum modulo P , that is, to predict c such that $a + b \equiv c \pmod{P}$.

The **Coin Flip** dataset is a reasoning task in which the model is given a natural language description of a sequence of fair coin flips and must predict the final outcome, either heads or tails.

The Edge Existence dataset from **GraphQA** is a graph reasoning task in which the model is given an undirected graph and must determine whether a specified edge is present in the graph.

The **MMLU** dataset is a multitask benchmark covering a wide range of subjects, including humanities, social sciences, and STEM. It evaluates a model's ability to perform multiple-choice question answering across diverse domains. We use the college-level tasks in the evaluation.

The **GSM8K** dataset is a grade-school math reasoning benchmark consisting of natural language word problems. Each example requires multi-step arithmetic reasoning, and the model is asked to produce the final numerical answer. We use this dataset to evaluate whether Algorithm 1 preserves the reasoning information needed for long-context mathematical inference.

Models. Our method is a model-agnostic method. In our experiments, we evaluate it on a diverse set of open-source language models that vary in scale and architecture, including **Qwen2.5-1.5B-Instruct**, **Qwen2.5-3B-Instruct**, **Qwen2.5-7B-Instruct**, and **Llama-3.1-8B-Instruct**. This setup allows us to assess the effectiveness and scalability of our method across models with different capacities, training paradigms, and usage settings.

Training budget. We train our method on a single

NVIDIA RTX A6000 GPU. We set the number of virtual tokens to 16, the SSM state dimension to 512, and the number of layer groups to 4. Training for 1 epoch in phase one and 2 epochs in phase two requires 1.15 GPU hours.

Baselines. We compare Algorithm 1 against full inference, StreamingLLM, LM-Infinite, Block Sparse Attention baselines, and DuoAttention. Then, we compare Algorithm 2 against baselines including BM25, top- k , BRIDGE, GRADCE.

Full Attention serves as our upper-bound baseline, performing standard autoregressive inference without KV cache compression. Every token attends to all preceding tokens, retaining the complete Key-Value cache across all layers and heads. While preserving maximum model capability, its linear memory scaling and quadratic pre-filling cost become prohibitively expensive for large demonstration pools.

StreamingLLM (Xiao et al., 2024) exploits the phenomenon of "attention sinks," where initial tokens receive disproportionately high attention scores regardless of semantic relevance. It maintains a fixed-size KV cache comprising only these initial sink tokens and a sliding window of recent tokens, evicting all intermediate context. This achieves constant memory usage but inherently degrades performance on tasks requiring fact retrieval from the middle of the input.

LM-Infinite (Han et al., 2024) addresses length generalization beyond the model’s pre-training window. To prevent attention degradation from out-of-distribution position indices, it discards tokens outside a prescribed local window and applies a Λ -shaped position remapping. This aligns the remaining tokens’ indices with the training distribution, enabling infinite context processing without fine-tuning. However, similar to StreamingLLM, it suffers from information loss by permanently evicting middle-context tokens.

DuoAttention (Xiao et al., 2025b) introduces a head-level compression strategy by categorizing attention heads into retrieval heads (attending to arbitrary positions) and streaming heads (attending to sinks and recent tokens). During a lightweight identification phase, a trainable gate $\alpha_{i,j}$ is assigned to each KV head and optimized via synthetic passkey-retrieval data. This minimizes output deviation from Full Attention under an ℓ_1 sparsity penalty. At deployment, binarized gates selectively apply the StreamingLLM eviction policy exclusively to streaming heads, while retrieval heads retain the

complete KV cache. This selectively reduces memory and latency while preserving long-range retrieval capabilities.

ICAE (Ge et al., 2024) compresses the input context into a small number of learned memory tokens through an autoencoding objective. It first encodes the original context into compact latent representations, and then conditions the language model on these memory tokens to recover the information needed for downstream generation.

Implicit In-context Learning (I2CL) (Li et al., 2025c) moves demonstrations from the token space into the model’s activation space. It extracts a vector from each demonstration, combines these vectors in a way that is independent of their order, and injects the resulting context vector into the model’s residual streams during inference. I2CL provides few-shot performance with inference cost close to that of zero-shot prediction. However, constructing its context vector still requires Transformer forward passes over the demonstrations.

The top- k method selects the k most similar candidates based on feature similarity, given a test input. We compute the cosine similarity in the top- k range between the last-layer hidden representations of queries and demonstration examples. The BM25 method utilizes a term frequency-based ranking function to retrieve the top- k candidates whose input texts are most relevant to the query, with a focus on lexical overlap rather than embedding-level similarity. GRADCE (Zhang et al., 2025b) first filters candidates by embedding similarity, then selects a random anchor and performs a single forward and backward pass on all validation samples to precompute the loss and gradient with respect to the input embedding at that anchor. For each remaining candidate, it computes the embedding difference from the anchor and takes the inner product with the precomputed gradient to approximate the candidate’s validation loss via a first-order Taylor expansion, selecting the k candidates with the lowest estimated loss. BRIDGE (Wan et al., 2025) iteratively alternates between using Bayesian optimization to select a small high-performing subset of ICL demonstrations from the candidate pool ("optimize") and using that subset as seed examples to re-generate reasoning paths on the training set back to the many-shot regime ("generate"), repeating this process over multiple rounds to progressively improve demonstration quality.

B.2 Efficient Implementation

Previous work (Xiao et al., 2024) shows that standard LLM inherently relies on initial tokens as “attention sinks” to stabilize the generation process. Thus, we explicitly retain a small number of exact initial KV states as sink tokens from the raw prefix. These states are concatenated with our generated virtual tokens to form the compressed context.

The standard SSM recurrence processes tokens sequentially:

$$s_t = A s_{t-1} + u_t, \quad u_t = B x_t, \quad s_0 = 0, \quad (19)$$

where $x_t \in \mathbb{R}^H$ is the input embedding at step t , $B \in \mathbb{R}^{D \times H}$ is a learned input projection, $A \in \mathbb{R}^{D \times D}$ is the frozen discretized HiPPO matrix, and $s_t \in \mathbb{R}^D$ is the hidden state. A naive implementation requires T sequential kernel launches, which may under utilize the GPU.

We observe that unrolling the recurrence over C consecutive steps yields a closed-form expression. Starting from an incoming state s_{in} , the state after processing inputs $u_0, u_1, \dots, u_{C-1}$ is:

$$s_{\text{out}} = A^C s_{\text{in}} + \sum_{t=0}^{C-1} A^{C-1-t} u_t. \quad (20)$$

The summation can be rewritten as a single batched contraction. Define the weight matrices $W_t = A^{C-1-t}$ for $t = 0, \dots, C-1$, so that $W_0 = A^{C-1}$ (oldest input, most decay) and $W_{C-1} = I$ (newest input, no decay). Stacking these into a tensor $\mathbf{W} \in \mathbb{R}^{C \times D \times D}$ and the chunk inputs into $\mathbf{U} \in \mathbb{R}^{C \times D}$, the entire summation reduces to:

$$\sum_{t=0}^{C-1} W_t u_t = \sum_{t=0}^{C-1} \mathbf{W}[t] \mathbf{U}[t], \quad (21)$$

which is evaluated as a single einsum operation on the GPU.

We partition the full prefix of length T into $\lceil T/C \rceil$ contiguous chunks. Processing proceeds as follows: for each chunk $k = 0, 1, \dots$, we first compute the powers $\{A^{C-1}, A^{C-2}, \dots, I\}$ iteratively using C matrix multiplications of size $D \times D$. Then, we evaluate the within-chunk contribution via the batched contraction. Finally, we advance the inter-chunk state as $s_{\text{in}}^{(k+1)} = A^C s_{\text{in}}^{(k)} + \text{contrib}^{(k)}$. Since A is frozen, the powers are computed on the fly and discarded after each chunk to avoid storing a $C \times D \times D$ buffer.

This reduces the number of sequential state updates from T to $\lceil T/C \rceil$, while each chunk-level

operation is a single, well-parallelised GPU kernel. The underlying SSM dynamics are unchanged—the final state s_T is mathematically identical to that of the token-by-token scan.

During training, we observe that directly optimizing the distillation objectives (logits and hidden states) without prior initialization degrades performance. The root cause is a bootstrap problem: randomly initialized virtual KV tokens produce degenerate attention patterns in the frozen Transformer, which in turn yield near-uniform output distributions. The resulting logit-level and hidden-state gradients carry little useful signal for the auxiliary, preventing meaningful learning. The first-stage KV alignment resolves this by constraining the virtual tokens to lie in a geometrically valid region of the KV representation space before any query-conditioned supervision is applied. Although the chunked mean-pooling targets used in this stage are only an approximation of the true cache, they do not preserve positional structure or the sparse attention patterns of individual tokens, they are sufficient to initialize the auxiliary into a regime where the subsequent distillation losses produce informative gradients and converge reliably.

Artifact use and licenses. We use publicly available datasets, model checkpoints, and baseline implementations only for research evaluation. We cite the original creators of these artifacts and follow their stated licenses and terms of use.

B.3 Omitted Experiment Results

We illustrate additional evaluation results across different model scales and sequence lengths to supplement the findings in the main text.

Sensitivity to fewer demonstrations. To verify that compression fidelity holds at shorter context lengths, Table 7 extends the relative output error analysis to the context window of $k = 25$ demonstrations. Across two different models (Qwen-1.5B, Qwen-3B) and four downstream tasks, the relative error between the full demonstration prefix and our compressed virtual tokens remains below 0.6%. These results confirm that Algorithm 1 consistently preserves the original predictive distribution regardless of the context scale or model architecture.

Generalization to smaller models. To test whether Algorithm 1 maintains its advantage at a smaller model scale, Table 10 details the computational cost, memory usage, and relative error

Table 6: To illustrate the scoring mechanism in our random ensemble method, we report the average score of demonstration samples from $\mathcal{D}_1$ and $\mathcal{D}_2$, given the query set generated from $\mathcal{D}_1$. We vary the number of subsets and find that the in-distribution samples always achieve high score than out-of-distribution samples.

# Subsets	10	50	100	200
Samples from $\mathcal{D}_1$	$0.22_{\pm 0.04}$	$0.55_{\pm 0.03}$	$0.57_{\pm 0.04}$	$0.56_{\pm 0.03}$
Samples from $\mathcal{D}_2$	$0.21_{\pm 0.01}$	$0.52_{\pm 0.04}$	$0.51_{\pm 0.06}$	$0.48_{\pm 0.04}$

Table 7: Relative error of the predictive output between the full demonstration prefix p and the compressed virtual tokens $\hat{p}$. Evaluations are reported across multiple downstream tasks for varying demonstration counts. Lower values indicate higher fidelity to the original uncompressed model output. We run three times to compute the mean and standard deviations.

Model	# Demo.	SST-2	CR	Poem Sentiment	Modular Addition
Qwen-1.5B	$k = 25$	$2.3_{\pm 0.2} \times 10^{-3}$	$1.9_{\pm 0.3} \times 10^{-3}$	$8.9_{\pm 2.5} \times 10^{-4}$	$1.1_{\pm 0.1} \times 10^{-3}$
Qwen-3B	$k = 25$	$5.3_{\pm 0.2} \times 10^{-3}$	$3.2_{\pm 0.3} \times 10^{-3}$	$2.0_{\pm 2.5} \times 10^{-3}$	$3.5_{\pm 0.1} \times 10^{-3}$

Table 8: Test accuracy (%) of Algorithm 2 with and without context compression. Each result reports the mean and standard deviation over three random seeds.

Method	SST-2	Poem Sentiment	Addition	Coin Flip	Edge Existence
Algorithm 2	$95.9_{\pm 0.7}$	$78.6_{\pm 1.1}$	$81.8_{\pm 1.2}$	$85.3_{\pm 1.2}$	$83.6_{\pm 0.6}$
Algorithm 2 w/o compression	$95.0_{\pm 0.5}$	$79.7_{\pm 0.8}$	$82.8_{\pm 1.0}$	$85.3_{\pm 1.2}$	$83.7_{\pm 1.3}$

for the Qwen-1.5B-Instruct model on the SST-2, Poem Sentiment, and Modular Addition datasets. Under the standard setting of $k = 50$ demonstrations, Algorithm 1 achieves the lowest relative error among all evaluated compression methods, including 0.3% on SST-2 and 0.1% on both Poem Sentiment and Addition. Furthermore, Algorithm 1 requires fewer prefill FLOPs than the leading baseline, BSA, while maintaining a comparable memory footprint. This verifies that the efficiency and accuracy advantages of the Algorithm 1 module generalize reliably to smaller model variants.

Coverage of remaining task categories. To confirm that compression fidelity extends to other reasoning types, Table 10 and Table 11 present the evaluation on the remaining reasoning datasets, Coin Flip and Edge Existence, using both the Qwen-1.5B-Instruct and Qwen-3B-Instruct models with $k = 50$ demonstrations. Consistent with earlier findings, Algorithm 1 achieves the lowest relative error among all baselines, recording 0.1% error on both datasets for the 1.5B model, and under 0.9% for the 3B model. This demonstrates that Algorithm 1 preserves strong predictive fidelity across diverse reasoning tasks while maintaining a highly efficient computational profile compared to sliding-window and block-sparse baselines.

Results regarding Algorithm 2. To separate the effect of the selection algorithm from that of context compression, we further evaluate Algorithm 2 using the original, uncompressed contexts. As shown in Table 8, removing compression changes the average accuracy from 85.04% to 85.30%, corresponding to an increase of 0.26 percentage points. This result shows that the accuracy gain cannot be explained by the compressed representation acting as a regularizer. The selection algorithm remains effective when candidates are evaluated using the full context.

Table 9: We report the compression results using more window lengths on MMLU, Addition datasets, and GSM8K with $k = 50$. The relative error is computed against actual inference results.

Method	Window Length	Error	MMLU		Modular Addition			Error	GSM8K	
			FLOPs	Memory	Error	FLOPs	Memory		FLOPs	Memory
StreamingLLM	32	2.5%	$7.72e^{13}$	15.88G	10.7%	$1.90e^{13}$	13.95G	6.4%	$1.25e^{14}$	14.05G
	128	0.7%	$7.75e^{13}$	15.95G	8.7%	$1.91e^{13}$	13.96G	5.1%	$1.26e^{14}$	14.07G
LM-Infinite	32	11.6%	$7.73e^{13}$	15.90G	14.3%	$1.90e^{13}$	12.96G	5.7%	$1.26e^{14}$	14.06G
	128	5.1%	$7.75e^{13}$	15.98G	8.7%	$1.91e^{13}$	12.99G	4.9%	$1.27e^{14}$	14.07G

Table 10: Supplementary compression results on Qwen-1.5B-Instruct. We compare long-context inference methods on SST-2, Poem Sentiment, Modular Addition, Coin Flip, Edge Existence, and MMLU with $k = 50$ demonstrations, and report relative output error with respect to full-prefix inference, together with FLOPs and peak memory usage.

Method	Window Length	Error	SST-2		Poem Sentiment			Modular Addition		
			FLOPs	Memory	Error	FLOPs	Memory	Error	FLOPs	Memory
StreamingLLM	32	11.3%	$9.42e^{12}$	6.94G	17.5%	$1.36e^{13}$	6.61G	12.8%	$9.53e^{12}$	7.21G
	64	7.6%	$9.43e^{12}$	6.94G	13.2%	$1.36e^{13}$	6.62G	11.9%	$9.54e^{12}$	7.22G
	128	5.2%	$9.46e^{12}$	6.96G	8.7%	$1.37e^{13}$	6.64G	8.4%	$9.56e^{12}$	7.24G
LM-Infinite	32	16.2%	$9.36e^{12}$	6.93G	7.0%	$1.37e^{13}$	6.61G	5.5%	$9.41e^{12}$	7.21G
	64	11.5%	$9.38e^{12}$	6.94G	3.7%	$1.37e^{13}$	6.61G	3.5%	$9.44e^{12}$	7.21G
	128	11.0%	$9.41e^{12}$	6.96G	2.9%	$1.37e^{13}$	6.64G	3.2%	$9.50e^{12}$	7.23G
BSA	~ 400	1.1%	$9.52e^{12}$	7.15G	0.7%	$1.53e^{13}$	6.96G	0.9%	$8.81e^{12}$	7.78G
Algorithm 1	16	0.3%	$6.30e^{12}$	7.19G	0.1%	$1.02e^{13}$	7.06G	0.1%	$5.28e^{12}$	8.06G
Method	Window Length	Error	Coin Flip		Edge Existence			Error	MMLU	
			FLOPs	Memory	Error	FLOPs	Memory		FLOPs	Memory
StreamingLLM	32	18.1%	$1.30e^{13}$	7.83G	3.8%	$5.99e^{13}$	14.63G	3.2%	$3.88e^{13}$	10.06G
	64	10.2%	$1.30e^{13}$	7.84G	2.7%	$6.01e^{13}$	14.64G	0.8%	$3.88e^{13}$	10.08G
	128	9.8%	$1.31e^{13}$	7.85G	2.5%	$6.03e^{13}$	14.65G	0.6%	$3.89e^{13}$	10.13G
LM-Infinite	32	28.7%	$1.32e^{13}$	7.84G	7.5%	$6.43e^{13}$	14.63G	1.4%	$3.88e^{13}$	10.07G
	64	26.8%	$1.32e^{13}$	7.84G	6.5%	$6.43e^{13}$	14.64G	0.7%	$3.89e^{13}$	10.08G
	128	23.9%	$1.33e^{13}$	7.86G	4.6%	$6.46e^{13}$	14.66G	0.5%	$3.89e^{13}$	10.13G
BSA	~ 400	0.7%	$1.35e^{13}$	8.28G	0.9%	$6.88e^{13}$	15.77G	0.5%	$3.95e^{13}$	10.36G
Algorithm 1	16	0.1%	$9.02e^{12}$	8.35G	0.1%	$4.11e^{13}$	14.73G	0.3%	$2.49e^{13}$	10.81G

Table 11: Supplementary compression results on Qwen-3B-Instruct. We compare long-context inference methods on the Coin Flip, Edge Existence, and Poem Sentiment tasks with $k = 50$ demonstrations, and report relative output error with respect to full-prefix inference, together with FLOPs and peak memory usage.

Method	Window Length	Error	Coin Flip		Edge Existence			Poem Sentiment		
			FLOPs	Memory	Error	FLOPs	Memory	Error	FLOPs	Memory
Dense	-	-	$1.49e^{15}$	14.58G	-	$9.77e^{15}$	24.57G	-	$2.53e^{15}$	12.70G
StreamingLLM	64	31.5%	$2.63e^{13}$	13.59G	1.9%	$1.28e^{14}$	20.40G	9.3%	$2.72e^{13}$	12.37G
	128	30.6%	$2.64e^{13}$	13.61G	1.8%	$1.28e^{14}$	20.42G	9.1%	$2.73e^{13}$	12.39G
LM-Infinite	64	32.8%	$2.64e^{13}$	13.60G	1.3%	$1.28e^{14}$	20.40G	14.2%	$2.72e^{13}$	12.38G
	128	30.6%	$2.64e^{13}$	13.61G	1.2%	$1.28e^{14}$	20.43G	13.8%	$2.73e^{13}$	12.40G
ICAE	128	1.4%	$1.36e^{14}$	14.27G	3.5%	$1.02e^{15}$	24.76G	3.2%	$5.40e^{13}$	12.32G
DuoAttention	64	10.5%	$2.70e^{13}$	14.02G	3.2%	$1.40e^{14}$	22.59G	7.6%	$2.75e^{13}$	12.50G
BSA	~ 400	2.4%	$2.75e^{13}$	14.07G	1.4%	$1.53e^{14}$	24.64G	4.2%	$2.95e^{13}$	12.85G
Algorithm 1	16	0.7%	$1.91e^{13}$	14.13G	0.8%	$9.97e^{13}$	25.02G	1.6%	$2.37e^{13}$	13.23G