Research Article

StreamAdapter: Efficient Test Time Adaptation from Contextual Streams

Dilxat Muhtar[1], Yelong Shen[2], Yaming Yang[2], Xiaodong Liu[2], Yadong Lu[2], Jianfeng Liu[2], Yuefeng Zhan[2], Hao Sun[2], Weiwei Deng[2], Feng Sun[2], Xueliang Zhang[1], Jianfeng Gao[2], Weizhu Chen[2], Qi Zhang[2]

1. Nanjing University, China; 2. Microsoft, Washington, United States

In-context learning (ICL) allows large language models (LLMs) to adapt to new tasks directly from the given demonstrations without requiring gradient updates. While recent advances have expanded context windows to accommodate more demonstrations, this approach increases inference costs without necessarily improving performance. To mitigate these issues, We propose StreamAdapter, a novel approach that directly updates model parameters from context at test time, eliminating the need for explicit in-context demonstrations. StreamAdapter employs context mapping and weight absorption mechanisms to dynamically transform ICL demonstrations into parameter updates with minimal additional parameters. By reducing reliance on numerous in-context examples, StreamAdapter significantly reduce inference costs and allows for efficient inference with constant time complexity, regardless of demonstration count. Extensive experiments across diverse tasks and model architectures demonstrate that StreamAdapter achieves comparable or superior adaptation capability to ICL while requiring significantly fewer demonstrations. The superior task adaptation and context encoding capabilities of StreamAdapter on both language understanding and generation tasks provides a new perspective for adapting LLMs at test time using context, allowing for more efficient adaptation across scenarios and more cost-effective inference.

Dilxat Muhtar[†] and Yelong Shen equally contributed to this work.

[†] Work done during internship at Microsoft.

Corresponding authors: Dilxat Muhtar, dmuhtar@smail.nju.edu.cn; Yelong Shen, yelong.shen@microsoft.com

1. Introduction

Large language models (LLMs) have emerged as a powerful tool in natural language processing, demonstrating exceptional performance across a diverse range of tasks, including text generation[1], question answering[2],

open-ended conversations[3], and mathematical problem-solving[4]. A key factor behind the success of LLMs is their ability to perform in-context learning (ICL)[5], where the model adapts to new tasks by conditioning on a small number of input-output demonstrations provided in the context. Without any gradient updates, ICL enables LLMs to acquire new knowledge and capabilities at test time, while also enabling LLMs to solve complex tasks through step-by-step guidance[6].

Despite its remarkable capabilities, ICL faces several limitations that hinder its full potential. Firstly, the effectiveness of ICL heavily depends on the quality and relevance of the provided demonstrations, making the selection of appropriate examples a challenging task that often requires domain expertise[7][8]. Moreover, the number of demonstrations that can be included is constrained by the model's context window size. While recent advancements have expanded these windows[9][10], accommodating more examples introduces significant computational overhead[11].

Although recent studies have attempted to use heuristic rules to select the most important subset of context to improve the robustness and efficiency of ICL[12][13], these methods inevitably cannot ensure that the discarded tokens that are currently unimportant will not become important in future decoding steps. Other investigations have focused on constructing meta-ICL approaches to enhance ICL's robustness and reduce reliance on perfect prompts[14]. Yet, these methods remain constrained by limited context length and often require hand-crafted prompt strategies, potentially leading to suboptimal performance. On the other hand, recent studies suggest that ICL is actually performing a meta-gradient update for adapting to new tasks given the context information[15][16]. These findings lead us to a crucial question: *Instead of implicitly "updating" model parameters to adapt to a new domain or task via context, is it possible to directly convert the context into parameter updates, thus updating the network at test time without any backpropagation and without requiring demonstrations in the context window?*

To answer this question, we propose StreamAdapter, a novel approach that leverages the inherent capabilities of LLMs to encode context information into their parameters. Instead of storing demonstrations explicitly in the input context, StreamAdapter dynamically maps these demonstrations into temporary parameter updates. This approach allows the model to benefit from context to adapt to new tasks similar to ICL at test-time, without consuming the context window or requiring backpropagation, thereby reducing the resource requirements of traditional ICL methods. StreamAdapter employs two key mechanisms to achieve this goal: a) Context Mapping: This mechanism utilizes intra-chunk cross-attention and inter-chunk recurrence to adaptively condense the variable cached context into a constant context state for each parameter in the linear layer of LLMs. b) Weight Absorption: The condensed context state interacts with two lightweight low-rank matrices to be absorbed into the original model parameters. This process updates the LLM's knowledge with

minimal additional learnable parameters and incurs no additional inference latency. By combining these mechanisms, StreamAdapter effectively distills the context into parameter updates, allowing for more efficient test-time adaptation (TTA). Comprehensive experiments across diverse language understanding and long-context generation tasks, with various model architectures and scales, demonstrate that StreamAdapter achieves comparable or superior adaptation capability to full context evaluation while outperforming other context compression variants and TTA methods. Moreover, StreamAdapter not only demonstrates constant inference generation time and lower memory consumption compared to full context generation, but also shows better scalability when provided with more adaptation context and improved robustness across various scenarios.

The contributions of our work can be summarized as follows:

- We propose a new TTA strategy, StreamAdapter, that directly maps the given context into parameter updates, rather than conditioning on the context. This method enables models to quickly adapt to new tasks or acquire new temporary knowledge at test time like ICL, but with fewer or no demonstrations in context, thereby reducing memory consumption and inference time.
- We design StreamAdapter with innovative context mapping and low-rank adaptation mechanisms. These allow StreamAdapter to map the context into parameter updates with minimal additional learning parameters and without inducing any additional inference latency.
- We validate StreamAdapter on both language understanding and language generation tasks across various model scales and architectures. The results demonstrate the effectiveness of StreamAdapter over ICL and other TTA methods in various adaptation scenarios. Analyses of efficiency and robustness further highlight StreamAdapter's advantages in terms of computational resources and generalization capabilities.

2. Related Work

2.1. In-Context Learning

ICL enables LLMs to acquire new knowledge or adapt to new tasks using in-context examples at test time without any gradient updates[5]. Recent studies show that with proper instruction and more demonstrations, ICL can surpass model fine-tuning and mitigate inherent biases in pre-trained LLMs[7][17]. This exceptional capability has inspired research into ICL's working mechanisms, leading to various hypotheses such as induction heads[18], task vectors[19][20], and structured task hypothesis[21]. A popular assumption posits that ICL performs meta-gradient descent during inference.[16] demonstrate how a linear attention-only transformer model can implicitly perform a gradient descent-like procedure, while[15] compare standard gradient descent-

based fine-tuning and ICL, revealing that transformer attention in ICL exhibits a dual form of gradient descent-based optimization. Inspired by these findings, our work seeks to develop a learning algorithm that directly performs parameter updates from the context without backpropagation at test time, aiming to achieve performance similar to ICL while requiring limited or no demonstrations in the context.

2.2. Test-Time Adaptation

Test-time adaptation (TTA) enhances model capabilities at inference by learning directly from test data[22]. In-context learning (ICL) represents a form of TTA where models adapt to new tasks using demonstrations within the context at test time. Recent TTA research primarily follows two directions: a) Condition Augmentation: This approach focuses on modifying the context conditioning to improve performance, either through heuristic rules for adjusting conditional prediction distributions[12][13] or through sampling strategies like best-of-N and reward-model based sampling[23][24][25]. b) Parameter Updates: This direction explores modifying model parameters at inference time. Early approaches build on fast weight programming[26], exemplified by fast weight programmers[27] and Hopfield networks[28], which update pre-trained weights using input-based products. Meta-learning approaches[29][30] employ hypernetworks to generate auxiliary parameters for test-time adaptation. TempLoRA[31] extends this concept by training chunk-specific low-rank adapters[32] for next-chunk prediction. Recent work[33] formalizes test-time parameter updates through self-supervised learning with TTT-Linear and TTT-MLP, treating model parameters as latent RNN states.

Our approach, StreamAdapter, aligns with parameter update methods but uniquely maps context directly into parameter updates at test time without backpropagation.

2.3. Low-Rank Adaptation

Inspired by the observation that pre-trained models have low intrinsic dimension during fine-tuning[34], low-rank adaptation (LoRA)[32] employs two trainable low-rank matrices to estimate the accumulated gradient updates, thereby adapting pre-trained models with minimal additional parameters. Given its lower inference latency and superior adaptation performance, LoRA has been widely adopted, with subsequent research enhancing its efficiency and stability through dynamic rank allocation across layers[35] and further matrix decomposition[36]. Our work also employs low-rank adaptation to adapt LLMs with minimal parameters. However, instead of training the adapter for specific tasks or datasets, StreamAdapter learns directly from previous context at test time, enabling more customized and flexible adaptation.

3. Method

We propose StreamAdapter to directly map contextual information into parameter updates, serving as a temporary weight-level associative memory that encodes new knowledge and adapts to new tasks without relying on full explicit context. The overall structure of StreamAdapteris presented in Figure 1. StreamAdapter utilizes intra-chunk cross-attention and inter-chunk gated recurrence to adaptively map sparse context information into constant-sized context states (context mapping). These states are then absorbed into pre-trained weights through low-rank adaptation.

In the following subsections, we will briefly describe the duality between ICL and model parameter updates through gradient descent, shedding light on the fundamental motivation behind StreamAdapter and its formalization. We will then explore the details of StreamAdapter context mapping and weight absorption mechanisms.

3.1. Duality between In-Context Learning and Weight Updates

Recent studies have highlighted the inherent similarities between ICL and parameter updates through gradient descent[15][16]. Specifically, let x_i be the current input token, $\mathbf{X}'$ be the previous context, and $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$ be the projection matrices of the self-attention (SA) layer. By approximating standard SA with linear attention, the output of single-head SA is formulated as:

$$\begin{aligned} \mathcal{F}_{\mathrm{ICL}}(x_i) &\approx \mathbf{W}_v[\mathbf{X}', x_i](\mathbf{W}_k[\mathbf{X}', x_i])^T \mathbf{W}_q x_i \\ &= \mathbf{W}_v x_i (\mathbf{W}_k x_i)^T \mathbf{W}_q x_i + \mathbf{W}_v \mathbf{X}' (\mathbf{W}_k \mathbf{X}')^T \mathbf{W}_q x_i \\ &= (\mathbf{W}_0 + \Delta \mathbf{W}_{\mathrm{ICL}}) \mathbf{W}_q x_i, \end{aligned} \tag{1}$$

where $\mathbf{W}_0 = \mathbf{W}_v x_i (\mathbf{W}_k x_i)^T$ are the initial result without any context, and $\Delta \mathbf{W}_{\mathrm{ICL}} = \mathbf{W}_v \mathbf{X}' (\mathbf{W}_k \mathbf{X}')^T$ represents the "parameter updates" obtained from the given context. Moreover, denoting $\Delta \mathbf{W}_k$ and $\Delta \mathbf{W}_v$ as the accumulated gradient updates from fine-tuning, the result of linear attention can be expressed as:

$$\begin{aligned} \mathcal{F}_{\mathrm{FT}}(x_i) &= (\mathbf{W}_v + \Delta \mathbf{W}_v) x_i x_i^T (\mathbf{W}_k + \Delta \mathbf{W}_k) \mathbf{W}_q x_i \\ &= (\mathbf{W}_0 + \Delta \mathbf{W}_{\mathrm{FT}}) \mathbf{W}_q x_i. \end{aligned} \tag{2}$$

From the similarity between $\mathcal{F}_{\mathrm{ICL}}(x_i)$ and $\mathcal{F}_{\mathrm{FT}}(x_i)$, it can be hypothesized that ICL actually functions as a meta-optimizer, updating the underlying parameters through context-level associations[15].

In this study, we delve deeper into the potential of leveraging context to directly update model parameters, thereby integrating context information into the model's weights. The objective of StreamAdapter is to learn a mapping function $\mathcal{F}$ that, given context $\mathbf{X}'$, maps the key-value (KV) caches $\mathbf{W}_k \mathbf{X}'$ and $\mathbf{W}_v \mathbf{X}'$ of $\mathbf{X}'$ to parameter update $\Delta \mathbf{W}$:

$$\mathcal{F}(\mathbf{W}_k\mathbf{X}', \mathbf{W}_v\mathbf{X}') \rightarrow \Delta\mathbf{W}. \tag{3}$$

We anticipate that updating the model parameters with $\Delta\mathbf{W}$ will achieve results comparable to full ICL without the need for complete demonstrations filling the context window.

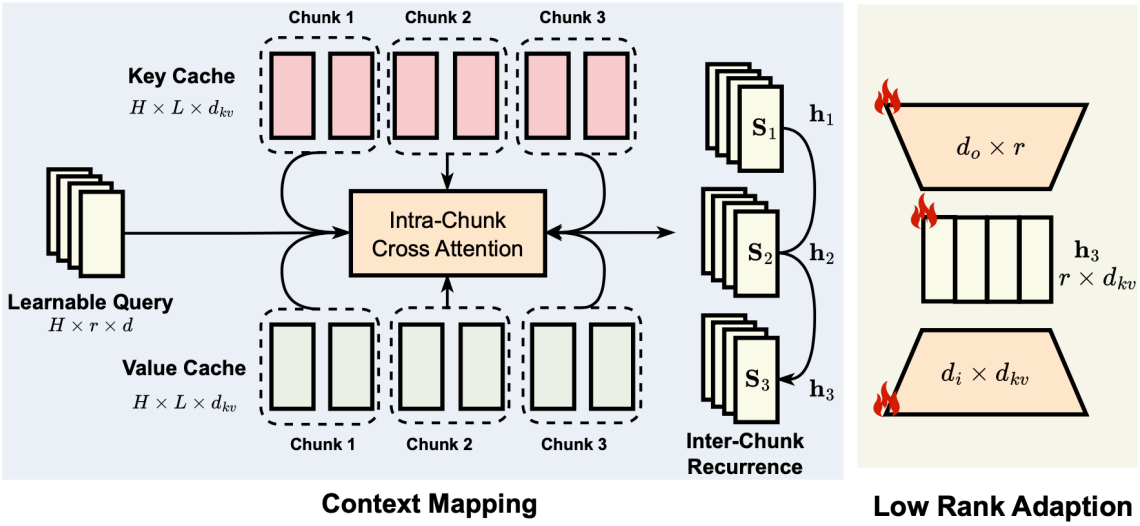

Figure 1. Overall structure of StreamAdapter. StreamAdapter maps the KV cache into a context state using intra-chunk cross-attention and inter-chunk recurrence, then connects two low-rank matrices through the context state to update the model parameters for absorbing context information into model weights

3.2. Context Mapping

The KV cache scales linearly with the context, whereas the model's parameters remain constant in size. Consequently, a context mapping strategy that condenses the cache information into a fixed-size state is essential for absorbing context information into the model's weights. The most straightforward approach to achieving this is to compress the KV cache into a latent hidden state, similar to recurrent models[37][38]. However, token-by-token recurrence requires substantial memory, as it necessitates materializing all time step states. To mitigate this issue, we propose splitting the KV cache into fixed-size chunks and leveraging a number of learnable queries to summarize each chunk of caches. We then perform inter-chunk recurrence across each chunk of summarized results to convert the cache into a constant-size context state. More specifically, let the KV cache be denoted as $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{H\times L\times d}$, where H is the number of heads, L is the length of cache, and d is the hidden dimension for each head. Let C be the predefined chunk size, and define $\mathbf{K}_{[i]} := \mathbf{K}_{iC+1:(i+1)C+1} \in \mathbb{R}^{H\times C\times d}$ as the key cache corresponding to the i-th chunk (with similar notation for $\mathbf{V}_{[i]}$). Suppose the learnable query is denoted as $\mathbf{Q} \in \mathbb{R}^{H\times r\times d}$, where r is a hyperparameter determining how many queries are used to summarize the KV cache in the current chunk. For each chunk,

doi.org/10.32388/G5H95J

StreamAdapter performs multi-head cross-attention between $\mathbf{Q}$ and $\mathbf{K}_{[i]}, \mathbf{V}_{[i]}$ to obtain the summarized result $\mathbf{S}_i$ for chunk i:

$$\mathbf{S}_i = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}_{[i]}^\top}{\sqrt{d}}\right)\mathbf{V}_{[i]} \in \mathbb{R}^{r\times d_{kv}}, \tag{4}$$

where $d_{kv} = H \times d$ is the hidden state dimension after concatenation across all heads, and $i \in \{0, 1, \ldots, L/C - 1\}$.

After obtaining the chunk-wise results $\{\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_{L/C-1}\}$, it is necessary to further aggregate the information across different chunks. This aggregation should consider locality, as the most recent information is likely to be more relevant to subsequent generation processes. Therefore, we employ a gated inter-chunk recurrence to aggregate this information:

$$\mathbf{h}_i = \alpha_i \mathbf{h}_{i-1} + \mathbf{S}_i \in \mathbb{R}^{r\times d_{kv}}, \tag{5}$$

where $\mathbf{h}_0$ is initialized to zero and α_i is a per-query scalar forget gate. Given recent research suggesting that data-dependent gating demonstrates more expressive power[39], StreamAdapter determines each gating factor α_i with the following parameterization:

$$\alpha_i = \sigma(\mathbf{S}_i\mathbf{W}_\alpha + b_\alpha)^{\frac{1}{\tau}} \in \mathbb{R}, \tag{6}$$

where $\mathbf{W}_\alpha \in \mathbb{R}^{d_{kv}\times 1}$, $\sigma(\cdot)$ is sigmoid function, and $\tau = 16$ is a temperature term that encourages the model to have a slower forgetting rate[40]. Through the data-dependent approach, the final context state $\mathbf{h}_{L/C-1}$ condenses the information from the KV cache. This condensed state, $\mathbf{h}_{L/C-1}$, is then integrated into the parameters of the pre-trained model using the low-rank adaptation method, thereby serving as the updated weight-level associative memory[28].

3.3. Weight Absorption

We expect the context states $\mathbf{h}$ to serve as newly learned knowledge from the context, which can be absorbed into the model's weights. Drawing inspiration from the low-rank adaptation method[32], StreamAdapter assigns learnable queries to each linear layer in the pre-trained model and maps the KV cache corresponding to the block where the current linear layer resides to the context state using these queries. The parameters of the linear layer are then updated by integrating the context state with two low-rank matrices in a sandwich-like structure (Figure 1).

Specifically, a typical transformer-based LLMs is built by stacking a series of identical blocks, each containing a multi-head self-attention (MHA) layer and a FFN layer. Each block stores the KV cache computed by its MHA layer. Therefore, for each parameter $\mathbf{W} \in \mathbb{R}^{d_i\times d_o}$ (where d_i and d_o denote the input and output dimensions,

respectively) of the linear layer in the l-th block, and the stored KV cache $\mathbf{K}^l, \mathbf{V}^l$ of that block, StreamAdapter assigns a unique learnable query $\mathbf{Q}$ to each $\mathbf{W}$ and summarize $\mathbf{K}^l$ and $\mathbf{V}^l$ into the context state $\mathbf{h}$, following Equations 4 and 5. This strategy of summarizing context with a unique query for each parameter allows the compression process to be adaptively learned from data for different weights. Finally, two low-rank learnable matrices, $\mathbf{W}_1 \in \mathbb{R}^{d_i \times d_{kv}}$ and $\mathbf{W}_2 \in \mathbb{R}^{r \times d_o}$, are connected through $\mathbf{h}$ to absorb the context information into $\mathbf{W}$:

$$\mathbf{W}' = \mathbf{W} + \mathbf{W}_1 \mathbf{h}^{\mathbf{T}} \mathbf{W}_2. \tag{7}$$

The second term, $\mathbf{W}_1 \mathbf{h}^{\mathbf{T}} \mathbf{W}_2$, can be interpreted as a simplified form of linear attention[41] with context-informed keys and a fixed value prototype[42]. From this interpretation, the input $x \in \mathbb{R}^{d_i}$ is first projected to the KV dimension d_{kv} via $\mathbf{W}_1$ and then used to compute the dot product similarity $x\mathbf{W}_1\mathbf{h}^T$ with the context state $\mathbf{h}$. This similarity weights the pre-learned prototype $\mathbf{W}_2$ and produces the output that is updated through the new context-informed weight.

In implementation, considering that the KV dimension d_{kv} is typically large in current LLMs (e.g., 1024 for LLaMA-3-8B), resulting in a significant number of new learnable parameters, StreamAdapter introduces an additional down-projection learnable parameter $\mathbf{W}_{down} \in \mathbb{R}^{H \times d \times d'}$ (where $d' \ll d$) to reduce the parameter size by down-projecting the $\mathbf{V}$ cache:

$$\mathbf{V}' = \mathbf{V}\mathbf{W}_{down} \in \mathbb{R}^{H \times L \times d'}. \tag{8}$$

As a result, the original dimension $d_{kv} = H \times d$ is projected to $d'_{kv} = H \times d'$, thereby reducing the number of the learnable parameters.

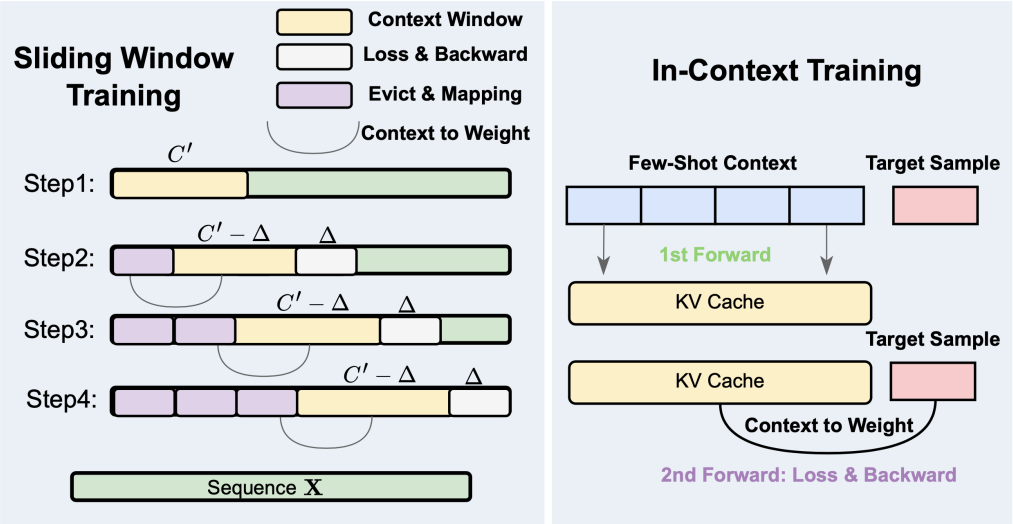

Figure 2. Training strategy of StreamAdapter. The sliding-window strategy accumulates loss from each step in a sequence and updates StreamAdapter's parameters after the entire sequence has been processed. The in-context training employs a 2-forward-1-backward strategy: the first forward pass computes the KV cache without gradient computation, while the second forward pass updates the model parameters using the KV cache from the first forward pass and calculates the loss to update the parameters introduced by StreamAdapter

3.4. Training Strategy

StreamAdapter's reliance on the KV cache for parameter updates necessitates a departure from conventional next-token prediction training methods. To address this, we have developed two distinct training strategies tailored to the specific requirements of language generation and language understanding tasks: sliding window training and in-context training (Figure 2).

3.4.1. Sliding Window Training

For general language generation tasks, we employ a sliding window strategy to train StreamAdapter for mapping context into parameter updates. This process begins with general language corpora, which we divide into sequences $\mathbf{X}$ of length L. For each sequence, we utilize a window size C' and a stride size Δ. It's important to note that C' here is larger than the context length C used in Section 3.2. We start by initializing the window with the first C' tokens of $\mathbf{X}$. Then, we begin an iterative process. In each step, we evict the earliest Δ tokens. The KV caches of these evicted tokens are then used to generate parameter updates. Simultaneously, we calculate the next token prediction loss for the incoming Δ tokens using the updated parameters. As we progress through the sequence, the loss is accumulated until the entire sequence $\mathbf{X}$ has been processed. Once

the sequence is fully seen, we update StreamAdapter's parameters using this accumulated loss. This sliding window approach enables StreamAdapter to efficiently process long sequences while maintaining a fixed context size. By continuously updating the window and accumulating loss, the model learns to utilize context information effectively across various positions in the input sequence.

3.4.2. In-Context Training

To adapt StreamAdapter for language understanding tasks, we employ an in-context training strategy using a selected set of tasks. For each sample in each task's training set, we first randomly sample k examples to form a few-shot context and store their KV caches (1st forward pass without gradient computation). We then update the base model parameters using this cache and compute the loss for the current sample to optimize the parameters introduced by StreamAdapter (2nd forward pass with backpropagation).

3.5. Inference Strategy

For model inference, inspired by context-locality[12][13], we adopt a hybrid approach tailored to different task types: For language understanding tasks, we convert most demonstrations into weight updates, retaining only a small portion of recent context. For long context generation tasks, we use a sliding window strategy with stride size Δ smaller than window size C'. We keep the most recent context intact while transforming the evicted context into temporary model updates via StreamAdapter. This adaptive strategy balances immediate context and adapted knowledge from earlier inputs, optimizing efficiency and performance across different scenarios.

4. Experiments and Results

We evaluate StreamAdapter across various model scales and architectures, focusing on both language understanding tasks and language generation tasks. We also explore the scaling ability of StreamAdapter with different numbers of in-context demonstrations across various tasks and lengths. Additionally, We evaluate StreamAdapter's efficiency and robustness through comprehensive ablation studies and in-depth analyses.

4.1. Experimental Setting

Base Model

we select TinyLlama-1.1B[43], LLaMA-3-8B, and Phi-3-Medium[44] as our base model. In all experiments, we froze the original model weights and only trained the parameters introduced by StreamAdapter.

Base Setting

Without explicit specialization, we apply StreamAdapter to every linear layer of the pre-trained model. The default chunk size C in Section 3.2 is set to 128, and the down-projected value dimension d'_{kv} in Equation 8 is set to 32 for all base models. When performing chunk-wise cross-attention, in cases where the input KV cache is not divisible by C, we performe an additional cross-attention operation on the remaining KV cache after division and concatenate the result with the chunk-wise result. The number of learnable queries in StreamAdapter is set to 16 for TinyLlama-1.1B and LLaMA-3-8B, and 48 for Phi-3-Medium.

4.2. Language Understanding Task

Training Details

For adapting StreamAdapter to language understanding tasks, we employ the in-context training approach introduced in Section 3.4.2. We carefully select several tasks for training. The tasks included BoolQ [45], CoPA [46], SST2 [47], CB [48], and RTE [49]. For each sample in training set across all tasks, we randomly select context examples from the training set for computing the KV cache in the first forward pass. The number of demonstrations tailored to each model's capacity: 30 samples for TinyLlama-1.1B, and 60 samples for both LLaMA-3-8B and Phi-3-Medium. For further training detailes, please refer to Appendix A.1.

Evaluation and Baseline

We evaluate StreamAdapter across a diverse set of language understanding tasks, including both those encountered during the training stage and unseen tasks. Our comparison encompasses several baseline methods: zero-shot prompting, ICL, and two heuristic context eviction strategies—SnapKV [12] and H_2O [13]. We also include TempLoRA [31], a test-time low-rank adaptation method, as a baseline. For a fair comparison, we additionally incorporate results obtained after fine-tuning the base model using LoRA [32] with the same number of learnable parameters as StreamAdapter. For detailed settings of the different methods, please refer to Appendix B.1.

		Seen Task						Unseen Task						
		BoolQ	CoPA	SST2	CB	RTE	Avg.	Hellaswag	Winogrande	OBQA	ARC-C	ARC-E	PIQA	Avg.
Zero-shot	Tiny-Llama-1.1B	57.03	**78.00**	69.61	14.29	51.99	54.18	59.01	58.88	21.80	27.56	60.31	73.34	50.15
$ICL_{10-shot}$		63.39	76.00	78.21	51.79	49.10	63.70	**59.46**	59.83	26.20	30.61	64.81	**73.78**	52.45
LoRA		74.28	**78.00**	86.58	80.36	69.31	77.71	59.01	56.35	24.80	27.90	57.15	72.14	49.56
TempLoRA		57.61	74.00	71.33	39.29	56.68	59.78	59.42	60.46	21.60	27.82	62.12	72.09	50.59
H_2O		62.72	77.00	75.80	23.21	47.23	57.19	58.86	59.12	23.20	30.20	62.58	72.96	51.15
SnapKV		63.29	75.00	63.30	44.64	46.93	58.63	58.09	59.83	23.40	28.75	62.92	72.80	50.97
StreamAdapter		**81.77**	76.00	**89.56**	**85.71**	**82.67**	**83.14**	59.45	**59.91**	**27.00**	**31.06**	**64.93**	73.29	**52.61**
Zero-shot	LLaMA-3-8B	81.11	88.00	67.09	51.79	67.87	71.17	79.15	72.93	35.60	50.09	80.35	79.60	66.29
$ICL_{10-shot}$		83.49	89.00	95.07	83.93	79.06	86.11	**81.46**	78.45	36.40	52.43	82.79	80.58	68.69
LoRA		89.24	84.00	96.22	96.43	88.09	90.80	80.78	69.14	35.40	51.62	80.77	78.67	66.06
TempLoRA		81.77	91.00	90.60	76.79	70.40	82.11	79.62	76.56	34.20	50.85	81.06	79.27	66.93
H_2O		82.48	88.00	92.43	73.21	75.81	82.39	80.57	75.77	33.80	51.45	82.62	79.98	67.37
SnapKV		84.46	88.00	94.41	75.00	74.01	83.18	80.41	76.95	35.20	50.77	82.74	79.65	67.62
StreamAdapter		**90.15**	**89.00**	**96.72**	**98.21**	**89.67**	**92.75**	80.87	**78.64**	**38.80**	**53.03**	**83.53**	**80.59**	**69.24**
Zero-shot	Phi-3-Medium	88.65	92.00	90.48	73.21	80.51	84.97	79.12	76.48	37.80	54.95	80.85	80.90	68.35
$ICL_{10-shot}$		88.23	**94.00**	95.07	83.93	77.62	87.77	80.70	**79.40**	**44.80**	62.80	86.87	**82.37**	72.82
LoRA		88.59	92.00	**96.22**	**92.86**	89.53	91.84	79.36	76.87	41.80	58.60	84.22	81.18	70.34
TempLoRA		89.27	92.00	93.81	71.43	78.34	84.97	79.30	76.56	44.40	60.58	86.28	80.61	71.29
H_2O		83.46	**94.00**	94.95	80.46	79.78	86.53	80.61	76.11	40.80	62.12	86.41	82.07	71.35
SnapKV		78.99	89.00	70.41	80.36	68.95	77.54	80.21	72.53	41.30	61.71	83.89	79.87	69.92
StreamAdapter		**90.24**	92.00	95.18	**92.86**	**89.89**	**92.03**	**82.03**	77.11	**44.80**	**64.33**	**86.91**	82.47	**72.94**

Table 1. Evaluation results on language understanding tasks after in-context training. OBQA: OpenbookQA. ARC-C: ARC-Challenge. ARC-E: ARC-Easy

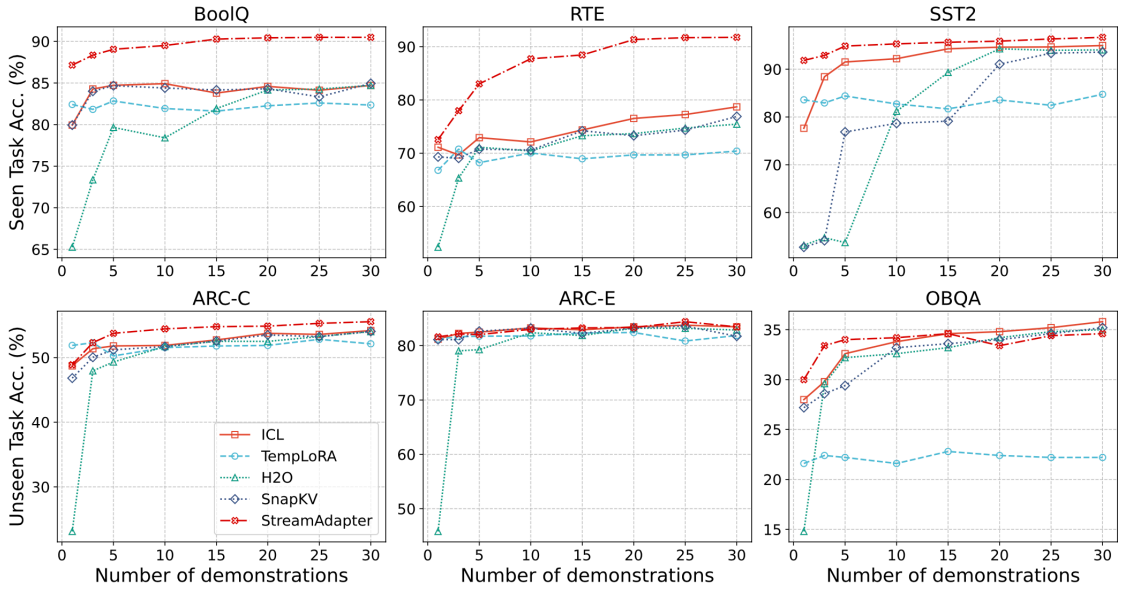

Figure 3. Comparison of various methods across different tasks with different numbers of demonstrations

Evaluation Result

The evaluation results in Table 1 show that StreamAdapter consistently outperforms LoRA on the test set across the seen tasks, despite using the same training recipe and parameter count. On unseen tasks, StreamAdapter also surpasses all other methods across the three models. Unlike context selection methods such as SnapKV and H_2O, which are upper-bounded by full ICL, StreamAdapter enhances model capability by absorbing context and even outperforms full ICL. Additionally, while LoRA exhibits performance degradation

doi.org/10.32388/G5H95J

on unseen tasks, indicating catastrophic forgetting, StreamAdapter maintains improved results, demonstrating its effectiveness and generalization capabilities.

Scaling Analysis

We examine the adaptation accuracy of various methods, including StreamAdapter, as the number of demonstrations increases across different tasks using the LLaMA-3-8B model. To ensure fair comparison across methods, we employ a consistent approach to demonstration selection. For each task, we generate a fixed set of demonstrations from its training set. All test samples are then evaluated using this same set of demonstrations across all methods. For detailed configuration information, please refer to Appendix B.2.

Figure 3 shows that StreamAdapter consistently improves with more demonstrations on both seen and unseen tasks. On both seen and unseen tasks, StreamAdapter significantly outperforms TempLoRA and achieves better results than ICL and other context eviction strategies. The increasing accuracy with more demonstrations, particularly on unseen tasks, suggests that StreamAdapter effectively leverages contextual information to encode knowledge into parameters rather than simply memorizing task-specific patterns.

These results highlight StreamAdapter's potential as a robust approach for TTA in language models, demonstrating its ability to generalize across diverse language understanding tasks.

4.2.1. Language Generation Task

Training Details

For training StreamAdapter on language generation tasks, we utilize the training set of the PG19 dataset[50], employing the sliding window strategy introduced in Section 3.4.1. The sequence length L is set to 8192 for TinyLlama-1.1B, and 16384 for LLaMA-3-8B and Phi-3-Medium. The SW size C' for all models is fixed at 1024, with a stride Δ of 512. For additional training hyperparameters, please refer to Appendix A.2.

		16K	32K	64K	128K	256K
TinyLlama-1.1B	Sliding Window	11.16	11.24	11.29	11.29	11.28
	TempLoRA	11.04	11.03	10.99	10.93	10.91
	StreamAdapter	**10.81**	**10.88**	**10.70**	**10.44**	**9.97**
LLaMA-3-8B	Sliding Window	9.44	9.52	9.51	9.50	9.50
	TempLoRA	9.73	9.81	9.81	9.83	9.84
	StreamAdapter	**9.27**	**9.33**	**9.23**	**9.22**	**9.21**

Table 2. Comparison on PG19 test set across varying maximum context lengths using sliding window evaluation strategy

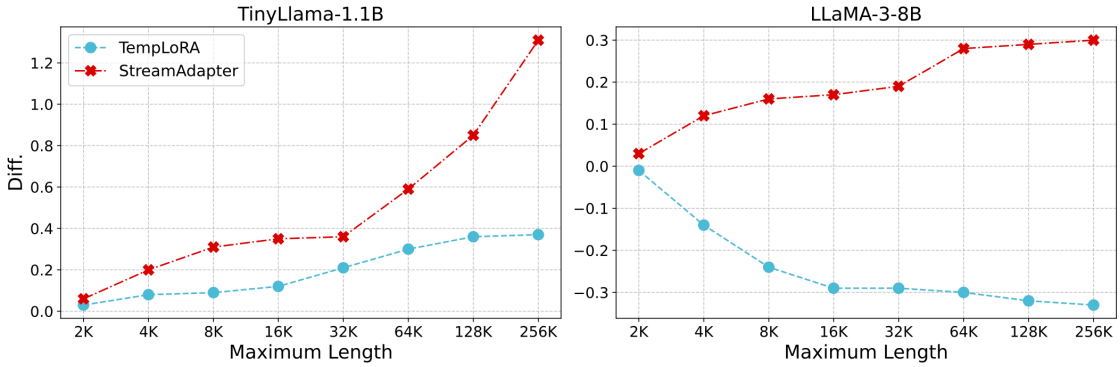

Figure 4. Perplexity gap between TTA methods and sliding window strategy across varying maximum context lengths on the PG19 test set

Evaluation and Baselines

We evaluate StreamAdapter on the PG19 test set using various maximum truncation lengths. For each sample, we employ the sliding window evaluation strategy with a window size C' of 1024 and a stride Δ of 512. Perplexity is computed in the incoming stride window, and we report the average perplexity across the entire test set. For comparison, we use two baselines: naive sliding window approach with identical settings, and TempLoRA. Detailed parameter settings for TempLoRA are provided in Appendix B.3.

Evaluation Result

The results are presented in Table 2. They clearly demonstrate that StreamAdapter outperforms both the sliding window approach and TempLoRA across all maximum context length. Notably, while TempLoRA achieves lower generation perplexity than sliding window with TinyLlama-1.1B, it shows inferior performance when evaluated with LLaMA-3-8B. We hypothesize that this discrepancy may be due to LLaMA-3-8B's training on high-quality corpora. TTA with TempLoRA on the current chunk might lead LLaMA-3 to overfit to that chunk, resulting in inaccurate predictions on subsequent context. In contrast, StreamAdapter exhibits superior generation performance on both TinyLlama-1.1B and LLaMA-3-8B models, showcasting its wide applicability across different model scales. Moreover, we visualize the perplexity gap between StreamAdapter and TempLoRA compared to the sliding window approach at different maximum lengths in Figure 4. The gap consistently widens as the context size increases for both TinyLlama-1.1B and LLaMA-3-8B models. The consistent improvement across increasing context lengths highlights StreamAdapter's ability to effectively leverage additional contextual information, regardless of the base model's scale. This scalability further

emphasizes StreamAdapter's robustness and adaptability in processing long-form text, making it particularly suitable for applications requiring efficient handling of extensive contextual data.

4.3. Analysis

Efficiency

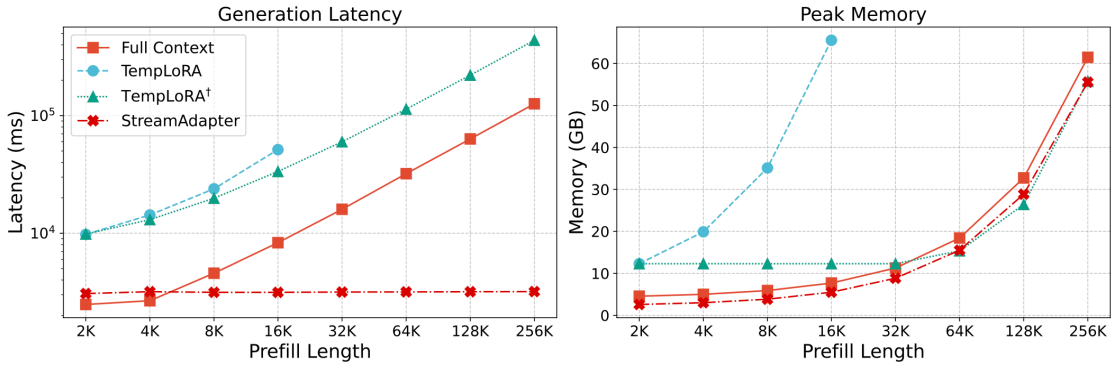

Figure 5. Generation latency and peak memory consumption across different prefill lengths. † indicates adaptation using sequential chunk-wise strategy, as directly mapping all prefill context leads to out-of-memory

We compare the end-to-end latency and peak memory consumption of model generation with TinyLlama-1.1B across various prefill context lengths. Our evaluation process begins by generating the KV cache for a given prefill context length, followed by measuring the latency of generating 128 tokens using three methods: full context, TempLoRA, and our StreamAdapter.

The hyperparameter settings for TempLoRA and StreamAdapter are consistent with those described in Section 4.2.1. All results are averaged across five runs with a single NVIDIA A100-80G GPU.

The results, presented in Figure 5, clearly demonstrate that StreamAdapter maintains constant generation latency across different prefill context lengths (i.e., different KV cache sizes). In contrast, the latency of full context generation and TTA with TempLoRA increases almost linearly with the context size. Moreover, TempLoRA's need for gradient backpropagation during adaptation leads to substantial GPU memory consumption as the prefill context increases. While this can be mitigated using sequential chunk-wise adaptation (with a chunk size of 2048 in our setting), this approach increases the generation latency. Conversely, StreamAdapter's recurrent design allows simultaneous mapping of all context without requiring sequential chunk-wise processing. Although StreamAdapter's peak memory consumption also increases with larger prefill contexts, we attribute this to the current implementation materializing all intermediate states. As

doi.org/10.32388/G5H95J

only the final state is needed, we believe further optimizations, similar to those in[38], could reduce StreamAdapter's memory demands.

Adaptation Ratio

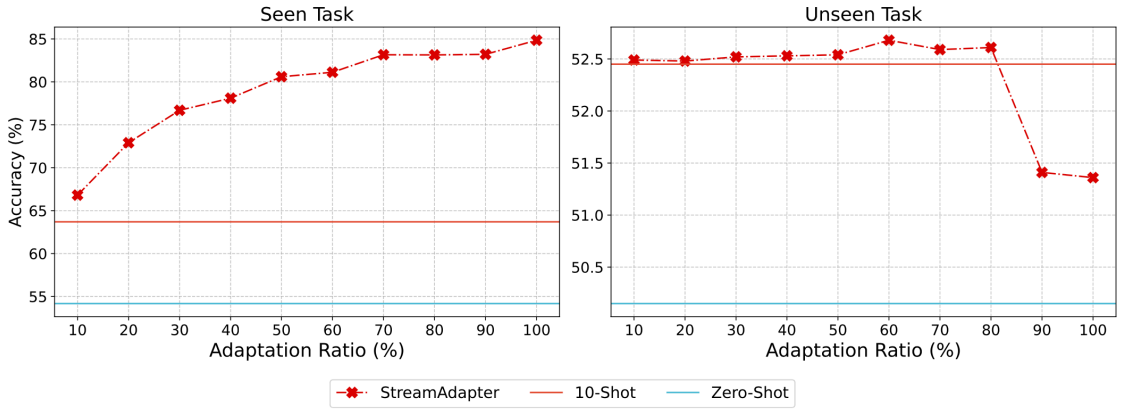

Figure 6. Average accuracy of StreamAdapter with TinyLlama-1.1B across different adaptation ratios on both seen and unseen tasks

In the language understanding tasks described in Section 4.2, we adapt a fixed ratio of context into model weights and evaluate the model with the remaining context in context. To explore the relationship between adaptation ratio and final accuracy on both seen and unseen tasks, we evaluate TinyLlama-1.1B with fixed 10-shot samples across different adaptation ratios.

The results of our adaptation ratio analysis are presented in Figure 6. For a more detailed breakdown of acuracy on each individual task, please refer to Appendix C.2. StreamAdapter generally performs better on seen tasks when adapting more demonstrations. For unseen tasks, StreamAdapter outperforms 10-shot ICL when adapting 10%-80% of demonstrations but shows a decline with extreme adaptation ratios (90% or 100%). Although teh adaptation accuracy remains better than zero-shot prompting, we hypothesize that retaining a small portion of demonstrations is necessary to guide adaptation direction on unseen tasks. This is likely because StreamAdapter learns mapping relations from a limited set of tasks and may adapt the base model in a direction different from the target unseen task. We posit that training StreamAdapter with a more diverse task set could address this issue, which we leave for future work.

Robustness

	Seen Task				Unseen Task			
	BoolQ	SST2	RTE	Avg.	ARC-C	ARC-E	PIQA	Avg.
$ICL_{10-shot}$	53.06	50.92	48.10	50.69	25.17	50.13	72.80	49.37
StreamAdapter	**71.38**	**50.92**	**82.03**	**68.11**	**25.43**	**50.67**	**72.96**	**49.69**

Table 3. Evaluation results on language understanding tasks with different prompt templates for in-context examples and evaluated samples

We evaluate the influence of using different templates for in-context examples and target evaluated samples to analyze the robustness of different TTA methods as patterns change. For this analysis, we use the TinyLlama-1.1B model trained from Section 4.2. We select three seen tasks (BoolQ, SST2, RTE) and three unseen tasks (ARC-Challenge[51], ARC-Easy[51], PIQA[52]) to verify the robustness of full in-context learning (ICL) and StreamAdapter. We fix the number of in-context examples at 10, with other details for StreamAdapter remaining the same as in Section 4.2.

The results, presented in Table 3, show that although both full ICL and StreamAdapter exhibit degraded accuracy compared to Table 1, StreamAdapter still outperforms ICL on both seen and unseen tasks. Moreover, as illustrated in Figure 7, ICL's average accuracy decreases as the number of in-context examples increases, suggesting that ICL primarily memorizes patterns and fails to adapt when these patterns change. Conversely, StreamAdapter consistently achieves higher accuracy with additional demonstrations, indicating that TTA with StreamAdapter leverages contextual information to enhance model capability rather than simply memorizing task-specific patterns.

doi.org/10.32388/G5H95J

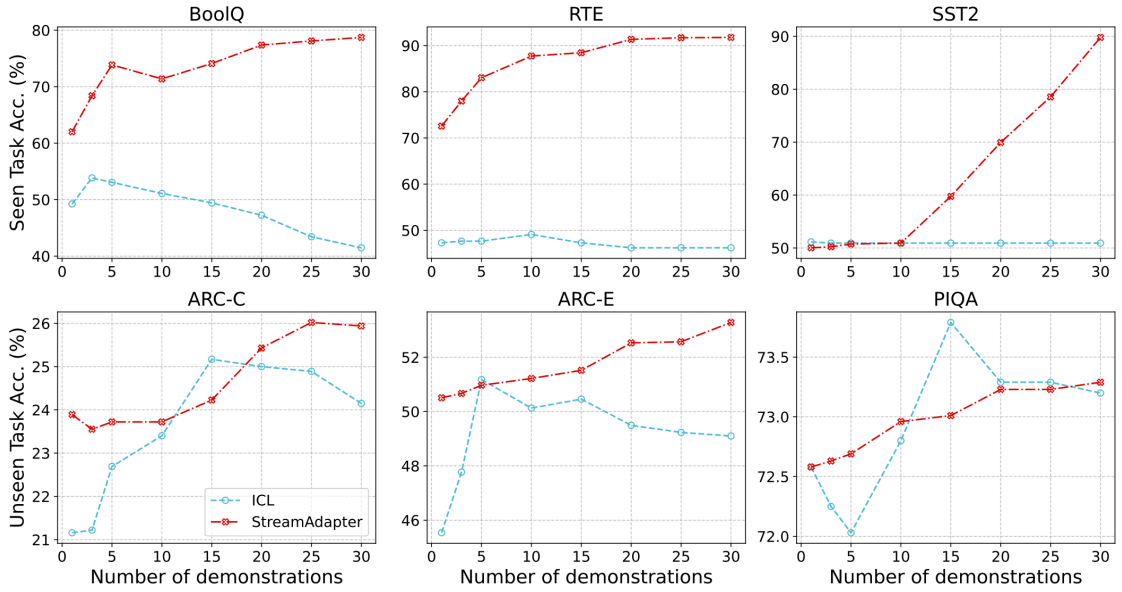

Figure 7. Evaluation with varying numbers of demonstrations, using different prompt templates for evaluated samples and in-context examples

ChunkWise	Seen Task Avg.	Unseen Task Avg.
No Chunking	82.84	51.46
64	82.23	**51.47**
128	**86.99**	51.92
256	86.10	51.71

Table 4. Evaluation on language understanding task with different chunk size

doi.org/10.32388/G5H95J

# of Query	# of Learnable Param. (%)	Seen Task Avg.	Unseen Task Avg.
16	4.99	83.14	**52.61**
32	5.55	86.50	52.01
64	6.66	87.21	51.99
128	8.81	**87.22**	52.05

Table 5. Evaluation on language understanding task with different number of learnable query for summarizing each chunk

# of Query	# Learnable Param.	Seen Task Avg.	Unseen Task Avg.
8	4.20	85.33	52.30
16	4.99	83.14	52.61
32	5.55	87.80	52.50

Table 5. Evaluation on language understanding task with different number of learnable query for summarizing each chunk

Chunk Size	# of Query	# Learnable Param.	Seen Task Avg.	Unseen Task Avg.
64	8	4.20	85.33	52.30
128	16	4.99	83.14	52.61
256	32	5.55	87.80	52.50

Table 6. Evaluation results on language understanding task with fixed chunk size / query ratio using TinyLlama-1.1B

4.4. Ablation

We examine the impact of different components and settings of StreamAdapter, focusing our analysis on the TinyLlama-1.1B model and evaluating its adaptation capability on language understanding tasks. Except for the

specific parameter settings under investigation, all other training and evaluation settings remain consistent with those described in Section 4.2.

We begin by examining the effectiveness of the chunk-wise design and the influence of chunk size on StreamAdapter's performance. For this analysis, we fix the number of queries used to compress each chunk at 16. In the absence of a chunk-wise design, we would directly summarize the entire KV cache using queries with cross-attention to generate new model weights, eliminating the need for inter-chunk recurrence. Table 4 shows that the chunk-wise approach outperforms the non-chunked version, with a chunk size of 128 achieving the best results on both seen (86.99%) and unseen tasks (51.92%). Smaller (64) and larger (256) chunk sizes show suboptimal results, indicating that 128 strikes the right balance in capturing contextual information with 16 queries.

Next, we examine the effect of different numbers of queries, with results presented in Table 5. As the number of queries per chunk increases, accuracy improves on seen tasks but declines on unseen tasks. We hypothesize that increasing the number of learnable parameters through additional queries causes StreamAdapter to trend towards memorizing fixed patterns from training tasks, resulting in poorer generalization to unseen tasks.

From the results in Tables 4 and 5, we hypothesize that the optimal ratio of context tokens per query is 128 / 16 = 8. To validate this hypothesis, we conduct experiments with different chunk sizes while maintaining this fixed ratio. The results from Table 6 show that maintaining this ratio does improve the adaptation accuracy on both seen and unseen tasks compared to the results from Table 4. However, the highest accuracy is still achieved with the original chunk size of 128 and 16 queries. We hypothesize that this optimal configuration may be related to the context length of each task's in-context examples (presented in Table 3). Further analysis of this relationship is left for future work.

5. Conclusion

We introduce StreamAdapter, a novel approach for adapting pretrained LLMs at test time directly from given context. StreamAdapter employs context mapping and weight absorption mechanisms to efficiently transform context tokens into parameter updates, achieving similar or superior results to full-context generation while reducing both memory consumption and inference time. Evaluations across diverse language understanding and generation tasks with various model scales demonstrate StreamAdapter's effectiveness in adapting to new tasks, outperforming fine-tuning and zero-shot prompting, while also surpassing full ICL. Analysis reveals StreamAdapter's superior scalability and robustness across varying context lengths and adaptation ratios, while maintaining constant inference time and memory consumption. These promising results open new avenues for efficient TTA of LLMs, paving the way for more flexible and customized language model

deployments. Future work could explore StreamAdapter's application to more diverse tasks and larger model scales, potentially extending its principles to other modalities.

Appendix A. Training Details

A.1. Language Understanding Task

We use the training sets of BoolQ[45], CoPA[46], SST2[47], CB[48], and RTE[49] for training on language understanding tasks. We construct each sample with pre-defined template, the template for each task is presented in Table 7.

For training StreamAdapter, we employ the WarmupCosine learning rate scheduler and the AdamW optimizer with $(\beta_1, \beta_2) = (0.9, 0.95)$ and weight decay 0.01 for 3 epochs. The hyperparameters vary across models: for TinyLlama-1.1B, we use a batch size of 16, learning rate of 5×10^{-5}, and 100 warmup steps; for LLaMA-3-8B, a batch size of 4, learning rate of 2×10^{-5}, and 500 warmup steps; and for Phi-3-Medium, a batch size of 2, learning rate of 1×10^{-5}, and 800 warmup steps.

For training LoRA, we apply the adapter to every linear layer of the pre-trained model and use the same learning rate scheduler, optimizer, and number of epochs as for training StreamAdapter. The rank and α of LoRA are both set to 64. The hyperparameters are adjusted for each model: TinyLlama-1.1B uses a batch size of 16, learning rate of 1×10^{-4}, and 100 warmup steps; LLaMA-3-8B uses a batch size of 8, learning rate of 8×10^{-5}, and 300 warmup steps; and Phi-3-Medium uses a batch size of 4, learning rate of 5×10^{-5}, and 500 warmup steps.

A.2. Language Generation Task

For training on language generation tasks, we utilize the training set of the PG19 dataset. We employ the WarmupCosine learning rate scheduler with 500 warmup steps and the AdamW optimizer with $(\beta_1, \beta_2) = (0.9, 0.95)$ and weight decay 0.01 for 1 epoch. The hyperparameters are adjusted for each model: TinyLlama-1.1B uses a batch size of 8 and a learning rate of 5×10^{-5}; LLaMA-3-8B uses a batch size of 4 and a learning rate of 2×10^{-5}; and Phi-3-Medium uses a batch size of 2 and a learning rate of 1×10^{-5}.

```
BoolQ:
{passage}\nQuestion: {question}\nAnswer:
CB:
{premise}\nQuestion: {hypothesis}. True, False, or Neither?\nAnswer:
CoPA:
{premise} so/because {candidate}
SST2:
{sentence}\nQuestion: Is this sentence positive or negative?\nAnswer:
RTE:
{premise}\nQuestion: {hypothesis} True or False?\nAnswer:
```

Table 7. Templates used for each task in training on language understanding tasks

B. Evaluation Details

B.1. Language Understanding Task

Unless otherwise specified, we use the task templates introduced in lm-evaluation-harness[53] for all our evaluations on language understanding tasks. We report the accuracy for task BoolQ, CoPA, SST2, CB, RTE, OpenbookQA, ARC-Challenge, Winogrande, PIQA, and ARC-Easy, while report the normalized accuracy for Hellaswag.

For a fair comparison when using multi-shot demonstration contexts, we generate the required number of demonstrations from the training set of each task. These same demonstrations are then used as context for evaluating all methods. This approach eliminates potential variability due to demonstration selection, allowing for a more direct comparison of different methods. The results we report are averaged from three independent runs.

TempLoRA: We apply LoRA to every linear layer of the base model and directly train it on the given in-context examples. For optimization, we use the AdamW optimizer with a OneCycleLR learning rate scheduler. The rank and α of LoRA are both set to 64 across all models. We use a fixed learning rate of 1×10^{-5} and train for 5 epochs.

H_2O: We retain 20% of the context, with both the heavy ratio and recent ratio set to 0.1.

SnapKV: For SnapKV, we allocate 10% of the context for the observation window and retain an additional 10% for inference, leading to a total context retention of 20%.

StreamAdapter: Unless otherwise specified, we convert 80% of the context into a parameter update, leaving the remaining 20% of the context unchanged.

Task Name	Context Length							
	1-shot	3-shot	5-shot	10-shot	15-shot	20-shot	25-shot	30-shot
BoolQ	275	475	744	1218	2219	2992	4190	4567
CoPA	18	44	86	145	232	316	374	450
SST2	28	91	121	290	412	470	705	872
CB	97	240	550	988	1533	1629	2703	3304
RTE	42	161	500	958	1240	1666	1984	2399
Hellaswag	63	298	382	567	1051	1470	1959	2129
Winogrande	25	77	129	231	322	416	527	638
OpenbookQA (OBQA)	20	53	76	164	267	344	393	437
ARC-Chanllege (ARC-C)	25	154	171	363	570	680	895	998
ARC-Easy (ARC-E)	29	129	194	367	426	802	983	1296
PIQA	23	191	257	371	598	817	973	1028

Table 8. The context length of different demonstration of different tasks using LLaMA-3-8B tokenizer

doi.org/10.32388/G5H95J

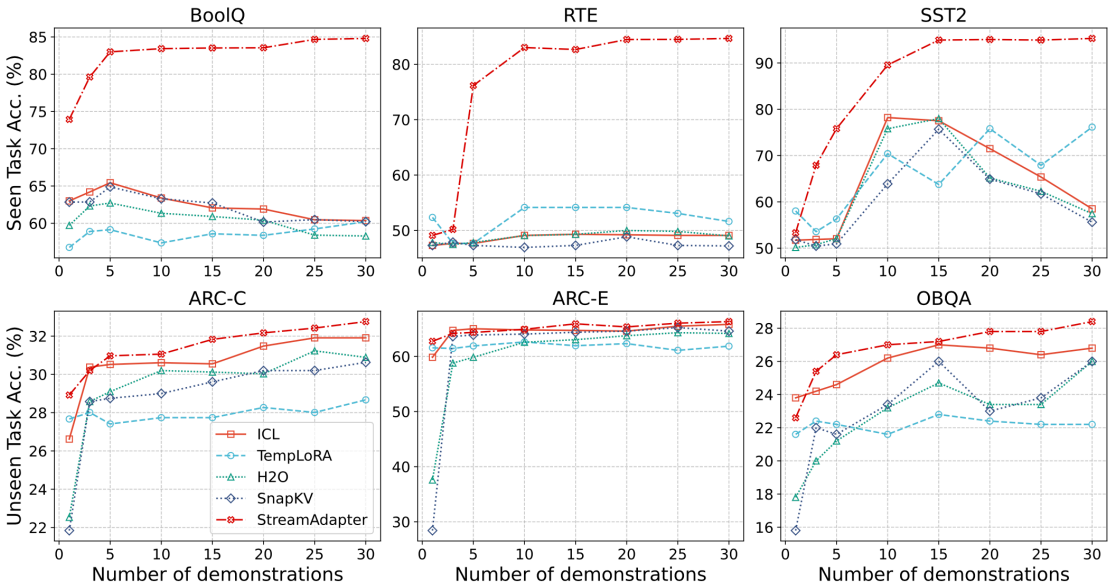

Figure 8. Comparison of various methods across different tasks using TinyLlama-1.1B with different numbers of demonstrations

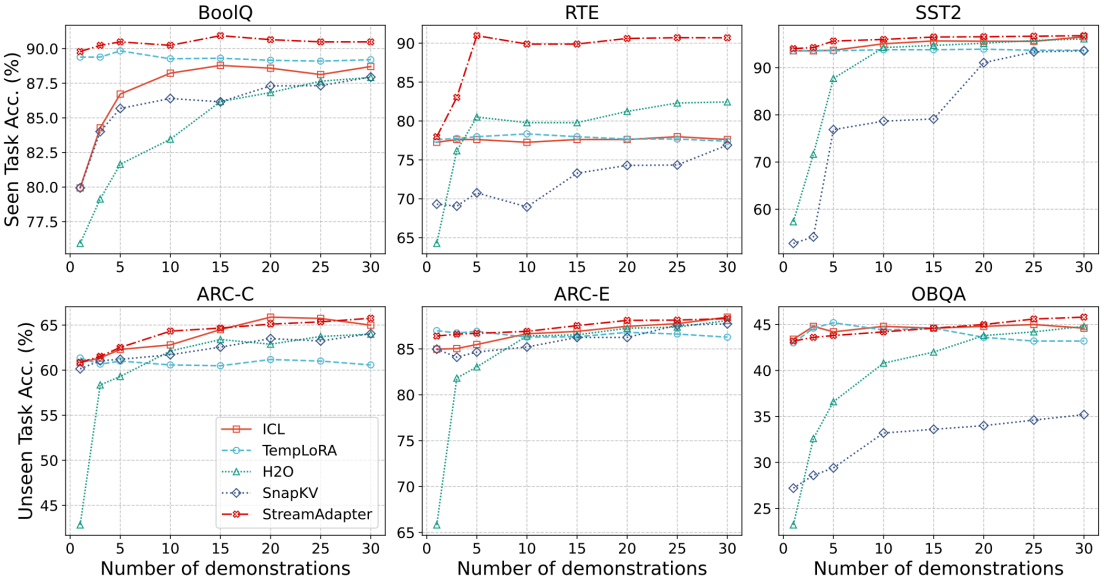

Figure 9. Comparison of various methods across different tasks using Phi-3 Medium with different numbers of demonstrations

doi.org/10.32388/G5H95J

B.2. Language Understanding Scaling Analysis

We evaluate different methods under varying numbers of demonstrations on six tasks: BoolQ, RTE, SST2, ARC-Challenge (ARC-C), ARC-Easy (ARC-E), and PIQA. To ensure a fair comparison, we employ a consistent approach across all methods. We first generate a fixed set of demonstrations for each task, which is then used as context for all methods being compared. Our evaluation covers 1, 3, 5, 10, 15, 20, 25, and 30-shot scenarios. The reported results are obtained by averaging three different runs, each utilizing a distinct set of generated demonstrations.

We also provide the average context length for each task across different numbers of demonstrations using the LLaMA-3-8B tokenizer in Table 8.

Adaptation Ratio	Seen Task						Unseen Task						
	BoolQ	CoPA	SST2	CB	RTE	Avg.	Hellaswag	Winogrande	OBQA	ARC-C	ARC-E	PIQA	Avg.
10%	70.15	76.00	80.25	55.64	51.99	66.81	59.62	60.46	26.00	30.12	65.03	73.72	52.49
20%	74.28	75.00	83.37	55.00	76.91	72.91	59.29	**60.69**	25.80	30.38	64.86	73.88	52.48
30%	77.83	**77.00**	84.17	62.86	81.59	76.69	59.39	60.38	26.00	30.38	64.90	74.05	52.52
40%	79.60	76.00	85.08	66.07	83.75	78.10	**59.83**	60.06	25.80	30.38	65.19	73.94	52.53
50%	80.06	75.00	83.14	81.07	**83.75**	80.60	59.28	59.80	26.00	30.55	**65.56**	74.05	52.54
60%	80.37	74.00	84.63	82.86	**83.75**	81.12	59.30	60.22	26.40	30.55	65.45	**74.16**	**52.68**
70%	81.47	75.00	89.79	85.77	**83.75**	83.16	59.67	59.35	**27.00**	**31.06**	65.07	73.39	52.59
80%	81.77	76.00	89.56	85.71	82.67	83.14	59.45	59.91	**27.00**	**31.06**	64.93	73.29	52.61
90%	82.36	75.00	90.31	85.72	82.67	83.21	59.54	59.04	23.00	29.60	64.10	73.18	51.41
100%	**83.36**	74.00	**93.81**	**90.43**	82.67	**84.85**	59.40	59.04	23.00	29.52	64.02	73.78	51.46

Table 9. Average accuracy of StreamAdapter on language understanding tasks using TinyLlama-1.1B, evaluated across different adaptation ratios

B.3. Language Generation Task

For TempLoRA, we apply the LoRA adapter to every linear layer, with the rank and α both set to 64.

B.4. Robustness Analysis

For evaluting the robustness of ICL and StreamAdapter adaption capability from different prompt template, we use different prompt template for in-contetx examples and targer sample. For in-context examples, we use the same prompt teamplte with lm-evaluation-harness[53], while the teamplte for the target sample are presented in Table 10.

```
BoolQ:
Given the passage: {passage}\nAnswer the following question: {question}?
SST2:
Could you tell me is this <{sentence}> positive or negative?
RTE:
{premise}\n{hypothesis} True or False?
ARC-Challenge:
{question}\nPlease answer the question.
ARC-Easy:
{question}\nPlease answer the question.
PIQA:
Answer the question: {question}.
```

Table 10. Templates used for each task in training on language understanding tasks

C. Additional Results

C.1. Scaling Analysis on Language Understanding Task

We further present the results on language understanding tasks with varying numbers of demonstrations for TinyLlama-1.1B and Phi-3-Medium in Figure 8 and Figure 9, respectively. These results further demonstrate that StreamAdapter clearly outperforms full ICL and other TTA methods. Moreover, StreamAdapter exhibits better scaling capability as the number of demonstrations increases.

C.2. Evaluation with Different Adaptation Ratio

Table 9 presents the detailed accuracy of StreamAdapteracross different adaptation ratios, as discussed in Section 4.3.

References

1. ^[Δ]*Yuan A, Coenen A, Reif E, Ippolito D (2022). "Wordcraft: Story Writing With Large Language Models". Proceeding s of the 27th International Conference on Intelligent User Interfaces.*

2. ^Kumar B, Lu C-C, Gupta G, Palepu A, Bellamy DR, Raskar R, Beam AL (2023). "Conformal prediction with large language models for multi-choice question answering". ArXiv. **abs/2305.18404**.
3. ^Zhang D, Li S, Zhang X, Zhan J, Wang P, Zhou Y, Qiu X (2023). "SpeechGPT: Empowering Large Language Models with Intrinsic Cross-Modal Conversational Abilities". In: Conference on Empirical Methods in Natural Language Processing.
4. ^Shao Z, Wang P, Zhu Q, Xu R, Song JM, Zhang M, Li YK, Wu Y, Guo D (2024). "DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models". ArXiv. **abs/2402.03300**. Available from: https://arxiv.org/abs/2402.03300.
5. a, b Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler DM, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, Amodei D (2020). "Language Models are Few-Shot Learners". arXiv. arXiv:2005.14165 [cs.CL].
6. ^Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, Chi E, Le Q, Zhou D (2023). "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". arXiv. arXiv:2201.11903 [cs.CL].
7. a, b Agarwal R, Singh A, Zhang LM, Bohnet B, Chan S, Anand A, Abbas Z, Nova A, Co-Reyes JD, Chu E, Behbahani FMP, Faust A, Larochelle H (2024). "Many-Shot In-Context Learning". ArXiv. **abs/2404.11018**. Available from: https://arxiv.org/abs/2404.11018.
8. ^Sahoo P, Singh AK, Saha S, Jain V, Mondal S, Chadha A (2024). "A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications". arXiv. arXiv:2402.07927.
9. ^Ding Y, Zhang LL, Zhang C, Xu Y, Shang N, Xu J, Yang F, Yang M (2024). "LongRoPE: Extending LLM Context Window Beyond 2 Million Tokens". ArXiv. **abs/2402.13753**.
10. ^Gemini Team, Georgiev P, Lei VI, Burnell R, Bai L, Gulati A, Tanzer G, Vincent D, Pan Z, Wang S, Mariooryad S, Ding Y, Geng X, Alcober F, Frostig R, Omernick M, Walker L, Paduraru C, Sorokin C, Tacchetti A, Gaffney C, Daruki S, Sercinoglu O, Gleicher Z, Love J, Voigtlaender P, Jain R, Surita G, Mohamed K, Blevins R, Ahn J, Zhu T, Kawintiranon K, Firat O, Gu Y, Zhang Y, Rahtz M, Faruqui M, Clay N, Gilmer J, Co-Reyes J, Penchev I, Zhu R, Morioka N, Hui K, Haridasan K, Campos V, Mahdieh M, Guo M, Hassan S, Kilgour K, Vezer A, Cheng H-T, de Liedekerke R, Goyal S, Barham P, Strouse D, Noury S, Adler J, Sundararajan M, Vikram S, Lepikhin D, Paganini M, Garcia X, Yang F, Valter D, Trebacz M, Vodrahalli K, Asawaroengchai C, Ring R, Kalb N, Soares LB, Brahma S, Steiner D, Yu T, Mentzer F, He A, Gonzalez L, Xu B, Kaufman RL, Shafey LE, Oh J, Hennigan T, van den Driessche G, Odoom S, Lucic M, Roelofs B, Lall S, Marathe A, Chan B, Ontanon S, He L, Teplyashin D, Lai J, Crone P, Damoc B, Ho L, Riedel S, Lenc K, Yeh C-K, Chowdhery A, Xu Y, Kazemi M, Amid E, Petrushkina A, Swersky K, Khodaei A, Chen G, Larkin C, Pinto M, Yan G, Badia AP, Patil P, Hansen S, Orr D, Arnold SMR, Grimstad J, Dai A, Douglas S, Sinha R, Yadav V, Chen X, Gribovskaya E, Austin J, Zhao J, Patel K, Komarek P, Austin S, Borgeaud S, Friso L, Goyal A, Caine B, Cao K, Chung D-W, Lamm M, Barth-Maron

G, Kagohara T, Olszewska K, Chen M, Shivakumar K, Agarwal R, Godhia H, Rajwar R, Snaider J, Dotiwalla X, Liu Y, Barua A, Ungureanu V, Zhang Y, Batsaikhan B-O, Wirth M, Qin J, Danihelka I, Doshi T, Chadwick M, Chen J, Jain S, L e Q, Kar A, Gurumurthy M, Li C, Sang R, Liu F, Lamprou L, Munoz R, Lintz N, Mehta H, Howard H, Reynolds M, Aro yo L, Wang Q, Blanco L, Cassirer A, Griffith J, Das D, Lee S, Sygnowski J, Fisher Z, Besley J, Powell R, Ahmed Z, Paulu s D, Reitter D, Borsos Z, Joshi R, Pope A, Hand S, Selo V, Jain V, Sethi N, Goel M, Makino T, May R, Yang Z, Schalkwyk J, Butterfield C, Hauth A, Goldin A, Hawkins W, Senter E, Brin S, Woodman O, Ritter M, Noland E, Giang M, Bolina V, Lee L, Blyth T, Mackinnon I, Reid M, Sarvana O, Silver D, Chen A, Wang L, Maggiore L, Chang O, Attaluri N, Thornto n G, Chiu C-C, Bunyan O, Levine N, Chung T, Eltyshev E, Si X, Lillicrap T, Brady D, Aggarwal V, Wu B, Xu Y, McIlroy R, Badola K, Sandhu P, Moreira E, Stokowiec W, Hemsley R, Li D, Tudor A, Shyam P, Rahimtoroghi E, Haykal S, Spre chmann P, Zhou X, Mincu D, Li Y, Addanki R, Krishna K, Wu X, Frechette A, Eyal M, Dafoe A, Lacey D, Whang J, Avra hami T, Zhang Y, Taropa E, Lin H, Toyama D, Rutherford E, Sano M, Choe H, Tomala A, Safranek-Shrader C, Kassne r N, Pajarskas M, Harvey M, Sechrist S, Fortunato M, Lyu C, Elsayed G, Kuang C, Lottes J, Chu E, Jia C, Chen C-W, Hu mphreys P, Baumli K, Tao C, Samuel R, dos Santos CN, Andreassen A, Rakićević N, Grewe D, Kumar A, Winkler S, Ca ton J, Brock A, Dalmia S, Sheahan H, Barr I, Miao Y, Natsev P, Devlin J, Behbahani F, Prost F, Sun Y, Myaskovsky A, Pi llai TS, Hurt D, Lazaridou A, Xiong X, Zheng C, Pardo F, Li X, Horgan D, Stanton J, Ambar M, Xia F, Lince A, Wang M, Mustafa B, Webson A, Lee H, Anil R, Wicke M, Dozat T, Sinha A, Piqueras E, Dabir E, Upadhyay S, Boral A, Hendrick s LA, Fry C, Djolonga J, Su Y, Walker J, Labanowski J, Huang R, Misra V, Chen J, Skerry-Ryan RJ, Singh A, Rijhwani S, Yu D, Castro-Ros A, Changpinyo B, Datta R, Bagri S, Hrafnkelsson AM, Maggioni M, Zheng D, Sulsky Y, Hou S, Paine TL, Yang A, Riesa J, Rogozinska D, Marcus D, Badawy DE, Zhang Q, Wang L, Miller H, Greer J, Sjos LL, Nova A, Zen H, Chaabouni R, Rosca M, Jiang J, Chen C, Liu R, Sainath T, Krikun M, Polozov A, Lespiau J-B, Newlan J, Cankara Z, Kwak S, Xu Y, Chen P, Coenen A, Meyer C, Tsihlas K, Ma A, Gottweis J, Xing J, Gu C, Miao J, Frank C, Cankara Z, Gana pathy S, Dasgupta I, Hughes-Fitt S, Chen H, Reid D, Rong K, Fan H, van Amersfoort J, Zhuang V, Cohen A, Gu SS, Mo hananey A, Ilic A, Tobin T, Wieting J, Bortsova A, Thacker P, Wang E, Caveness E, Chiu J, Sezener E, Kaskasoli A, Ba ker S, Millican K, Elhawaty M, Aisopos K, Lebsack C, Byrd N, Dai H, Jia W, Wiethoff M, Davoodi E, Weston A, Yagati L, Ahuja A, Gao I, Pundak G, Zhang S, Azzam M, Sim KC, Caelles S, Keeling J, Sharma A, Swing A, Li Y, Liu C, Bostoc k CG, Bansal Y, Nado Z, Anand A, Lipschultz J, Karmarkar A, Proleev L, Ittycheriah A, Yeganeh SH, Polovets G, Faust A, Sun J, Rrustemi A, Li P, Shivanna R, Liu J, Welty C, Lebron F, Baddepudi A, Krause S, Parisotto E, Soricut R, Xu Z, B loxwich D, Johnson M, Neyshabur B, Mao-Jones J, Wang R, Ramasesh V, Abbas Z, Guez A, Segal C, Nguyen DD, Sven sson J, Hou L, York S, Milan K, Bridgers S, Gworek W, Tagliasacchi M, Lee-Thorp J, Chang M, Guseynov A, Hartman AJ, Kwong M, Zhao R, Kashem S, Cole E, Miech A, Tanburn R, Phuong M, Pavetic F, Cevey S, Comanescu R, Ives R, Y ang S, Du C, Li B, Zhang Z, Iinuma M, Hu CH, Roy A, Bijwadia S, Zhu Z, Martins D, Saputro R, Gergely A, Zheng S, Jia D, Antonoglou I, Sadovsky A, Gu S, Bi Y, Andreev A, Samangooei S, Khan M, Kocisky T, Filos A, Kumar C, Bishop C, Y u A, Hodkinson S, Mittal S, Shah P, Moufarek A, Cheng Y, Bloniarz A, Lee J, Pejman P, Michel P, Spencer S, Feinberg

V, Xiong X, Savinov N, Smith C, Shakeri S, Tran D, Chesus M, Bohnet B, Tucker G, von Glehn T, Muir C, Mao Y, Kaza wa H, Slone A, Soparkar K, Shrivastava D, Cobon-Kerr J, Sharman M, Pavagadhi J, Araya C, Misiunas K, Ghelani N, Laskin M, Barker D, Li Q, Briukhov A, Houlsby N, Glaese M, Lakshminarayanan B, Schucher N, Tang Y, Collins E, Li m H, Feng F, Recasens A, Lai G, Magni A, Cao ND, Siddhant A, Ashwood Z, Orbay J, Dehghani M, Brennan J, He Y, Xu K, Gao Y, Saroufim C, Molloy J, Wu X, Arnold S, Chang S, Schrittwieser J, Buchatskaya E, Radpour S, Polacek M, Gior dano S, Bapna A, Tokumine S, Hellendoorn V, Sottiaux T, Cogan S, Severyn A, Saleh M, Thakoor S, Shefey L, Qiao S, Gaba M, Chang S-y, Swanson C, Zhang B, Lee B, Rubenstein PK, Song G, Kwiatkowski T, Koop A, Kannan A, Kao D, Schuh P, Stjerngren A, Ghiasi G, Gibson G, Vilnis L, Yuan Y, Ferreira FT, Kamath A, Klimenko T, Franko K, Xiao K, Bh attacharya I, Patel M, Wang R, Morris A, Strudel R, Sharma V, Choy P, Hashemi SH, Landon J, Finkelstein M, Jhakra P, Frye J, Barnes M, Mauger M, Daun D, Baatarsukh K, Tung M, Farhan W, Michalewski H, Viola F, de Chaumont Qu itry F, Le Lan C, Hudson T, Wang Q, Fischer F, Zheng I, White E, Dragan A, Alayrac J-b, Ni E, Pritzel A, Iwanicki A, Is ard M, Bulanova A, Zilka L, Dyer E, Sachan D, Srinivasan S, Muckenhirn H, Cai H, Mandhane A, Tariq M, Rae JW, W ang G, Ayoub K, FitzGerald N, Zhao Y, Han W, Alberti C, Garrette D, Krishnakumar K, Gimenez M, Levskaya A, Sohn D, Matak J, Iturrate I, Chang MB, Xiang J, Cao Y, Ranka N, Brown G, Hutter A, Mirrokni V, Chen N, Yao K, Egyed Z, Ga lilee F, Liechty T, Kallakuri P, Palmer E, Ghemawat S, Liu J, Tao D, Thornton C, Green T, Jasarevic M, Lin S, Cotruta V, Tan Y-X, Fiedel N, Yu H, Chi E, Neitz A, Heitkaemper J, Sinha A, Zhou D, Sun Y, Kaed C, Hulse B, Mishra S, Georgaki M, Kudugunta S, Farabet C, Shafran I, Vlasic D, Tsitsulin A, Ananthanarayanan R, Carin A, Su G, Sun P, V G, Carvaja l G, Broder J, Comsa I, Repina A, Wong W, Chen WW, Hawkins P, Filonov E, Loher L, Hirnschall C, Wang W, Ye J, Burn s A, Cate H, Wright DG, Piccinini F, Zhang L, Lin C-C, Gog I, Kulizhskaya Y, Sreevatsa A, Song S, Cobo LC, Iyer A, Tek ur C, Garrido G, Xiao Z, Kemp R, Zheng HS, Li H, Agarwal A, Ngani C, Goshvadi K, Santamaria-Fernandez R, Fica W, Chen X, Gorgolewski C, Sun S, Garg R, Ye X, Eslami SMA, Hua N, Simon J, Joshi P, Kim Y, Tenney I, Potluri S, Thiet LN, Yuan Q, Luisier F, Chronopoulou A, Scellato S, Srinivasan P, Chen M, Koverkathu V, Dalibard V, Xu Y, Saeta B, An derson K, Sellam T, Fernando N, Huot F, Jung J, Varadarajan M, Raul A, Le M, Habalov R, Clark J, Jalan K, Bullard K, Singhal A, Luong T, Wang B, Rajayogam S, Eisenschlos J, Jia J, Finchelstein D, Yakubovich A, Balle D, Fink M, Agarw al S, Li J, Dvijotham D, Pal S, Kang K, Konzelmann J, Beattie J, Dousse O, Wu D, Crocker R, Elkind C, Jonnalagadda S R, Lee J, Holtmann-Rice D, Kallarackal K, Liu R, Vnukov D, Vats N, Invernizzi L, Jafari M, Zhou H, Taylor L, Prendki J, Wu M, Eccles T, Liu T, Kopparapu K, Beaufays F, Angermueller C, Marzoca A, Sarcar S, Dib H, Stanway J, Perbet F, Tr din N, Sterneck R, Khorlin A, Li D, Wu X, Goenka S, Madras D, Goldshtein S, Gierke W, Zhou T, Liu Y, Liang Y, White A, Li Y, Singh S, Bahargam S, Epstein M, Basu S, Lao L, Ozturel A, Crous C, Zhai A, Lu H, Tung Z, Gaur N, Walton A, D ixon L, Zhang M, Globerson A, Uy G, Bolt A, Wiles O, Nasr M, Shumailov I, Selvi M, Piccinno F, Aguilar R, McCarthy S, Khalman M, Shukla M, Galic V, Carpenter J, Villela K, Zhang H, Richardson H, Martens J, Bosnjak M, Belle SR, Sei bert J, Alnahlawi M, McWilliams B, Singh S, Louis A, Ding W, Popovici D, Simicich L, Knight L, Mehta P, Gupta N, Shi C, Fatehi S, Mitrovic J, Grills A, Pagadora J, Petrova D, Eisenbud D, Zhang Z, Yates D, Mittal B, Tripuraneni N, Assael

Y, Brovelli T, Jain P, Velimirovic M, Akbulut C, Mu J, Macherey W, Kumar R, Xu J, Qureshi H, Comanici G, Wiesner J, G ong Z, Ruddock A, Bauer M, Felt N, GP A, Arnab A, Zelle D, Rothfuss J, Rosgen B, Shenoy A, Seybold B, Li X, Mudigon da J, Erdogan G, Xia J, Simsa J, Michi A, Yao Y, Yew C, Kan S, Caswell I, Radebaugh C, Elisseeff A, Valenzuela P, McKi nney K, Paterson K, Cui A, Latorre-Chimoto E, Kim S, Zeng W, Durden K, Ponnapalli P, Sosea T, Choquette-Choo C A, Manyika J, Robenek B, Vashisht H, Pereira S, Lam H, Velic M, Owusu-Afriyie D, Lee K, Bolukbasi T, Parrish A, Lu S, Park J, Venkatraman B, Talbert A, Rosique L, Cheng Y, Sozanschi A, Paszke A, Kumar P, Austin J, Li L, Salama K, K im W, Dukkipati N, Baryshnikov A, Kaplanis C, Sheng X, Chervonyi Y, Unlu C, de Las Casas D, Askham H, Tunyasuv unakool K, Gimeno F, Poder S, Kwak C, Miecnikowski M, Mirrokni V, Dimitriev A, Parisi A, Liu D, Tsai T, Shevlane T, Kouridi C, Garmon D, Goedeckemeyer A, Brown AR, Vijayakumar A, Elqursh A, Jazayeri S, Huang J, Mc Carthy S, H oover J, Kim L, Kumar S, Chen W, Biles C, Bingham G, Rosen E, Wang L, Tan Q, Engel D, Pongetti F, de Cesare D, Hw ang D, Yu L, Pullman J, Narayanan S, Levin K, Gopal S, Li M, Aharoni A, Trinh T, Lo J, Casagrande N, Vij R, Matthey L, Ramadhana B, Matthews A, Carey C, Johnson M, Goranova K, Shah R, Ashraf S, Dasgupta K, Larsen R, Wang Y, V uyyuru MR, Jiang C, Ijazi J, Osawa K, Smith C, Boppana RS, Bilal T, Koizumi Y, Xu Y, Altun Y, Shabat N, Bariach B, K orchemniy A, Choo K, Ronneberger O, Iwuanyanwu C, Zhao S, Soergel D, Hsieh C-J, Cai I, Iqbal S, Sundermeyer M, C hen Z, Bursztein E, Malaviya C, Biadsy F, Shroff P, Dhillon I, Latkar T, Dyer C, Forbes H, Nicosia M, Nikolaev V, Gree ne S, Georgiev M, Wang P, Martin N, Sedghi H, Zhang J, Banzal P, Fritz D, Rao V, Wang X, Zhang J, Patraucean V, Du D, Mordatch I, Jurin I, Liu L, Dubey A, Mohan A, Nowakowski J, Ion V-D, Wei N, Tojo R, Raad MA, Hudson DA, Kesha va V, Agrawal S, Ramirez K, Wu Z, Nguyen H, Liu J, Sewak M, Petrini B, Choi D, Philips I, Wang Z, Bica I, Garg A, Wilk iewicz J, Agrawal P, Li X, Guo D, Xue E, Shaik N, Leach A, Khan SM, Wiesinger J, Jerome S, Chakladar A, Wang AW, O rnduff T, Abu F, Ghaffarkhah A, Wainwright M, Cortes M, Liu F, Maynez J, Terzis A, Samangouei P, Mansour R, Kęp a T, Aubet F-X, Algymr A, Banica D, Weisz A, Orban A, Senges A, Andrejczuk E, Geller M, Santo ND, Anklin V, Merey MA, Baeuml M, Strohman T, Bai J, Petrov S, Wu Y, Hassabis D, Kavukcuoglu K, Dean J, Vinyals O. "Gemini 1.5: Unloc king multimodal understanding across millions of tokens of context". arXiv [cs.CL]. 2024. Available from: https://ar xiv.org/abs/2403.05530.

11. ^*Fu Y. Challenges in deploying long-context transformers: A theoretical peak performance analysis. ArXiv. 2024; **a bs/2405.08944**. Available from: https://arxiv.org/abs/2405.08944.*
12. a, b, c, d*Li Y, Huang Y, Yang B, Venkitesh B, Locatelli A, Ye H, Cai T, Lewis P, Chen D (2024). "SnapKV: LLM Knows Wh at You are Looking for Before Generation". **arXiv**. Available from: arXiv:2404.14469.*
13. a, b, c, d*Zhang Z, Sheng Y, Zhou T, Chen T, Zheng L, Cai R, Song Z, Tian Y, Ré C, Barrett C, Wang Z, Chen B (2023). "H $_2$O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models". arXiv:2306.14048 [cs.LG].*
14. ^*Coda-Forno J, Binz M, Akata Z, Botvinick M, Wang JX, Schulz E (2023). "Meta-in-context learning in large langua ge models". arXiv. arXiv:2305.12907.*

15. ^a, b, c, dDai D, Sun Y, Dong L, Hao Y, Sui Z, Wei F (2023). "Why Can GPT Learn In-Context? Language Models Secretly Perform Gradient Descent as Meta-Optimizers". ArXiv. **abs/2212.10559**.
16. ^a, b, cvon Oswald J, Niklasson E, Randazzo E, Sacramento J, Mordvintsev A, Zhmoginov A, Vladymyrov M. "Transformers learn in-context by gradient descent". In: International Conference on Machine Learning; 2022.
17. ^Li T, Zhang G, Do QD, Yue X, Chen W (2024). "Long-context LLMs Struggle with Long In-context Learning". arXiv. arXiv:2404.02060.
18. ^Olsson C, Elhage N, Nanda N, Joseph N, DasSarma N, Henighan T, Mann B, Askell A, Bai Y, Chen A, Conerly T, Drain D, Ganguli D, Hatfield-Dodds Z, Hernandez D, Johnston S, Jones A, Kernion J, Lovitt L, Ndousse K, Amodei D, Brown T, Clark J, Kaplan J, McCandlish S, Olah C (2022). "In-context Learning and Induction Heads". arXiv. 2209.11895.
19. ^Hendel R, Geva M, Globerson A (2023). "In-Context Learning Creates Task Vectors". arXiv. arXiv:2310.15916 [cs.CL].
20. ^Zheng B, Ma M, Lin Z, Yang T (2024). "Distributed Rule Vectors is A Key Mechanism in Large Language Models' In-Context Learning". arXiv. arXiv:2406.16007 [cs.CL].
21. ^Li J, Hou Y, Sachan M, Cotterell R (2024). "What Do Language Models Learn in Context? The Structured Task Hypothesis". ArXiv. **abs/2406.04216**. Available from: https://arxiv.org/abs/2406.04216.
22. ^Niu S, Miao C, Chen G, Wu P, Zhao P (2024). "Test-time model adaptation with only forward passes". arXiv preprint arXiv:2404.01650.
23. ^Cobbe K, Kosaraju V, Bavarian M, Chen M, Jun H, Kaiser L, Plappert M, Tworek J, Hilton J, Nakano R, et al. (2021). "Training verifiers to solve math word problems". arXiv preprint arXiv:2110.14168.
24. ^Chen G, Liao M, Li C, Fan K (2024). "AlphaMath Almost Zero: process Supervision without process". arXiv preprint arXiv:2405.03553.
25. ^Yao S, Yu D, Zhao J, Shafran I, Griffiths T, Cao Y, Narasimhan K (2024). "Tree of thoughts: Deliberate problem solving with large language models". Advances in Neural Information Processing Systems. **36**.
26. ^Hinton GE, Plaut DC (1987). "Using fast weights to deblur old memories". In: Proceedings of the ninth annual conference of the Cognitive Science Society. pp. 177--186.
27. ^Schlag I, Irie K, Schmidhuber J (2021). "Linear Transformers Are Secretly Fast Weight Programmers". International Conference on Machine Learning.
28. ^a, bRamsauer H, Schafl B, Lehner J, Seidl P, Widrich M, Gruber L, Holzleitner M, Pavlovi'c M, Sandve GK, Greiff V, Kreil DP, Kopp M, Klambauer G, Brandstetter J, Hochreiter S (2020). "Hopfield Networks is All You Need". ArXiv. **abs/2008.02217**.
29. ^Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning. PMLR; 2017. p. 1126-1135.

30. ^Beck J, Jackson MT, Vuorio R, Whiteson S. "Hypernetworks in meta-reinforcement learning." In: Conference on Robot Learning. PMLR; 2023. p. 1478-1487.

31. a, bWang Y, Ma D, Cai D (2024). "With Greater Text Comes Greater Necessity: Inference-Time Training Helps Long Text Generation". arXiv. arXiv:2401.11504 [cs.CL].

32. a, b, c, dHu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang S, Wang L, Chen W (2021). "LoRA: Low-rank adaptation of large language models". arXiv preprint arXiv:2106.09685.

33. ^Sun Y, Li X, Dalal K, Xu J, Vikram A, Zhang G, Dubois Y, Chen X, Wang X, Koyejo S, et al. (2024). "Learning to (learn at test time): Rnns with expressive hidden states". arXiv preprint arXiv:2407.04620.

34. ^Aghajanyan A, Zettlemoyer L, Gupta S (2020). "Intrinsic dimensionality explains the effectiveness of language model fine-tuning". arXiv preprint arXiv:2012.13255. Available from: https://arxiv.org/abs/2012.13255.

35. ^Zhang Q, Chen M, Bukharin A, Karampatziakis N, He P, Cheng Y, Chen W, Zhao T (2023). "AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning". arXiv preprint arXiv:2303.10512. arXiv:2303.10512.

36. ^Liu SY, Wang CY, Yin H, Molchanov P, Wang YC, Cheng KT, Chen MH (2024). "DoRA: Weight-decomposed low-rank adaptation". arXiv preprint arXiv:2402.09353.

37. ^Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. **9** (8): 1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

38. a, bGu A, Dao T (2023). "Mamba: Linear-time sequence modeling with selective state spaces". arXiv preprint arXiv:2312.00752.

39. ^Beck M, Pöppel K, Spanring M, Auer A, Prudnikova O, Kopp M, Klambauer G, Brandstetter J, Hochreiter S (2024). "xLSTM: Extended Long Short-Term Memory". arXiv. **cs.LG**: 2405.04517.

40. ^Yang S, Wang B, Shen Y, Panda R, Kim Y (2023). "Gated Linear Attention Transformers with Hardware-Efficient Training". ArXiv. ***abs/2312.06635***. Available from: https://arxiv.org/abs/2312.06635.

41. ^Schlag I, Irie K, Schmidhuber J (2021). "Linear Transformers Are Secretly Fast Weight Programmers". arXiv. arXiv:2102.11174 [cs.LG].

42. ^Caron M, Misra I, Mairal J, Goyal P, Bojanowski P, Joulin A (2021). "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments". arXiv. arXiv:2006.09882.

43. ^Zhang P, Zeng G, Wang T, Lu W. TinyLlama: An Open-Source Small Language Model. 2024. Available from: arXiv:2401.02385.

44. ^Abdin M, Aneja J, Awadalla H, Awadallah A, Awan AA, Bach N, Bahree A, Bakhtiari A, Bao J, Behl H, Benhaim A, Bilenko M, Bjorck J, Bubeck S, Cai M, Cai Q, Chaudhary V, Chen D, Chen D, Chen W, Chen YC, Chen YL, Cheng H, Chopra P, Dai X, Dixon M, Eldan R, Fragoso V, Gao J, Gao M, Gao M, Garg A, Giorno AD, Goswami A, Gunasekar S, Haider E, Hao J, Hewett RJ, Hu W, Huynh J, Iter D, Jacobs SA, Javaheripi M, Jin X, Karampatziakis N, Kauffmann P, Khademi

M, Kim D, Kim YJ, Kurilenko L, Lee JR, Lee YT, Li Y, Li Y, Liang C, Liden L, Lin X, Lin Z, Liu C, Liu L, Liu M, Liu W, Liu X, Luo C, Madan P, Mahmoudzadeh A, Majercak D, Mazzola M, Mendes CCT, Mitra A, Modi H, Nguyen A, Norick B, Patra B, Perez-Becker D, Portet T, Pryzant R, Qin H, Radmilac M, Ren L, de Rosa G, Rosset C, Roy S, Ruwase O, Saarikivi O, Saied A, Salim A, Santacroce M, Shah S, Shang N, Sharma H, Shen Y, Shukla S, Song X, Tanaka M, Tupini A, Vaddamanu P, Wang C, Wang G, Wang L, Wang S, Wang X, Wang Y, Ward R, Wen W, Witte P, Wu H, Wu X, Wyatt M, Xiao B, Xu C, Xu J, Xu W, Xue J, Yadav S, Yang F, Yang J, Yang Y, Yang Z, Yu D, Yuan L, Zhang C, Zhang C, Zhang J, Zhang LL, Zhang Y, Zhang Y, Zhang Y, Zhang Y, Zhou X (2024). "Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone". arXiv. Available from: https://arxiv.org/abs/2404.14219.

45. [a, b] *Clark C, Kenton L, Chang M-W, Kwiatkowski T, Collins M, Toutanova K. "BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions." In: NAACL; 2019.*
46. [a, b] *Roemmele M, Bejan CA, Gordon AS (2011). "Choice of plausible alternatives: An evaluation of commonsense causal reasoning". In: 2011 AAAI Spring Symposium Series.*
47. [a, b] *Socher R, Perelygin A, Wu J, Chuang J, Manning CD, Ng A, Potts C (2013). "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank". In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle, Washington, USA: Association for Computational Linguistics; p. 1631-1642.*
48. [a, b] *De Marneffe M-C, Simons M, Tonhauser J (2019). "The CommitmentBank: Investigating projection in naturally occurring discourse."*
49. [a, b] *Bentivogli L, Dagan I, Dang HT, Giampiccolo D, Magnini B (2009). "The Fifth PASCAL Recognizing Textual Entailment Challenge". TAC.*
50. [^] *Rae JW, Potapenko A, Jayakumar SM, Hillier C, Lillicrap TP (2019). "Compressive Transformers for Long-Range Sequence Modelling". arXiv preprint. 2019.*
51. [a, b] *Clark P, Cowhey I, Etzioni O, Khot T, Sabharwal A, Schoenick C, Tafjord O (2018). "Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge". arXiv. arXiv:1803.05457.*
52. [^] *Bisk Y, Zellers R, Le Bras R, Gao J, Choi Y. "PIQA: Reasoning about Physical Commonsense in Natural Language." In: Thirty-Fourth AAAI Conference on Artificial Intelligence; 2020.*
53. [a, b] *Gao L, Tow J, Abbasi B, Biderman S, Black S, DiPofi A, Foster C, Golding L, Hsu J, Le Noac'h A, Li H, McDonell K, Muennighoff N, Ociepa C, Phang J, Reynolds L, Schoelkopf H, Skowron A, Sutawika L, Tang E, Thite A, Wang B, Wang K, Zou A (2024). "A framework for few-shot language model evaluation". Zenodo. doi:10.5281/zenodo.12608602. Available from: https://zenodo.org/records/12608602.*

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.