LCIRC: A Recurrent Compression Approach for Efficient Long-form Context and Query Dependent Modeling in LLMs

Sumin An[1]   **Junyoung Sung[1]**   **Wonpyo Park[2]**
Chanjun Park[1,†]   **Paul Hongsuck Seo[1,†]**
[1]Dept. of CSE, Korea University   [2]Google
{suminan, jys7451, bcj1210, phseo}@korea.ac.kr
wppark@google.com

Abstract

While large language models (LLMs) excel in generating coherent and contextually rich outputs, their capacity to efficiently handle long-form contexts is limited by fixed-length position embeddings. Additionally, the computational cost of processing long sequences increases quadratically, making it challenging to extend context length. To address these challenges, we propose Long-form Context Injection with Recurrent Compression (LCIRC), a method that enables the efficient processing long-form sequences beyond the model's length limit through recurrent compression without retraining the entire model. We further introduce query dependent context modeling, which selectively compresses query-relevant information, ensuring that the model retains the most pertinent content. Our empirical results demonstrate that Query Dependent LCIRC (QD-LCIRC) significantly improves LLM's ability to manage extended contexts, making it well-suited for tasks that require both comprehensive context understanding and query relevance.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in generating coherent and contextually rich outputs (Zhao et al., 2023; Wang et al., 2024). However, a major limitation remains in their ability to handle long-form contexts efficiently (Liu et al., 2024; Li et al., 2024). Transformer-based architectures, which underlie most LLMs, are constrained by fixed-length position embeddings, limiting the number of tokens they can process in a single pass (Chen et al., 2021; Lin et al., 2022). Once this limit is exceeded, crucial information from earlier parts of the input is truncated, leading to reduced performance in tasks

---

†Co-corresponding authors.

requiring extended context, such as document summarization (Koh et al., 2022) or long-form question answering (Fan et al., 2019). Also, the computational cost of processing longer input sequences grows quadratically with sequence length, making it impractical to simply increase the context window (Fournier et al., 2023; Chen et al., 2023).

To address these challenges, we propose Long-form Context Injection with Recurrent Compression (LCIRC), a method designed to extend LLM's ability to process long-form inputs efficiently. LCIRC compresses the context beyond the model's length limit into compact representations, which are injected back into the model, allowing it to retain essential information without retraining the entire model. By recurrently compressing the input sequence, our approach maintains a balance between preserving context and minimizing computational overhead, enabling the model to generate outputs grounded in long-span contexts (Valmeekam et al., 2023; Chen et al., 2024; Lester et al., 2024).

In many real-world applications, LLMs need to process input in response to specific queries or instructions (Ouyang et al., 2022; Naveed et al., 2023). Compressing all available information indiscriminately can lead to irrelevant data being retained, reducing the model's effectiveness (Mulc and Steele, 2024; Franceschelli et al., 2024). To address this, we introduce query dependent context modeling, which selectively compresses information based on its relevance to the query. This ensures that the model focuses on the most pertinent parts of the input, enhancing its performance in tasks like multiple choice and long-form question answering, where query relevance is crucial.

Our results show that LCIRC, combined with query dependent modeling, significantly improves the ability of LLMs to handle long-form contexts in a scalable and efficient manner, making it well-suited for applications that require both extensive context understanding and precise query relevance.

In summary, our primary contributions are as follows: (1) We propose LCIRC, a method that extends the context window of LLMs through recurrent compression, allowing efficient handling of long-form inputs without retraining the entire model. (2) We introduce query dependent context modeling, which selectively compresses query-relevant information, enhancing performance in tasks that require the comprehensive understanding of extended contexts. (3) Our approach significantly improves the efficiency of LLMs in long-form contexts, reducing computational costs while demonstrating performance improvements quantitatively through various benchmarks.

2 Related Work

Recurrent Transformers Various methods have been explored to extend the capacity of transformer-based language models for handling longer input sequences through the use of recurrence mechanisms. These approaches typically divide the input into segments and recurrently store and reuse information across them. Transformer-XL (Dai, 2019) and Compressive Transformer (Rae et al., 2019) employ segment-level recurrence, while models such as Recurrent Memory Transformer (Bulatov et al., 2022) and Memformer (Wu et al., 2020) utilize memory modules to retain past information. Additionally, Recurrent Attention Network (RAN) (Li et al., 2023) and Segmented Recurrent Transformer (SRformer) (Long et al., 2023) integrate recurrent attention to aggregate information across segments. Despite these advancements, a common limitation of these methods is the need for full model re-pretraining, which is computationally costly and resource-intensive.

Sparse Attention As input sequence lengths increase, the computational overhead associated with full attention mechanisms grows quadratically. Sparse attention mechanisms have been proposed as an efficient alternative to mitigate this computational burden. Notable models such as Sparse Transformer (Child et al., 2019), Longformer (Beltagy et al., 2020), Reformer (Kitaev et al., 2020), Big Bird (Zaheer et al., 2020), Routing Transformer (Roy et al., 2021), and MInference (Jiang et al., 2024) implement sparse attention to reduce computational costs for longer contexts. While these methods improve efficiency, they do so at the cost of partial information loss, resulting in performance that often falls short of full attention

mechanisms, and they are still constrained by the limited context length of LLMs, making it challenging to process long-form sequences.

Prompt Compression Prompt compression techniques aim to reduce the computational challenges of processing large prompts by creating more compact representations. Gisting (Mu et al., 2024) compresses the entire user instruction into a set of "gist tokens" in one step, providing a compact representation of the prompt. However, this approach is typically optimized for shorter contexts, limiting its effectiveness when handling inputs that exceed the context window of the underlying LLM. Additionally, several studies have explored KV cache compression (Nawrot et al., 2024; Zhang et al., 2024b) to enhance the efficiency of long-form context inference. While these approaches improve computational efficiency, they are primarily designed to optimize context modeling within the context length of the underlying LLM and have limited capability in extending the sequence length. ICAE (Ge et al., 2023) and AutoCompressor (Chevalier et al., 2023) address this limitation by segmenting long contexts into smaller chunks and compressing each individually. Although these methods represent progress, they have yet to demonstrate compression capabilities for extremely long sequences, such as those involving hundreds of thousands of tokens, and still face limitations in terms of inference costs and context window size due to the simultaneous inference and compression processes.

3 Efficient Long-form Context Modeling

In this section, we present an efficient approach to handle long-form inputs in pretrained LLMs. We introduce a method that enables models to process lengthy contexts by compressing and injecting the relevant information back into the model in a computationally efficient manner. Additionally, we outline the training strategy used to optimize the proposed components while maintaining the foundational capabilities of the model.

3.1 Preliminaries: Transformer-Based LLMs

Modern LLMs are built on the transformer architecture, where the model autoregressively estimates the probability of each subsequent token $x_{n+1:N}$ given the preceding tokens $x_{1:n}$, as formulated by:

$$P(x_{n+1:N}|x_{1:n}) = \prod_{i=n+1}^{N} P(x_i|x_{1:i-1}) \quad (1)$$
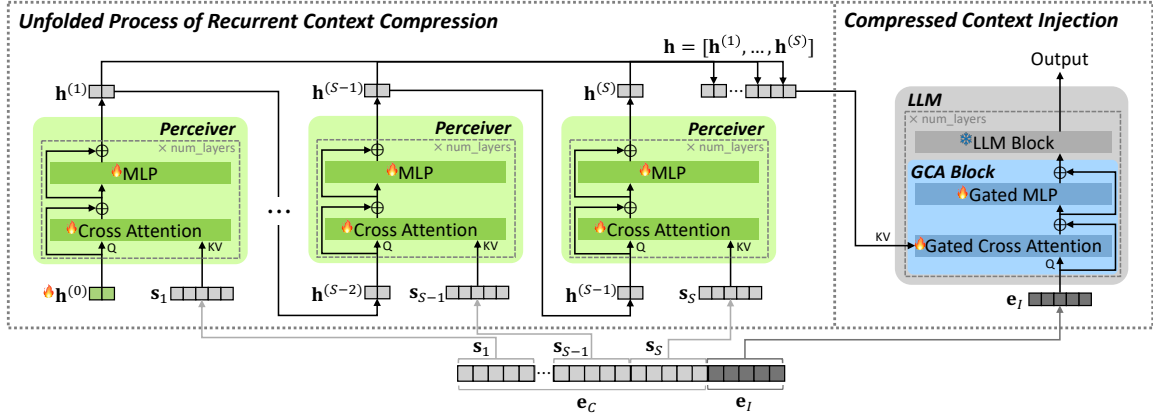
Figure 1: **The overall process of the proposed Long-form Context Injection with Recurrent Compression (LCIRC)**. LCIRC comprises two components: Recurrent Context Compression (left) and Compressed Context Injection (right). In the i-th step of Recurrent Context Compression, the previously compressed features $\mathbf{h}^{(i-1)}$ and the segment embeddings $\mathbf{s}_i$ are fed into the Perceiver module as query and input features, respectively. The compressed features $\mathbf{h}^{(i)}$ are then generated and reinjected as query features for the subsequent recurrence step. The initial query features $\mathbf{h}^{(0)}$ are learnable parameters. In Compressed Context Injection, the concatenated compressed features $\mathbf{h}$ serve as input to the Gated Cross Attention layer. Layers indicated with a fire symbol represent trained layers, while layers marked with a snow symbol denote frozen layers.

Here, $x_{j:k}$ refers to a subsequence of tokens $(x_j, x_{j+1}, \ldots, x_k)$, and $x_{1:N}$ denotes the full token sequence. During generation, the model samples one token at a time, appending it to the input to predict the next token.

Accurately modeling token order is essential in autoregressive generation because it enables the model to effectively capture the underlying structure and meaning of language. Transformer-based LLMs employ learned positional embeddings to represent token positions, but these embeddings impose a fixed input length, limiting the model's ability to handle sequences longer than the maximum number of embeddings (M). Consequently, transformers are constrained in their capacity to process input sequences exceeding this limit ($N > M$).

Incorporating long-form context into LLMs is critical for generating outputs that remain grounded in extended input sequences. However, extending the input length requires full retraining of the model, and the computational cost scales quadratically with the context length, making long-form input processing computationally prohibitive in standard transformer architectures.

3.2 Long-form Context Injection with Recurrent Compression

To address the limitations of processing lengthy inputs ($N \gg M$) in pretrained LLMs, we propose Long-form Context Injection with Recurrent Compression (LCIRC), an approach that enables efficient handling of long-form contexts. A sim-

ple truncation of the first $N - M$ tokens would discard essential contextual information, whereas our method restores access to this context through recurrent context compression and compressed context injection, detailed below. An overview of the proposed method is provided in Figure 1.

3.2.1 Recurrent Context Compression

We introduce a recurrent context compression mechanism that effectively reduces the long-form context into a compact sequence of embeddings, which can be efficiently processed by the model.

Given an input sequence $x_{1:N}$, where $x_{1:N-M}$ represents the truncated context x_C, it is often the case that $N \gg M$, for example, $N = 192\text{K}$ in InfiniteBench (Zhang et al., 2024a) compared to $M = 4\text{K}$ in Llama (Touvron et al., 2023). To handle this, the recurrent compressor produces a compact feature sequence $(h_1, \ldots, h_K) \in \mathbf{h}$ from x_C, where $K \ll N - M$.

We employ the Perceiver architecture (Jaegle et al., 2021b,a) for the compressor, which consists of stacked Perceiver blocks. Each block includes a cross-attention layer followed by a two-layer MLP with residual connections (Figure 1). The cross-attention mechanism aggregates input features based on query features, enabling efficient compression by using a compact sequence as the query and the longer context as the input features.

In particular, we use the token embeddings $\mathbf{e}_C$ of the truncated context x_C as the input features and a set of learnable query vectors $\mathbf{h}^{(0)}$ of length K as

the query features. The compressed features $\mathbf{h}$ are obtained through the Perceiver module as follows:

$$\mathbf{h} = \text{Perceiver}(\mathbf{h}^{(0)}, \mathbf{e}_C) \qquad (2)$$

where $\text{Perceiver}(q, x)$ represents the Perceiver module with query q and input features x.

However, compressing such an extensive context in a single step is computationally expensive, thus we introduce a recurrent compression process. The long context $\mathbf{e}_C$ is split into S disjoint segments $\mathbf{s}_1, \ldots, \mathbf{s}_S$, where $\mathbf{s}_i = \mathbf{e}_{n_{i-1}+1:n_i}$ represents the i-th segment. These segments are sequentially fed into the Perceiver module, with the compressed features from the previous segment serving as the query features for the next segment:

$$\mathbf{h}^{(i)} = \text{Perceiver}(\mathbf{h}^{(i-1)}, \mathbf{s}_i) \qquad (3)$$

Here, $\mathbf{h}^{(i)}$ compresses both the current segment $\mathbf{s}_i$ and the cumulative information from all previous segments, enabled by the recurrent mechanism. The initial query vectors $\mathbf{h}^{(0)}$ consists of learnable parameters.

Finally, the compressed representations $\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(S)}$ of all segments are concatenated to form the overall compressed representation of the long-form context:

$$\mathbf{h} = [\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(S)}] \qquad (4)$$

where $[\cdots]$ denotes concatenation. This recurrent approach ensures efficient long-form context representation, enabling LLMs to process extended inputs beyond their native length limitations.

3.2.2 Compressed Context Injection

After obtaining the compressed representation $\mathbf{h}$ for the long-form context, we inject this compressed information into the pretrained transformer using gated cross-attention layers with residual connections (Alayrac et al., 2022). For the truncated input sequence $x_{N-M+1:N}$ of length M, denoted as x_I, we first compute the embedding sequence $\mathbf{e}_I^{(l)}$ at the l-th transformer block. The embeddings are then contextualized through Gated Cross Attention Block (GCA Block in Figure 1) as follows:

$$
\begin{aligned}
\dot{\mathbf{e}}_I^{(l)} &= \alpha^{(l)} \cdot \text{CA}(\mathbf{e}_I^{(l)}, \mathbf{h}) + \mathbf{e}_I^{(l)} \\
\alpha^{(l)} &= \tanh(a^{(l)}) \\
\ddot{\mathbf{e}}_I^{(l)} &= \beta^{(l)} \cdot \text{MLP}(\dot{\mathbf{e}}_I^{(l)}) + \dot{\mathbf{e}}_I^{(l)} \\
\beta^{(l)} &= \tanh(b^{(l)})
\end{aligned}
\qquad (5)
$$

where $\text{CA}(q, x)$ denotes the cross-attention layer with queries q and key-value inputs x. Unlike standard transformer layers, we pass the modified embeddings $\ddot{\mathbf{e}}_I^{(l)}$ to the next transformer block instead of the original embeddings $\mathbf{e}_I^{(l)}$. The scalar parameters $a^{(l)}$ and $b^{(l)}$ are learnable and initialized to 0, preserving the pretrained LLM's performance at the start of training. The compressed context representation enables efficient context injection, minimizing computational overhead.

3.2.3 Optimization Strategy

To fully leverage the pretrained LLM's foundational capabilities, we optimize only the additional components using a corpus of long-form texts.

Given a long-form training sequence $x_{1:N}$, the model is trained by minimizing the negative log-likelihood (NLL) loss:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \log P(x_i | x_{1:i-1}). \qquad (6)$$

The long-form input is randomly segmented, with each segment limited to a maximum length R. The probability $P(x_i | x_{1:i-1})$ is estimated within each segment, and prior segments are processed by the recurrent context compressor. The input $x_{k:i-1}$, where k is the starting index of the current segment, is treated as the regular input for the LLM.

Although the recurrent architecture enables memory-efficient inference, the space complexity during training scales linearly with the input length N due to backpropagation through time (BPTT). To mitigate this, we employ truncated BPTT, where gradient computation is restricted to the last T segments. Since gradient calculation is unnecessary for earlier segments beyond the last T, we cache the compressed features $\mathbf{h}$ and reuse them for predicting subsequent tokens within each segment.

3.2.4 Inference

LLMs perform inference by conditioning on a given input prompt and generating a coherent and contextually relevant output sequence. Given an input token sequence of length N and an output token sequence of maximum length P, if their combined length remains within the context window M of the underlying LLM ($N + P \le M$), the our recurrent compression mechanism is not activated, as LCIRC is specifically designed for long-form contexts. In this case, the model operates in the same manner as the inference process of the pretrained

LLMs. In contrast, when the total maximum length of input and output sequences exceeds the context window of the LLM ($N + P > M$), two distinct cases must be considered. When the maximum output length remains within the context window of the LLM ($P \leq M$), the LLM can generate the entire output sequence in a single pass given the compressed input sequence by the proposed recurrent compression mechanism. Conversely, if the maximum output length exceeds the context window ($P > M$), our model iteratively compresses the earlier portions of the generated tokens (e.g., the first $M/2$ tokens) by performing additional recurrent steps using the compression mechanism. This process removes the compressed tokens from the context of the underlying LLM, thereby creating space for generating new next tokens while preserving coherence and continuity. Notably, this additional process remains highly efficient, as it only involves the compression steps while maintaining the original window size of the underlying LLM.

4 Query Dependent Context Modeling

LLMs frequently process context based on specific instructions or queries. To enhance the ability of LLMs to handle long-form context, we extend our method to incorporate query dependent context compression. This allows the model to selectively focus on the context most relevant to the given query, thus improving the overall efficiency and relevance of the model's responses.

Unlike the vanilla recurrent context compression, which merges all the information in the current segment $\mathbf{s}_i$ into the compressed features $\mathbf{h}^{(i)}$ as shown in Eq. (3), query dependent compression selectively injects information that is most relevant to the user query. This selective compression is achieved through the addition of a Gated Cross Attention Block to our recurrent context compression method as shown in Figure 2.

As illustrated in Figure 2, the model integrates the user query into the compression pipeline. At each compression step, the learnable query vectors or the previously compressed features $\mathbf{h}^{(i-1)}$ used as the query features for the Perceiver module are transformed into a query dependent representation. This is done through the newly introduced gated cross attention block, which takes $\mathbf{h}^{(i-1)}$ as the query features, and the user query embedding $\mathbf{e}_{\text{query}}$ as the input features. The query dependent
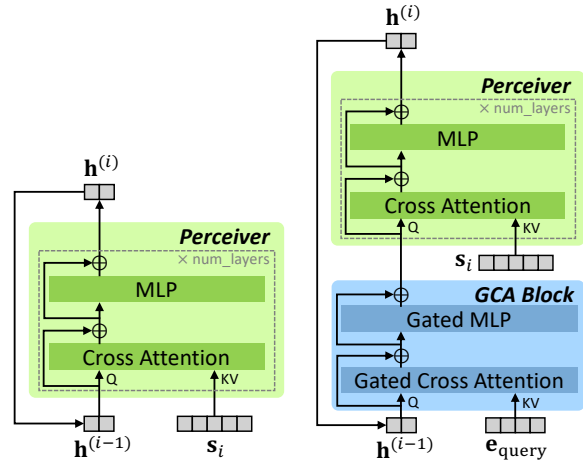

Figure 2: **Comparison of the recurrent context compression module with and without query dependent modeling.** In addition to the regular context compression module (left), we add additional cross attention module (blue box) to inject query information into the compressed feature $\mathbf{h}^{(i-1)}$ (right).

features $\ddot{\mathbf{h}}^{(i-1)}$ are computed through the same process in Eq. (5) with $\mathbf{h}^{(i-1)}$ and $\mathbf{e}_{\text{query}}$.

The query dependent compressed feature $\mathbf{h}^{(i)}$ is then produced using the Perceiver module, with the query dependent feature $\ddot{\mathbf{h}}^{(i-1)}$ and the segment embeddings $\mathbf{s}_i$ through the same process in Eq. (3). Subsequently, these query dependent compressed features $\mathbf{h}$ are then used during inference as described in Eq. (5), enabling the model to focus on information relevant to the query while handling long-form inputs.

Training and Efficiency Enhancements Query Dependent LCIRC (QD-LCIRC) builds upon the pre-trained LCIRC architecture by adding a gated cross-attention layer to introduce query dependency. To train QD-LCIRC, we minimize the following negative log-likelihood (NLL) loss:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \log P(x_i | x_{1:i-1}, x_{\text{query}}) \quad (7)$$

This objective ensures that the model learns to predict each token in the input sequence conditioned on both the preceding tokens and the query, facilitating query dependent context modeling.

As shown in Figure 3, we employ Random Selective BPTT to train the QD-LCIRC efficiently. Unlike vanilla BPTT, which computes gradients across all timesteps, or truncated BPTT, which only computes gradients for the last T timesteps, Random Selective BPTT randomly selects a subset of
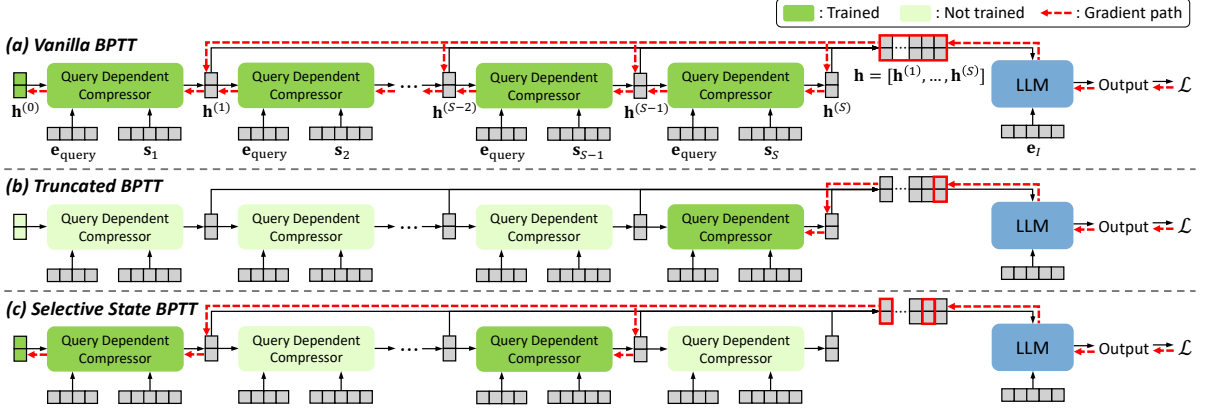
Figure 3: **Comparisons of the proposed Selective State BPTT with vanilla and truncated BPTT.** Green boxes represent timesteps where gradients are computed in BPTT whereas the light green ones indicate the timesteps without gradient computation. Finally, dotted red lines illustrate the gradient flows. (a) Vanilla BPTT computes the full gradients through the entire timesteps in recurrence but is computationally infeasible with a large N. The gradients for $\mathbf{h}^{(i)}$ receives upstream gradients both through the recurrent connection and through the direct connection from $\mathbf{h}$. (b) Truncated BPTT backprobagates gradients to the last T timesteps only significantly reducing computational costs. However, it does not transfer gradient flows to timesteps further than T (marked with light green color) and fails to learn long-term QD modeling. (c) Our proposed Selective State BPTT selects several random timesteps and transfer gradient flows directly through the direct connection from $\mathbf{h}$, which enables efficient learning of long-term QD modeling capabilities.

timesteps for gradient computation. This allows the model to efficiently learn long-term query dependent context modeling without excessive computational overhead. Additionally, we cache the compressed features $\mathbf{h}$ from earlier segments, further reducing the memory and computational requirements for training.

Inference The only distinction between QD-LCIRC and LCIRC lies in the incorporation of query dependent compression, facilitated by the GCA block, within the recurrent compression step. Consequently, the inference process of QD-LCIRC differs from that of LCIRC only by the addition of query dependent modeling through an extra GCA block in the recurrent compression step, as demonstrated in Section 3.2.4 and Figure 2.

5 Experiments

5.1 Datasets and Metrics

FineWeb-Edu FineWeb-Edu (Lozhkov et al., 2024) comprises 1.3T tokens, filtered from the FineWeb dataset, which contains 15T tokens. To facilitate long-form context modeling, we selected texts with a minimum length of 4K tokens, resulting in a dataset that includes texts up to 339K tokens in length. The statistics of the training data are provided in the Table 1. For evaluation, we curated 1,000 texts of 128K tokens each to assess

Token Length	# of Samples	Proportion (%)
6K ≤ Data < 8K	410,876	36.64
8K ≤ Data < 16K	418,332	37.30
16K ≤ Data < 32K	207,503	18.50
32K ≤ Data < 64K	69,396	6.19
64K ≤ Data < 128K	13,659	1.22
128K ≤ Data	1,679	0.15
Total	1,121,445	100.00

Table 1: **Data distribution of the training set extracted from FineWeb-Edu.** Our training set is constructed focusing on long-context modeling.

perplexity.

FineWeb-LQA FineWeb-LQA, a long-form QA dataset, was automatically generated from FineWeb-Edu to support the training of our query-dependent model, following a data construction process similar to An et al. (2024). We extract random 128-token text segments and utilize the Llama-3.1-70B-Instruct-FP8 model to generate corresponding QA pairs. The extracted segments are then reintegrated into the original long-form context, which serves as the basis for evaluating long-form QA tasks.

InfiniteBench InfiniteBench (Zhang et al., 2024a) is a benchmark suite designed to assess the capability of LLMs in handling ultra-long contexts, exceeding 100K tokens. We focus on

two tasks: LongBook QA (En.QA) and LongBook Multiple Choice (En.MC), both of which test the model's ability to answer identical questions in open-ended and multiple-choice formats, respectively. Evaluation is conducted using the F1 score for En.QA and accuracy for En.MC.

LongBench LongBench (Bai et al., 2023) evaluates long-form context modeling through a suite of tasks across real-world and synthetic categories. In this study, we focus on six English tasks, consisting of both single-document and multi-document QA tasks. The average context length is 18K tokens, with a maximum of 82K tokens. Performance is measured using the F1 score.

L-Eval L-Eval (An et al., 2023) includes a dataset of 508 long documents spanning various domains, divided into 20 subtasks. It comprises over 2K human-annotated query-response pairs, with context lengths ranging from 3K to 200K tokens. We evaluate the models on four open-ended QA tasks using the F1 score and four multiple-choice QA tasks using accuracy.

5.2 Models

We build upon Llama2-7B (Touvron et al., 2023) as our baseline model, augmenting its long-form context modeling capabilities. In line with (Chevalier et al., 2023), we implement an extended version of Llama2 that supports full attention over longer contexts (ExtendedFA), extending the token length limit to 8K by modifying the RoPE θ (Su et al., 2024). Due to the quadratic complexity of full attention, ExtendedFA is restricted to a maximum of 8K tokens. We compare this approach against AutoCompressor (Chevalier et al., 2023), a state-of-the-art method that models context through prompt compression by recurrently feeding segmented inputs to the model while leveraging compressed tokens for subsequent segments. Additionally, we evaluate our proposed method with and without query dependent modeling.

All models are trained on FineWeb-Edu to ensure fair comparisons in language modeling. For QD-LCIRC, we initialize the model with the pretrained weights of LCIRC on FineWeb-Edu and fine-tune it on FineWeb-LQA for query dependent modeling. Note that FineWeb-LQA is automatically generated from FineWeb-Edu for this purpose.

For baseline models, context exceeding the token length limit is truncated. Llama2 and Extend-

| Models | Total Token Length (N) | | | |
	4k	8k	64k	128k
Llama-2-7B	5.472	-	-	-
ExtendedFA	**5.442**	5.319	-	-
AutoCompressor	6.127	6.010	6.188	-
LCIRC (Ours)	5.472	5.313	5.312	5.312
QD-LCIRC (Ours)	5.472	**5.299**	**5.298**	**5.298**

Table 2: **Perplexity scores on the FineWeb-Edu test set.** Each long-form text is truncated from the beginning to adhere to the total token length, which encompasses both the context and the last 2K target tokens used for measuring perplexity.

| Models | Total Token Length (N) | | | |
	4k	8k	64k	128k
ExtendedFA	63	143	3,118	10,739
AutoCompressor	61	125	1,350	-
LCIRC (Ours)	63	77	97	120
QD-LCIRC (Ours)	63	77	98	122

Table 3: **Computational complexities for different models in TeraFLOPs.** We compute the TFLOPs of ExtendedFA under the assumption that the model is extended to process input tokens of the specified length. AutoCompressor is unable to process inputs with 128K tokens.

edFA handle up to 4K and 8K tokens, respectively, while AutoCompressor supports up to 84K tokens, based on the experimental setup from Chevalier et al. (2023). In contrast, our method imposes no explicit length limit, as the compressed context is injected via cross-attention, allowing us to process sequences up to 815K tokens in length.

5.3 Implementation Details

All models are based on Llama2-7B and trained using a batch size of 64 with the Adam optimizer (Kingma, 2014). QD-LCIRC is trained with a learning rate of 2e-5 and 300 warmup steps, utilizing Selective State BPTT with 8 random selections. Other models are trained with a learning rate of 5e-5. The length (K) of the initial learnable queries $\mathbf{h}^{(0)}$, which also corresponds to the length of each compressed features $\mathbf{h}^{(i)}$, is set to 64 based on our preliminary experiments with a smaller model OPT-2.7B. In these experiments, we observed no significant performance difference between $K = 64$ and $K = 256$, leading us to adopt the more efficient setting of $K = 64$. All training procedures are conducted using eight NVIDIA H100 80GB GPUs.

	FW-LQA	QD	InfiniteBench			LongBench						
			En.MC	En.QA	Avg	NQA	Qasper	MFQA	HQA	2WQA	MSQ	Avg
Llama-2-7B	✗	✗	6.99	3.95	5.47	**13.04**	12.08	14.68	16.27	7.10	4.41	11.26
ExtendedFA	✗	✗	15.72	3.88	9.80	12.21	18.23	18.23	17.81	14.18	8.25	14.82
AutoCompressor	✗	✗	18.34	4.46	11.40	12.60	16.89	19.93	19.00	16.36	8.84	15.60
LCIRC (Ours)	✗	✗	**21.40**	**5.26**	**13.33**	10.67	**18.32**	**21.71**	**21.66**	**16.55**	**9.09**	**16.33**
ExtendedFA	✓	✗	28.38	4.55	16.47	**18.96**	13.73	23.48	20.78	17.24	8.29	17.08
AutoCompressor	✓	✗	31.00	5.35	18.18	13.69	18.63	**33.55**	15.01	14.13	8.98	17.33
QD-LCIRC (Ours)	✓	✓	**38.86**	**5.80**	**22.33**	15.31	**20.57**	33.25	**28.19**	**19.00**	**12.39**	**21.45**

Table 4: **Per-task performance on InfiniteBench and LongBench.** The following abbreviations are used: **NQA** denotes NarrativeQA, **MFQA** represents MultiFieldQA-en, **HQA** refers to HotpotQA, **2WQA** to 2WikiMQA, and **MSQ** to MuSiQue. **Avg** indicates the average score across all subtasks within respective benchmarks. **FW-LQA** indicates whether the model is fine-tuned on FineWeb-LQA. Our QD-LCIRC consistently outperforms competing methods, achieving the highest average score by incorporating query dependent modeling, as indicated in the **QD** column.

	FW-LQA	QD	CS	QALIT	TOEFL	SF	LFQA	NQA	NQ	Qasper	Avg
Llama-2-7B	✗	✗	10.47	25.74	0.00	29.69	22.30	8.66	22.64	2.29	15.22
ExtendedFA	✗	✗	16.86	33.66	17.10	22.66	13.89	10.76	33.24	10.96	19.89
AutoCompressor	✗	✗	18.60	31.68	17.47	22.66	11.73	4.82	26.42	6.64	17.50
LCIRC (Ours)	✗	✗	22.09	27.23	10.04	27.34	14.66	9.89	26.40	9.44	18.39
ExtendedFA	✓	✗	15.70	**34.16**	13.75	20.31	28.27	16.91	**35.68**	8.51	21.66
AutoCompressor	✓	✗	20.35	32.18	**26.39**	16.41	29.33	19.08	28.67	15.38	23.47
QD-LCIRC (Ours)	✓	✓	**25.58**	30.20	12.27	**37.50**	**31.63**	**20.92**	34.37	**16.92**	**26.17**

Table 5: **Per-task performance on L-Eval**. The following abbreviations are used: **CS** denotes Coursera, **QALIT** refers to QuALITY, **SF** represents SFiction, **LFQA** refers to LongFQA, and **NQA** to NarrativeQA. **Avg** indicates the mean performance score across all subtasks within the respective benchmark. **FW-LQA** indicates whether the model has been fine-tuned on FineWeb-LQA, while **QD** denotes whether query dependent modeling.

	InfBench	LongBench	L-Eval
Truncated BPTT	21.26	20.73	25.45
Selective State BPTT	**22.33**	**21.45**	**26.17**

Table 6: **Impact of different BPTT variations on benchmark performance.** This table presents the performance of the model trained with Truncated BPTT, and Selective State BPTT ($T = 8$) across three benchmarks: InfiniteBench, LongBench, and L-Eval. The results highlight the differences in benchmark scores for each method, demonstrating the effectiveness of Selective State BPTT.

5.4 Results on Language Modeling

Following Chevalier et al. (2023), we first measure the perplexity of the last 2K tokens given its context while varying the total token length between 4K and 128K, which refers to the combined length of both the context and target tokens. To ensure that the same tokens are used for perplexity measurement for comparisons, we truncate the inputs from the beginning based on the total context length (N), retaining the last N tokens, of which the final 2K tokens are used for the perplexity computation.

Note that all the models except ours impose limits on context lengths.

Table 2 presents the perplexity results of the models on the FineWeb-Edu test set. We first observe that all the methods, except for AutoCompressor, perform similarly when $N = 4K$. AutoCompressor significantly alters Llama's generation mechanism resulting in a notable drop in perplexity. This finding aligns with the observations from experiments conducted with Llama in Chevalier et al. (2023). In contrast, our method preserves the original strong capabilities of Llama with short contexts, as the frozen LLM remains unchanged.

With $N > 4K$, Llama is unable to process the full context. In contrast, all other models exhibit improved perplexity scores by utilizing additional context compared to their scores with $N = 4K$. Note, however, that both ExtendedFA and AutoCompressor still impose strict limits on the total context length. Moreover, the perplexity of AutoCompressor increases at $N = 64K$, indicating challenging optimization for long-form context modeling. In contrast, the proposed LCIRC model consistently improves perplexity and maintains this improved

performance even as the context further lengthens.

We evaluate the computational complexities of the models for processing long-term contexts with varying total token lengths at inference, as shown in Table 3. As the token length increases, the results show a prohibitively large increase in complexity with ExtendedFA, which requires full attention across all tokens. In contrast, AutoCompressor effectively reduces complexity compared to ExtendedFA; a reduction rate increases with the number of tokens and showing 66% reduction with 64K tokens. However, AutoCompressor cannot process tokens longer than 64K with segments of 2K token length. In contrast, LCIRC can handle 128K tokens, while achieving a 99% complexity reduction compared to ExtendedFA. Note that LCIRC improved perplexity while maintaining this significant reduction in complexity. Finally, QD-LCIRC introduces some additional complexities, but they are marginal compared to the overall reduction rate.

5.5 Results on Long Context Benchmarks

We also evaluate the methods on multiple QA benchmarks that require long-form context understanding. In these benchmarks, models are asked to answer a question, requiring to understand the input text under the context of the question. For QD-LCIRC, we use the input question as the query.

Table 4 presents performances on InfiniteBench and LongBench. Since the QA instances in these benchmarks require an understanding of long-form context documents, Llama exhibits poor performance across all tasks. ExtendedFA improves this by leveraging additional context, but it is still limited to 8K tokens, sharing the same underlying issue as Llama. Both AutoCompressor and LCIRC further enhance performance over ExtendedFA by enabling access to much longer contexts, with LCIRC achieving much greater improvements. When LCIRC is combined with our query-dependent modeling technique (QD-LCIRC), it leads to significant performance gains, yielding approximately 308% and 90% relative improvements in average scores over the base Llama model on InfiniteBench and LongBench, respectively. To ensure fair comparisons, we further fine-tuned ExtendedFA and AutoCompressor on FineWeb-LQA. Despite this, our QD-LCIRC still achieves significant improvements over these models, consistently delivering the best performance on most tasks and resulting in the highest average score. This further underscores the effectiveness of incorporating query dependent modeling.

We also evaluate the models on L-Eval in Table 5. Note that, although L-Eval is a benchmark designed to assess long-form context understanding capabilities, its context lengths are relatively shorter, with an average length of 19K compared to 218K in InfiniteBench. Based on this, Llama, which can process up to 4K tokens, demonstrates strong performance on several tasks. When extended to capture longer contexts using various methods—namely ExtendedFA, AutoCompressor, and LCIRC—all models show improved average performance compared to the base model. Finally, our QD-LCIRC achieves the highest average performance on L-Eval as well, demonstrating significant gains through query dependent modeling, with a relative improvement of approximately 11.5% compared to the best-performing baseline, ExtendedFA finetuned on FineWeb-LQA.

In Table 6, we compare the proposed selective state BPTT with truncated BPTT for QD-LCIRC on InfiniteBench, LongBench and L-Eval. The results show that our selective state BPTT allows higher scores compared to truncated BPTT across all three benchmarks. Note that truncated BPTT only backpropagates gradients to a limited number of timesteps in recurrence, restricting optimization for long-term context modeling. In contrast, our selective state BPTT enables the model to receive gradients from any timesteps and in consequence, the trained model better models long inputs.

6 Conclusion

We propose Long-form Context Injection with Recurrent Compression (LCIRC) to address challenges LLMs face with extended inputs. LCIRC efficiently compresses long-form contexts, expanding context length while reducing computational overhead. By incorporating query dependent context modeling, it selectively retains relevant information, improving performance in tasks requiring long-context comprehension and query relevance. Our experiments demonstrate significant advancements in both areas. Future work will focus on extending LCIRC to multilingual settings to integrate context across diverse languages and cultures.

Limitations

Although our proposed method, LCIRC, demonstrates notable improvements in handling long-form contexts, several limitations remain. First, the

training and evaluation of query dependent context modeling were restricted to question-answering tasks, leaving its generalizability to other query-driven applications, such as information retrieval and dialogue systems, unexplored. Future work is needed to assess its broader applicability. Furthermore, despite the reduction in computational complexity during inference, LCIRC incurs substantial training costs due to the introduction of cross-attention layers, which increases both the computational load and the amount of training data required. This constraint may limit the method's deployment in resource-constrained environments. Lastly, the experimental evaluation was primarily conducted on English-language datasets, limiting the conclusions that can be drawn about LCIRC performance in multilingual or non-English contexts. Further experimentation on diverse linguistic datasets is necessary to evaluate the model's cross-lingual capabilities.

Ethics Statement

All experiments conducted in this study were performed with fairness and transparency in mind. The datasets used were sourced from publicly available data, and no personally identifiable information (PII) was involved. Our experimental design ensured that all models were trained and evaluated under consistent conditions, allowing for fair comparisons. We commit to maintaining high ethical standards in the use and development of our models, promoting responsible research practices.

Acknowledgments

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736.

Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2023. L-eval: Instituting standardized evaluation for long context language models. *arXiv preprint arXiv:2307.11088*.

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, and Jian-Guang Lou. 2024. Make your llm fully utilize the context. *arXiv preprint arXiv:2404.16811*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091.

Pu-Chin Chen, Henry Tsai, Srinadh Bhojanapalli, Hyung Won Chung, Yin-Wen Chang, and Chun-Sung Ferng. 2021. A simple and effective positional encoding for transformers. *arXiv preprint arXiv:2104.08698*.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Yinpeng Chen, DeLesley Hutchins, Aren Jansen, Andrey Zhmoginov, David Racz, and Jesper Andersen. 2024. Melodi: Exploring memory compression for long contexts. *arXiv preprint arXiv:2410.03156*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*.

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

Zihang Dai. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering. *arXiv preprint arXiv:1907.09190*.

Quentin Fournier, Gaétan Marceau Caron, and Daniel Aloise. 2023. A practical survey on faster and lighter transformers. *ACM Computing Surveys*, 55(14s):1–40.

Giorgio Franceschelli, Claudia Cevenini, and Mirco Musolesi. 2024. Training foundation models as data compression: On information, model weights and copyright law. *arXiv preprint arXiv:2407.13493*.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. 2021a. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. 2021b. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *arXiv preprint arXiv:2407.02490*.

Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Huan Yee Koh, Jiaxin Ju, Ming Liu, and Shirui Pan. 2022. An empirical survey on long document summarization: Datasets, models, and metrics. *ACM computing surveys*, 55(8):1–35.

Brian Lester, Jaehoon Lee, Alex Alemi, Jeffrey Pennington, Adam Roberts, Jascha Sohl-Dickstein, and Noah Constant. 2024. Training llms over neurally compressed text. *arXiv preprint arXiv:2404.03626*.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*.

Xianming Li, Zongxi Li, Xiaotian Luo, Haoran Xie, Xing Lee, Yingbin Zhao, Fu Lee Wang, and Qing Li. 2023. Recurrent attention networks for long-text modeling. *arXiv preprint arXiv:2306.06843*.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2022. A survey of transformers. *AI open*, 3:111–132.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yinghan Long, Sayeed Shafayet Chowdhury, and Kaushik Roy. 2023. Segmented recurrent transformer: An efficient sequence-to-sequence model. *arXiv preprint arXiv:2305.16340*.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024. Fineweb-edu.

Jesse Mu, Xiang Li, and Noah Goodman. 2024. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36.

Thomas Mulc and Jennifer L Steele. 2024. Compressing search with language models. *arXiv preprint arXiv:2407.00085*.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo Ponti. 2024. Dynamic memory compression: Retrofitting LLMs for accelerated inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 37396–37412. PMLR.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

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

Chandra Shekhara Kaushik Valmeekam, Krishna Narayanan, Dileep Kalathil, Jean-Francois Chamberland, and Srinivas Shakkottai. 2023. Llmzip: Lossless text compression using large language models. *arXiv preprint arXiv:2306.04050*.

Xindi Wang, Mahsa Salmani, Parsa Omidi, Xiangyu Ren, Mehdi Rezagholizadeh, and Armaghan Eshaghi. 2024. Beyond the limits: A survey of techniques to extend the context length in large language models. *arXiv preprint arXiv:2402.02244*.

Qingyang Wu, Zhenzhong Lan, Kun Qian, Jing Gu, Alborz Geramifard, and Zhou Yu. 2020. Memformer: A memory-augmented transformer for sequence modeling. *arXiv preprint arXiv:2010.06891*.

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

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, et al. 2024a. ∞bench: Extending long context evaluation beyond 100k tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277.

Yichi Zhang, Bofei Gao, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, Wen Xiao, et al. 2024b. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.