Training-Free Long-Context Scaling of Large Language Models

Chenxin An [*1 2] **Fei Huang** [2] **Jun Zhang** **Shansan Gong** [1] **Xipeng Qiu** [3] **Chang Zhou** [2] **Lingpeng Kong** [1]

Abstract

The ability of Large Language Models (LLMs) to process and generate coherent text is markedly weakened when the number of input tokens exceeds their pretraining length. Given the expensive overhead of finetuning large-scale models with longer sequences, we propose Dual Chunk Attention (DCA), which enables LLAMA2 70B to support context windows of more than 100k tokens without continual training. By decomposing the attention computation for long sequences into chunk-based modules, DCA manages to effectively capture the relative positional information of tokens within the same chunk (Intra-Chunk) and across distinct chunks (Inter-Chunk), as well as integrates seamlessly with Flash Attention. In addition to its impressive extrapolation capability, DCA achieves performance on practical long-context tasks that is comparable to or even better than that of finetuned models. When compared with proprietary models, our training-free 70B model attains 94% of the performance of `gpt-3.5-16k`, indicating it is a viable open-source alternative. All code and data used in this work are released at https://github.com/HKUNLP/ChunkLlama.

1. Introduction

The ability to comprehend and process long-context information is essential for large language models (LLMs) (OpenAI, 2023; Touvron et al., 2023a;b; Bai et al., 2023; Anthropic, 2023) to cater to a wide range of applications effectively. These include analyzing and responding to inquiries within sizable PDFs, retaining extended dialogue history, and empowering interactive chatbots (Wei et al., 2023; Lee et al., 2023; Rula & D'Souza, 2023; Saad-Falcon et al., 2023; Lv et al., 2024).

[*]Work done during internship at Alibaba Group [1]The University of Hong Kong [2]Alibaba Group [3]Fudan University. Correspondence to: Chenxin An <cxan23@connect.hku.hk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

Recent advances have shown that the long-context ability can be improved by further training a short-context model on long text sequences (Ruoss et al., 2023; Rozière et al., 2023). The impressive performance of Llama2 Long (Xiong et al., 2023), which is trained from a mix of long text data and the original Llama2 (Touvron et al., 2023b) pre-training corpus, stands as a testament to this approach. Nevertheless, due to the limited accessibility of these training corpora and the prohibitive cost of long-context finetuning, current open-source models often fall short in performance when compared to the proprietary counterparts, and are generally available in smaller sizes (e.g., 7B/13B).

Given these constraints, approaches that do not require additional training for context scaling in LLMs become particularly attractive. Recent training-free methods, including LM-infinite (Han et al., 2023) and StreamingLLM (Xiao et al., 2023), have shown that LLMs trained on a limited context window can efficiently process text of infinite length (Zhang et al., 2023; 2024; Xiao et al., 2024; Qin et al., 2024). Assuming that LLMs are unable to generalize to texts longer than the training length, these models handle extended sequences by selectively retaining essential local information. Such paradigms effectively maintain a low Perplexity (PPL), yet they lose long-range dependencies. To retain the global information, another perspective is to effectively extrapolate to sequence lengths that surpass those encountered during their training (Sun et al., 2022; Kazemnejad et al., 2023; Liu et al., 2023b; Chi et al., 2023). Popular techniques for Llama-based models, including Position Interpolation (PI) (Chen et al., 2023b) and NTK-Aware RoPE (NTK) (LocalLLaMA, 2023b;a), are adaptations of Rotary Positional Encodings (RoPE) (Su et al., 2022). These scaled positional encodings necessitate fewer finetuning steps compared to the original RoPE, and their training costs can be further reduced via methods such as YaRN (Peng et al., 2023) and CLEX (Chen et al., 2023a). However, in a training-free setting, we find that these approaches usually lead to a notable increase in PPL especially in input lengths that are more than twice the training length (§4, Table 1).

In this paper, we introduce Dual Chunk Attention (DCA), a new training-free framework to extrapolate the context window of LLMs. We avoid linearly downscaling the position indices or increasing the base frequency in RoPE (Su et al., 2022). Instead, we opt to reuse the original position

indices with their embeddings from the pretrained model, yet to redesign the construction of the relative position matrix in such a way that it can accurately reflect the relative position of two tokens as faithfully as possible. Inspired by efficient chunk-based attention patterns (Child et al., 2019; Song et al., 2023; Ratner et al., 2023; He et al., 2024), DCA segments self-attention computations for a long sequence into small chunks, each chunk being smaller than the size of the pretraining window. DCA consists of three components: (1) intra-chunk attention, tailored for processing tokens within the same chunk; (2) inter-chunk attention, for processing tokens between distinct chunks; and (3) successive chunk attention, for processing tokens in successive, distinct chunks. These respective treatments help the model effectively capture both long-range and short-range dependencies in a sequence. In addition to that, the chunk-based attention calculation can be seamlessly integrated with Flash Attention 2 (Dao et al., 2022; Dao, 2023), a key element for long-context scaling in the open-source community.[1]

We present a comprehensive evaluation of our models on a diverse range of tasks that include language modeling, passkey retrieval, and real-world long-context applications that span question answering (Pang et al., 2022; Kočiský et al., 2018; Dasigi et al., 2021; An et al., 2023) and summarization (Zhong et al., 2021). Unlike previous work that is usually limited to verification on 7B/13B models, the significant training efficiency of our method makes it possible to validate on 70B models, ensuring robust conclusions. To verify the model's long-context ability independent of potential data exposure during pretraining, we used this paper itself as the input and crafted a series of questions for the models.[2] Our empirical results reveal the following insights:

1. **Extrapolation**. On language modeling, DCA marks a significant advance for training-free approaches. It first shows that LLMs with a 4k context window can be expanded to more than 32k without training, maintaining a negligible increase in PPL, whereas previous methods typically falter at context lengths beyond 8k. Furthermore, we demonstrate that Llama2 70B, when integrated with DCA, showcases an exceptional extrapolation capability to handle context sizes exceeding 100k tokens.
2. **Orthogonality**. DCA is orthogonal to existing popular scaled positional encodings such as PI (Chen et al., 2023b) and NTK (LocalLLaMA, 2023b;a). We empirically show that existing long-context LLMs, which have already supported a 32k context window, can further extrapolate to a 192k context length while maintaining high passkey retrieval accuracy and low perplexity.
3. **Long-Context Understanding**. We evaluate DCA on a suite of long-context understanding benchmarks in both zero-shot and few-shot settings. The results suggest that our training-free models achieve performance comparable to, or even surpassing, that of existing state-of-the-art models built through costly continual training.

2. Background

2.1. Positional Encoding

The original positional embedding from the Transformer model (Vaswani et al., 2017) maps absolute position indices to a d-dimensional feature space, and incorporates this into the input layer. The input $\mathbf{x}$, associated with the position index i, is expressed as: $\mathbf{x}_i = \mathbf{x} + f(i)$, where $f : \mathbb{N} \to \mathbb{R}^d$ is the (positional) embedding function.

One of the most prevalent positional encoding methods for LLMs is the Rotary Positional Encoding (RoPE) (Su et al., 2022). RoPE eschews the conventional approach of infusing positional information into the input layer. Instead, it directly incorporates this information into the attention layer. For a sequence of l tokens, we denote the position indices for keys and queries[3] as follows:

$$P_{\mathbf{k}} = P_{\mathbf{q}} = [0, 1, \ldots, l-1]. \tag{1}$$

We abuse the notation f for the embedding function of RoPE, which accepts a query vector $\mathbf{q}$ or a key vector $\mathbf{k}$, and the respective position index as arguments. For example, we have $\mathbf{q}_i = f(\mathbf{q}, P_{\mathbf{q}}[i])$ and $\mathbf{k}_j = f(\mathbf{k}, P_{\mathbf{k}}[j])$, where $[i]$ denotes the i-th element of the list. In the most straightforward case, we have $P[i] = i$. The function f[4]outputs a modified query or key vector that encapsulates the position index, ensuring that the inner product between the i-th query and the j-th key (for $i \geq j$) captures the relative positional information $P_{\mathbf{q}}[i] - P_{\mathbf{k}}[j]$. Although RoPE takes absolute position indices as input, the result of the inner product of $\mathbf{q}, \mathbf{k}$ only contains relative position information (i.e., we have $\mathbf{q}_i^\top \mathbf{k}_j = \mathbf{q}_m^\top \mathbf{k}_n$ whenever $m - n = i - j$). The relative position matrix M introduced by RoPE during self-attention can be described as a Toeplitz matrix, as shown in Figure 1. Each element $M[i][j] = P_{\mathbf{q}}[i] - P_{\mathbf{k}}[j]$ signifies the relative position between $\mathbf{q}_i$ (the i-th query) and $\mathbf{k}_j$ (the j-th key).

2.2. Extrapolation of RoPE

Recent work (Chen et al., 2023b; Chowdhury & Caragea, 2023; Chen et al., 2023a) has demonstrated that LLMs with the original RoPE lack robust length extrapolation capabilities, typically resulting in performance degradation when

[1]Without Flash Attention, the maximum input tokens for Llama2 7B/13B is about 16k, and for Llama2 70B, it is 5k when tested on two A100 80G GPUs in our experiments

[2]We invite interested readers to examine the results in Tables 6,7

[3]Queries and keys are usually derived by projecting the input $\mathbf{x}$ through a learnable linear layer.

[4]A typical implementation of f can be found in modeling_llama.py Line 211 `apply_rotary_pos_emb()`

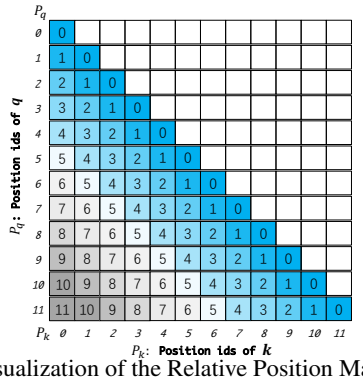

Figure 1. Visualization of the Relative Position Matrix M utilizing standard RoPE. The pretraining context window is 6 and the input sequence length is 12. The x-axis $P_{\mathbf{k}}$ indicates the position indices of keys, while the y-axis $P_{\mathbf{q}}$ corresponds to the position indices of queries. Each matrix entry $M[i][j]$ represents the relative positional offset $P_{\mathbf{q}}[i] - P_{\mathbf{k}}[j]$.

tested on input sequences longer than those seen during pretraining (Li et al., 2023b; Zhu et al., 2023). Recent studies (Chen et al., 2023b; Su, 2023; Jin et al., 2024) mainly attribute this limitation to the presence of unseen relative positions in pretraining phase and propose to redesign the relative position matrix. As illustrated in the example in Figure 1, the model is trained on sequences of 6 tokens, while inference is carried out on a sequence of 12 tokens. This discrepancy can lead to a high PPL because relative positions beyond 6 were never trained. Previous approaches, such as PI and NTK, aim to mitigate this issue by reducing the magnitude of $M[i][j]$ to ensure it falls within the scope of the observed context length during training. For instance, applying PI in this example would adjust the position indices by scaling: $P_{\mathbf{q}}[i] \Rightarrow P_{\mathbf{q}}[i]/2$ and $P_{\mathbf{k}}[j] \Rightarrow P_{\mathbf{k}}[j]/2$. Consequently, the relative position matrix is also scaled: $M[i][j] = M[i][j]/2$. Here, a scaling factor $2 = \frac{12}{6}$ is employed to scale down the relative positions, leading to inferior resolution of the position information and weak extrapolation ability.

3. Method

In this section, we describe our new training-free framework Dual Chunk Attention in detail. A running example of dual chunk attention is shown in figure 2. Our method starts from the intra-chunk attention (Figure 2 (a)) which is a chunk-based efficient attention pattern (Child et al., 2019; Song et al., 2023). The position embedding of each chunk ranges from 0 to chunk size where the chunk size is set to be smaller than pretraining length. The intra-chunk attention pattern practically means directly truncating the input from left to the chunk size discarding information from previous chunks. Such truncation usually brings low perplexity (Xiao et al., 2023) but loses long-range information. To address this limitation, we implement inter-chunk attention (Figure 2 (b)) that enables attention calculations between different chunks, albeit with less precision for distant token positions. Finally, we introduce successive-chunk attention, a variant of inter-chunk attention depicted in Figure 2 (c), which is specifically applied when two chunks are adjacent in order to preserve locality. An ablation study to show how these attention mechanisms influence PPL and passkey retrieval accuracy can be found in Figure 4.

3.1. Intra-Chunk Attention

Intra-Chunk Attention is employed to calculate the inner product of queries and keys within the same chunk. For a long sequence of length l, we partition the sequence into $n = \frac{l}{s}$ chunks, ensuring that the position indices within each chunk will not exceed the chunk size s. Figure 2 (a) illustrates the process of segmenting a sequence of 12 tokens exceeding the pretraining length 10 into 2 chunks, with each chunk comprising $s = 6 < 10$ tokens. Then the position indices for keys and queries are scaled within the chunk size 6. Concretely, we have position indices for keys $P_{\mathbf{k}} = [\underbrace{0, 1, 2, 3, 4, 5}_{\text{chunk 0}}, \underbrace{0, 1, 2, 3, 4, 5}_{\text{chunk 1}}]$ and $P_{\mathbf{q}}^{\text{Intra}} = P_{\mathbf{k}}$, where $P_{\mathbf{q}}^{\text{Intra}}$ means position indices for queries during intra-chunk attention. To formalize, in intra-chunk attention, we adjust the position indices for queries and keys as follows:

$$P_{\mathbf{q}}^{\text{Intra}} = P_{\mathbf{k}} = [0, 1, \ldots, l-1] \mod s. \tag{2}$$

For the absolute indices i and j within the same chunk i.e., $\lfloor i/s \rfloor = \lfloor j/s \rfloor$, satisfying $0 \leq j \leq i < l$, the element $M[i][j]$ is defined as the difference between the positional encodings of the query and the key:

$$M[i][j] = P_{\mathbf{q}}^{\text{Intra}}[i] - P_{\mathbf{k}}[j]. \tag{3}$$

When $\lfloor i/s \rfloor = \lfloor j/s \rfloor$, we calculate $M[i][j]$ follows Eq. 3. The computed M of the previous example where we have a sequence length of 12 and a chunk size of 6, is illustrated in Figure 2 (a). The intra-chunk attention score for the interaction between the i-th query and the j-th key is then calculated as:

$$\mathbf{q}_i^\top \mathbf{k}_j = f(\mathbf{q}, P_{\mathbf{q}}^{\text{Intra}}[i])^\top f(\mathbf{k}, P_{\mathbf{k}}[j]). \tag{4}$$

3.2. Inter-Chunk Attention

To aggregate information from other chunks, we introduce Inter-Chunk Attention. In Llama-based LLMs, the position indices for queries are greater than those of the keys to reflect the left-to-right information flow, i.e, we have $P_{\mathbf{q}}[i] \geq P_{\mathbf{k}}[j]$ whenever $i \geq j$. Using $P_{\mathbf{q}} = P_{\mathbf{q}}^{\text{Intra}}$ and $P_{\mathbf{k}}$ for attention calculation between different chunks clearly violates this property. For example, considering $\mathbf{q}_s$ and $\mathbf{k}_1$ where s is the chunk size, their relative distance given by $P_{\mathbf{q}}^{\text{Intra}}[s] = 0$ and $P_{\mathbf{k}}[1] = 1$ is -1. We maintain the position

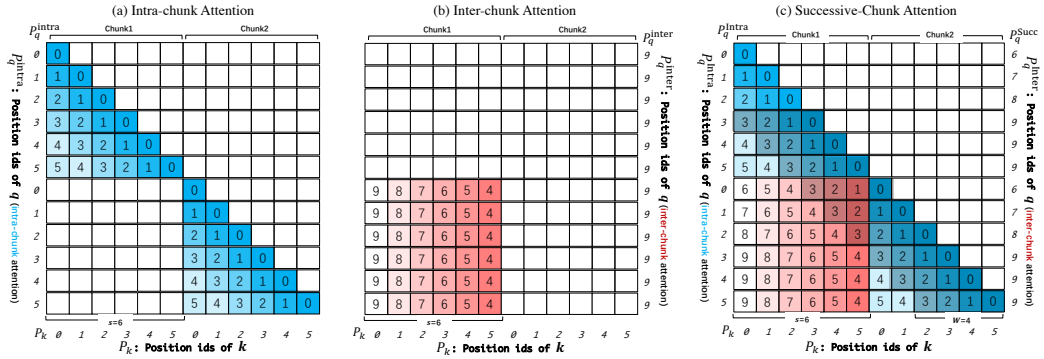

Figure 2. Visualization of the Relative Position Matrix M employing Dual Chunk Attention (DCA), with chunk size $s = 6$, pretraining window size $c = 10$, and local window size $w = 4$ noted by the shadow in (c). The sequence is segmented into chunks to ensure that relative positions do not exceed 9. The matrix element $M[i][j]$ represents the relative position between the i-th query vector $\mathbf{q}$ and the j-th key vector $\mathbf{k}$. Unlike the original position indices for $\mathbf{q}, \mathbf{k}$ in RoPE, DCA utilizes distinct position index sets $P_{\mathbf{k}}$, $P_{\mathbf{q}}^{\text{Intra}}$ (defined in Eq. 2), $P_{\mathbf{q}}^{\text{Inter}}$ (defined in Eq. 5), $P_{\mathbf{q}}^{\text{Succ}}$ (defined in Eq. 7) to compute the relative distances within different sections of M.

indices for keys $P_{\mathbf{k}}$ considering KV cache and seek for a new set of $P_{\mathbf{q}}$ during inter-chunk attention, noted as $P_{\mathbf{q}}^{\text{inter}}$. Given Eq. 2, the position indices for keys are cyclically repeated with the maximum position index $\max(P_{\mathbf{k}}) = s - 1$. To ensure that the queries have larger position indices than all keys from previous chunks, A simple strategy to distinguish distant queries is to assign them a considerably large position index, such as the maximum position index during pretraining $c - 1 > \max(P_{\mathbf{k}})$, where c is the pretraining context length:

$$P_{\mathbf{q}}^{\text{Inter}} = [\underbrace{c-1, c-1, \ldots c-1}_{l \text{ elements}}], \tag{5}$$

When $\lfloor i/s \rfloor \neq \lfloor j/s \rfloor$, we can give the relative position matrix M with $\mathbf{q}_i$ and $\mathbf{k}_j$ from distinct chunks as:

$$M[i][j] = P_{\mathbf{q}}^{\text{Intra}}[i] - P_{\mathbf{k}}[j] = c - 1 - P_{\mathbf{k}}[j] \geq c - s. \tag{6}$$

As reflected in Figure 2 (b), we assign $P_{\mathbf{q}}^{\text{Inter}}$ with a constant value of $c - 1 = 9$ for all positions, which is larger than the maximum position index $s - 1 = 5$ in $P_{\mathbf{k}}$. We complete the rest part of the matrix M left blank by intra-chunk attention with Eq. 6.

3.3. Successive-Chunk Attention

Successive-Chunk Attention can be viewed as a special case for inter-chunk attention, proposed to maintain the locality of LLMs where locality means LLMs tend to heavily rely on the neighboring tokens to predict the next token (Xiao et al., 2023; Han et al., 2023). Simply using inter-chunk attention may no longer keep the precise relative position between neighboring tokens, leading to performance degradation.

As shown in Figure 2(b), where the chunk size is $s = 6$ and the pretraining length is $c = 10$, the last key of the first chunk, $\mathbf{k}_5$, with $P_{\mathbf{k}}[5] = 5$, is followed by the first query of the second chunk, $\mathbf{q}_6$, with the position index $P_{\mathbf{q}}^{\text{Inter}}[6] = 9$. Despite their absolute distance being 1, the relative distance between $\mathbf{q}_6$ and $\mathbf{k}_5$ is $P_{\mathbf{q}}^{\text{Inter}}[6] - P_{\mathbf{k}}[5] = 4$. This configuration challenges the model's ability to maintain locality in its attention mechanism.

Fortunately, this issue only occurs between successive chunks, so we introduce a new successive-chunk attention to deal with this case. Concretely, we propose to maintain the locality of w neighboring tokens via adjusting the first w position indices in for $P_{\mathbf{q}}^{\text{Inter}}$. For example, in Figure 2 (c), given pretraining context $c = 10$, chunk size $s = 6$, and $P_{\mathbf{q}}^{\text{Inter}} = [\underbrace{9,9,9,9,9,9}_{\text{chunk 0}}, \underbrace{9,9,9,9,9,9}_{\text{chunk 1}}]$, the position indices $P_{\mathbf{q}}^{\text{Succ}}$ can be set to $[\underbrace{6,7,8,9,9,9}_{\text{chunk 0}}, \underbrace{6,7,8,9,9,9}_{\text{chunk 1}}]$ for attention calculation between successive chunks, if we keep a local window of $w = 4$. Formally, given chunk size s, pretraining size c and local window w we have:

$$P_{\mathbf{q}}^{\text{Succ}} = [\underbrace{\overbrace{s, s+1, \ldots, s+w-1}^{w \text{ elements}}, c-1, \ldots, c-1}_{\text{the same for all chunks}}], \tag{7}$$

where w means the local window size and can be directly set to the difference between pretraining length and chunk size $c - s$. For i, j from successive chunks, the calculation results of $M[i][j]$ using $P_{\mathbf{q}}^{\text{Succ}}andP_{\mathbf{k}}$ are reflected in Figure 2 (c) where the shadow means the resulting local window. Eq 7 ensures that the neighboring w keys have the closest distance to the current query.

By combining intra-chunk, inter-chunk, and successive-

chunk attention, we finally calculate $M[i][j]$ as:

$$M[i][j] = \begin{cases} P_{\mathbf{q}}^{\text{Intra}}[i] - P_{\mathbf{k}}[j] & \text{if } \lfloor i/s \rfloor - \lfloor j/s \rfloor = 0 \\ P_{\mathbf{q}}^{\text{Succ}}[i] - P_{\mathbf{k}}[j] & \text{if } \lfloor i/s \rfloor - \lfloor j/s \rfloor = 1 \\ P_{\mathbf{q}}^{\text{Inter}}[i] - P_{\mathbf{k}}[j] & \text{if } \lfloor i/s \rfloor - \lfloor j/s \rfloor > 1. \end{cases}$$

The inner product of $\mathbf{q}, \mathbf{k}$ in DCA is consequently defined as:

$$\mathbf{q}_i^T \mathbf{k}_j = \begin{cases} f(\mathbf{q}, P_{\mathbf{q}}^{\text{Intra}}[i])^T f(\mathbf{k}, P_{\mathbf{k}}[j]), & \text{if } \lfloor i/s \rfloor - \lfloor j/s \rfloor = 0 \\ f(\mathbf{q}, P_{\mathbf{q}}^{\text{Succ}}[i])^T f(\mathbf{k}, P_{\mathbf{k}}[j]), & \text{if } \lfloor i/s \rfloor - \lfloor j/s \rfloor = 1 \\ f(\mathbf{q}, P_{\mathbf{q}}^{\text{Inter}}[i])^T f(\mathbf{k}, P_{\mathbf{k}}[j]), & \text{if } \lfloor i/s \rfloor - \lfloor j/s \rfloor > 1, \end{cases} \quad (8)$$

3.4. Normalization

Softmax layer The inner product calculations within the DCA are formalized as shown in Equation 8. Subsequently, a softmax function is applied to normalize the computed inner products:

$$\mathbf{p}_i = \text{softmax}(\left[\frac{\mathbf{q}_i^\top \mathbf{k}_0}{\sqrt{d}}, \frac{\mathbf{q}_i^\top \mathbf{k}_1}{\sqrt{d}}, \ldots, \frac{\mathbf{q}i^\top \mathbf{k}_i}{\sqrt{d}}\right]). \quad (9)$$

where d denotes the dimension of hidden states.

Flash Attention The PyTorch-style pseudocode for how integrating DCA with Flash Attention 2 (Dao, 2023), can be found in Algorithm 1. The explanation and complexity analysis of the code can be found in Appendix §A.3. With Flash Attention, DCA attains comparable GPU memory usage and inference speed to the original self-attention in Llama. Results can be found in Figure 3.

4. Experiments

We evaluate our framework, DCA, on various variants of Llama2 (Touvron et al., 2023b), specifically the 7B, 13B, and 70B models, along with their chat counterparts, which have a 4k pretraining context. Our Llama2-based model is denoted as **CHUNKLLAMA2**. Additionally, we apply DCA to two popular open-source long context models: (1) Together-32k (Together, 2023)[5]: This model uses Positional Interpolation (PI) as its positional encoding. The DCA-enhanced version of this model is referred to as ChunkTogether. (2) CodeLlama (Rozière et al., 2023)[6]: This model applies NTK-Aware RoPE. Following the application of DCA, the resulting model is termed ChunkCodeLlama.

4.1. Experimental Setup

DCA can be implemented by a monkey patch to replace the inference code of the original `LlamaAttention`. Thanks to Flash Attention 2 (Dao, 2023), for the 7B/13B variants of CHUNKLLAMA2, we only need one single NVIDIA A100-80G GPU for the inference. When scaling up to 70B models, two A100 GPUs are enough to manage inference within a 16k context length. The chunk size s can be typically set to $\frac{3}{4}$ training length and for Llama2, this value is 3072. The number of chunks depends on the input sequence length.

In addition to training-free evaluations, we also provide finetuned models from 7B/13B Llama2 checkpoints. This finetuning process leverages only long conversations with 16k input tokens, following Vicuna (LMSYS, 2023) and LongChat (Li et al., 2023a). The training dataset is sourced from ShareGPT[7] and AlpacaGPT4 (Taori et al., 2023). For the data derived from ShareGPT, we specifically curate a subset by extracting responses generated by GPT-4, and dialogues that exceed 4k tokens in length. This selection results in a compilation of 5,405 training instances.

We adhere to the training hyperparameters as specified in the LongChat repository[8]. We further finetune Llama2 with over 16k steps with a batch size of 1. The finetuning process amounts to approximately 40 GPU hours for the 7B model and 60 GPU hours for the 13B variant.

Datasets We evaluate the long sequence language modeling performance of our CHUNKLLAMA2 on the book corpus dataset PG19 (Rae et al., 2020), with context lengths ranging from 4k to 192k tokens. For the 7B and 13B models, we employ a sliding window of 256, in line with previous work (Peng et al., 2023; Chen et al., 2023c). For 70B models, we adjust the sliding window size to 2048 and when dealing with contexts that exceed 96k tokens, we adjust the sliding window to be half of the input length considering the running time. For few-shot experiments, we follow the settings in Llama2 Long (Xiong et al., 2023). Concretely, we evaluate 0-shot performance of CHUNKLLAMA2 on NarrativeQA (Kočiský et al., 2018), 1-shot on QMSum (Zhong et al., 2021), 2-shot on QuALITY (Pang et al., 2022) , and 2-shot for Qasper (Dasigi et al., 2021). For zero-shot experiments, we test CHUNKLLAMA2 on 4 closed-ended tasks from L-Eval (An et al., 2023): TOFEL, QuALITY (cleaned from Pang et al. (2022)), Coursera, SFiction. We also validate our model on passkey retrieval used in Mohtashami & Jaggi (2023). Evaluations on passkey retrieval (Mohtashami & Jaggi, 2023) can be found in Appendix A.1.

Baselines We compare with popular open-source long-context models available in Huggingface Transformers[9]. *Base Models*: Focused Transformer 3B (Tworkowski et al., 2023), CLEX 7B (Chen et al., 2023a), YaRN 7B/13B (Peng et al., 2023), MPT 30B (MosaicML, 2023b;a), Together 7B (Together, 2023), CodeLlama 7B (Rozière et al., 2023),

[5] https://huggingface.co/togethercomputer/LLaMA-2-7B-32K

[6] https://huggingface.co/codellama

[7] https://sharegpt.com/

[8] https://github.com/DachengLi1/LongChat

[9] prior to December 1, 2023

Longlora 13B/70B (Chen et al., 2023c), and Llama2 Long 7B/13B/70B (Xiong et al., 2023). *Chat Models*: LongChat-v1.5-32k 7B (Li et al., 2023a), Vicuna-v1.5-16k (LMSYS, 2023) 7B/13B, Longlora-Chat 70B (Chen et al., 2023c), and Llama2 Long-Chat 70B (Xiong et al., 2023).

4.2. Long-Sequence Language Modeling

Table 1 presents the Perplexity (PPL) scores on the PG19 validation set for various training-free and finetuned models. All these baselines are Llama-based. We demonstrate that the previously best training-free method fails with a context length of 16k. However, CHUNKLLAMA2 can extrapolate to a context window of more than 32k, with only an increase of 0.02 in PPL. We further demonstrate that CHUNKLLAMA2 surpasses the results of finetuned models within a 16k context length. Notably, the 70B variant of CHUNKLLAMA2 exhibits consistency in performance across a range of context lengths, achieving a PPL score that only marginally rises from 5.18 to 5.59.

We also reveal that DCA can be integrated with models that have been further trained on longer contexts with PI (Chen et al., 2023b) or NTK-Aware RoPE (LocalLLaMA, 2023b;a) and support a context length of 192k in Table 2. The encouraging outcomes observed with 64k input tokens motivate us to test CHUNKLLAMA2 on even longer contexts. We progressively tested the model with input token lengths extending from 32k to 192k (Table 2). For Llama2 70B, DCA has proven effective in extending the context window to 96k tokens. This extension is achieved with only a minor increase of 0.56 PPL compared to its original performance at a 4k context length. Alongside evaluating CHUNKLLAMA2, we also applied DCA to existing long-context models that utilize different positional encodings. Integrating DCA with existing long-context models requires only an adjustment of the chunk size within the DCA framework. We show that CodeLlama and Together's Llama2 fork can be efficiently scaled to a 192k context length using DCA with a chunk size of 24k. We further validated the performance of our model on the passkey retrieval task (Mohtashami & Jaggi, 2023). The results also indicate that by integrating DCA with existing long-context models, the enhanced system maintains a 90% retrieval accuracy with an extended context length of up to 192k tokens (Figure 7).

4.3. Practical Tasks

In contrast to previous studies that typically validate their methods based on PPL, we also apply our framework to both base models and instruction-finetuned chat models on real-world benchmarks.

Few-shot Results We validate DCA on models that have not undergone instruction tuning in a few-shot learning setting. The results are summarized in Table 3. Experimental settings are the same as those in Xiong et al. (2023). If the input prompts exceed an input length of 16k tokens, they are truncated from the left side. Most test cases within NarrativeQA (Kočiský et al., 2018) and QMSum (Zhong et al., 2021) have input lengths exceeding 16k tokens, while the lengths of test cases in Qasper (Dasigi et al., 2021) and QuALITY (Pang et al., 2022) are generally under 8k tokens. Without any training cost, both the 7B/13B variants of CHUNKLLAMA2 achieve results comparable to popular finetuned baselines such as YaRN (Peng et al., 2023), MPT (MosaicML, 2023b), Together (Together, 2023), which are based on previous scaled RoPE (Chen et al., 2023b; LocalLLaMA, 2023b) or Alibi (Press et al., 2022).

Unlike previous studies that usually verify their techniques on smaller versions of Llama2, we also present results for DCA paired with Llama2 70B, where DCA improves performance by an average of more than 8.0 points over the original Llama2 model with a 4k training length. Given the increasing cost of long-context finetuning for 70B models, we did not find many open-source 70B baselines. We compare our training-free method against the robust 70B baseline, Longlora (Chen et al., 2023c), which employs LoRA-based (Hu et al., 2021) efficient tuning based on the Redpajama dataset (Computer, 2023) for 1000 steps supporting a 32k context window. The results demonstrate that our

Table 1. Perplexity (PPL) evaluation on PG19 (Rae et al., 2020) validation set. The results highlighted in red indicate the Perplexity has increased by more than 1.0 compared to its original value at the pretraining context length of 4096. ReRoPE (Su, 2023) encounters OOM (Out of Memory) problems with 16k input tokens as it is currently not compatible with Flash Attention. The scaling factors in PI and NTK are dynamically changed.

Model	Evaluation Context Window				
	4096	8192	16384	32768	65536
Finetuned					
Longlora-32k 7B	**8.14**	**7.85**	**7.70**	7.80	91.79
Together-32k 7B	8.21	7.95	7.76	**7.64**	$>10^2$
CodeLlama-16k 7B	8.93	8.64	8.44	8.36	**8.65**
CLEX-16k 7B	8.84	7.66	7.43	7.57	8.73
Training-free					
Llama2 7B	**7.87**	$>10^2$	$>10^2$	$>10^2$	$>10^2$
Llama2-ReRoPE 7B	7.94	7.75	OOM	OOM	OOM
Llama2-PI 7B	7.87	9.19	15.11	$>10^2$	$>10^2$
Llama2-PI-Yarn 7B	7.87	8.80	11.75	42.42	$>10^2$
Llama2-NTK 7B	7.87	11.98	26.12	58.91	$>10^2$
Llama2-NTK-Yarn 7B	7.87	8.06	9.82	11.74	41.57
CHUNKLLAMA2 7B	7.87	7.67	7.64	7.89	15.87
CHUNKLLAMA2 13B	7.15	6.95	6.99	7.90	15.14
CHUNKLLAMA2 70B	**5.24**	**5.18**	**5.21**	**5.30**	**5.59**
Llama3					
Llama3 8B	9.04	8.71	78.88	$>10^2$	$>10^2$
Llama3 70B	5.36	5.16	$>10^2$	$>10^2$	$>10^2$
CHUNKLLAMA3 8B	9.04	8.71	8.61	8.62	8.95
CHUNKLLAMA3 70B	**5.36**	**5.16**	**5.14**	**5.14**	**5.21**

Table 2. Perplexity evaluation on PG19 (Rae et al., 2020) validation set with context lengths of up to 192k tokens. We test DCA on Llama2 70B together with 2 popular further pretrained models using PI and NTK. The results highlighted in red indicate the PPL has increased by more than 1.0 compared to its original value at the pretraining context length of 4096.

Model	Position Emb	Training context	Evaluation Context Window 4k	32k	64k	96k	128k	160k	192k
Llama2 7B	RoPE	4k	7.87	$>10^2$	$>10^2$	$>10^2$	$>10^2$	$>10^2$	$>10^2$
CHUNKLLAMA2 7B	RoPE	4k	7.87	7.89	15.87	43.57	96.21	$>10^2$	$>10^2$
Llama2 70B	RoPE	4k	**5.24**	$>10^2$	$>10^2$	$>10^2$	$>10^2$	$>10^2$	$>10^2$
CHUNKLLAMA2 70B	RoPE	4k	**5.24**	**5.30**	**5.59**	**5.80**	**6.12**	6.52	7.05
Llama3 8B	RoPE	8k	9.04	$>10^2$	$>10^2$	$>10^2$	$>10^2$	$>10^2$	$>10^2$
CHUNKLLAMA3 8B	RoPE	8k	9.04	8.61	8.62	8.95	9.43	10.04	10.66
Llama3 70B	RoPE	8k	**5.36**	$>10^2$	$>10^2$	$>10^2$	$>10^2$	$>10^2$	$>10^2$
CHUNKLLAMA3 70B	RoPE	8k	**5.36**	**5.14**	**5.14**	**5.21**	**5.32**	**5.40**	**5.45**
CodeLlama 7B	NTK	16k	8.93	8.36	8.65	9.14	9.87	15.68	24.78
ChunkCodeLlama 7B	NTK	16k	8.93	8.36	8.13	8.33	8.66	9.30	9.83
Together 7B	PI	32k	8.21	**7.64**	$>10^2$	$>10^2$	$>10^2$	$>10^2$	$>10^2$
ChunkTogether 7B	PI	32k	8.21	**7.64**	**7.59**	**7.64**	**7.67**	**7.74**	**7.83**

Table 3. Comparison between popular open-source base models (first block) and proprietary models (last block) across four research benchmarks on their validation set. We underline the best results in each block. Results exceeding the previous best open-source finetuned model are in **bold**. Llama2 Long has been trained with a total of 400B tokens over 100,000 steps. The maximum allowed prompt length is set to 16,384 tokens. †: results are taken from Xiong et al. (2023) We use the simplest prompt: *long-document Question:... Answer:*. In-context examples are randomly selected from the training set, and we also have a discussion on the selection of in-context examples in Appendix §A.4.

Model	Further training	Training context	**NarrativeQA** F1 (0-shot)	**Qasper** F1 (2-shot)	**QuALITY** EM (2-shot)	**QMSum** R-g (1-shot)	**Avg**
FoT 3B†	✓	8k	16.3	15.4	20.5	10.6	15.7
Yarn 7B†	✓	128k	20.9	26.2	32.3	11.4	22.7
Together 7B†	✓	32k	23.3	27.3	41.2	12.6	26.1
Yarn 13B †	✓	128k	23.4	27.1	46.4	11.9	27.2
Longlora 13B	✓	32k	25.8	26.4	48.9	15.1	29.1
MPT 30B†	✓	8k	22.9	29.0	41.5	10.3	25.9
Llama2-DynNTK 70B	✗	4k	11.1	27.8	60.9	7.8	26.9
Llama2 70B†	✗	4k	25.7	27.5	53.0	11.9	29.5
Longlora 70B	✓	32k	34.2	29.0	69.9	15.6	37.2
CHUNKLLAMA2 7B	✗	4k	20.0	28.2	35.6	14.7	24.6
CHUNKLLAMA2 13B	✗	4k	26.3	**29.3**	47.9	15.2	29.7
CHUNKLLAMA2 70B	✗	4k	32.5	**29.6**	**73.2**	**16.0**	**37.8**
CHUNKLLAMA3 8B	✗	8k	27.4	30.5	52.6	15.4	31.5
CHUNKLLAMA3 70B	✗	8k	**33.7**	**33.1**	**75.4**	**16.0**	**39.5**
proprietary models							
Llama2 Long 7B†	✓	32k	21.9	27.8	43.2	14.9	27.0
Llama2 Long 13B†	✓	32k	25.6	31.2	57.6	15.7	32.5
Llama2 Long 70B†	✓	16k	30.9	35.7	79.7	16.5	40.7

70B DCA model achieves comparable performance (37.8 vs. 37.2) requires no training steps.

Compared to the strong proprietary baseline, Llama2 Long (Xiong et al., 2023), which has been trained with a total of 400 billion tokens (Llama2 pretraining corpus and new long text data) over 100,000 steps, the performance gaps for all sizes of models are generally within a 3-point range. The in-context examples used in this experiment are randomly selected from the training set. We have also tried other ways to select the examples, and the details are included in Appendix A.4.

Zero-shot Results In addition to verifying DCA on base models, we also apply DCA on the **chat version** of Llama2 (with instruction tuning) in a zero-shot learning scenario. Specifically, we test our models on four closed-ended tasks from L-Eval (An et al., 2023) with diverse input lengths ranging from 3k to 27k. All these datasets adopt Exact Match (EM) as the evaluation metric. Overall, the conclusions are similar to the few-shot evaluation. Our training-free 7B/13B models show comparable performance with open-source models with further training. Notably, in zero-shot experiments, we demonstrate a significant improvement over the Chat version of Longlora 70B (Chen et al., 2023c). Furthermore, when compared with proprietary models such as GPT-3.5 with a 16k token context and the chat

Table 4. Comparison with open-source **chat** models (first block) and proprietary models (last block) on 4 closed-ended tasks with various input lengths from L-Eval (An et al., 2023). We underline the best results in each block. Results exceeding previous the best open-source finetuned model are in **bold**. 'dialogues' means the mix of ShareGPT and AlpacaGPT4 used in our training. Llama2-PI-SFT and Llama2-NTK-SFT are models trained with the same data and training steps with CHUNKLLAMA2. ‡: results are taken from Xiong et al. (2023).

Model	Finetuning corpus	Training context	TOFEL (3k∼5k)	QuALITY (4k∼9k)	Coursera (5k∼17k)	SFiction (6k∼27k)	Avg
Llama2-Chat 7B	✗	4k	51.67	37.62	29.21	60.15	48.74
Llama2-DynNTK 7B	✗	4k	52.27	30.69	13.95	57.02	38.48
Longchat-v1.5-32k 7B	ShareGPT	32k	39.77	37.62	32.99	57.02	41.85
Llama2-PI-SFT 7B	Dialogues	16k	56.13	38.61	36.19	53.90	46.20
Llama2-NTK-SFT 7B	Dialogues	16k	53.90	38.11	34.01	64.06	47.51
Vicuna-v1.5-16k 7B	ShareGPT	16k	55.39	39.60	38.66	60.15	48.45
Llama2-Chat 13B	✗	4k	60.96	42.57	35.75	54.68	48.99
Llama2-DynNTK 13B	✗	4k	62.45	33.16	37.06	60.93	48.40
Vicuna-v1.5-16k 13B	ShareGPT	16k	68.40	53.96	40.69	61.71	56.19
Longlora-Chat 70B	LongAlpaca	32k	71.37	55.45	44.76	67.96	59.88
Training-free							
CHUNKLLAMA2-Chat 7B	✗	4k	57.62	35.14	32.12	61.72	46.64
CHUNKLLAMA2-Chat 13B	✗	4k	66.54	43.06	41.56	57.03	52.04
CHUNKLLAMA2-Chat 70B	✗	4k	**82.15**	**60.39**	**48.54**	61.72	**63.20**
Llama3							
CHUNKLLAMA3-Instruct 8B	✗	8k	83.27	63.86	56.24	70.31	68.42
CHUNKLLAMA3-Instruct 70B	✗	8k	84.75	82.17	76.88	75.78	79.89
Finetuned							
CHUNKLLAMA2-Chat 7B	Dialogues	16k	62.08	41.58	39.68	64.06	51.85
CHUNKLLAMA2-Chat 13B	Dialogues	16k	65.42	53.96	44.76	65.62	57.94
proprietary models							
GPT3.5-16k-0613	Unkown	–	78.43	**61.38**	63.51	64.84	67.03
Claude1.3-100k	Unkown	–	**83.64**	60.03	**73.76**	**72.65**	**72.52**
Llama2 Long-Chat 70B‡	Long doc+diag	16k	81.8	–	52.9	–	–

version of Llama2 Long, the results suggest that the Llama2 70B chat model can be directly scaled to a 16k context window without additional training with DCA, achieving 94% of the performance of `gpt-3.5-turbo-16k`. We also demonstrate that our model's performance can be enhanced through additional finetuning on long dialogue data following the approach used by Vicuna (LMSYS, 2023) and Longchat (Li et al., 2023a), both of which are popular finetuned baselines utilizing ShareGPT. With further training, CHUNKLLAMA2-Chat outperforms the previously best 13B model, Vicuna-v1.5-13b-16k, by a significant margin of 1.75 points.

4.4. Analysis

Efficiency In figure 3, the inference time and GPU memory of (a) the original self-attention mechanism as implemented in PyTorch, Flash Attention (Dao, 2023), and our proposed DCA (integrated with Flash Attention) are evaluated across various prompt lengths. These experiments are run on a single NVIDIA 80G A100 GPU using Llama2 7B. The input long prompt is from NarrativeQA (Kočiský et al., 2018). We conduct 20 trials and report the average performance. Without Flash Attention, we observe that the maximum input length manageable by a single GPU is roughly between 12k and 16k tokens. DCA sustains similar GPU memory consumption and inference speed, without adding considerable overhead, with the original Flash attention.

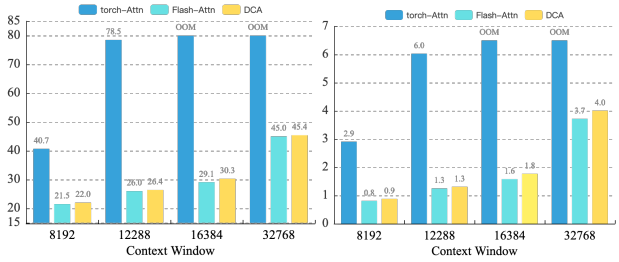

Figure 3. Inference time and GPU memory of (a) the original self-attention implemented by Pytorch, (b) Flash Attention (Dao, 2023), and (c) DCA (this work).

Ablation Study To validate the three attention mechanisms proposed in this work, we present an ablation study for DCA in Figure 4, focusing on language modeling and passkey retrieval tasks. We consider three experimental conditions: (1) Employing only intra-chunk attention. (2) Utilizing both intra-chunk and inter-chunk attention. (3) Combining all three types of attention: intra-chunk, inter-chunk, and successive chunk attention. From the results in language modeling, we observe that using intra-chunk attention which disregards information from previous chunks, is able to maintain a very low PPL but hinders the model's ability to retrieve passkeys from other chunks. Introducing inter-chunk attention, we notice an improvement in passkey retrieval performance at an input length of 12k.

However, the loss of locality causes a significant increase in the model's PPL. By integrating successive chunk attention, we achieve both a low PPL and high retrieval accuracy.

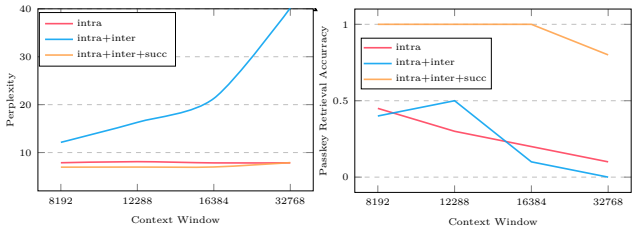

Figure 4. Ablation study of DCA on language modeling (left) and passkey retrieval (right). We test the three attention mechanisms with input sequences from 8k to 32k.

5. Conclusion

In this paper, we present Dual Chunk Attention (DCA) as a novel and efficient approach to overcoming the context length limitations inherent in LLMs. By ingeniously leveraging the model's existing position indices and introducing a multi-faceted attention mechanism, DCA allows for extrapolating more than 8x the training length without the need for costly and time-consuming further training.

Impact Statement

Numerous studies have emerged targeting to expand the supported context length of LLMs; however, due to high training costs and incompatibilities with technologies such as Flash Attention, the industry mainly relies predominantly on expanding the base frequency of RoPE or PI. Our Dual Chunk Attention (DCA) method is compatible with Flash Attention and requires only modifications to the inference code, negating the need for extensive retraining. DCA preserves model performance within the training length, and only benefits it beyond this range, offering compatibility with models that have already undergone long-context fine-tuning. Consequently, our approach may have a substantial impact on the industry, providing a cost-effective solution for managing long-context scenarios in LLM applications. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

Acknowledgements

We thank Yukang Chen and Hang Yan for their helpful comments and open-source code. This research was supported in part by the joint research scheme of the National Natural Science Foundation of China (NSFC) and the Research Grants Council (RGC) under grant number N_HKU714/21.

References

An, C., Gong, S., Zhong, M., Li, M., Zhang, J., Kong, L., and Qiu, X. L-eval: Instituting standardized evaluation for long context language models. *arXiv preprint arXiv:2307.11088*, 2023.

Anthropic. Introducing 100K Context Windows, 2023. URL https://www.anthropic.com/index/100k-context-windows.

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., Hui, B., Ji, L., Li, M., Lin, J., Lin, R., Liu, D., Liu, G., Lu, C., Lu, K., Ma, J., Men, R., Ren, X., Ren, X., Tan, C., Tan, S., Tu, J., Wang, P., Wang, S., Wang, W., Wu, S., Xu, B., Xu, J., Yang, A., Yang, H., Yang, J., Yang, S., Yao, Y., Yu, B., Yuan, H., Yuan, Z., Zhang, J., Zhang, X., Zhang, Y., Zhang, Z., Zhou, C., Zhou, J., Zhou, X., and Zhu, T. Qwen technical report, 2023.

Chen, G., Li, X., Meng, Z., Liang, S., and Bing, L. Clex: Continuous length extrapolation for large language models, 2023a.

Chen, S., Wong, S., Chen, L., and Tian, Y. Extending context window of large language models via positional interpolation, 2023b.

Chen, Y., Qian, S., Tang, H., Lai, X., Liu, Z., Han, S., and Jia, J. Longlora: Efficient fine-tuning of long-context large language models. *arXiv:2309.12307*, 2023c.

Chi, T.-C., Fan, T.-H., Rudnicky, A. I., and Ramadge, P. J. Dissecting transformer length extrapolation via the lens of receptive field analysis, 2023.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Chowdhury, J. R. and Caragea, C. Monotonic location attention for length generalization, 2023.

Computer, T. Redpajama: an open dataset for training large language models, 2023. URL https://github.com/togethercomputer/RedPajama-Data.

Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022.

Dasigi, P., Lo, K., Beltagy, I., Cohan, A., Smith, N. A., and Gardner, M. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter*

of the Association for Computational Linguistics: Human Language Technologies, pp. 4599–4610, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.365. URL https://aclanthology.org/2021.naacl-main.365.

Han, C., Wang, Q., Xiong, W., Chen, Y., Ji, H., and Wang, S. Lm-infinite: Simple on-the-fly length generalization for large language models, 2023.

He, Z., Feng, G., Luo, S., Yang, K., He, D., Xu, J., Zhang, Z., Yang, H., and Wang, L. Two stones hit one bird: Bilevel positional encoding for better length extrapolation, 2024.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021.

Jin, H., Han, X., Yang, J., Jiang, Z., Liu, Z., Chang, C.-Y., Chen, H., and Hu, X. Llm maybe longlm: Self-extend llm context window without tuning, 2024.

Kazemnejad, A., Padhi, I., Ramamurthy, K. N., Das, P., and Reddy, S. The impact of positional encoding on length generalization in transformers, 2023.

Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018. doi: 10.1162/tacl_a_00023. URL https://aclanthology.org/Q18-1023.

Lee, G., Hartmann, V., Park, J., Papailiopoulos, D., and Lee, K. Prompted llms as chatbot modules for long open-domain conversation. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.277. URL http://dx.doi.org/10.18653/v1/2023.findings-acl.277.

Li, D., Shao, R., Xie, A., Sheng, Y., Zheng, L., Gonzalez, J. E., Stoica, I., Ma, X., and Zhang, H. How long can open-source llms truly promise on context length. 2023a.

Li, S., You, C., Guruganesh, G., Ainslie, J., Ontanon, S., Zaheer, M., Sanghai, S., Yang, Y., Kumar, S., and Bhojanapalli, S. Functional interpolation for relative positions improves long context transformers, 2023b.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts, 2023a.

Liu, X., Yan, H., Zhang, S., An, C., Qiu, X., and Lin, D. Scaling laws of rope-based extrapolation, 2023b.

LMSYS. Vicuna: An open-source chatbot impressing gpt-4 with 90 URL https://lmsys.org/blog/2023-03-30-vicuna/.

LocalLLaMA. Dynamically scaled rope further increases performance of long context llama with zero fine-tuning, July 2023a. URL https://www.reddit.com/r/LocalLLaMA/comments/14mrgpr/dynamically_scaled_rope_further_increases/.

LocalLLaMA. Ntk-aware scaled rope allows llama models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation., June 2023b. URL https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have/.

Lv, K., Liu, X., Guo, Q., Yan, H., He, C., Qiu, X., and Lin, D. Longwanjuan: Towards systematic measurement for long text quality, 2024.

Mohtashami, A. and Jaggi, M. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023.

MosaicML. Introducing mpt-30b: Raising the bar for open-source foundation models, 2023a. URL www.mosaicml.com/blog/mpt-30b. Accessed: 2023-06-22.

MosaicML. Introducing mpt-7b: A new standard for open-source, ly usable llms, 2023b. URL www.mosaicml.com/blog/mpt-7b.

OpenAI. Gpt-4 technical report, 2023.

Pang, R. Y., Parrish, A., Joshi, N., Nangia, N., Phang, J., Chen, A., Padmakumar, V., Ma, J., Thompson, J., He, H., and Bowman, S. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5336–5358, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.391. URL https://aclanthology.org/2022.naacl-main.391.

Peng, B., Quesnelle, J., Fan, H., and Shippole, E. Yarn: Efficient context window extension of large language models, 2023.

Press, O., Smith, N. A., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation, 2022.

Qin, Z., Sun, W., Li, D., Shen, X., Sun, W., and Zhong, Y. Lightning attention-2: A free lunch for handling unlimited sequence lengths in large language models. *ArXiv*, abs/2401.04658, 2024. URL https://api.semanticscholar.org/CorpusID:266900042.

Rae, J. W., Potapenko, A., Jayakumar, S. M., Hillier, C., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SylKikSYDH.

Ratner, N., Levine, Y., Belinkov, Y., Ram, O., Magar, I., Abend, O., Karpas, E., Shashua, A., Leyton-Brown, K., and Shoham, Y. Parallel context windows for large language models, 2023.

Robertson, S., Zaragoza, H., et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M., Ferrer, C. C., Grattafiori, A., Xiong, W., Défossez, A., Copet, J., Azhar, F., Touvron, H., Martin, L., Usunier, N., Scialom, T., and Synnaeve, G. Code llama: Open foundation models for code, 2023.

Rula, A. and D'Souza, J. Procedural text mining with large language models, 2023.

Ruoss, A., Delétang, G., Genewein, T., Grau-Moya, J., Csordás, R., Bennani, M., Legg, S., and Veness, J. Randomized positional encodings boost length generalization of transformers, 2023.

Saad-Falcon, J., Barrow, J., Siu, A., Nenkova, A., Yoon, D. S., Rossi, R. A., and Dernoncourt, F. Pdftriage: Question answering over long, structured documents, 2023.

Song, K., Wang, X., Cho, S., Pan, X., and Yu, D. Zebra: Extending context window with layerwise grouped local-global attention, 2023.

Su, J. Rectified rotary position embeddings. https://github.com/bojone/rerope, 2023.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding, 2022.

Sun, Y., Dong, L., Patra, B., Ma, S., Huang, S., Benhaim, A., Chaudhary, V., Song, X., and Wei, F. A length-extrapolatable transformer, 2022.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Together. Llama-2-7b-32k-instruct — and fine-tuning for llama-2 models with together api, 2023. URL https://together.ai/blog/llama-2-7b-32k-instruct.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Tworkowski, S., Staniszewski, K., Pacek, M., Wu, Y., Michalewski, H., and Miłoś, P. Focused transformer: Contrastive training for context scaling, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2017.

Wang, L., Yang, N., and Wei, F. Learning to retrieve in-context examples for large language models, 2024.

Wei, J., Kim, S., Jung, H., and Kim, Y.-H. Leveraging large language models to power chatbots for collecting user self-reported data, 2023.

Xiao, C., Zhang, P., Han, X., Xiao, G., Lin, Y., Zhang, Z., Liu, Z., Han, S., and Sun, M. Infllm: Unveiling the intrinsic capacity of llms for understanding extremely long sequences with training-free memory, 2024.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks, 2023.

Xiong, W., Liu, J., Molybog, I., Zhang, H., Bhargava, P., Hou, R., Martin, L., Rungta, R., Sankararaman, K. A., Oguz, B., Khabsa, M., Fang, H., Mehdad, Y., Narang, S., Malik, K., Fan, A., Bhosale, S., Edunov, S., Lewis, M., Wang, S., and Ma, H. Effective long-context scaling of foundation models. *CoRR*, abs/2309.16039, 2023. doi: 10.48550/ARXIV.2309.16039. URL https://doi.org/10.48550/arXiv.2309.16039.

Ye, J., Wu, Z., Feng, J., Yu, T., and Kong, L. Compositional exemplars for in-context learning. *arXiv preprint arXiv:2302.05698*, 2023.

Zhang, J., Jiang, S., Feng, J., Zheng, L., and Kong, L. Linear attention via orthogonal memory. *ArXiv*, abs/2312.11135, 2023. URL https://api.semanticscholar.org/CorpusID:266359128.

Zhang, P., Liu, Z., Xiao, S., Shao, N., Ye, Q., and Dou, Z. Soaring from 4k to 400k: Extending llm's context with activation beacon. *ArXiv*, abs/2401.03462, 2024. URL https://api.semanticscholar.org/CorpusID:266844488.

Zhong, M., Yin, D., Yu, T., Zaidi, A., Mutuma, M., Jha, R., Awadallah, A. H., Celikyilmaz, A., Liu, Y., Qiu, X., and Radev, D. QMSum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5905–5921, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.472. URL https://aclanthology.org/2021.naacl-main.472.

Zhu, D., Yang, N., Wang, L., Song, Y., Wu, W., Wei, F., and Li, S. Pose: Efficient context window extension of llms via positional skip-wise training, 2023.

A. Appendix

A.1. Passkey retrieval

In addition to practical tasks, we evaluate the long-context capability of LLMs to perform the passkey retrieval task as defined in Mohtashami & Jaggi (2023). This task challenges a language model to locate a simple passkey (e.g., a five-digit random number) embedded within a lengthy and otherwise nonsensical text sequence. The primary purpose of this task is to determine if a Large Language Model (LLM) can maintain awareness of information distributed throughout a lengthy input sequence. To assess retrieval accuracy, we randomly place the passkey at various document depths which are distributed uniformly. For each document depth, we run 20 times with different passkeys and we test the input sequence length from 4k to 20k. We compare the performance of DCA with 2 popular extension methods: PI (Chen et al., 2023b), NTK-Aware (LocalLLaMA, 2023b;a), on the Llama2 13B model with a 4k pretraining context window. The performance results are depicted in Figure 5. Notably, within a context length of 18k tokens, our model CHUNKLLAMA2 consistently achieved a 100% passkey retrieval accuracy across all depths tested.

We expanded the scope of the passkey retrieval tasks by incrementally increasing the input token count from 2k to 192k. For each input context length, the model is evaluated 20 times, with the passkey's position randomly varied in each test. Additionally, we also verify the Together-32k 7B model (Together, 2023), which supports a 32k token context window, and its ChunkTogether 7B counterpart. The outcomes for both the baseline and DCA-enhanced variants of these models are illustrated in Figure 7. With only a 4k training context length, CHUNKLLAMA2 maintains high retrieval accuracy up to a 32k context length. By integrating these findings with existing long-context models, we can feasibly extend the supported context window to an impressive 192k tokens using a learning-free approach.

lost in the beginning: An intriguing observation is that the failure cases of PI appear to be largely unrelated to the document's depth, while the NTK-based approach typically excels when the passkey is positioned near the beginning of the document. However, its effectiveness significantly diminishes—with accuracy dropping to between 40% and 80%—when the passkey is placed in the middle sections. This trend aligns with findings reported by Liu et al. (2023a). Conversely, as the input context is expanded, CHUNKLLAMA2 demonstrates improved performance in the middle sections but the first place where a drop in accuracy occurs is at the beginning of the text.

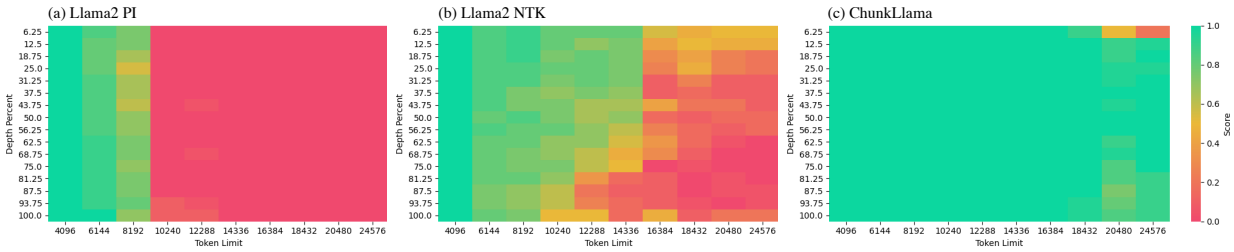

Figure 5. Testing Different Learning-Free Extension Methods with a 24K Context ("Needle in a Haystack" Passkey Retrieval). All the models have a 4k pretraining context and are not further trained. The X-axis represents the input context length, and the Y-axis indicates the depth of the passkey within the document. For each depth, we run 20 different test cases.

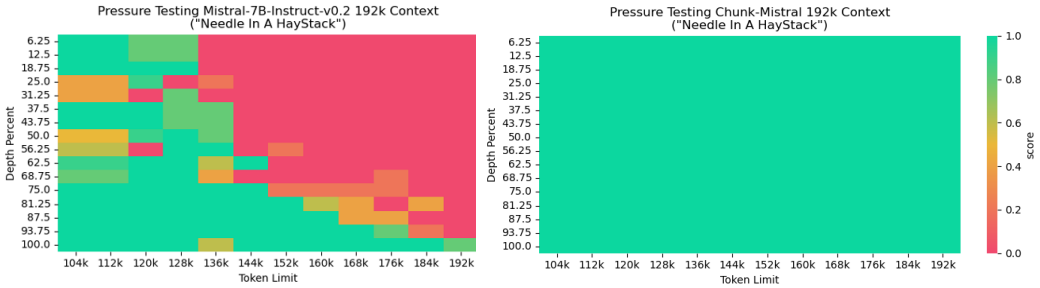

Figure 6. Pressure testing Mistral-7B-Instruct-v0.2 over a 192k context length for and its DCA enhanced version (("Needle In A HayStack")).

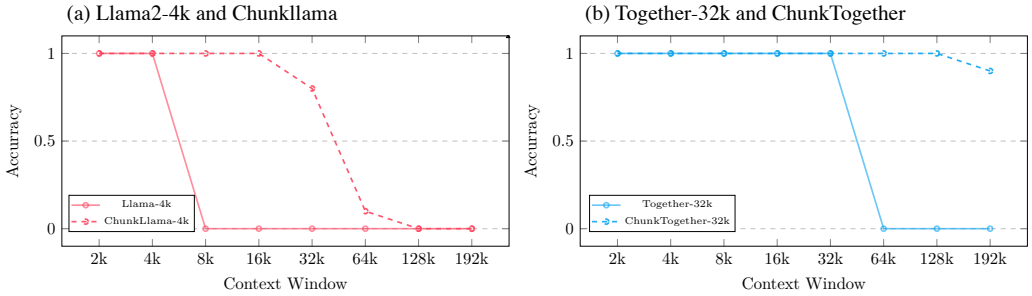

Figure 7. Passkey retrieval over a 192k context length for Llama2 13B, Together-32k 7B and their DCA enhanced versions.

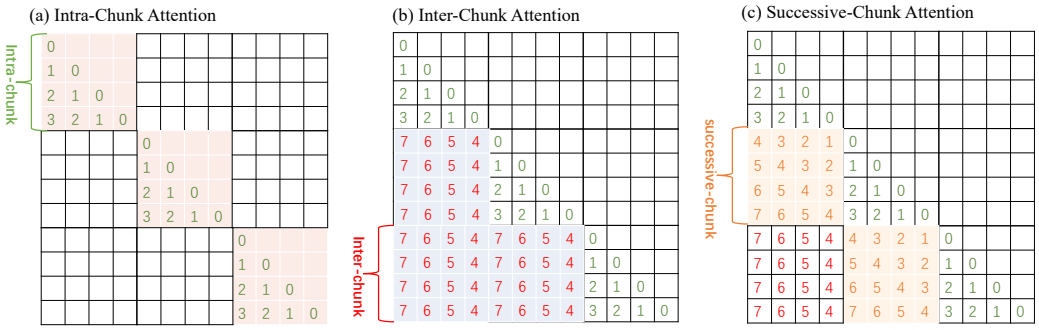

Figure 8. Visualization of the Relative Position Matrix M employing Dual Chunk Attention (DCA) by splitting the whole sequence into **3 chunks** and the chunk size $s = 4$. In this case, we have the pretraining window size $c = 8$, and local window size $w = 3$. The sequence is segmented into 3 chunks to ensure that relative positions do not exceed 7. The matrix element $M[i][j]$ represents the relative position between the i-th query vector $\mathbf{q}$ and the j-th key vector $\mathbf{k}$.

A.2. More Examples

In this section, we give an example of handling a 12-token sequence but the pre-training length is only 8. Based on Llama2, the key/query position indices will be initialized as:

$$P_{\mathbf{q}} = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]$$
$$P_{\mathbf{k}} = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11].$$

When $\mathbf{query}[11]$ and $\mathbf{query}[0]$ perform an inner product, their relative position is 11, which exceeds the pre-training size.

In DCA, we set a chunk size, which is a hyperparameter smaller than the pre-training length. In this example, we can set the chunk size to 4, which means we split the entire input into 3 chunks and give new position indices for keys:

$$[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] \Rightarrow [0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]$$

We then give different position indices for queries.

Intra-chunk Attention: Calculating the attention for tokens in the same chunk

$$P_{\mathbf{q}}^{\text{intra}} = [0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]$$
$$P_{\mathbf{k}} = [0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]$$

The resulting position matrix is shown in Appendix Figure 8 (a) and the maximum relative position is $3 - 0 = 3$.

Inter-chunk Attention: Calculating the attention for tokens in the different chunks When the pre-training length is 8, we have the maximum position index=7.

$$P_{\mathbf{q}}^{\text{inter}} = [7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7]$$
$$P_{\mathbf{k}} = [0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]$$

The resulting position matrix after inter-chunk attention is shown in Appendix Figure 8 (b).

Successive-chunk Attention: Calculating the attention for tokens in successive chunks. We change the first $w = 3$ (a hyperparameter) elements in $P_{\mathbf{q}}^{\text{inter}}$:

$$P_{\mathbf{q}}^{\text{succ}} = [4, 5, 6, 7, 4, 5, 6, 7, 4, 5, 6, 7]$$
$$P_{\mathbf{k}} = [0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3]$$

The position matrix after successive-chunk attention is shown in Appendix Figure 8 (c).

A.3. Flash Attention

We divide the standard self-attention into 3 separate flash attention calculations respectively obtaining the output from intra-chunk attention, inter-chunk-attention, and successive chunk-attention. Algorithm 1 showcases how the 3 attention introduced in DCA integrate with Flash Attention. We illustrate with the i-th query vector $\mathbf{q}_i$ and it needs to calculate the inner product with all keys $\mathbf{k}_j$ with $j \leq i$. We have $n = \lfloor i/s \rfloor$ chunks before the current chunks. DCA calls 3 separate Flash Attention operations with complexity $O(i - n * s)$(intra-chunk attention), $O(s)$ (succssive-chunk attention) and $O(s * (n - 1))$.

Algorithm 1 Pseudocode of DCA with FlashAttention

```
# q: 1 x d query vector (tensor with shape [1, d])
# i: the absolute index of q (integer)
# K, V: i x d matrices for keys and values (tensors with shape [i, d])
# s: chunk size (integer)
# P_k, P_q_intra, P_q_succ, P_q_inter: position ids (lists of integers)
n = math.floor(i/s) # Number of chunks before the current chunk
# Apply rotary position embeddings to the entire key matrix K
K = apply_rotary_pos_emb(K, P_k) # K is [i, d] after embedding

# ------------- Intra-chunk Attention, casual=True -------------
q_intra = apply_rotary_pos_emb(q, P_q_intra[i]) # q_intra is [1, d]
# Select intra-chunk keys and values
K_intra = K[s*n:i] # K_intra is [(i - s*n), d]
V_intra = V[s*n:i] # V_intra is [(i - s*n), d]
# Compute output and softmax attention map for intra-chunk attention
o_intra, map_intra = Flash(q_intra, K_intra, V_intra) # o_intra is [1, d], map_intra is [1, i - s*n]

# ------------- Successive-chunk Attention, casual=False -----------
q_succ = apply_rotary_pos_emb(q, P_q_succ[i]) # q_succ is [1, d]
# Select successive-chunk keys and values
K_succ = K[s*(n-1):s*n] # K_succ is [s, d]
V_succ = V[s*(n-1):s*n] # V_succ is [s, d]
# Compute output and softmax attention map for successive-chunk attention
o_succ, map_succ = Flash(q_succ, K_succ, V_succ) # o_succ is [1, d], map_succ is [1, s]

# ------------- Inter-chunk Attention, casual=False -----------
q_inter = apply_rotary_pos_emb(q, P_q_inter[i]) # q_inter is [1, d]
# Select inter-chunk keys and values
K_inter = K[:s*(n-1)] # K_inter is [s*(n-1), d]
V_inter = V[:s*(n-1)] # V_inter is [s*(n-1), d]
# Compute output and softmax attention map for inter-chunk attention
o_inter, map_inter = Flash(q_inter, K_inter, V_inter) # o_inter is [1, d], map_inter is [1, s*(n-1)]

# Normalization
# Sum the attention maps for each attention type to get normalizers
sum_intra = map_intra.sum(-1) # sum_intra is a scalar
sum_inter = map_inter.sum(-1) # sum_inter is  a scalar
sum_succ = map_succ.sum(-1) # sum_succ is a scalar
normalizer = sum_intra + sum_inter + sum_succ # normalizer is a scalar

# Concatenate attention outputs and divide by normalizer
output = (sum_intra*o_intra, sum_succ*o_succ, sum_inter*o_inter) / normalizer # output is [1, d]
```

A.4. In-Context Examples Selection

We opt to select in-context examples from the training set which is a practical and common way to obtain the examples (Ye et al., 2023; Wang et al., 2024). We experimented with 2 different methods for this selection process: (1)Random Selection: randomly selecting examples from the training set. (2) Retrieval-Based Selection: Using the current query, we employ

Table 5. Comparison of few-shot results using different in-context examples

Models	**In-Context Examples**	Qasper F1 (2-shot)	QuALITY EM (2-shot)	QMSum R-g (1-shot)
CHUNKLLAMA2 7B	EXAMPLE BEST	27.3	33.9	15.0
CHUNKLLAMA2 7B	EXAMPLE RANDOM	28.2	35.6	14.7
CHUNKLLAMA2 7B	EXAMPLE WORST	28.4	35.9	14.3
CHUNKLLAMA2 13B	EXAMPLE BEST	28.5	46.2	15.6
CHUNKLLAMA2 13B	EXAMPLE RANDOM	29.3	47.9	15.2
CHUNKLLAMA2 13B	EXAMPLE WORST	29.0	47.5	15.5

retrieval algorithms such as BM25 (Robertson et al., 2009) to find the most relevant examples from the training set. We refer to the in-context examples with the highest retrieval score as EXAMPLE BEST and those with the lowest as EXAMPLE WORST. The performance of different selection approaches based on CHUNKLLAMA2 7B/13B is shown in Table 5. The performance on the summarization dataset QMSum (Zhong et al., 2021) generally is less likely to be influenced by prompt selection. However, on the 2 question-answering datasets, we find that using the closest examples, paradoxically, leads to the poorest outcomes and the performance of both random selection and choosing the worst example is relatively similar. A possible explanation for this phenomenon is that when the example is highly similar, LLMs tend to copy the response given in the example which usually leads to a wrong answer.

A.5. Performance on Unseen Data

Currently, almost all benchmarks for LLMs fail to thoroughly address the potential of data contamination, which implies that the test data might have already been used in pretraining or finetuning phases. To demonstrate ChunkLlama's performance on previously unseen long-document data, we directly used the Latex code of **this paper** as a test case while omitting the title, abstract, and conclusion sections. After tokenization, the total input length is 19388. We initiate the evaluation with a set of straightforward questions that do not necessitate prior knowledge for accurate responses (see Table 6). This was followed by a series of more challenging questions designed to assess the comprehension of the proposed DCA (refer to Table 7).

The results indicate that, compared to the NTK, CHUNKLLAMA2 demonstrates a superior ability to correctly interpret instructions and provide accurate responses. However, the precision of the 13B model's answers remains suboptimal, even for relatively simple inquiries. For instance, when queried about the finetuning corpus used by DCA, it erroneously cites the Llama2 corpus rather than the correct corpora, which are ShareGPT and AlpacaGPT4.

On these elementary questions, ChunkLlama 70B exhibits a remarkably high accuracy rate. Moreover, ChunkLlama 70B shows promising outcomes on more challenging queries. It is able to articulate the rationale behind our design of the inter-chunk and successive-chunk strategies accurately. However, when we pose questions that require a global understanding of this method, it still faces difficulties.

Table 6. QA (Easy) results of CHUNKLLAMA2 13B/70B and Dynamic-NTK with Llama2 13B, using the content of this paper as input. Most of the questions do not require a deep understanding of this paper. The input prompt has 19388 tokens. Dynamic-NTK is unable to give the correct answer for all test cases. Low-quality generation results are marked with ✗.

Prompt	[INST] <<SYS>> Please read the long paper and answer the questions << /SYS>> 1. Introduction \n\n The ability to comprehend and process long-context information is essential for large language models (LLMs) to cater to a wide range of applications effectively. These include analyzing and responding to inquiries within sizable PDFs, retaining extended dialogue history, and empowering interactive chatbots... [/INST]
Question	Can you give a title for this paper
CHUNKLLAMA270B	Sure, here are some potential titles for the paper: 1. "Dual Chunk Attention: A Training-Free Framework for Long-Context Language Models" 2. "Efficient Long-Context Language Modeling with Dual Chunk Attention" 3. "Scaling Long-Context Language Models without Training: A Dual Chunk Attention Approach" 4. "DCA: A Training-Free Method for Long-Context Language Modeling" 5. "Long-Context Language Modeling with Dual Chunk Attention: A Training...
CHUNKLLAMA2 13B	Sure! Dual Chunk Attention: A Training-Free Framework for Long-Context Scaling of Large Language Models. This title highlights the key contributions of the paper: the proposed Dual Chunk Attention (DCA) framework, which enables long-context scaling without additional training, and its application to various tasks. The phrase "training-free" is also included to emphasize the main innovation of the paper.
Dynamic NTK 13B	The proposed Dual Chunk Attention (DCA) framework improves the long-context scaling of Lama2 by more than 8 times, achieving a PPL of 5.59 at a context window of 2k. ✗(ignore the instruction)
Question	Which model can be scaled up to 192k context with DCA?
CHUNKLLAMA270B	The Llama2 70B model can be scaled up to a 192k context window using DCA, achieving a PPL score of 5.59
CHUNKLLAMA2 13B	The Llama2 70B model can be scaled up to 192k context with DCA.
Dynamic NTK 13B	`{\end{enumerate}\end{figure}[ht!]` `\end{document}\end{document}\begin{table}`✗(error)
Question	In addition to Llama2, have the authors validated their method on other models?
CHUNKLLAMA270B	Yes, the authors have validated their method on other models, including Together-32k, CodeLlama, which are all based on Llama. They have also compared their method to a variety of open-source and proprietary models.
CHUNKLLAMA2 13B	The authors have validated their method on the following models: * Together-32k * CodeLlama * Llama2 Long ✗(We did not verify DCA on Llama2 Long)
Dynamic NTK 13B	The paper introduces a new training-free framework called Dual Chunk Attention (DCA) that extends the context window by more than 8 times for 7B/13B Llama2 variants and 20 times for 70B models✗(ignore the instruction)
Question	Describe the finetuning corpus used by DCA
CHUNKLLAMA270B	The finetuning corpus used by DCA is a combination of long dialogue data from ShareGPT and AlpacaGPT4
CHUNKLLAMA2 13B	The finetuning corpus used by DCA is the same as the one used by the original Llama2 model, which is a combination of the training set and the validation set of the original Llama2 model ✗(We did not use training set of Llama)
Dynamic NTK 13B	`{\end{begin}\begin{table}\end{figure}` `[ht!]} \end{document}\end{document}\end{document}` ✗(error)
Question	How many training steps does Llama2 Long need?
CHUNKLLAMA270B	Llama2 Long needs 100,000 training steps
CHUNKLLAMA2 13B	Llama2 Long needs 100,000 training steps.
Dynamic NTK 13B	Llama2 Long needs 40,000 steps for a 4k context window. ✗(not 40,000)
Question	Do the authors validate the model on L-Eval? and list the tasks
CHUNKLLAMA270B	Yes, the authors validate their model on L-Eval and list the tasks in Table 4. They evaluate their model on four closed-ended tasks from L-Eval with diverse input lengths ranging from 3k to 27k. The tasks include TOFEL, QuALITY, Coursera, and SFiction.
CHUNKLLAMA2 13B	Yes, the authors do validate the model on L-Eval and list the tasks. In the section titled "Experiments", the authors present an extensive evaluation of their model on a diverse set of tasks, including NarrativeQA, Qasper, Quality, and QMSum. ✗(These are not from L-Eval)
Dynamic NTK 13B	The paper presents a new training-free framework called Dual Chunk Attention (DCA) that extends the context window by more than 8 times for 7B/13B Llama2 variants and... ✗(ignore the instruction)

Table 7. QA (hard) results of CHUNKLLAMA2 13B/70B and Dynamic-NTK with Llama2 13B, using the content of this paper as input. Questions in this table typically require readers to understand this method. The input prompt has 19388 tokens. Dynamic-NTK is unable to give the correct answer for all test cases. Low-quality generation results are marked with ✗.

Prompt	[INST] <<SYS>> Please read the long paper and answer the questions << /SYS>> 1. Introduction \n\n The ability to comprehend and process long-context information is essential for large language models (LLMs) to cater to a wide range of applications effectively. These include analyzing and responding to inquiries within sizable PDFs, retaining extended dialogue history, and empowering interactive chatbots... [/INST]
Question	Why do the authors introduce successive-chunk attention? (Hard)
CHUNKLLAMA270B	The authors introduce successive-chunk attention to address the issue of **losing locality** in the attention mechanism when using inter-chunk attention. Inter-chunk attention can lead to a situation where the relative position between two tokens in different chunks is not accurately captured, resulting in a high perplexity. Successive-chunk attention helps to maintain locality by adjusting the position indices of the first w tokens in each chunk to ensure that the relative position between neighboring tokens is accurately captured. This allows the model to better handle long-context tasks that require the ability to capture long-range dependencies.
CHUNKLLAMA2 13B	The authors introduce successive-chunk attention to address the issue of long-range dependencies in long-context language models. The original positional encoding methods, such as RoPE and NTK-aware RoPE, do not capture long-range dependencies effectively, leading to poor performance in language modeling and passkey retrieval tasks. By introducing successive-chunk attention, the model can capture both local and long-range dependencies.. ✗(misunderstand successive-chunk attention)
Dynamic NTK 13B	The paper introduces a new training-free framework called Dual Chunk Attention (DCA) that extends the context window by more than 8 times for 7B/13B Llama2 variants...✗(ignore the instruction)
Question	Why do the authors introduce inter-chunk attention? (Hard)
CHUNKLLAMA270B	The authors introduce inter-chunk attention to address the issue of poor **long-range dependencies** in the intra-chunk attention module, which uses a fixed context window for attention computation. The inter-chunk attention mechanism allows the model to attend to tokens from different chunks, enabling it to capture long-range dependencies and improve performance on tasks that require processing long sequences.
CHUNKLLAMA2 13B	13b: The author introduces inter-chunk attention to improve the passkey retrieval performance of the model. The author notes that using only intra-chunk attention, which disregards information from previous chunks, can maintain a low PPL but hinder the model's ability to retrieve passkeys from other chunks. ✗(not accurate enough)
Dynamic NTK 13B	The paper presents a new training-free framework called Dual Chunk Attention (DCA) that extends the context window by more than 8 times for 7B/13B Llama2 variants...✗(ignore the instruction)
Question	Why the relative position in DCA will not exceed the pretraining context length? (Hard)
CHUNKLLAMA270B	The relative position in DCA is calculated using the position indices of the keys and queries, which are both less than or equal to the pretraining context length. Therefore, the relative position will not exceed the pretraining context length. (not clear enough)
CHUNKLLAMA2 13B	The relative position in DCA will not exceed the pretraining context length because the model only considers the relative positions within the same chunk. The chunks are segmented based on the pretraining context length, and the relative positions are calculated within each chunk. (not clear enough)
Dynamic NTK 13B	The relative position in Dual Chunk Attention (DCA) does not exceed the pretraining context length because it is designed to maintain locality. (not clear enough)
Question	What are the key differences between this paper and earlier papers? (Hard)
CHUNKLLAMA270B	The key differences between this paper and earlier papers on large language models (LLMs) are: 1. The paper proposes a new method called Dual Chunk Attention (DCA) that enables LLMs to process long-context information without additional training costs. 2. DCA is designed to handle long-context information by segmenting the input sequence into smaller chunks and computing attention within each chunk, which reduces the computational cost and allows for efficient scaling of the context window...
CHUNKLLAMA2 13B	This paper proposes a new training-free framework called Dual Chunk Attention (DCA) that enables large language models (LLMs) to process long-context information without additional training. DCA introduces a new attention mechanism that scales the context window of LLMs without additional training, allowing for more efficient use of computational resources.
Dynamic NTK 13B	DCA is orthogonal to popular extrapolation methods such as PI and NTK-Aware RoPE, and can be integrated with existing long-context models to scale up significantly...