Advancing Transformer Architecture in Long-Context Large Language Models: A Comprehensive Survey

Yunpeng Huang[1], **Jingwei Xu**[1], **Zixu Jiang**[1], **Junyu Lai**[1], **Zenan Li**[1], **Yuan Yao**[1], **Taolue Chen**[2], **Lijuan Yang**[3], **Zhou Xin**[3], and **Xiaoxing Ma**[1]

[1]State Key Lab of Novel Software Technology, Nanjing University, China
[2]School of Computing and Mathematical Sciences, Birkbeck, University of London, UK
[3]Baidu.inc, China
{hyp, junyu_lai, lizn}@smail.nju.edu.cn,{jingweix,y.yao,xxm}@nju.edu.cn,
jzxthlw@gmail.com,t.chen@bbk.ac.uk, {yanglijuan04, xinzhou}@baidu.com

Abstract

With the bomb ignited by ChatGPT, Transformer-based Large Language Models (LLMs) have paved a revolutionary path toward Artificial General Intelligence (AGI) and have been applied in diverse areas as knowledge bases, human interfaces, and dynamic agents. However, a prevailing limitation exists: many current LLMs, constrained by resources, are primarily pre-trained on shorter texts, rendering them less effective for longer-context prompts, commonly encountered in real-world settings. In this paper, we present a comprehensive survey focusing on the advancement of model architecture in Transformer-based LLMs to optimize long-context capabilities across all stages from pre-training to inference. We firstly delineate and analyze the problems of handling long-context input and output with the current Transformer-based models. Then, we mainly offer a holistic taxonomy to navigate the landscape of Transformer upgrades on architecture to solve these problems. Afterward, we provide the investigation on wildly used evaluation necessities tailored for long-context LLMs, including datasets, metrics, and baseline models, as well as some amazing optimization toolkits like libraries, systems, and compilers to augment LLMs' efficiency and efficacy across different stages. Finally, we further discuss the predominant challenges and potential avenues for future research in this domain. Additionally, we have established a repository where we curate relevant literature with real-time updates at https://github.com/Strivin0311/long-llms-learning.

1 Introduction

In recent years, leveraging techniques from deep learning [93], especially the surge of Transformer-based models like BERT [45], GPT [134, 135, 17] and their variants [97, 105, 137], Natural Language Processing (NLP) has significantly advanced, empowering machines to understand and generate human language [170, 98], thus revolutionizing numerous tasks in Natural Language Understanding (NLU) like sentiment analysis [206], Natural Language Generation (NLG) like document summarization [51], as well as other domains like computer vision [81] and autonomous driving [67]. Furthermore, in the wake of ChatGPT [121], PaLM [36], GPT4 [123, 122], etc, the Transformer-based Large Language Models (LLMs) which scale up to 1B∼100B parameters to empower emergence abilities [183], have shown a new exhilarating path towards Artificial General Intelligence (AGI) [18], and been rapidly adopted in a myriad of human-interactive applications, like chatbots [146, 95], programming assistants [184, 196] and educational tutors [1, 117].

Transformer is an elaborate deep neural network model, which incorporates many great preceding designs [8, 65, 7] and comprises diverse novel components to solve the sequence-to-sequence language modeling problem in machine translation at the very beginning [175]. The contemporary LLMs mostly derive their foundation from the Transformer architecture by employing its full or part modules [45, 134, 137]. Among those components, Transformer-based LLMs own the success mainly due to the core well-designed attention mechanism that captures global dependencies of each pair of tokens across the whole input, enabling the model to handle sequences with intricate relations. While the attention mechanism offers remarkable performance, its quadratic time and space complexities with respect to input sequence length lead to a significant computational resource bottleneck, which imposes limitations on not only the permissible input text length during training, but also the effective context window of prompts due to unsatisfactory efficiency and expensive caching memory consumption as the generated

tokens increase during inference. To be worse for inference, the LLMs also suffer from performance degeneration when facing sequences longer than the ones in training, owning to poor generalizable mechanism design for input length.

However, with LLMs deeply ingrained in various applications that require long-context comprehension [193, 87] and generation [106, 68], the demand for long-context LLMs capable of comprehending and generating extremely long sequences effectively and efficiently becomes increasingly indispensable and urgent. Consequently, researchers have devoted significant efforts to enhancing the Transformer architecture to address the long-context problem in LLMs, including optimization on the efficiency of attention (Sec.3), context window extension with extra memory mechanisms (Sec.4), effective length generalization with extrapolative positional embeddings (Sec.5), context pre/post-processing (Sec.6), and other miscellaneous methods (Sec.7) such as specific pre-training objectives, mixture of experts, quantization, parallelism, etc.

Existing surveys. The field of long-context LLMs has become one of the hottest and most rapidly developing research areas on LLMs recently, with some existing surveys [86, 168, 102, 212, 49] summarizing the related work of literature. Among those, [86] offers a cursory overview of long document summarization yet refrains from delving deeply into the intrinsic techniques of long text modeling. In [168] and [102], they both primarily concentrate only on augmenting the computational efficiency of Transformers in long-text scenarios. Although [212] underscores the challenges LLMs face when engaging with extensive sequences, its discussed methods predominantly align with efficient Transformers, akin to [168] and [102]. A more recent contribution [49] bears the closest resemblance to our study, introducing approaches in long-text modeling and applications with Transformers, covering pre-processing techniques, part of efficient Transformers, and special characteristics of lengthy documents. However, there is still a lack of comprehensive studies to review the literature on the advancement in breaking the barriers of context length across all stages for more intricate and scalable Transformer-based LLMs by exploring the *Transformer's architecture from an operational perspective*.

The **objective of this survey** is to thoroughly review the panorama of literature on architecture evolution for scaling the effective context window length of present Transformer-based LLMs from a methodological perspective. The key contributions are as follows:

- We build a holistic taxonomy to categorize *five* pieces by breaking down the Transformer architecture and then delving into the existing methodologies in enhancing long-context LLMs during each stage, including pre-training, fine-tuning, inference, and pre/post-processing.
- We explore the widely-used evaluation necessities, comprising datasets, metrics, and baseline specifically assessing the long-context capabilities of LLMs, followed by some popular toolkits to optimize LLMs' efficiency and effectiveness for both training and inference, such as libraries, systems, and compilers.
- We identify key challenges to revamping the Transformer structure for handling extensive contexts, with corresponding future directions to further push the frontier.
- Considering the extremely rapid growth of this field and the survey might fall behind soon, we build a repository that gathers relevant literature within this specific domain. We shall keep it updated continuously to help readers keep pace with the newest advancements.

Organization of this survey. Sec.2 gives an overview of long-context LLMs, including the preliminaries about objectives and stages for language modeling and critical components of Transformer-based LLMs, the structure limitation analyses for LLMs to deal with lengthy contexts and the taxonomy of existing efforts on advancing Transformer architecture. Then, we mainly delve into the discussion of each part of methodologies from the taxonomy in next five sections 3, 4, 5, 6,7, corresponding to related modules in Transformer architecture. In Sec.8, we also summarize the necessities for long-context capabilities evaluation and collect some popular optimization toolkits to augment LLMs about effectiveness and efficiency during training and inference. Then in Sec.9, we explore the critical challenges and corresponding potential avenues lighted up by them, as well as draw insights from existing breakthroughs. Finally, Sec.10 closes this survey with overarching conclusions regarding the the panorama of this domain as well the motivation of this study.

2 Overview

In this section, we start with the preliminaries (Sec.2.1) from the fundamental language modeling objectives, typical modeling stages to critical architecture modules found in Transformer-based decoder-only LLMs, as depicted in Fig.1 ***(a)***. Subsequently, we offer brief analyses of the architecture limitations when LLMs encounter extensive

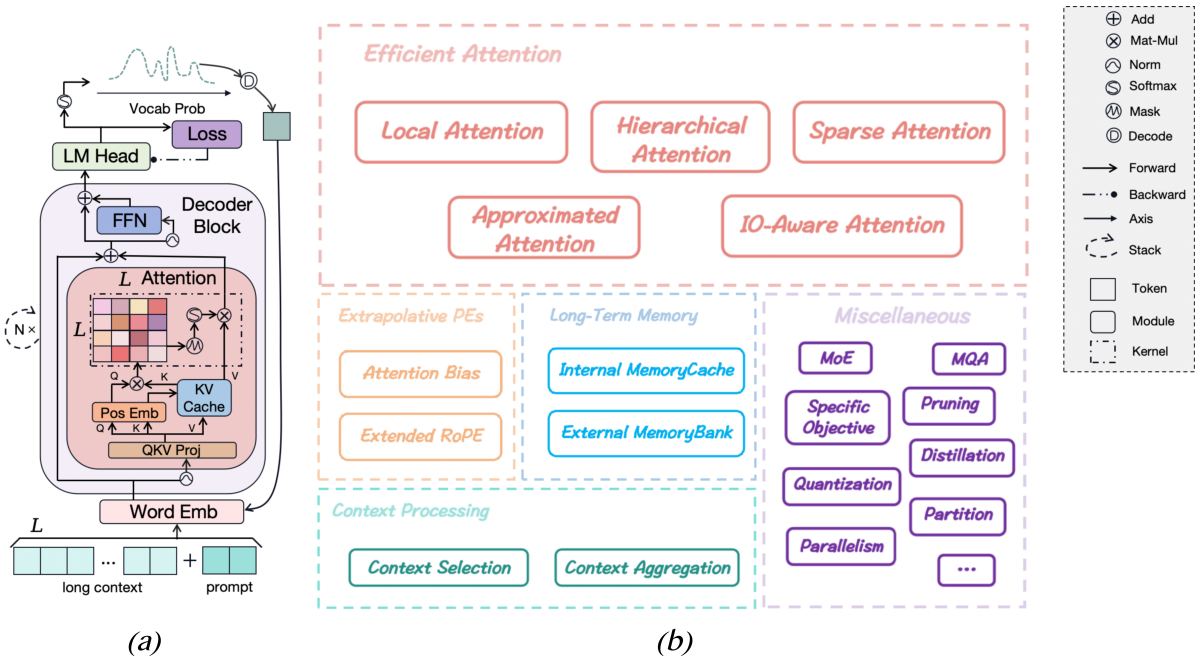

Figure 1: The overview of the core of this survey: ***(a)*** The typical architecture anatomy diagram of contemporary Transformer-based decoder-only LLMs, with the legend on the far top right; ***(b)*** The taxonomy of methodologies for enhancing Transformer architecture modules (corresponding to ***(a)*** by color): Efficient Attention (submodule of attention kernel), Long-Term Memory (targeting KV cache), Extrapolative PEs (against the positional embedding module), Context Processing (related to context pre/post-processing), and Miscellaneous (general for the whole Decoder Block as well as the Loss module).

context windows (Sec.2.2). Finally, we present a comprehensive methodological taxonomy (Sec.2.3) aimed at enhancing the long-context capabilities of LLMs through architectural innovations (see Fig.1 ***(b)***). This taxonomy serves as a guideline for the next five sections 3, 4, 5, 6,7.

2.1 Preliminaries

Language Modeling. The heart of LLMs lies the foundational task of language modeling, which aims to empower neural networks to comprehend and generate human language. From a mathematical perspective, the essence of neural language modeling is to approximate the precise log-probability of the occurrence of any given text, denoted as $\log P(X_{1\sim L};\theta)$, where θ stands for the network parameters to learn, and $X_{1\sim L} := \{x_1, .., x_L\}$ comprises a sequence of L elements representing natural language including words, punctuation, mathematical symbols and more. However, this task encounters a significant practical hurdle known as the *curse of dimensionality*, which stems from the exponential growth of possibilities as L increases. To avoid the impracticality, LLMs employ variations mainly including *masked language modeling* (MLM) and *causal language modeling* (CLM). The former MLM is to predict masked tokens based on the bidirectional remaining unmasked tokens, whose objective can be written as Eq.1, that maximizes the conditional probability of the i-th token x_i given all others, where $\mathcal{M}$ denotes the index set of the masked tokens. In contrast, the objective of CLM is to predict the next token, i.e., maximize the conditional probability of each token, given the unidirectional preceding ones (see Eq.2). In this setup, casual LLMs can effectively leverage the temporal dependencies inherent in natural language sequences, enabling LLMs to generate coherent and contextually relevant text.

$$\text{MLM}: \arg\max_{\theta} \sum_{i\in\mathcal{M}} \log P(x_i|X_{1\sim i\text{-}1,i+1\sim L};\theta) \tag{1}$$

$$\text{CLM}: \arg\max_{\theta} \sum_{i=1}^{L} \log P(x_i|X_{1\sim i\text{-}1};\theta) \tag{2}$$

Modeling Stages. Nowadays, the *operation* of a LLM often undergoes a multi-stage modeling process. Initially,

during the pre-processing stage, raw text data is segmented and tokenized into individual (sub)words called *tokens* predefined in a vocabulary, using algorithms like BPE [148]. Then, in the pre-training stage, the model is trained on vast text corpora, with the object of either MLM or CLM, to capture semantic patterns and linguistic structures of natural language. Once pretrained, the model proceeds to the fine-tuning stage, where it is further trained with a few epochs on task-specific data with extra heads to learn sometimes. Finally, the fine-tuned model is deployed into downstream scenarios to predict expected answers in an inference mode. Particularly, the casual LLMs are pre-trained and fine-tuned with the same CLM objective but a different corpus. During the inference step, the model predicts from the probability distribution of the vocabulary by some decoding strategy such as *greedy search*, *beam search*, *nucleus sampling* [177], to generate contextually coherent responses to prompts in a token-by-token *autoregressive* paradigm.

Decoder Block. The vanilla Transformer architecture proposed in [175] mainly comprises an Encoder and a Decoder, each stacked with multiple identical blocks. The skeleton of each block is mostly compatible with the one portrayed in Fig.1 ***(a)***. In general, the first block will take the tokenized sequence encoded by a word embedding layer, following a *multi-head scaled-dot self-attention* (MHA) layer with an attention mask corresponding to specific language modeling objectives and a feed-forward network (FFN) layer. Both the MHA and FFN layers are enriched with layer normalization [7] and residual connections [65] at every entrance/exit of the block. Then, each higher-level block takes the output hidden states from the previous block as input, represents them with its MHA and FFN layers, and feeds them to the next block. The final output hidden state from the last block is fed into a linear layer called language modeling head, and the output *logits* will be transformed into a probability distribution over the target vocabulary through *softmax* operation. Notably, the slight difference between the Encoder block and Decoder block in an Encoder-Decoder Transformer is that the latter additionally interfaces with the Encoder's output via an *cross-attention* (CA) layer before feeding into the FFN layer.

However, such a binary structure was originally designed for sequence-to-sequence modeling in machine translation tasks. Subsequently, it has given rise to several variations aimed at more general language modeling objectives like MLM and CLM. The BERT series [45, 105] harnesses only the Encoder with MLM to enhance bidirectional information, serving as a discriminative model. Conversely, the GPT series [134, 17, 135] utilizes only the Decoder with CLM, focusing on unidirectional generative models. T5 [137] and BART [97] variants, however, treat each NLP task as a text-to-text conversion, leveraging both Encoder and Decoder. The decoder-only generative model architecture has recently become the predominant choice for current LLMs. Notable examples include GPT4 [123], PaLM [36, 6], LLaMA [173, 174], and GLM [50, 203], among others.

Attention Mechanism. The attention mechanism [8], as the core design of the Transformer implemented in the MHA layer, computes a weighted representation of each token in the input sequence based on its relevance to any other ones. More specifically, as illustrated in Fig.1 ***(a)***, the word-embedded token sequence $X \in \mathbb{R}^{L\times d_{in}}$, concatenating long contexts and user prompts with total length L, will derive three embedding matrices with a linear projection layer (see Eq.3): *query* $Q \in \mathbb{R}^{L\times d_q}$, *key* $K \in \mathbb{R}^{L\times d_k}$ and *value* $V \in \mathbb{R}^{L\times d_v}$. Then, as for the attention kernel operations in Eq.4, first the unnormalized relevance matrix $P \in \mathbb{R}^{L\times L}$ are calculated by matrix multiplication of Q, K^{T}, where each entry serves as relevance for corresponding pair of tokens. Then, the normalized attention score matrix $A \in \mathbb{R}^{L\times L}$ is computed as: a scaling operation by factor $\sqrt{d_k}$, an element-wise mask operation with $M \in \mathbb{R}^{L\times L}$, and a row-wise *softmax*. Finally, the output hidden states $O \in \mathbb{R}^{L\times d_o}$ are generated by a weighted sum of V with attention weights in each row of A, usually with an extra linear transformation.

Note that the embedding dimensions of Q, K, V, O can be the same or not, and even though subscripts are used here to distinguish them for generality, we default set $d = d_q = d_k = d_v = d_o$ in the rest of the paper. As for the mask matrix M, it is typically used for masking padding tokens to align all batched input sequences and also applies casual mask operation of causal language modeling for generative LLMs. Furthermore, to capture diverse relationships, the model often employs multi-head attention instead of single-head one, performing the attention process in parallel with differently weighted Q_h, K_h, V_h sets by dividing learnable parameters like $W_{q,k,v} \in \mathbb{R}^{d_{in}\times(3\times d)}$ into $W^{mh}_{q,k,v} \in \mathbb{R}^{d_{in}\times(3\times H\times d_{head})}$, where H denotes the number of heads. Similar to embedding dimensions, the number of heads can be specific for Q, K, V, which vary in different LLMs, yet we consider them the same by default.

$$\text{QKV Projection}: \; Q, K, V := \text{split}\left(X \times W_{q,k,v}\right), \quad W_{q,k,v} \in \mathbb{R}^{d_{in}\times(d_q+d_k+d_v)} \tag{3}$$

$$\text{Attention Kernel}: \; P := Q \times K^{\mathrm{T}}, \;\; A := \text{softmax}[\frac{P}{\sqrt{d_k}} \odot M], \;\; O := (A \times V) \times W_o, \quad W_o \in \mathbb{R}^{d_v\times d_o} \tag{4}$$

Positional Embeddings. PEs are minute but essential in the Transformer, especially for NLP tasks. Unlike recurrent neural networks(RNNs) [198], Transformers process input tokens in parallel as a bag-of-words and lack an inherent sense of sequence order. To preserve the sequential information, the vanilla Transformer presents a novel

Sinusoidal PE (SinPE) [175]. As depicted in Eq.5, every entry in each dimension of SinPEs is derived based on trigonometric functions like sin, cos, and the periods vary exponentially with respect to absolute token positions, where *base* is a large integer manually set to 10000 according to the original paper without further explanation, and d is the unit embedding dimension of hidden states. Some variants have recently emerged, including trainable embeddings [26] to learn an embedding mapping and relative embeddings [149] based on relative positions. Among them, Rotary PE (RoPE) [161] applies a rotation operation on a complex field (see Eq.6) instead of an addition to Q, K based on absolute positions, where it shares the same basis function as SinPE. According to the property described in Eq.7, not only RoPEs ensure the magnitude of $\mathbf{q}, \mathbf{k}$ remains unchanged due to unitary transformation, but also every entry in P, i.e., each pair of $\mathbf{q}, \mathbf{k}$, will only be tagged with embeddings in terms of their relative distance in the sequence. RoPE provides a more stable scheme to handle longer sequences. It captures relative positional patterns with absolute position awareness, thus widely used in state-of-the-art open-source LLMs like LLaMA [173, 174] and GLM [50, 203].

It is worth noting that SinPEs are initially applied on the word embeddings before entering the Encoder or Decoder blocks by addition. In contrast, as shown in Fig.1 ***(a)***, RoPEs are applied to Q, K in each attention layer before the kernel operations by equivalent element-wise vector multiplication to save registered buffer memory.

$$\text{SinPE}(n) := \begin{bmatrix} sin(n\theta^0) \\ cos(n\theta^0) \\ sin(n\theta^1) \\ cos(n\theta^1) \\ \cdots \\ sin(n\theta^{\frac{d}{2}-1}) \\ cos(n\theta^{\frac{d}{2}-1}) \end{bmatrix}, \quad where \;\; \theta = base^{-\frac{2}{d}}, \; n \in \{0, 1, 2, .., L-1\} \tag{5}$$

$$\text{RoPE}(n) := \begin{bmatrix} R_n^{(0)} & & & \\ & R_n^{(1)} & & \\ & & \cdots & \\ & & & R_n^{(\frac{d}{2}-1)} \end{bmatrix}, \quad where \;\; R_n^{(i)} := \begin{bmatrix} cos(n\theta^i) & -sin(n\theta^i) \\ sin(n\theta^i) & cos(n\theta^i) \end{bmatrix} \tag{6}$$

$$\text{RoPE } properties: \; ||R_i\mathbf{q}|| = ||\mathbf{q}||, \;\; P_{i,j} := \langle R_i\mathbf{q}, R_j\mathbf{k} \rangle = \mathbf{q}^{\mathrm{T}} R_i^{\mathrm{T}} R_j \mathbf{k} = \mathbf{q}^{\mathrm{T}} R_{j\text{-}i} \mathbf{k} \tag{7}$$

Key-Value Cache. In a narrow sense, the Key-Value (KV) cache is a list of tensors that stores the $\mathbf{k}, \mathbf{v}$ embeddings for all previous tokens in the attention layer for each block, utilized and updated during the autoregressive generation process of causal LLMs. As shown in Fig.1 ***(a)***, before the first token is generated, all KV caches are initialized empty and will be filled with L (key, value) pairs after the heave attention computation with L queries and L keys. Then, the first generated token will also be considered as input, extending the whole sequence to $L+1$ tokens. To avoid redundant calculations, the real input will contain only the latest generated token, deriving one new triplet of (query, key, value). But to compute equivalently, the new query has to attend and apply to all $L+1$ previous keys and values, thus the new (key, value) have to concatenate with past L pairs stored in the KV cache, and update themselves into it for the next generated token to attend. However, in a broad sense, we can consider the KV cache as the memory storage of LLMs, whose occupation grows linearly as the generated tokens increase. That directly causes one of the limitations below about the lack of efficient memory and suggests the approaches to enhance the *long-term memory* mechanisms for LLMs in Sec.4.

2.2 Limitation Analyses

Attention Complexity. In typical scenarios where $L \gg d$, the computational complexity of MHA can be concisely summarized as follows: It involves $O(L^2 d)$ time complexity, comprising $O(Ld^2)$ for QKV projection, $O(L^2 d)$ for the computation of P, $O(L^2)$ for the *softmax* operation to obtain A, $O(L^2 d)$ for the multiplication of A and V, and $O(Ld^2)$ for the output projection of O. And it incurs $O(L^2)$ space complexity, involving $O(Ld)$ for embeddings of Q, K, V, O, and additional $O(L^2)$ buffers for storing weights P and A. Consequently, both temporal and spatial computational costs exhibit a quadratic increase with the expansion of the sequence length, which can be burdensome for both training and inference.

In-context Memory. LLMs lack an explicit memory mechanism, relying solely on the KV cache to store representations of all previous tokens in a list. This design implies that once querying is completed in one call, the Transformer does not retain or recall any previous states or sequences in subsequent calls unless the entire history is reloaded token by token into the KV cache. Consequently, the Transformer possesses only an in-context working

memory during each call, as opposed to an inherent memory mechanism such as Long Short-Term Memory (LSTM) [198]. This statelessness offers computational advantages in terms of parallelism but presents challenges in tasks like chatbot applications [83], where long-term memory retention is essential.

Max-Length Constraint. During the training phase, engineers typically need to determine a crucial hyper-parameter *max-length*, denoted as L_{max} throughout this paper. This hyperparameter represents the upper bound on sequence length for any training sample in a batch. It is commonly set to values such as *1k*, *2k*, or *4k* based on the available computational resources to avoid Out-of-Memory (OOM) errors on GPUs. However, during inference, LLMs service providers must also either restrict the length of user prompts or automatically truncate them to align with the predefined L_{max}, even though inference resources are typically more abundant than during training. Note that none of the Transformer modules inherently require such restrictions since all learned weights depend solely on dimension sizes. So, theoretically, Transformers can process sequences of any length as long as the resources are sufficient. Unfortunately, current Language Models have shown noticeable performance degradation when handling input sequences exceeding L_{max}, often resulting in repetitive and implausible outputs.

2.3 Taxonomy

Building upon the foundational insights presented in Sec.2.1 and the limitations discussed in Sec.2.2, there are multiple avenues to explore for advancing the Transformer structure to endow LLMs with long-context capabilities, such as reducing attention complexity during training, designing efficient memory mechanisms, and enhancing the ability for *length extrapolation*, as outlined in [129], where the model is trained on short sequences but tested on longer ones during inference. Consequently, in this paper, we provide a comprehensive review of recent advancements in methodologies aimed at improving the long-context capabilities of LLMs throughout various stages, and we organize them into a unified taxonomy, as illustrated in Fig.1 ***(b)***. Specifically, these methods are categorized into five main clusters as follows:

- *Efficient Attention* (Sec.3): these methods focus on implementing efficient attention mechanisms with reduced computational demands, even achieving linear complexity. By doing so, they enable the extension of the effective context length boundary of LLMs during inference by directly increasing L_{max} in the pre-training stage.
- *Long-Term Memory* (Sec.4): to address the limitations of the in-context working memory, some approaches aim to design explicit memory mechanisms that compensate for the lack of efficient and effective long-term memory in LLMs.
- *Extrapolative PEs* (Sec.5): recent efforts have been made to enhance the length generalization capability of LLMs by improving the extrapolative properties of existing positional encoding schemes.
- *Context Processing* (Sec.6): in addition to methods that enhance specific low-level Transformer modules, some approaches involve wrapping off-the-shelf LLMs with additional context pre/post-processing. These methods ensure that the input fed to LLMs in each call always meets the maximum length requirement and breaks the context window limit by introducing multiple calling overheads.
- *Miscellaneous* (Sec.7): this section explores various general and valuable methods that do not neatly fit into the previous four categories, offering a broader perspective on advancing long-context capabilities in LLMs.

3 Efficient Attention

The first category of methods is dedicated to optimizing attention mechanisms, especially focusing on the kernel operations that make the module the computational bottleneck of the Transformer (see Eq.4). This approach enables the expansion of the effective context length boundary for LLMs during inference by directly increasing the hyperparameter L_{max} in the pre-training stage. We further categorize these methods into five distinct strategies, each with a specific focus: *Local Attention* (Sec.3.1), *Hierarchical Attention* (Sec.3.2), *Sparse Attention* (Sec.3.3), *Approximated Attention* (Sec.3.4), and *IO-Aware Attention* (Sec.3.5).

3.1 Local Attention

The traditional attention mechanism is characterized by its global and full attention nature, wherein every token is expected to attend to every other token, resulting in quadratic time and space complexities. Considering the

significance of local context in certain applications [191], various approaches have been introduced to implement local attention mechanisms in recent years. These mechanisms restrict each token's attention to its neighboring tokens instead of all tokens, and the variations among these approaches arise from the heuristic criteria to determine who qualifies as a token's neighbor, as depicted in Fig.2.

Block-wise Attention. One straightforward approach to implementing local attention involves segmenting the input sequence into non-overlapping blocks. As proposed in BlockBERT [132], within each block of fixed size B, tokens are only allowed to attend to other tokens within the same block. This block-wise attention involves performing full attention calculations within each $B \times B$ block for $\frac{L}{B}$ iterations, resulting in a time complexity of $O(LBd)$ and a memory complexity of $O(LB)$. However, this approach restricts the global receptive field and can limit the ability to model long-term dependencies. To address this limitation, Bi-BloSAN [152] introduces an inter-block attention mechanism to capture long-range dependencies. Sinkhorn [167] employs a differentiable ranking network to sort blocks, allowing each token to attend to tokens in the newly sorted block, thus enabling a quasi-global receptive field. SPADE [213] augments state space models (SSMs) [58] to address long-range dependency limitations. Additionally, Landmark Attention [119] introduces a new token called the *landmark token* for each block, enabling block-wise representations by training the attention mechanism to use it to select relevant blocks. In the fine-tuning stage, LongLoRA [29] introduces shift short attention (S^2-Attn) on top of LoRA [66], shifting tokens by half the block size in half of the attention heads to ensure information flow between neighboring blocks.

Sliding Window Attention. Inspired by convolutional neural networks (CNNs) [94, 89], another approach is to use sliding-window techniques, as demonstrated in Longformer [11]. In this method, each token is assigned a consecutive fixed window and is allowed to attend only to the previous adjacent $w \ll L$ tokens as its neighbors. To extend the receptive field similar to dilated convolution [197], the window is dilated with gaps of size dilation d, enabling each token to attend to tokens as far as $w \times d + 1$ away. To aggregate global information without additional computation, global attention is also applied to a few pre-selected positions where special tokens like `[CLS]` are located, decreasing computation complexity to $O(Lwd)$.

Global-Local Hybrid Attention. A similar global-local attention mechanism has also been adopted in ETC [4] and LongT5 [59], which explicitly or implicitly construct auxiliary global tokens to represent the segment information with global attention while only applying local attention to tokens in the source. We can also consider it as a hierarchical organization of attention receptive fields related to Sec.3.2. Another intriguing technique of the global token comes from the very recent streamLLM [188], where they observe an interesting phenomenon, namely *attention sink*, that not only keeping the KV of initial tokens during inference will largely recover the performance of sliding window attention, but adding a placeholder token during pre-training can further improve streaming deployment as well. They demonstrate that the emergence of attention sink is due to the strong attention scores towards initial tokens as a "sink" even if they are not semantically important. Similar strategies to keep attending to the starting tokens have also been proposed in the very recent Lm-infinite [62], where they propose a Λ-shaped mask and a positional distance constraint.

LSH Attention. In contrast to direct positional adjacency, Reformer [84] utilizes a neighbor token selection mechanism based on k-Nearest-Neighbor (kNN) and Locality-Sensitive Hashing (LSH) algorithms [75]. LSH attention allows each query $\mathbf{q}_i$ to attend to a set of keys $S_i := \{\mathbf{k}_{j \le i} : h(\mathbf{q}_i) = h(\mathbf{k}_j)\}$ within a single hash bucket. The hashing function h is designed to assign the same hash with high probability to two vectors that are similar and vice versa. This approach ensures that each token can access a fixed number $K \ll L$ of neighboring keys, and the primary computational cost of LSH attention arises from bucket sorting, with a complexity of $O(L \log Ld)$.

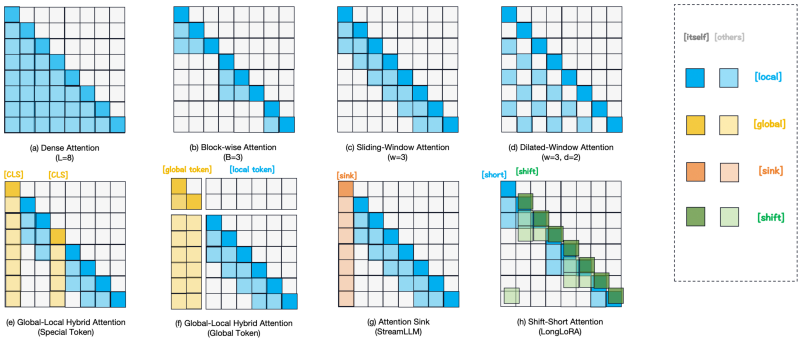

Figure 2: The visualization of various typical local causal attention mechanisms. As the legend on the right indicates, tokens are distinguished by colors, with shades denoting attention to itself (darker) or attention to the preceding others (lighter).

3.2 Hierarchical Attention

To further think of either the global token techniques [11, 4, 59], or the inter-block attention [152] mentioned above, we can regard them as introducing some hierarchical features to self-attention to compensate with more global information from the higher-level attention while keeping the low computation cost from the low-level local attention at the same time. From this view, many works have explored various hierarchical mechanisms that introduce a structured hierarchy into self-attention, leveraging both higher-level global information and lower-level local attention for multi-scaled contextual receptive fields.

Two-Level Hierarchy. HAN [194] pioneers the use of a two-level attention mechanism. It first applies self-attention to word features to obtain a sentence representation, and then employs self-attention on sentence-level features to generate document-level features. This hierarchical approach improves efficiency and performance in document classification tasks. Subsequently, similar hierarchical attention mechanisms have led to significant advancements in various document-level tasks, including machine translation [113, 143, 187] and document summarization [207, 204, 38].

Multi-Level Hierarchy. In contrast to the typical binary level structure above, BPT [195] introduces a more elaborated fine-to-coarse attention mechanism that operates on multi-scale spans through binary partitioning. Token nodes can attend to smaller-scale spans for closer context and larger-scale spans for more distant context. This approach formalizes the hierarchical structure as a graph neural network and updates it using graph self-attention [176]. A simpler variation is seen in Adaptive Span Transformer [162], which employs a soft attention masking function to non-increasingly map relative distances to real values in the range $[0, 1]$. This function controls the span of attention for each head, allowing the model to attend to different context spans.

Building upon the hypothesis, the attention matrix in many NLP tasks holds a hierarchical low-rank structure, H-Transformer-1D [211] introduces hierarchical attention that partitions the attention matrix into different blocks with varying low-rank ranges, enabling different levels of approximation. This approach reduces the overall runtime and memory cost complexity to a linear scale of $O(Ld)$, with the number of hierarchy levels denoted as M. Viewing full-attention as a conditional expectation over embeddings at each location, Combiner [142] approximates this conditional distribution with structured factorization on token regions. Tokens can then attend to others either directly or through indirect attention to *abstractions*, which are conditional expectations from corresponding factorized local regions. This approach also leverages sparse attention patterns, as discussed in the next Sec.3.3, to provide sub-quadratic low computation and memory complexity while maintaining full-attention expressiveness.

Generally speaking, hierarchical attention mechanisms derive from the same principles of contextual locality present in natural languages as local attention. However, they incorporate a more elaborated structure, often designed heuristically, to strike a balance between capturing long-range contextual dependencies and maintaining low-level computational efficiency.

3.3 Sparse Attention

While some approaches have introduced heuristics for achieving locality and hierarchical structure within self-attention, another direction explores the sparsity patterns inherent in full attention matrices [32, 199]. These methods aim to introduce a sparse attention mask, denoted as $M_{\mathcal{S}}$, where each row i assigns a sparse set of indices $\mathcal{S}_i \subseteq \{j | j < i\}$ that the i-th token attends to. These sparsity-based attention mechanisms offer both computational efficiency and the ability to capture global context information. Figure 3 provides a visualization of these sparse attention mechanisms.

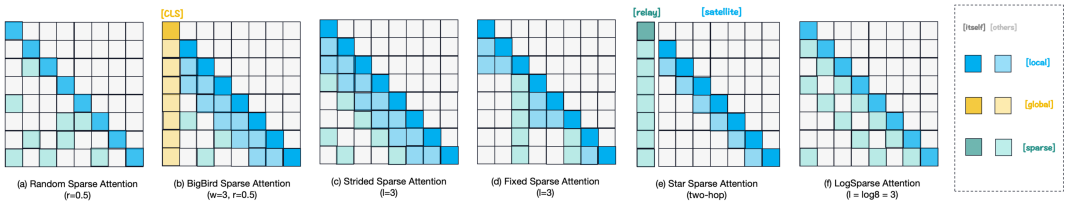

Figure 3: The visualization of some typical causal sparse attention patterns. The legend on the right distinguishes token types based on their colors, where darker shades indicate attending to themselves while lighter ones represent attention to other previous tokens.

Fixed Sparsity Patterns. To start with Sparse Transformer [32], it draws inspiration from attention patterns learned on CIFAR-10 [88] and proposes a row-column factorized attention scheme. This approach results in faster

computations while still maintaining global context awareness. Formally, it employs a chosen stride l that is close to $\sqrt{L}$. Each query $\mathbf{q}_i$ applies one *row attention* for local context information (i.e., local attention) and another *column attention* that summarizes previous locations and propagates information to all future tokens, resembling a form of global attention. The authors provide two specific patterns for row and column attention, corresponding to the *stride* and *fixed* ones respectively illustrated in Fig.3 ***(c),(d)***. This strategy reduces the total computational complexity to $O(L\sqrt{L}d)$. The intuition of Sparse Transformer is to attribute a stride $l \approx \sqrt{L}$ to equally distribute $\sqrt{L}$ tokens to attend to. In contrast, LogSparse [101] employs an exponentially sparse attention pattern by dispatching only $\log L$ tokens for each location to attend to. This method ensures that any pair of tokens can eventually exchange information with each other through a path spanning $\log L$ layers. This results in an overall memory usage of $O(L(\log L)^2)$. The recent LongNet [46] further improves computational efficiency by introducing *dilated attention*, which expands the attentive field exponentially as the distance between tokens increases. It incorporates mixed dilate rates to model both long and short-range dependencies, ultimately reducing the computation complexity to $O(Ld)$ while successfully scaling to sequences of up to one billion tokens.

Adaptive Sparsity Patterns. Instead of fixed sparse indices set only dependent on locations, some approaches seek sparsity adaptively in a learnable manner, taking into account embedding values. Expire-Span [163] introduces a learnable scalar in the range $[0, 1]$ for each previous token, allowing the model to retain tokens with the most important information while expiring those that are no longer relevant, similar to the *forget gate* in LSTM-based RNNs [198]. Routing Transformer [144] leverages k-means clustering to identify the top-k most relevant *centroid* vectors in Q, K and assigns each query to the keys with the same cluster membership, reducing the overall complexity of attention to $O(L\sqrt{L}d)$. Inspired by Differentiable Architecture Search(DARTS) [103], SparseBERT [153] introduces a differentiable attention mask using Gumbel relaxation techniques [108], allowing the model to learn to guide attention pattern selection by importance. It incorporates a predefined sparsity ratio ρ, resulting in computational complexity of $O((1-\rho^2)L^2d)$.

Graph Sparsification. Furthermore, some other works treat full attention as a fully connected graph, with nodes representing embeddings of each token and edges denoting connections through attention. These approaches frame sparsity as a *graph sparsification problem*. For instance, Star-Transformer [60] introduces a star-shaped topology, where each *satellite* node attends to local neighbors with a ring connection and a virtual *relay* node with the radial connection. In contrast, BigBird [201] incorporates sparsity based on random graph theory, allowing each query to attend to a random number of keys with a fixed probability. It also absorbs sliding-window local attention with window size w and global token techniques in its design. This approach reduces the quadratic dependency to linear complexity, specifically $O((w+r+1)Ld)$, stacked with three efficient attention mechanisms.

3.4 Approximated Attention

In addition to heuristic approaches aimed at restricting full attention computation, some research explores the mathematical essence behind attention kernel computations. These studies use estimation methods based on the sparsity or low-rank properties of attention matrices to approximate attention with linear complexity, albeit at the cost of precision. We introduce several of these approximation techniques below.

Note that we have not distinguished whether the methods are employed for BERT-like encoder-only LLMs or GPT-like decoder-only LLMs in previous sections since most of them can be trivially transferred from the BERT setting to the GPT setting with a causal attention mask. However, the casual mask is often non-trivial for many approximation strategies. So to facilitate later discussions, we first define a general weighted causal function $\xi_{\mathbf{w}}$ in Eq.8, where $\mathbf{w} \in \mathbb{R}^L$ represents a weights vector for each row. This function will substitute the causal attention mask operation thus, we omit the mask M in all of the attention equations below for simplification.

$$\xi_{\mathbf{w}}(Q, K, V) := \left[w_i \cdot \mathbf{q}_i^{\mathrm{T}} \sum_{j=1}^{i} \mathbf{k}_j \mathbf{v}_j^{\mathrm{T}} \right]_{i=1}^{L} \tag{8}$$

Low-Rank Approximation. Linformer [179] employs Singular Value Decomposition (SVD) to approximate the attention matrix A with a low-rank matrix $\widetilde{A}$, leveraging its low-rank property as proved in its *Theorem 1*. This approach involves two learnable projection matrices E and F of dimensions $L \times k$, where $k = O(\frac{d}{\epsilon^2}) \ll L$. The process includes projecting K, V using E, F respectively, followed by standard MHA kernel on Q with the projected $\widetilde{K}, \widetilde{V}$. According to its *Theorem 2*, this low-rank technique approximates full attention with linear complexity $O(Lkd)$ while allowing for an error of ϵ.

Nested Attention. Luna [107] decouples the attention kernel into two nested attention approaches, both of which have linear complexity with relative to L. Specifically, it firstly applies *pack attention* as Eq.9 to get *packed*

context $\widetilde{S}$, where S is an extra side-input sequence with constant length $k \ll L$, and the activation function $\mathrm{elu}(\cdot)$ is the exponential linear unit [37]. Then it secondly applies *unpack attention* as Eq.10 to get *unpacked output* $\widetilde{O}$, with the causal mask function defined in Eq.8. Afterwards, they pass $\widetilde{O}$ and $\widetilde{S}$ to next attention layer, again denoted as X and S, to propagate packed contextual information via S without leakage of future information. It is noteworthy that the pack attention can be regarded as a generalization of the linear projection in Linformer [179] with the same complexity $O(Lkd)$ but the advantage to model sequences with various lengths since the projection matrices are not dependent to L as projection matrix E, F.

$$A_s := \mathrm{elu}\left(\frac{Q_s \times K^{\mathrm{T}}}{\sqrt{d_k}}\right), \quad \widetilde{S} := A_s \times V, \ \ where \ \ Q_s := S \times W_q \tag{9}$$

$$A_u := \mathrm{softmax}\left(\xi_{\mathbf{w}_{inv}}(Q, V, A_s^{\mathrm{T}})\right), \ \widetilde{O} := \xi_{\mathbf{w}_{inv}}(A_u, A_s^{\mathrm{T}}, V), \ \ where \ \ \mathbf{w}_{inv} := \left[i^{-1}\right]_{i=1}^{L} \tag{10}$$

Kernelized Approximation. Except for relying on low-rankness prior, some works offer approximation based on generalized kernelizable attention shown in Eq.11, where the kernel function $\mathcal{K}(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to R_+$ is applied row-wise to each pair of $\mathbf{q}_i, \mathbf{k}_j$ in Q, K, and D is the normalization factor defined in Eq.12. From this view, the vanilla softmax attention just implements a specific kernel function as $\mathcal{K}(Q,K) = \exp(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}})$, which explicitly derives a $L \times L$ attention matrix. But if we carefully choose another kernel function to be factorizable as the condition in the second step of Eq.11,12, then simply applying the associative property, we can compute matrix multiplication of K, V and $K, \mathbf{1}_L$ ahead with lower complexity $O(Ld^2)$. However, the drawback is that similar to Luna [107], one has to compute iteratively across each query $\mathbf{q}_i$, which does not fully make use of the parallelism, as shown in the third step of Eq.11,12.

$$O := \left(D^{-1} \times \mathcal{K}(Q,K)\right) \times V \overset{\mathcal{K}(Q,K)=\widetilde{Q}\times\widetilde{K}^{\mathrm{T}}}{\underset{associative}{=\!=\!=}} D^{-1} \times \widetilde{Q} \times \left(\widetilde{K}^{\mathrm{T}} \times V\right) \overset{causal}{=\!=\!=} D^{-1} \times \xi_{\mathbf{1}_L}(\widetilde{Q}, \widetilde{K}, V) \tag{11}$$

$$where \quad D := \mathrm{diag}\left[\mathcal{K}(Q,K) \times \mathbf{1}_L\right] \overset{\mathcal{K}(Q,K)=\widetilde{Q}\times\widetilde{K}^{\mathrm{T}}}{\underset{associative}{=\!=\!=}} \mathrm{diag}\left[\widetilde{Q} \times \left(\widetilde{K} \times \mathbf{1}_L\right)\right] \overset{causal}{=\!=\!=} \mathrm{diag}\left[\xi_{\mathbf{1}_L}(\widetilde{Q}, \widetilde{K}, \mathbf{1}_L)\right] \tag{12}$$

For instance, Linear Transformer [80] designs a simple feature map φ_{Li} based on the *elu* kernel as shown in Eq.13. Therefore, it avoids quadratic attention matrix and diminishes the time-space complexity to $O(Ld^2)$ and $O(Ld)$. In contrast, Performer [34] achieves unbiased and low-variance estimation based on orthogonal random features (ORFs) mapping $\varphi_{Pe}(\cdot) : \mathbb{R}^d \to \mathbb{R}^r$, as shown in Eq.14, where $r = m \times l \ll L$, $b_1, .., b_l : \mathbb{R} \to \mathbb{R}$ are l basis functions, $h : \mathbb{R}^d \to \mathbb{R}_+$ is the norm function, and $\omega_1, ..., \omega_m \in \mathbb{R}^d$ are m orthogonalized random features $i.i.d$ sampled from a distribution $\mathcal{D} \in \mathcal{P}(\mathbb{R}^d)$. Performer provides multiple configurations of these parameterized functions, such as $h := 1, l := 1, \mathcal{D} := \mathcal{N}(0, \mathbf{I}_d)$ called PNG-kernels [35], and $h := 1, l := 2, b_1 := sin, b_2 := cos, \mathcal{D} := \mathcal{N}(0, \sigma^2\mathbf{I}_d)$ leading to shift-invariant Gaussian kernel [138], which is also adopted in RFA [127]. Hence the time complexity can be declined to $O(Lrd)$, so as the space complexity dropped to $O((d+r)L)$.

$$\mathcal{K}_{Li}(\mathbf{q}, \mathbf{k}) := \varphi_{Li}(\mathbf{q}) \times \varphi_{Li}(\mathbf{k})^{\mathrm{T}}, \ \ where \ \ \varphi_{Li}(\mathbf{x}) = \mathrm{elu}(\mathbf{x}) + 1 \tag{13}$$

$$\mathcal{K}_{Pe}(\mathbf{q}, \mathbf{k}) := \mathbb{E}_{\omega}\left[\varphi_{Pe}(\mathbf{q}) \times \varphi_{Pe}(\mathbf{k})^{\mathrm{T}}\right], \ \ where \ \ \varphi_{Pe}(\mathbf{x}) = \frac{h(\mathbf{x})}{\sqrt{m}}\left[b_1(\omega_1^{\mathrm{T}}\mathbf{x}), .., b_1(\omega_m^{\mathrm{T}}\mathbf{x}), .., b_l(\omega_1^{\mathrm{T}}\mathbf{x}), .., b_l(\omega_m^{\mathrm{T}}\mathbf{x})\right] \tag{14}$$

Sparse-Kernelized Hybrid. Furthermore, inspired by Robust-PCA [21], the recent Scatterbrain [23] provides a more accurate yet efficient approximation by combining LSH-based sparse matrices S like Reformer's [84] and low-rank kernelized decomposition with randomized feature maps like Performer's [34], as simplified in Eq.15, where we omitted the normalization step and the causal mask applying function. Not only does this method unify two approximation techniques to achieve linear time complexity of $O(Lrd)$ with higher precision, but it also offers flexibility to leverage various low-rank and sparse approximation methods as sub-components, more than just the example combination of Reformer and Performer.

$$\widetilde{O} := \left(\widetilde{Q} \times \widetilde{K}^{\mathrm{T}} + S\right) \times V = \widetilde{Q} \times (\widetilde{K}^{\mathrm{T}} \times V) + S \times V \tag{15}$$

3.5 IO-Aware Attention

All of the methods above in pursuit of efficient attention can be considered as trading off high attention quality for low computation complexity, based on some theoretical or empirical properties of attention matrix and NLP

tasks, including locality, sparsity, low-rankness, and other heuristic or mathematical tricks. In comparison, these IO-aware attention mechanisms below collectively represent efforts to optimize attention computations by considering the memory bottleneck while preserving the exactness of attention kernel calculations.

Memory-Efficient Attention. This simple method is firstly proposed in [133], which utilizes the *lazy softmax* algorithm [76] and tracks normalization factor to compute standard and numerically stable attention by sequentially processing each single/chunked-query attention. Such a simple method only needs constant working memory with respect to sequence length, while the time complexity is still quadratic.

Flash Attention. To Push it forward, the recent work Flash Attention [42, 41] manages to reduce time and memory consumption while applying exact attention computation by making it IO-aware between GPU high bandwidth memory(HBM) and GPU on-chip SRAM. It has already reached wide adoption and has been incorporated directly into Pytorch v2.0 [131]. To be more specific, for forward pass, it utilizes tiling technique [190] to decompose large softmax attention into smaller blocks, loading block by block from HBM [78] to SRAM, so as to perform all the attention computation steps on-chip to reduce HBM accesses, and incrementally update the output back to HBM by rescaling, leveraging the online softmax technique [116]. As Eq.16 illustrates, $P^{(n)} := Q_{i_n} \times K_{j_n}^{\mathrm{T}} \in \mathbb{R}^{B_r \times B_c}$ denotes one pair of block-wise matrix multiplication from Q, K, which is tiny enough to be loaded and computed in SRAM, and the rescaling factor $\alpha^{(n)}$ only comprises some statistics about $P^{(n)}$. As for backward pass in training stages, it can both decrease the required memory and speed up due to reduced HBM accesses as well, by just storing the output O and normalization statistics to recompute the intermediate results P, A in SRAM, similar to the gradient checkpointing technique [28, 155]. The authors analyze that flash attention only requires $O(L^2 d^2 M^{-1})$ HBM accesses compared to standard $O(Ld + L^2)$, where $M \gg d^2$ is the size of SRAM, which leads to both faster execution (up to 7.6× speedup on GPT2 [135]) and lower memory footprint (up to 20× more memory efficient), according to the experiments results [42].

$$O := \mathrm{softmax}\left(\begin{bmatrix} P^{(1)} & P^{(2)} \end{bmatrix}\right) \begin{bmatrix} V^{(1)} \\ V^{(2)} \end{bmatrix} = \alpha^{(1)}\mathrm{softmax}(P^{(1)})V^{(1)} + \alpha^{(2)}\mathrm{softmax}(P^{(2)})V^{(2)} \tag{16}$$

SCFA. Although Flash Attention can be easily extended to support block-sparse structures [42], it may lack flexibility for handling other sparse strategies with irregular structures and arbitrary attention masks. The effort of SCFA [124] extends the Flash Attention GPU kernel to accommodate a broad range of attention sparsity patterns, including key/query dropping and hashing-based attention like Reformer [84]. This extension leads to a training speedup of 2.0 to 3.3 times without sacrificing perplexity, according to the report from the paper.

Paged Attention. While Flash Attention has effectively tackled the training memory bottleneck, LLMs still face challenges related to the memory consumption of the KV cache during inference, which grows dynamically with batched requests. Recognizing the memory wastage due to fragmentation and redundancy, vLLM proposes Paged Attention [90]. This technique efficiently manages KV cache memory to minimize waste and allows flexible sharing across batched requests, drawing inspiration from memory paging techniques [82] in virtual memory operating systems [44].

4 Long-Term Memory

Transformer architectures often struggle with capturing long-term dependencies due to in-context working memory, as highlighted in Sec.2.2. Researchers have explored two main avenues to address this challenge without compromising the advantages of full attention. First, inspired by RNNs, some have introduced recurrent mechanisms into attention by incorporating internal memory caches accessible through attention layers. This approach enables the model to maintain and retrieve information over longer sequences, compensating for the inherent lack of built-in long-term memory. Second, an alternative approach involves leveraging existing models as interfaces to external knowledge bases, such as specific documents or datasets. During inference, the model can read from these knowledge bases to enrich its contextual input and write to them from the user's response to refresh its long-term memory. By integrating external knowledge in this manner, the model gains access to a broader range of context, enhancing its ability to handle long-term dependencies effectively.

4.1 Internal MemoryCache

Recalling the temporality of natural language representations instead of the success of full parallelism in Transformer, we introduce the concept of *Internal MemoryCache* based on recurrence mechanisms. It divides long text into a stream of fixed-length segments and enhances the query Q_t^n of the current t-th segment in the n-th layer with

more contextual information $\widetilde{K}_t^n, \widetilde{V}_t^n$. This contextual information is obtained from cached or distilled information from previous segments, stored in a memory cache denoted as Mem, as shown in Eq.17. To facilitate later explanations, we assume that each segment has the same length l, and the models consist of N layers of transformer blocks. The notation $[\circ]$ represents the concatenation operation along the length dimension. It's worth noting that the variables in the memory cache Mem are usually detached from the computation graph, eliminating the need for gradient computation, which we denote with a hat accent, such as $\widehat{X}$.

$$Q_t^n,\ \widetilde{K}_t^n,\ \widetilde{V}_t^n := X_t^n W_q,\ \widetilde{X}_t^n W_k,\ \widetilde{X}_t^n W_v,\ \ where\ \ X_t^n := O_t^{n-1},\ \widetilde{X}_t^n := \left[\mathrm{Mem}(n,t,..) \circ O_t^{n-1}\right] \tag{17}$$

Segment-Level Recurrence. The segment-level recurrence is initially introduced into Transformer from Transformer-XL [40]. As illustrated in Eq.18, it caches the output of m previous consecutive segments in the last layer and concatenates them into the current segment in the present layer to extend the context for the current query. Such mechanism allows for extending the largest possible dependency distance to $O(Nml)$, where m can be set as far as GPU memory allows. Building upon Transformer-XL, Segatron [9] introduces the segment-aware mechanism by enhancing the token-level PEs combined with sentence-level and even paragraph-level ones. To further extend the dependency with multi-grained memory caching, Compressive Transformer [136] stores the first FIFO fine-grained memory queue for m_1 previous segments as Transformer-XL does. However, instead of discarding old memory, it applies a *compression function* f_c with the rate c to compress it along the length dimension and pushes it into a secondary FIFO coarse-grained *compressive memory* queue of size m_2. Combining these two types of memories, one can obtain longest context dependency as $O(Nl(m_1 + cm_2))$, as shown in Eq.19.

$$\mathrm{Mem}_{XL}(n,t,m) := \left[\widehat{O}_{t-m}^{n-1} \circ .. \circ \widehat{O}_{t-1}^{n-1}\right] \tag{18}$$

$$\mathrm{Mem}_{Comp}(n,t,m_1,m_2,c) := \left[\mathrm{Mem}_{f_c} \circ \mathrm{Mem}_{XL}(n,t,m_1)\right], \tag{19}$$

$$where\ \ \mathrm{Mem}_{f_c} := \left[f_c(\widehat{O}_{t-m_1-m_2}^{n-1}) \circ .. \circ f_c(\widehat{O}_{t-m_1-1}^{n-1})\right]$$

Retrospective Recurrence. Note that both Transformer-XL and Compressive Transformer deploy a shifting-one-layer-downwards recurrence by default, thus the maximum effective context length is limited by N. To address it, similar to Feedback Transformer [53], ERNIE-Doc [48] proposes an enhanced recurrence mechanism, a drop-in replacement by concatenating the output hidden states of previous segments in the same layer, instead of the last layer, simply formalized as Eq.21. In this manner, no only the maximum effective context length can be implicitly expanded, but the past higher-level representations can be exploited to enrich future lower-level representations as well. Additionally, it employs a *retrospective feed* mechanism by feeding the segments twice, where the first time only skims each segment while the second one retrospects to enable bi-directional information flow, which resembles the mechanism in READTWICE [202].

$$\mathrm{Mem}_{\infty} := \widetilde{X}(s) = B^{\mathrm{T}}\Phi(s),\ \ s.t.\ \ \widetilde{X}(s_i) \approx X_i,\ s_i := i/L,\ \forall i \in [1,..,L] \tag{20}$$

$$\mathrm{Mem}_{Ernie}(n,t) := \widehat{O}_{t-1}^n \tag{21}$$

Continuous-Signal Memory. To draw a comparison with LSTM [198], we can view the compressed memory in Compressive Transformer as a finite-sized discrete version of the long-term cell memory in LSTM, while the first queue stores the short-term one. To achieve unbounded long-term memory like LSTM, as Eq.20 suggests, ∞-former [111] transfers the L token-wise discrete embeddings $X \in \mathbb{R}^{L\times d}$ into a continuous signal $\widetilde{X}(s) : [0,1] \to \mathbb{R}^d$. This signal is expressed as a linear combination of m radial basis functions (RBFs), denoted as $\Phi(s) \in \mathbb{R}^m$ with the coefficient matrix $B \in \mathbb{R}^{m\times d}$, which is fitted by multivariate ridge regression [16]. This continuous signal representation allows for unbounded context representation with fixed memory storage, independent of the context length, similar to LSTM. However, as the memory cache is stored as a continuous signal, it cannot simply prepend to the current segment but has to transform back to embeddings via continuous attention [110].

$$\widetilde{O}_t^N := \mathrm{Transformer}(\widetilde{X}_t^0),\ \widetilde{X}_t^0 := \left[X_t^{mem} \circ X_t^0 \circ X_t^{mem}\right], \tag{22}$$

$$where\ \ X_t^{mem} := O_{t-1}^{write},\ \left[O_{t-1}^{read} \circ O_{t-1}^N \circ O_{t-1}^{write}\right] := \widetilde{O}_{t-1}^N$$

$$(\widetilde{K}_t^N, \widetilde{V}_t^N) := \left[\mathrm{retr}(Q_t^N, m, k) \circ (K_t^N, V_t^N)\right], \tag{23}$$

$$where\ \ \mathrm{retr}(Q_t^N, m, k) := \mathrm{kNN}\left[Q_t^N, \{(K_{t-\tau}^N, V_{t-\tau}^N)\}_{\tau=1}^m\right]$$

Alternate Cache Designs. Except following the prepending style of *Mem* as the memory cache, RMT [20] formalizes the memory cache as special `[mem]` tokens, prepended both at the start and the end of each segment, as shown in Eq.22. After processing each segment, the read/write tokens will be split from the output embeddings and the write tokens will be taken as the `[mem]` tokens for next segment. By leveraging such recurrence mechanism with global memory tokens, RMT is demonstrated to scale effective context size to 1M tokens [19]. And Memorizing Transformer [186] applies (key,value) memory cache only for the top attention layer, but with a large cache size without compression. Besides, instead of a simple FIFO cache to read memory, they use kNN algorithm to retrieve top-k most similar (key,value) pairs for each query to prepend to the local ones, as Eq.23 indicates. In contrast, Memformer [185] reads and writes the memory cache fully leveraging variants of self-attention with a *forgetting mechanism*, to retrieve and retain the most significant information through long-range timesteps.

4.2 External MemoryBank

The previously discussed mechanisms enhance the vanilla stateless Transformer model with sequential recurrence by prepending extra hidden states of previous inputs from an internal memory cache. However, these mechanisms have certain drawbacks. Firstly, a slight change in the memory mechanism may necessitate retraining the model from scratch, not fully utilizing pre-trained LLMs that already possess a good representation of dependency across the context window, albeit not long enough. Secondly, as highlighted in Memorizing Transformer [186], they often encounter the problem of *memory staleness*, where older hidden states in memory cache may exhibit distributional shifts from the latest ones during training, limiting the effectiveness of memory augmentation.

As a solution, another retrieval-augmented mechanisms decouples the model itself from its long-term memory storage. They leverage the part before the language head as a well-performing contextual information encoder to store long sequences as an external memory bank in the form of embeddings. And during queries, the model retrieves information from this memory bank based on certain criteria and concatenates it to comprise in-context working memory in real-time.

Cosine-Based Retrieval Criteria. LangChain [22, 172], an hot open-source framework for developing applications like chatbots, accepts user-specified local documentation in common readable formats, then vectorizes this documentation using off-the-shelf LLMs into a memory bank. During each user interaction, it retrieves the top-relevant contexts based on dot-product *cosine similarity* between user prompt embeddings and the stored contexts. It then prepends these external contexts to the prompt, providing the LLMs with a more related input to generate responses. LangChain offers an effective pipeline to harness the capabilities of off-the-shelf LLMs and enhance their long-term memory with a cost-effective, flexible, and dynamic mechanism. Several similar works have also designed external memory banks for QA-like tasks or chatbot applications.

Heuristic Retrieval Criteria. Except for leveraging the cosine similarity, RETRO [15] retrieves from the BERT-embedded KV memory bank through kNN search based on L_2 distance. Also based on kNN search, Unlimiformer [12] designs a general approach for any existing pretrained encoder-decoder transformer to index unlimited input sequences for the decoder to retrieve its top-k keys to apply cross-attention. In contrast, SiliconFriend [208] proposes an enhanced *Memory Bank* mechanism to keep tracking the long-chat history with the user and provide specialized responses, consisting of dialogue logging with timestamps, events distillation into a high-level summary, awareness of user's personality portrait and memory refreshment with a rate of forgetting. Similar to this text-based memory, RecurrentGPT [209] enables recurrent prompting and defines the recurrent computation graph with ChatGPT [121] by simulating the Long Short-Term Memory mechanism in LSTM [198]. Moreover, RecallM [91] organizes and updates the memory as a dynamic concept-aware knowledge graph to improve more complex continual learning and temporal reasoning of the knowledge during chat. Inspired by the *Davidsonian semantics* [43], Ret-LLM [118] stores and retrieves knowledge from the memory bank in the form of triplets like $\langle A, B, R\rangle$ which means *"A and B have a relationship of R"*, as a general read-write memory unit and employs fine-tuned Alpaca [166] to treat memory read/write as text-based API calls.

Learnable Retrieval Criteria. Despite these heuristic designs, REALM [61] pre-trains a latent *neural knowledge retriever* using MLM as the learning signal, that takes charge of retrieving knowledge from the large textual corpus. LongMem [180] trains another Transformer-based *SideNet* to decouple the memory retrieval and fusion process from the pre-trained LLMs which are only responsible for encoding the (key, value) pairs into the memory bank.

5 Extrapolative PEs

Recognizing the need to push the inference length boundary beyond L_{max}, the research community has made significant efforts in this direction. Notably, according to [5], they have determined that *distractors* are the primary cause of failures in length generalization in the case of *parity* task. These issues, however, can be mitigated considerably through approaches such as scratchpad prompting [120]. Nevertheless, in this section, our focus remains on the undeniable role that current PEs play in length generalization in more general scenarios.

5.1 Enhancing Understanding

Before entering into the concrete approaches, we would love to provide some insights below to enhance the understanding of this minute but essential design in Transformer for sequential modeling tasks.

Rethinking PEs as β-Encoding. Su [157] revisits the sine and cosine basis functions of Sinusoidal PE and RoPE, considering them as approximated terms for the β-encoding system to represent any position number n, as shown in Eq.24. This approach employs $\frac{d}{2}$ fixed β-bits, where $\beta := \theta^{-1} = base^{2/d}$ represents the power basis of the wavelength or period of the trigonometric basis functions, which increases as a geometric series $\{\beta^i\}_{i=0}^{d/2}$ with the dimension i goes deeper.

To gain a deeper understanding of this concept, we can draw a comparison between Eq. 24 and Eq.5,6. It becomes evident that the i-th β-bit of the representation of n involves the division of the i-th power of β, followed by some sort of periodical operations (mod in Eq. 24 and sin, cos in Eq.5,6).

$$n(\beta) := \{ \lfloor \frac{n}{\beta^i} \rfloor \bmod \beta \}_{i=0}^{\lceil \log_\beta n \rceil - 1} \tag{24}$$

Length Extrapolation Dilemma. Prior to the era of Transformers, RNN-based language models were trained on shorter sequences but were expected to generalize effectively to longer contexts, a phenomenon referred to as *length extrapolation* or *length generalization* [114, 115]. Unfortunately, recent studies [129, 27, 5] have highlighted a significant shortcoming of *length extrapolation* ability for Transformer-based language models. This causes the insufficient context length limit during inference when applying to real-world applications, as analysed in Sec.2.2.

In the original Transformer paper [175], there is few discussion regarding the design insights or theoretical interpretation of their Sinusoidal PE. This has led many researchers to question its necessity and effectiveness, especially the blame on the extrapolation deficit, which points to the same trigonometry-based RoPE [161] as well. To understand the bad extrapolation caused by current trigonometric PEs, we investigate and summarize two insights from distint views as below:

- From a mathematical view, as Su [160] explains in his blog, extrapolation, which involves inferring the whole from local information, depends on the high-order *smoothness* of the function. However, to accommodate sufficient positional information, these PEs are designed as combinations of high-frequency oscillatory trigonometric basis functions. This choice makes it challenging for the models to generalize without specific learning during training stages.

- From a training view, due to the wavelength or period of the basis functions increases exponentially, proportional to $\{\beta^i\}_{i=0}^{d/2}$, training samples constrained by currently supported L_{max} are typically too short for the rear low-frequency dimensions to span the entire periodic cycle. This suggests only a few dimensions perceive complete periodic information thus receiving sufficient training for extrapolation, and the boundary is defined as *critical dimension* in [104] (e.g. for Llama2-4k [174], the critical dimension is only 92). Consequently, direct extrapolation becomes prone to failure when relying on these poor-learned low-frequency components.

5.2 Attention Bias

As alternative mechanisms to explicitly encoding positional information, *attention bias* have been explored to capture the sequentiality and temporality of natural language incorporated into attention kernel. As shown in Eq.25, the attention bias is depicted as a matrix, denoted as B, which is added to the unnormalized attention weights matrix P before applying the softmax operation. Each element of this matrix, indexed by (i, j), carries positional information encoded by a function $\mathcal{B}$. Thus, it is reasonable to regard the attention bias as a form of relative PEs.

$$\widetilde{P} := P + B, \quad B \in \mathbb{R}^{L \times L}, \quad where \;\; B_{ij} := \mathcal{B}(i,j), \;\; \forall i,j \in \{0,1,..,L-1\} \tag{25}$$

Early approaches like T5 [137] employ learnable attention bias, denoted as $\mathcal{B}_\theta(i,j)$, which is independent for each head in each attention layer. However, they did not explicitly address the problem of length extrapolation. The breakthrough in recognizing and addressing the extrapolation problem comes with ALiBi [129]. ALiBi introduces a negative causal attention bias heuristically, as shown in Eq.26, where $\lambda^{(h)}$ is a head-specific slope fixed before training and decreases geometrically with the head index h. ALiBi successfully maintains low perplexity levels when extrapolating inference tokens beyond L_{max} up to 16×.

Following the success of ALiBi, several variants emerged in the quest to improve extrapolative PEs for Transformer-based LLMs. KERPLE [30] extended the ALiBi-style attention bias by considering it as a composition triangle kernel to self-attention. Two extra learnable scalar parameters were introduced to generalize the bias kernel, as shown in Eq.27. The authors of Sandwich [31] reused the Sinusoidal PEs to form the attention bias in a RoPE-style, as illustrated in Eq.28, with λ as a hyper-parameter to tune. Interestingly, another method discussed by Su [160] in his blog utilizes a *super-baseline* approach during inference, as illustrated in Eq.29. This method relies on a local causal attention mask, where each query attends to keys whose distances have not exceeded L_{max} while still applying RoPE. According to Su's experiments, this approach proves to be simple, low-cost and performs sufficiently well compared to the more elaborate designs mentioned earlier, thus referred as a *super-baseline*.

$$\mathcal{B}^{(h)}_{ALiBi}(i,j) := -\lambda^{(h)} \cdot |i-j|, \;\; \lambda^{(h)} := \frac{1}{2^h} \; or \; \frac{1}{2^{h/2}} \tag{26}$$

$$\mathcal{B}_{KERPLE}(i,j) := \begin{cases} -r_1 \log(1 + r_2|i-j|), & r_1, r_2 > 0 \\ -r_1|i-j|^{r_2}, & r_1 > 0, r_2 \in (0,2] \end{cases} \tag{27}$$

$$\mathcal{B}_{Sandwitch}(i,j) := \lambda \cdot \langle \text{SinPE}(i), \text{SinPE}(j) \rangle \tag{28}$$

$$\mathcal{B}_{super\text{-}baseline}(i,j) := \begin{cases} 0, & |i-j| \in [0, max\text{-}length] \\ -\infty, & otherwise \end{cases} \tag{29}$$

5.3 Extended RoPE

RoPE, as introduced in Sec.2.1, is a widely-used positional encoding scheme utilized in popular LLMs such as Llama, GLM, PaLM. It offers advantages such as relative distance decay, training stability, compatibility with linear attention, and better length extrapolation capabilities compared to the traditional Sinusoidal PE, as demonstrated in various experiments [129, 27], albeit not that satisfactory. Therefore, several research works have aimed to extend RoPE using various strategies to enhance its length extrapolation capabilities.

Scaling Strategies. Recent approaches by simply scaling it to extrapolate the inference context length with minimal or no fine-tuning [27, 14], has gained prominence within the community. In LEX [164], the authors introduce an extended causal RoPE called XPOS, which incorporates an additional exponential decay term to scale RoPE, as illustrated in Eq.30, where $\gamma \in (0,1)$ is a scalar hyper-parameter. Similar techniques have been previously employed in PermuteFormer [25] to adapt to the linear attention introduced by Performer [34]. Positional Interpolation (PI) [27] applies linear scaling on each position number from n to $\frac{n}{\kappa}$, densifying the representation space to extend the farthest length boundary by κ times (see Eq.31). This strategy is experimentally shown to be more stable and require fewer fine-tuning steps than direct extrapolation.

However, it's evident that simple linear scaling may hinder the network's ability to distinguish the order and positions of closely spaced tokens, as their distances are compressed by a ratio of κ. Borrowing from the Neural Tangent Kernel theory (NTK) [74], which suggests that deep neural networks struggle to learn high-frequency information when the input dimension is low and corresponding embeddings lack high-frequency components, NTK-aware Scaling RoPE (NTK-RoPE) [14] combines *high-frequency extrapolation and low-frequency interpolation*. It scales β using a coefficient c_κ to achieve equivalence when applying interpolation by a ratio of κ for the lowest frequency term, while keeping the scale for high-frequency terms (see Eq.32). Surprisingly, this nonlinear scaling can be directly applied to LLMs pre-trained with RoPE, such as Llama, without any further fine-tuning to extend the context length boundary. This approach has already been deployed in open-source LLMs like CodeLlama [145].

Inspired by NTK-RoPE, several enhanced scaling methods have emerged. To avoid performance degradation when L is still within the "max length," Dynamic-NTK [52, 126] delays the application of the scaling trick until L exceeds the current supported context length. It gradually increases the ratio κ dynamically as L increases. This

idea has also been implemented in models like Qwen-7B [79] and the latest version of Llama2 [56]. Another approach, NTK-mix RoPE [158], introduces multiple coefficients $c_\kappa^{(0)} \geq c_\kappa^{(1)} \geq ... \geq c_\kappa^{(d/2-1)}$ for β to interpolate less as the frequency increases. In contrast, NTK-by-parts [13, 126] opts not to interpolate the higher frequency dimensions at all while always interpolating the lower ones. The authors of NTK-RoPE also propose YaRN [126], which combines NTK-by-parts with a "length scaling" trick that scales Q and K by a constant temperature factor t. This method claims to outperform all previous methods based on NTK-RoPE, whether fine-tuned or not. Additionally, Giraffe [125] introduces another scaling strategy known as "Power Scaling" (see Eq.33), where the exponent $\kappa > 0$ controls the decay ratio of low frequencies. This approach ensures that high-frequency elements of the basis are less affected than poorly learned low-frequency elements.

Inspired by NTK-RoPE, several enhanced scaling methods have emerged. To avoid performance degradation when L is still within the L_{max}, Dynamic-NTK [52, 126] delays the applying of the NTK scaling trick until L exceeds the current supported context length. And it gradually increases the ratio κ dynamically as L increases. This idea has also been implemented in models like Qwen-7B [79] and the latest version of Llama2 [56]. And to generalize β scaling for each dimension, NTK-mix RoPE [158] introduces multiple coefficients $c_\kappa^{(0)} \geq c_\kappa^{(1)} \geq .. \geq c_\kappa^{(d/2-1)}$ for β to interpolate less as the frequency increases. In contrast, NTK-by-parts [13, 126] opts not to interpolate the higher frequency dimensions at all while always interpolating the lower ones. Recently the authors of NTK-RoPE further proposes YaRN [126], which combines NTK-by-parts with a *length scaling* trick that scales Q, K by a constant temperature factor t. This method claims to outperform all previous methods based on NTK-RoPE, whether fine-tuned or not. Additionally, Giraffe [125] introduces another scaling strategy, namely *Power Scaling*, as shown in Eq.33, where the exponent $\kappa > 0$ controls the decay ratio of low frequencies.This approach ensures that high-frequency elements of the basis are less affected than poorly learned low-frequency elements. Despite these manual designs, CLEX [24] also exploits a neural ordinary differential equation (ODE) to learn a continuous scaling series as a continuous dynamical system.

$$\text{XPOS}: \ P_{i,j} := \langle \widetilde{\mathbf{q}}_i, \widetilde{\mathbf{k}}_j \rangle = \gamma^{i-j}(\mathbf{q}^{\mathrm{T}} R_{j-i}\mathbf{k}), \ \ \textit{where} \ \ \widetilde{\mathbf{q}}_i := \gamma^i(R_i\mathbf{q}), \ \widetilde{\mathbf{k}}_j := \gamma^{-j}(R_j\mathbf{k}), \ i \geq j \tag{30}$$

$$\text{PI}: \ \widetilde{P}_{i,j} := \langle R_{i/\kappa}\mathbf{q}, R_{j/\kappa}\mathbf{k} \rangle = \mathbf{q}^{\mathrm{T}} R_{\frac{j-i}{\kappa}}\mathbf{k} \tag{31}$$

$$\text{NTK}: \ \widetilde{\beta} := c_\kappa \cdot \beta, \ \ s.t. \ \ \frac{n}{\widetilde{\beta}^{d/2-1}} = \frac{n/\kappa}{\beta^{d/2-1}} \Rightarrow c_\kappa = \kappa^{2/(d-2)} \tag{32}$$

$$\text{Power Scaling}: \ \widetilde{\beta}^i := \beta^i/(1-2i/d)^\kappa \tag{33}$$

Truncation Strategies. Based upon the idea of *high-frequency extrapolation and low-frequency interpolation* from NTK-RoPE, *Su* further proposes two simple truncation strategies in his blog [159], named *ReRoPE* and *Leaky ReRoPE* after the activation function Rectified Linear Unit (ReLU) and its leaky variant. As shown in Eq.34, the main idea behind this *Rectified Truncation* approach is to set a local window with size w, and for each token, no scaling is applied as long as the tokens to attend are inside the window. However, linear scaling, akin to Leaky ReLU, is used to increase the position by step $1/\kappa$ when the token is located outside of the window (Leaky-ReRoPE). This method combines high-frequency extrapolation and low-frequency interpolation more directly and ensures that L_{max} is not exceeded by tuning w and κ carefully. Furthermore, if the ratio κ is set to infinity, it applies a constant PE of position number w to any pair of $(\mathbf{q}_i, \mathbf{k}_j)$ as long as $|i-j| \geq w$, potentially accommodating infinite contexts (ReRoPE). According to *Su*'s and our own elementary experiments, ReRoPE performs very well without any fine-tuning on perplexity metric and QA tasks, even outperforming NTK-based schemes.

However, both Leaky ReRoPE and ReRoPE involve connecting two stages of scaling, and their gap cannot be bridged by any linear transformation. Therefore, they require two attention matrix computations for each stage and utilize a boolean matrix to stitch them together, significantly increasing inference cost and limiting the real length boundary. Worse still, they are currently not compatible with the Flash Attention [42, 41] to mitigate the high computational cost. To adapt ReRoPE with Flash Attention, we have re-implemented the Flash Attention forward kernel to incorporate ReRoPE based on the Triton framework [171], somewhat alleviating its high computational cost[1]. Additionally, Giraffe [125] introduces another truncation strategy, namely *Basis Truncation*, as illustrated in Eq.35, where a, b are cutoff thresholds. This approach preserves the high-frequency components of the basis while cutting off the low-frequency elements to near-zero constant values ($\rho \approx 0$) and even zeros when low enough, resulting in less complex extrapolation for the low frequencies.

[1]The experimental implementation can be accessed at the following URL: https://github.com/Strivin0311/long-llms-learning/blob/main/notebooks/flash_rerope.ipynb.

$$\text{Rectified Truncation}: \ \widetilde{P}_{i,j} := \langle R_{\alpha(i,j,w,\kappa)}\mathbf{q},\ \mathbf{k}\rangle, \tag{34}$$
$$where\ \ \alpha(i,j,w,\kappa) := \begin{cases} \min\{i-j, w+\frac{i-j-w}{\kappa}\}, & 0<\kappa<\infty \text{ (Leaky ReRoPE)} \\ \min\{i-j,w\} & \kappa\to\infty \text{ (ReRoPE)} \end{cases}$$

$$\text{Basis Truncation}: \ \widetilde{\theta}^i := \begin{cases} \theta^i, & \theta^i \geq b \\ \rho, & \theta^i \in (a,b) \\ 0, & \theta^i \leq a \end{cases} \tag{35}$$

Rearrangement Strategies. Based on the insights discussed in Sec.5.1, it is clear that the PEs of rear positions are updated fewer times than the front position embeddings. Such imbalance may lead to improperly trained rear positions. Recently, this issue has been addressed to some extent by some simple but effective works. SHAPE [85] randomly shifts absolute positions during training to achieve *shift invariance*. Random Padding [165] involves moving a random number of padding tokens to the front of the input sequence during fine-tuning, balancing the updating times across all positions. Additionally, Randomized PE [147] randomly sub-samples an ordered set of positions from a much larger range of positions $\{0, 1, .., \mathcal{L}\text{-}2, \mathcal{L}\text{-}1\}$ than the sequence length ($\mathcal{L} \gg L$) during training, allowing the model to handle a broader range of PEs more robustly. PoSE [210] fine-tunes the model to adapt all relative positions of the target context window (e.g. 128k) by adding a distinct *skipping bias term* to the position indices of training samples (e.g. 2k) to simulate longer inputs.

In summary, research on extrapolative PEs is a promising and rapidly-developing field, aiming to enhance the LLMs' ability to infer long contexts in real-world scenarios with an available L_{max} setting during training.

6 Context Processing

Many of the methodologies discussed earlier propose intricate designs around the attention module in Transformer architecture, including efficient attention kernels (Sec.3), long-term memory mechanisms (Sec.4), and extrapolative PEs (Sec.5). In contrast, there exist simpler and more straightforward approaches that view pre-trained LLMs as black-box or gray-box models. These methods tackle the challenge of handling long-context inputs that exceed the model's length limitation by making multiple calls to the model, ensuring that the actual input provided to the LLM in each call does not exceed L_{max}. While these methods do not explicitly enhance the LLMs' inherent ability to process long-contexts, they leverage the LLMs' remarkable in-context learning capabilities to address the issue, albeit at the cost of increased computation and potentially less precise answers.

6.1 Context Selection

These methods primarily involve partitioning lengthy texts into multiple segments, followed by the selection of specific segments based on a predefined strategy. The goal is to ensure that the chosen segments can comfortably fit within the context window of LLMs, all while minimizing the loss of pertinent information from the original lengthy text that is relevant to the posed query. These approaches differ in two key aspects. Firstly, they diverge in how they define the selection criteria, which are used to assign a priority score to each segment. Secondly, they vary in their selection strategies, with some methods involving the simultaneous sorting of all segments based on their scores, while others employ an iterative, greedy selection approach, considering segments one by one.

LangChain [22] employs three strategies when dealing with retrieved context that exceeds the maximum context length of LLMs. One of these strategies is referred to as *Map Rerank*, in which LLMs are required to independently output answers for each segment, along with a confidence score. The answer with the highest confidence score is selected as the final output. For a more nuanced approach, CogLTX [47] introduces a multi-step reasoning mechanism known as *MemRecall*. During each reasoning step, two models are sequentially employed to score the context segments in a coarse-to-fine manner. The top-k segments with the highest scores are added to the final candidate queue, while the remaining segments are deferred to the next reasoning step until the candidate queue is filled. In contrast, LoBART [109] utilizes the ROUGE-2 score [55] to select the top-k contexts during training. For inference, it trains an additional Hierarchical RNN model [33, 38] to generate surrogate priority scores for context selection.

6.2 Context Aggregation

In contrast to the selection-based methods, approaches of this nature take into account the contributions of all context segments to the final answer, rather than selecting just one. Initially, these methods extract relevant information from each segment individually. Subsequently, they employ various fusion strategies to aggregate the retrieved information, ultimately arriving at the final answer. Notably, these approaches exhibit variations in two key aspects. The first pertains to the manner in which information is extracted from each segment, while the second revolves around the diverse fusion strategies employed to integrate information across all segments.

In the context of Encoder-Decoder architecture LLMs like T5 and BART, there exists a category of methods known as Fusion-in-Decoder (FiD) [72]. These methods leverage both the encoder to extract information in the form of embedded hidden states and the decoder to attend to all contextualized representations in order to generate the final output. To illustrate, consider the example of SLED [71]. In SLED, the process of creating a contextualized representation involves overlapping a small portion of each segment with the neighboring segments, effectively forming what can be termed as *context paddings*. Subsequently, each segment is independently encoded through the Encoder, and the resulting embeddings are concatenated to generate embeddings for the entire extended document, with the exclusion of the context paddings. Finally, the Decoder integrates these locally contextualized embeddings through cross-attention, achieving a coherent fusion of information.

For decoder-only LLMs, LangChain [22] introduces two additional aggregation techniques, in addition to the selection strategy known as *Map ReRank*. The first of these is referred to as *Map Reduce*, which involves the simultaneous processing of each segment to obtain answers in parallel. These answers are then passed on to another LLM, which synthesizes them into a final summary. In contrast, the second approach, named *Refine*, operates by progressively refining answers throughout the processing of each segment. In this strategy, the answers obtained from the previous segments are cascaded with the current segment, serving as the prompt for further refinement. This iterative refinement process continues until the final segment is processed.

In addition to LangChain, another recent method, PCW [141, 63], employs a similar approach for handling long-context inputs. PCW partitions the extended context into multiple smaller context windows, each with a maximum length denoted as $C := L - T$, with L representing the total length of the context and T being the length of the task-related tokens in the query, along with the maximum number of new tokens to be generated. Within each context window, tokens attend to each other in parallel, with their position indices isolated within the range of $[0, C-1]$. Subsequently, the task-related tokens, which include the last context position indices within the range $[C, L-1]$, attend to all the context tokens. This process aggregates the parallel information from each context window and fuses it to generate the final answer.

Similarly, another approach proposed by *Su* is NBCE [156], which treats the parallel context windows as a series of independent conditions denoted as $C_1, C_2, \ldots, C_n$, aiming to approximate the logarithmic posterior probability $\log P(T|C_1, C_2, \ldots, C_n)$ for the tokens to be generated. To achieve this, *Su* leverages the classic *Naive Bayes* algorithm [182] to simplify the formulation. As deduced in Eq.36, $P(T|C_i)$ represents the *likelihood* conditioned on the i-th context window, $P(T)$ represents the *prior*, and *const* is a constant that solely depends on $C_1, C_2, \ldots, C_n$. The computation of this formula is straightforward using LLMs, provided that access to their logits is available. Furthermore, *Su* extends this formulation to a more general case, as expressed in Eq.37, introducing a hyper-parameter μ and a pooling operation denoted as *pool*, which can be either *average* or *max*. In this extended form, the original *Naive Bayes* formula can be considered as a specific instance where μ is set to $n-1$ and *pool* is defined as *average*.

$$\log P(T|C_1, .., C_n) \underset{NB}{=\!=} \log \left[\frac{P(T) \prod_{i=1}^{n} \frac{P(T|C_i)P(C_i)}{P(T)}}{P(C_1, .., C_n)} \right] = \sum_{i=1}^{n} \log P(T|C_i) - (n-1) \log P(T) + const \tag{36}$$

$$\Rightarrow (\mu + 1)\text{pool}[\log P(T|C_i)] - \mu \log P(T) + const \tag{37}$$

NBCE and PCW can be seamlessly applied to any readily available open-access LLMs, allowing for significant extensions of the context length. However, it's important to note that both methods operate under the assumption that the relationships among these context windows are negligible and can be treated uniformly in an unordered fashion. Consequently, their performance may suffer when confronted with tightly interconnected context windows exhibiting sequential relations or when dealing with an excessive number of windows to process in parallel.

7 Miscellaneous

This section offers a concise overview of miscellaneous solutions that extend the previous four categories discussed, providing a broader perspective on advancing effective context window of LLMs or the efficiency when utilizing off-the-shelf LLMs. It is important to note that the literature presented here may not be exhaustive or tailored for Transformer-based models. Indeed, many of these techniques are applicable universally to any model equipped with deep neural networks, though they are particularly crucial for large-scale LLMs. Therefore, we hope this section serves as a valuable guide to these diverse directions.

Specific Objectives. In contrast to the conventional pre-training objectives such as MLM or CLM discussed in Sec.2.1, recent research explores tailored approaches to adapt pre-training for specific tasks, aiming to enhance LLMs' efficacy in capturing intricate long-range dependencies and discourse structures in longer texts compared to shorter ones [49]. These approaches involve alternative loss functions, additional terms, and specialized pre-processing techniques. For instance, XLNet [192] introduces a permutation objective that excels in various NLP tasks. ERNIE-Doc [48] extends this approach to long documents with a *Segment-Reordering Objective* to model long-range relationships. DANCE [57] employs a *divide-and-conquer* pre-processing strategy for summarization tasks, breaking the long document and its summary into multiple source-target pairs. PEGASUS [205] introduces the Gap Sentence Generation (GSG) objective for abstractive summarization, while PRIMERA [189] extends it across multi-documents using the *Entity Pyramid* method.

Mixture of Experts. The concept of Mixture of Experts (MoE) [73, 151] stands as a powerful augmentation for giant LLMs, by substituting the dense FFN layer with a MoE layer, which incorporates multiple specialized *experts*. Each expert excels in handling specific input types or tasks, and a dynamic *gating mechanism* arranges the selection of the most suitable expert for a given input. This approach can be implemented in various ways, including employing expert modules optimized for specific tasks [73], utilizing sparse activation [77, 151] and sharding to multiple devices [96], and adapting mixture weights through training to determine the contribution of each expert, as seen in Soft MoE [130]. Then, routing mechanisms are responsible for selecting the top-k experts for each token based on their gate values, where $k > 1$ determines the number of experts [151]. However, in Switch Transformer [54], they have found that setting $k = 1$, referred as *Switch Routing*, can preserve model quality while reducing routing computation. Finally, the output is obtained through a weighted summation of the contributions from these selected experts. MoE techniques can significantly enhance versatility, reduce computational demands, and elevate the efficiency and effectiveness of modeling large-scale contexts.

Parallelism. Leveraging modern aggregated GPU memory within and across nodes, recent research has introduced various parallelism strategies to scale up model sizes and extend sequence length. We summarize commonly-used parallelism paradigms with brief introductions as follows:

a) *Data Parallelism* [99], widely integrated into PyTorch [70], is the most commonly-used way to accelerate training in a distributed manner across multiple devices. It replicates the model on each device to generate gradients independently and communicates them at each iteration to maintain consistency.

b) *Tensor Parallelism* [154] introduces tensor splitting, where individual layers of the model are horizontally partitioned over multiple devices.

c) *Pipeline Parallelism* [69, 64] splits the model layers vertically along the batch dimension into different partitions of micro-batches on separate devices. Each device processes one micro-batch received from the previous one in a pipeline fashion.

d) *Sequence Parallelism* [100] divides the input sequence into multiple chunks and feeds each chunk into its corresponding device. It incorporates ring-style communication for computing the attention output.

e) *Expert Parallelism* [54], as discussed earlier in MoE, places different experts on different GPUs and executes them in parallel. Classic *all-to-all* communication primitives are often used to implement this form of parallelism [139].

Note that, the combination of **a)**,**b)**,**c)** is often referred to as *3D parallelism* [169], which efficiently scales models to trillions of parameters. However, 3D parallelism is primarily designed for larger model sizes rather than longer sequences. To train large-scale models with long sequences, integrating **a)**,**b)**,**c)** and **d)** into *4D parallelism* is a promising approach. Additionally, memory optimization strategies like ZeRO [140, 178] can further enhance efficiency by reducing redundancies in conventional data/tensor parallelism.

Alternate Memory-Efficient Inference. Other approaches have been developed to enhance memory efficiency in large-scale Transformer-based LLMs during inference. These methods encompass weight pruning [112],

weight factorization [92], weight quantization [200], weight partitioning [128] and knowledge distillation [181]. Among these, quantization strategies are particularly vital for practical deployment of massive LLMs, as they reduce parameter precision to alleviate memory demands and accelerate inference. Additionally, there are simpler strategies to alleviate the large KV cache during inference like Multi-Query Attention (MQA) [150] and Grouped-Query Attention (GQA) [3]. In particular, they reduce the number of heads for keys and values, and share them equally across multiple query heads. These approaches have already been applied into state-of-the-art LLMs such as GLM and PaLM, to the best of our knowledge.

8 Evaluation Necessity & Optimization Toolkit

In this section, we have explored the evaluation necessities commonly used for assessing the long-context capabilities of LLMs, which include datasets, metrics, and baseline models. Moreover, we have also investigated several popular optimization toolkits, such as libraries, systems, and compilers, aimed at enhancing the efficiency and effectiveness of LLMs throughout their development stages. For a concise and clear presentation, we have organized detailed information in corresponding tables located in the appendix A, B, C, D. Here, we provide an overview of these tables and their content.

Datasets. This part presents a summary of the evaluation datasets used to assess the long-context capabilities of LLMs. Detailed dataset information, including language, task types, length statistics, quality, splits, file size, sample count, and format, is provided in the Tab.1 located in Appendix A. We illustrate the meta information about the table as follows:

- *Language*: Language information for each dataset is represented using abbreviations, such as *en* for English, *zh* for Chinese, and *code* for programming languages. Multiple languages within a dataset are concatenated using "+".
- *Task Amount and Types*: This property defines each dataset's tasks and divides them into 10 categories, including language modeling (*LM*), multi-choice question-answering (*MCQA*), extractive question-answering (*ExtQA*), summarization (*Summ*), text classification (*Class*), reasoning tasks (*Math*), and natural language generation (*Cloze, Gen*[2]).
- *Length*: The average (*avg*), minimum (*min*), and maximum (*max*) sample lengths are provided in kilo "words"[3] for each dataset, where "words" are defined based on sample content[4].
- *Quality*: Quality assessment is simply based on two sub-dimensions: *Human Labeled* (labels generated by humans) and *Model Gen* (prompts or labels generated by LLMs).
- *Splits*: This indicates how datasets are partitioned, including conventional triple-split formats like *train/test/val*, a single *test* split only for evaluation purposes such as LongBench [10].
- *Size and Count*: These two provide statistics on file size (in bytes, B) and sample count for each split.
- *Format*: It tags the file format of samples, including common formats like *jsonl*, *json*, *csv*, *txt*, and others.

Metrics. This section provides a concise summary of 9 categories of general evaluation metrics commonly used across the 10 NLP task types categorized earlier. The detailed metrics for each task are presented in a Tab.2 located in Appendix B. Below, we offer a brief introduction to these metrics.

- *CE/PPL (Cross-Entropy/Perplexity)*: CE is the most commonly-used loss function for pretrained language models, which quantifies the KL divergence between predicted distributions and the true distribution approximated from the large training text corpus. And PPL measures how well a language model predicts a sequence, simply formalized as $\exp(loss)$, where $loss$ denotes the cross-entropy loss for the test set.
- *BPC/BPW (Bits per Character/Bits per Word)*: They measure the average number of bits required to encode characters or words, simply calculated by $\text{avg}_{\text{T}}(loss)$, where T is the number of characters or words respectively. These metrics assess the efficiency for a model to represent or compress text.

[2] *Cloze* is to predict and fill the short blanks in some long contexts like code filling, and *Gen* is more about the tasks to open writing like story generation, whose context may not be that long but the generated part is usually quite extensive.

[3] In the context of our study, "words" are approximately considered to be separated by spaces in English and code, while individual Chinese characters are treated as words.

[4] For example, if one typical sample has the prompt template like *"Read this {context}, and answer the question below: {question}"*, we will calculate the number of words in both context and question part, ignoring the fixed remaining part in the template.

- *Acc/F1 (Accuracy/F1 Score)*: Accuracy measures the proportion of correct predictions in tasks with objective answers like classification, MCQA, while the F1 Score balances precision and recall for imbalanced datasets.
- *EM (Exact Matching)*: It evaluates whether a generated sequence matches the reference exactly, used in tasks that require absolute precision like code auto-completion.
- *ROUGE-1/-2/-L* [55]: ROUGE scores assess the similarity between text pairs by comparing n-grams overlapping, typically setting n=1 (unigram), 2 (bigram), and L (longest). They are widely used in tasks EM may fail, such as summarization.
- *BLEU/METEOR/TER* [2]: These metrics are specific in machine translation tasks. BLEU measures the overlap of generated and reference translations based on n-grams. METEOR evaluates translation quality by considering various linguistic factors. TER quantifies the edit distance between the generated and reference translations.
- *EntMent (Entity Mention)* [39]: It evaluates the coverage and correctness of important entities, such as people, organizations, locations, etc, mentioned in the generated text, like the summary in the summarization task.
- *Pass@k*: This metric evaluates whether a generated answer ranks within the top-k answers provided by a model, commonly used in code generation and some math tasks with multiple possible solutions.
- *Human/Model Judge*: This involves human or power models like GPT-4 to score text quality based on fluency, coherence, and other subjective criteria suitable for tasks like story generation.

Baselines. In this part, we gather pretrained or fine-tuned LLMs frequently used in the literature to serve as baselines for assessing long-context capabilities in some downstream tasks. We provide an overview of these models about their basic information (name, base architecture, L_{max}, parameter sizes), special features (training techs and fine-tuning task), and available links for publication (huggingface, GitHub, homepage, and paper) to Tab.3 in Appendix C to facilitate researchers' convenience.

Toolkits In this section, we present a collection of valuable toolkits, encompassing libraries/packages, compilers, systems/frameworks, etc, designed to enhance the efficiency and effectiveness of LLMs throughout their development life-cycle. For a quick overview of these toolkits, please refer to Tab.4 provided in Appendix D.

9 Discussion

While significant progress has been made in this field as we discussed in sections 3, 4, 5, 6, several challenges still remain. Therefore, in this section, we delve into the these pivotal challenges and propose potential directions for future research and development in the context of enhancing long-context capabilities within Transformer-based LLMs, with a specific focus on architectural enhancements.

Attention Trade-off. As discussed in Sec.3, the efficient attention methods we have explored often involve a delicate trade-off between maintaining full-scale attention dependencies (such as local attention) or achieving higher attention score precision (e.g., through approximated attention) to mitigate the computational demands of standard attention kernels. However, with longer contexts, the discourse structure and interrelated information become increasingly complex, necessitating the ability to capture global, long-range dependencies while preserving precise relevancy. Addressing this challenge requires finding an optimal equilibrium between computational efficiency and retaining as much precision in attention patterns as possible. Thus it remains an ongoing pursuit in the field of long-context LLMs. Fortunately, some recent innovations, such as Flash Attention [42, 41], explore IO-aware solutions beyond algorithmic level, which extremely improve efficiency in both runtime and memory overhead without any loss of attention precision. This is an exhilarating potential avenue to tackle this issue in practical applications. ~~And~~ furthermore, one can explore the integration of preceding efficient strategies with these drop-in replacements, leveraging powerful GPU kernel programming tools like Cuda, or the more light-weighted Triton [171], as exemplified in SCFA [124].

Memory Effectiveness and Efficiency. As discussed earlier in Sec.2.1, 2.2, we have outlined the limitations arising from the absence of an explicit memory mechanism, relying solely on in-context working memory, as well as the significant increase in KV cache memory consumption during extended context interactions. These challenges collectively underscore the need for more effective and efficient memory mechanisms within the realm of Transformer-based LLMs. While we introduced various long-term memory mechanisms in Sec.4, they are constrained by the additional memory overhead introduced by their intricate heuristic design, thus leading to potential performance degradation over time due to frequent refreshment. To tackle this challenge, researchers can investigate more

efficient strategies for organizing memory storage and enhancing read/write throughput, drawing inspiration from recent advancements such as Paged Attention [90].

Length Extrapolation Mining. In Section 5, we conduct a thorough analysis of the challenges associated with length extrapolation in Transformer-based models, with a primary focus on the prevalent design of positional embeddings. And we offer a holistic overview of the very recent breakthroughs, particularly the extended strategies applied on the RoPE [14, 52, 13, 126, 159, 24], which, as we believe, holds significant promise in addressing the extrapolation limitation. However, it's important to note that these advancements often rely on simplified observations of complex high-dimensional positional embedding properties and incorporate straightforward heuristic adjustments. This prompts us to question the theoretical foundations of modeling sequentiality using high-dimensional embeddings and to explore the potential resurgence of learnable embeddings, guided by these heuristic designs with many hyperparameters to tune. We believe that future research should delve deeper into this area, as exemplified in CLEX [24], especially in terms of developing a robust theoretical framework for modeling sequentiality under Transformer settings.

Specific yet Universal Objective. While we have discussed specific objectives tailored for long-text modeling, it is worth noting that many of them are limited to certain types of tasks or are only compatible with the MLM objective rather than the more common CLM objective nowadays. This highlights the need for specific yet universally applicable causal language modeling objectives that can effectively capture long-range dependencies from the early stages of model training. This could potentially be achieved by aligning such objectives with the mining of an effective PE scheme, as mentioned earlier.

Reliable Metric Demand. Speaking of evaluation metrics, we have investigated many options in Sec.8. However, drawing from our prior experience in evaluation, it is evident that commonly-used metrics such as ROUGE scores often exhibit significant disparities when compared to human judgment scores, which can be seen as the oracle. With the rapid deployment of LLMs in real-world scenarios, there is an increasingly pressing need for more dependable metrics to assess long-context capabilities, particularly in generative tasks where precise ground truth is elusive. One promising avenue involves leveraging the robustness of state-of-the-art LLMs like GPT4 to serve as substitutes for human judges, although the associated high costs continue to pose challenges for wider adoption within the research community.

By addressing these challenges and exploring future potential solutions, researchers can pave the way for more capable LLMs that excel in understanding and processing long-context information, opening up new opportunities across various real-world applications.

10 Conclusion

In this survey, we comprehensively navigate the landscape of architectural advancement in Transformer-based LLMs, to enhance the capabilities handling extensive context windows across various development stages, with a holistic taxonomy that categorizes the these methodologies targeting different module designs in Transformer. Then we explore evaluation necessities specific for long-text tasks and some optimization toolkits that integrate many tools to augment LLMs' efficiency and efficacy. We further identify key challenges with corresponding future directions. In addition, our repository ensures that readers stay updated with the latest research in this dynamic field. As LLMs continue to evolve rapidly, we sincerely hope our survey serve as a valuable resource for researchers seeking to harness their power in building powerful long-context LLMs, ultimately advancing the pursuit of the era of AGI.

Acknowledgement

We would like to express our gratitude to Zenan Li and Hao Gao for their helpful discussions and feedback during the early stages of this paper. We also extend our appreciation to our Baidu colleagues, including Hao Chen, Linyun Liu, PengHao Zhao, and others, for their contributions and insights in the initial research phases.

Additionally, we acknowledge the generous support from the Baidu AI Cloud Group (ACG). We are especially thankful to Dou Shen, the Executive Vice President and Head of ACG, for the great idea and gracious invitation of the first session of Baidu ACG Summer Camp. This opportunity has been instrumental in shaping our research and providing valuable experiences.

References

[1] Alaa Abd-Alrazaq, Rawan AlSaad, Dari Alhuwail, Arfan Ahmed, Padraig Mark Healy, Syed Latifi, Sarah Aziz, Rafat Damseh, Sadam Alabed Alrazak, Javaid Sheikh, et al. Large language models in medical education: Opportunities, challenges, and future directions. *JMIR Medical Education*, 9(1):e48291, 2023.

[2] Abhaya Agarwal and Alon Lavie. Meteor, m-bleu and m-ter: Evaluation metrics for high-correlation with human rankings of machine translation output. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 115–118, 2008.

[3] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

[4] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. Etc: Encoding long and structured inputs in transformers. *arXiv preprint arXiv:2004.08483*, 2020.

[5] Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556, 2022.

[6] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[9] He Bai, Peng Shi, Jimmy Lin, Yuqing Xie, Luchen Tan, Kun Xiong, Wen Gao, and Ming Li. Segatron: Segment-aware transformer for language modeling and understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12526–12534, 2021.

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

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

[12] Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R Gormley. Unlimiformer: Long-range transformers with unlimited length input. *arXiv preprint arXiv:2305.01625*, 2023.

[13] bloc97. Add ntk-aware interpolation "by parts" correction, 2023. https://github.com/jquesnelle/yarn/pull/1, Jul 2023.

[14] bloc97. Ntk-aware scaled rope allows llama models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation. https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have/, Jun 2023.

[15] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR, 2022.

[16] Philip J Brown and James V Zidek. Adaptive multivariate ridge regression. *The Annals of Statistics*, 8(1):64–74, 1980.

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

[18] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023.

[19] Aydar Bulatov, Yuri Kuratov, and Mikhail S Burtsev. Scaling transformer to 1m tokens and beyond with rmt. *arXiv preprint arXiv:2304.11062*, 2023.

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

[21] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):1–37, 2011.

[22] Harrison Chase. Langchain. https://github.com/langchain-ai/langchain, 10 2022.

[23] Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention approximation. *arXiv preprint arXiv:2110.15343*, 2021.

[24] Guanzheng Chen, Xin Li, Zaiqiao Meng, Shangsong Liang, and Lidong Bing. Clex: Continuous length extrapolation for large language models. *arXiv preprint arXiv:2310.16450*, 2023.

[25] Peng Chen. Permuteformer: Efficient relative position encoding for long sequences. *arXiv preprint arXiv:2109.02377*, 2021.

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

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

[28] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[29] Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2023.

[30] Ta-Chung Chi, Ting-Han Fan, Peter J Ramadge, and Alexander Rudnicky. Kerple: Kernelized relative positional embedding for length extrapolation. *Advances in Neural Information Processing Systems*, 35:8386–8399, 2022.

[31] Ta-Chung Chi, Ting-Han Fan, Alexander Rudnicky, and Peter Ramadge. Dissecting transformer length extrapolation via the lens of receptive field analysis. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13522–13537, 2023.

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

[33] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[34] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[35] Krzysztof M Choromanski, Mark Rowland, and Adrian Weller. The unreasonable effectiveness of structured random orthogonal embeddings. *Advances in neural information processing systems*, 30, 2017.

[36] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[37] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[38] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. *arXiv preprint arXiv:1804.05685*, 2018.

[39] Hongliang Dai, Siliang Tang, Fei Wu, and Yueting Zhuang. Entity mention aware document representation. *Information Sciences*, 430:216–227, 2018.

[40] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[41] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[42] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[43] Donald Davidson. The logical form of action sentences. In Nicholas Rescher, editor, *The Logic of Decision and Action*, pages 81–120. Univ. of Pittsburgh Press, 1967.

[44] Peter J Denning. Virtual memory. *ACM Computing Surveys (CSUR)*, 2(3):153–189, 1970.

[45] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[46] Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, and Furu Wei. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*, 2023.

[47] Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. Cogltx: Applying bert to long texts. *Advances in Neural Information Processing Systems*, 33:12792–12804, 2020.

[48] Siyu Ding, Junyuan Shang, Shuohuan Wang, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. Ernie-doc: A retrospective long-document modeling transformer. *arXiv preprint arXiv:2012.15688*, 2020.

[49] Zican Dong, Tianyi Tang, Lunyi Li, and Wayne Xin Zhao. A survey on long text modeling with transformers, 2023.

[50] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. Glm: General language model pretraining with autoregressive blank infilling. *arXiv preprint arXiv:2103.10360*, 2021.

[51] Wafaa S El-Kassas, Cherif R Salama, Ahmed A Rafea, and Hoda K Mohamed. Automatic text summarization: A comprehensive survey. *Expert systems with applications*, 165:113679, 2021.

[52] emozilla. Dynamically scaled rope further increases performance of long context llama with zero fine-tuning. https://www.reddit.com/r/LocalLLaMA/comments/14mrgpr/dynamically_scaled_rope_further_increases/, Jun 2023.

[53] Angela Fan, Thibaut Lavril, Edouard Grave, Armand Joulin, and Sainbayar Sukhbaatar. Addressing some limitations of transformers with feedback memory. *arXiv preprint arXiv:2002.09402*, 2020.

[54] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022.

[55] Kavita Ganesan. Rouge 2.0: Updated and improved measures for evaluation of summarization tasks. *arXiv preprint arXiv:1803.01937*, 2018.

[56] gante. Llama/gptneox: add rope scaling. https://github.com/huggingface/transformers/pull/24653, Jul 2023.

[57] Alexios Gidiotis and Grigorios Tsoumakas. A divide-and-conquer approach to the summarization of long documents. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:3029–3040, 2020.

[58] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[59] Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. Longt5: Efficient text-to-text transformer for long sequences. *arXiv preprint arXiv:2112.07916*, 2021.

[60] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. Star-transformer. *arXiv preprint arXiv:1902.09113*, 2019.

[61] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Realm: Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR, 2020.

[62] Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*, 2023.

[63] Yaru Hao, Yutao Sun, Li Dong, Zhixiong Han, Yuxian Gu, and Furu Wei. Structured prompting: Scaling in-context learning to 1,000 examples, 2022.

[64] Aaron Harlap, Deepak Narayanan, Amar Phanishayee, Vivek Seshadri, Nikhil Devanur, Greg Ganger, and Phil Gibbons. Pipedream: Fast and efficient pipeline parallel dnn training. *arXiv preprint arXiv:1806.03377*, 2018.

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[66] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[67] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17853–17862, 2023.

[68] Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization, 2021.

[69] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

[70] Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. Pytorch. *Programming with TensorFlow: Solution for Edge Computing Applications*, pages 87–104, 2021.

[71] Maor Ivgi, Uri Shaham, and Jonathan Berant. Efficient long-text understanding with short-text models. *Transactions of the Association for Computational Linguistics*, 11:284–299, 2023.

[72] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*, 2020.

[73] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[74] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[75] Omid Jafari, Preeti Maurya, Parth Nagarkar, Khandker Mushfiqul Islam, and Chidambaram Crushev. A survey on locality sensitive hashing algorithms and their applications. *arXiv preprint arXiv:2102.08942*, 2021.

[76] Hanhwi Jang, Joonsung Kim, Jae-Eon Jo, Jaewon Lee, and Jangwoo Kim. Mnnfast: A fast and scalable system architecture for memory-augmented neural networks. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 250–263, 2019.

[77] Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Lukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. Sparse is enough in scaling transformers. *Advances in Neural Information Processing Systems*, 34:9895–9907, 2021.

[78] Zhe Jia, Marco Maggioni, Benjamin Staiger, and Daniele P Scarpazza. Dissecting the nvidia volta gpu architecture via microbenchmarking. *arXiv preprint arXiv:1804.06826*, 2018.

[79] JianxinMa. Introducing qwen-7b: Open foundation and human-aligned models (of the state-of-the-arts). https://github.com/QwenLM/Qwen/blob/main/tech_memo.md, Aug 2023.

[80] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[81] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

[82] Tom Kilburn, David BG Edwards, Michael J Lanigan, and Frank H Sumner. One-level storage system. *IRE Transactions on Electronic Computers*, pages 223–235, 1962.

[83] Jin K Kim, Michael Chua, Mandy Rickard, and Armando Lorenzo. Chatgpt and large language model (llm) chatbots: the current state of acceptability and a proposal for guidelines on utilization in academic medicine. *Journal of Pediatric Urology*, 2023.

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

[85] Shun Kiyono, Sosuke Kobayashi, Jun Suzuki, and Kentaro Inui. Shape: Shifted absolute position embedding for transformers. *arXiv preprint arXiv:2109.05644*, 2021.

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

[87] Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge, 2017.

[88] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009.

[89] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[90] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. *arXiv preprint arXiv:2309.06180*, 2023.

[91] Brandon Kynoch and Hugo Latapie. Recallm: An architecture for temporal context understanding and question answering. *arXiv preprint arXiv:2307.02738*, 2023.

[92] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[93] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[94] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[95] Gibbeum Lee, Volker Hartmann, Jongho Park, Dimitris Papailiopoulos, and Kangwook Lee. Prompted llms as chatbot modules for long open-domain conversation. *arXiv preprint arXiv:2305.04533*, 2023.

[96] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

[97] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[98] Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. Pretrained language models for text generation: A survey, 2022.

[99] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, et al. Pytorch distributed: Experiences on accelerating data parallel training. *arXiv preprint arXiv:2006.15704*, 2020.

[100] Shenggui Li, Fuzhao Xue, Chaitanya Baranwal, Yongbin Li, and Yang You. Sequence parallelism: Long sequence training from system perspective. *arXiv preprint arXiv:2105.13120*, 2021.

[101] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

[102] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 2022.

[103] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[104] Xiaoran Liu, Hang Yan, Shuo Zhang, Chenxin An, Xipeng Qiu, and Dahua Lin. Scaling laws of rope-based extrapolation. *arXiv preprint arXiv:2310.05209*, 2023.

[105] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[106] Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, et al. Codexglue: A machine learning benchmark dataset for code understanding and generation. *arXiv preprint arXiv:2102.04664*, 2021.

[107] Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. Luna: Linear unified nested attention. *Advances in Neural Information Processing Systems*, 34:2441–2453, 2021.

[108] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[109] Potsawee Manakul and Mark JF Gales. Long-span summarization via local attention and content selection. *arXiv preprint arXiv:2105.03801*, 2021.

[110] André Martins, António Farinhas, Marcos Treviso, Vlad Niculae, Pedro Aguiar, and Mario Figueiredo. Sparse and continuous attention mechanisms. *Advances in Neural Information Processing Systems*, 33:20989–21001, 2020.

[111] Pedro Henrique Martins, Zita Marinho, and André FT Martins. ∞-former: Infinite memory transformer. *arXiv preprint arXiv:2109.00301*, 2021.

[112] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019.

[113] Lesly Miculicich, Dhananjay Ram, Nikolaos Pappas, and James Henderson. Document-level neural machine translation with hierarchical attention networks. *arXiv preprint arXiv:1809.01576*, 2018.

[114] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010.

[115] Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 234–239. IEEE, 2012.

[116] Maxim Milakov and Natalia Gimelshein. Online normalizer calculation for softmax, 2018.

[117] Silvia Milano, Joshua A McGrane, and Sabina Leonelli. Large language models challenge the future of higher education. *Nature Machine Intelligence*, 5(4):333–334, 2023.

[118] Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. Ret-llm: Towards a general read-write memory for large language models. *arXiv preprint arXiv:2305.14322*, 2023.

[119] Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023.

[120] Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

[121] OpenAI. Openai: Introducing chatgpt. https://openai.com/blog/chatgpt, 2022.

[122] OpenAI. Gpt-4 technical report, 2023.

[123] OpenAI. Openai: Gpt-4, 2023. https://openai.com/research/gpt-4, 2023.

[124] Matteo Pagliardini, Daniele Paliotta, Martin Jaggi, and François Fleuret. Faster causal attention over large sequences through sparse flash attention. *arXiv preprint arXiv:2306.01160*, 2023.

[125] Arka Pal, Deep Karkhanis, Manley Roberts, Samuel Dooley, Arvind Sundararajan, and Siddartha Naidu. Giraffe: Adventures in expanding context lengths in llms. *arXiv preprint arXiv:2308.10882*, 2023.

[126] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.

[127] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. *arXiv preprint arXiv:2103.02143*, 2021.

[128] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5, 2023.

[129] Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

[130] Joan Puigcerver, Carlos Riquelme, Basil Mustafa, and Neil Houlsby. From sparse to soft mixtures of experts. *arXiv preprint arXiv:2308.00951*, 2023.

[131] Pytorch. Pytorch 2.0. https://pytorch.org/get-started/pytorch-2.0/, 2023.

[132] Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. *arXiv preprint arXiv:1911.02972*, 2019.

[133] Markus N Rabe and Charles Staats. Self-attention does not need o(n2) memory. *arXiv preprint arXiv:2112.05682*, 2021.

[134] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[135] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

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

[137] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[138] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[139] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International Conference on Machine Learning*, pages 18332–18346. PMLR, 2022.

[140] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.

[141] Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. Parallel context windows for large language models, 2023.

[142] Hongyu Ren, Hanjun Dai, Zihang Dai, Mengjiao Yang, Jure Leskovec, Dale Schuurmans, and Bo Dai. Combiner: Full attention transformer with sparse computation cost. *Advances in Neural Information Processing Systems*, 34:22470–22482, 2021.

[143] Tobias Rohde, Xiaoxia Wu, and Yinhan Liu. Hierarchical learning for generation with long source sequences. *arXiv preprint arXiv:2104.07545*, 2021.

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

[145] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

[146] Jürgen Rudolph, Shannon Tan, and Samson Tan. War of the chatbots: Bard, bing chat, chatgpt, ernie and beyond. the new ai gold rush and its impact on higher education. *Journal of Applied Learning and Teaching*, 6(1), 2023.

[147] Anian Ruoss, Grégoire Delétang, Tim Genewein, Jordi Grau-Moya, Róbert Csordás, Mehdi Bennani, Shane Legg, and Joel Veness. Randomized positional encodings boost length generalization of transformers. *arXiv preprint arXiv:2305.16843*, 2023.

[148] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[149] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[150] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

[151] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[152] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. Bi-directional block self-attention for fast and memory-efficient sequence modeling. *arXiv preprint arXiv:1804.00857*, 2018.

[153] Han Shi, Jiahui Gao, Xiaozhe Ren, Hang Xu, Xiaodan Liang, Zhenguo Li, and James Tin-Yau Kwok. Sparsebert: Rethinking the importance analysis in self-attention. In *International Conference on Machine Learning*, pages 9547–9557. PMLR, 2021.

[154] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020.

[155] Nimit S Sohoni, Christopher R Aberger, Megan Leszczynski, Jian Zhang, and Christopher Ré. Low-memory neural network training: A technical report. *arXiv preprint arXiv:1904.10631*, 2019.

[156] Jianlin Su. Nbce: Handling length in context expansion of llm with naive bayes. https://spaces.ac.cn/archives/9617, May 2023.

[157] Jianlin Su. Transformer upgrade roadmap:10. rope is a beta-base encoding. https://spaces.ac.cn/archives/9675, Jul 2023.

[158] Jianlin Su. Transformer upgrade roadmap:11. taking beta-base encoding to the limit. https://spaces.ac.cn/archives/9706, Jul 2023.

[159] Jianlin Su. Transformer upgrade roadmap:12. rerope for infinite extrapolation? https://spaces.ac.cn/archives/9708, Aug 2023.

[160] Jianlin Su. Transformer upgrade roadmap:7. length extrapolation and local attention. https://spaces.ac.cn/archives/9431, Jan 2023.

[161] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

[162] Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers. *arXiv preprint arXiv:1905.07799*, 2019.

[163] Sainbayar Sukhbaatar, Da Ju, Spencer Poff, Stephen Roller, Arthur Szlam, Jason Weston, and Angela Fan. Not all memories are created equal: Learning to forget by expiring. In *International Conference on Machine Learning*, pages 9902–9912. PMLR, 2021.

[164] Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. A length-extrapolatable transformer. *arXiv preprint arXiv:2212.10554*, 2022.

[165] Mingxu Tao, Yansong Feng, and Dongyan Zhao. A frustratingly easy improvement for position embeddings via random padding. *arXiv preprint arXiv:2305.04859*, 2023.

[166] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

[167] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pages 9438–9447. PMLR, 2020.

[168] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey, 2022.

[169] DeepSpeed Team and Rangan Majumder. Deepspeed: Extreme-scale model training for everyone, 2020.

[170] M Therasa and G Mathivanan. Survey of machine reading comprehension models and its evaluation metrics. In *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, pages 1006–1013. IEEE, 2022.

[171] Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19, 2019.

[172] Oguzhan Topsakal and Tahir Cetin Akinci. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences*, volume 1, pages 1050–1056, 2023.

[173] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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

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

[176] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

[177] Patrick von Platen. How to generate text: using different decoding methods for language generation with transformers. https://huggingface.co/blog/how-to-generate, March 2020.

[178] Guanhua Wang, Heyang Qin, Sam Ade Jacobs, Connor Holmes, Samyam Rajbhandari, Olatunji Ruwase, Feng Yan, Lei Yang, and Yuxiong He. Zero++: Extremely efficient collective communication for giant model training. *arXiv preprint arXiv:2306.10209*, 2023.

[179] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[180] Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *arXiv preprint arXiv:2306.07174*, 2023.

[181] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788, 2020.

[182] Geoffrey I Webb, Eamonn Keogh, and Risto Miikkulainen. Naive bayes. *Encyclopedia of machine learning*, 15(1):713–714, 2010.

[183] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022.

[184] Michel Wermelinger. Using github copilot to solve simple programming problems. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, pages 172–178, 2023.

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

[186] Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. *arXiv preprint arXiv:2203.08913*, 2022.

[187] Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite transformer with long-short range attention. *arXiv preprint arXiv:2004.11886*, 2020.

[188] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[189] Wen Xiao, Iz Beltagy, Giuseppe Carenini, and Arman Cohan. Primera: Pyramid-based masked sentence pre-training for multi-document summarization. *arXiv preprint arXiv:2110.08499*, 2021.

[190] Chang Xu, Steven R Kirk, and Samantha Jenkins. Tiling for performance tuning on different models of gpus. In *2009 Second International Symposium on Information Science and Engineering*, pages 500–504. IEEE, 2009.

[191] Baosong Yang, Longyue Wang, Derek F Wong, Shuming Shi, and Zhaopeng Tu. Context-aware self-attention networks for natural language processing. *Neurocomputing*, 458:157–169, 2021.

[192] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[193] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018.

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

[195] Zihao Ye, Qipeng Guo, Quan Gan, Xipeng Qiu, and Zheng Zhang. Bp-transformer: Modelling long-range context via binary partitioning. *arXiv preprint arXiv:1911.04070*, 2019.

[196] Burak Yetiştiren, Işık Özsoy, Miray Ayerdem, and Eray Tüzün. Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt. *arXiv preprint arXiv:2304.10778*, 2023.

[197] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[198] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[199] Chulhee Yun, Yin-Wen Chang, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. O (n) connections are expressive enough: Universal approximability of sparse transformers. *Advances in Neural Information Processing Systems*, 33:13783–13794, 2020.

[200] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pages 36–39. IEEE, 2019.

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

[202] Yury Zemlyanskiy, Joshua Ainslie, Michiel de Jong, Philip Pham, Ilya Eckstein, and Fei Sha. Readtwice: Reading very large documents with memories. *arXiv preprint arXiv:2105.04241*, 2021.

[203] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022.

[204] Haopeng Zhang, Xiao Liu, and Jiawei Zhang. Hegel: Hypergraph transformer for long document summarization. *arXiv preprint arXiv:2210.04126*, 2022.

[205] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR, 2020.

[206] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253, 2018.

[207] Xingxing Zhang, Furu Wei, and Ming Zhou. Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization. *arXiv preprint arXiv:1905.06566*, 2019.

[208] Wanjun Zhong, Lianghong Guo, Qiqi Gao, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. *arXiv preprint arXiv:2305.10250*, 2023.

[209] Wangchunshu Zhou, Yuchen Eleanor Jiang, Peng Cui, Tiannan Wang, Zhenxin Xiao, Yifan Hou, Ryan Cotterell, and Mrinmaya Sachan. Recurrentgpt: Interactive generation of (arbitrarily) long text. *arXiv preprint arXiv:2305.13304*, 2023.

[210] Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. Pose: Efficient context window extension of llms via positional skip-wise training. *arXiv preprint arXiv:2309.10400*, 2023.

[211] Zhenhai Zhu and Radu Soricut. H-transformer-1d: Fast one-dimensional hierarchical attention for sequences. *arXiv preprint arXiv:2107.11906*, 2021.

[212] Atabay Ziyaden, Amir Yelenov, and Alexandr Pak. Long-context transformers: A survey. In *2021 5th Scientific School Dynamics of Complex Networks and their Applications (DCNA)*, pages 215–218, 2021.

[213] Simiao Zuo, Xiaodong Liu, Jian Jiao, Denis Charles, Eren Manavoglu, Tuo Zhao, and Jianfeng Gao. Efficient long sequence modeling via state space augmented transformer. *arXiv preprint arXiv:2212.08136*, 2022.

A Datasets

Table 1: Fundamental information about existing datasets specific for long-text NLP tasks.

Dataset	Language	Task Amount	Task Types										Lengths (kilo words)			Quality		Splits	Size (B)	Count	Format
			LM	*MCQA*	*ExtQA*	*Summ*	*Class*	*Match*	*Math*	*Cloze*	*Gen*	*MT*	*Avg*	*Min*	*Max*	*Human Labeled*	*Model Gen*				
ChapterBreak	en	1	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	25.4	2.3	405.8	✓	✗	train	1350M	9.6K	json
HotpotQA	en	1	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	0.9	0.01	2	✓	✗	train/dev	540M/90M	90K/14.8K	json
Nature Questions	en	1	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	9.8	0.2	169.3	✓	✗	train/dev	16.2G/6.3G	307K/7.8K	json
CommonsenseQA	en	1	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.04	0.02	0.09	✓	✗	train/test/dev	3.7M/416K/464K	9.7K/1.2K/1.1K	jsonl
OpenBookQA	en	1	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	0.03	0.02	0.09	✓	✗	train/test/dev	4.8M/220K/224K	11k/500/500	jsonl
WikiHow	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.4	0	12.7	✓	✗	test	215K	591M	csv
Reddit TIFU	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.1	0	0.6	✓	✗	test	80K	640M	jsonl
SummScreen	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	7.3	1.6	24.0	✓	✗	train/test/dev	22.5K/2.1K/2.1K	1.7G	jsonl
CNN/DailyMail	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.8	0	2.9	✓	✗	test	312K	4.6G	txt
NewsRoom	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	0.7	0	178.5	✓	✗	train/test/dev	995K/109K/109K	5.4G	jsonl
Multi-News	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	2.1	0.1	464.2	✓	✗	train/test/val	45K/5.6K/5.6K	686M	txt
ArXiv + PubMed	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	5.2	0	157.3	✓	✗	train/test/val	323K/13.1K/13.1K	19G	jsonl
BigPatent	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	3.2	0.2	83.2	✓	✗	train/test/val	1.2M/67K/67K	13G	json
GovReport	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	43.5	0.2	1386.2	✓	✗	test	19K	1.2G	json
QMSum	en	1	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	10.8	1.7	26.8	✓	✗	train/test/val	162/35/35	17M	jsonl
IMDB	en	1	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	0.2	0	2.5	✓	✗	train/test	25K/25K	490M	txt
20 NewsGroups	en	1	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	0.3	0	11.8	✓	✗	test	20K	91M	txt
THUCnews	zh	1	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	0.9	0	79.5	✓	✗	test	83.6K	4.0G	txt
ECtHR	en	1	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	2.2	0.01	51.3	✓	✗	train/test/dev	116M/43M/24M	7.3k/3k/1.3k	jsonl
AGnews	en	1	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	0.1	0.3	1	✓	✗	train/test	30M/1.8M	120k/7.6k	csv
QQP	en	1	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	0.02	0	0.3	✓	✗	test	404.3K	56M	tsv
CAIL2019-SCM	zh	1	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	2.0	1.8	2.6	✓	✗	train/test/val	5.1K/1.5K/1.5K	43M	jsonl
GSM8k	en	1	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	0.1	0	0.3	✓	✗	train/test	7.5K/1.3K	14M	jsonl
ContractNLI	en	1	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	2.0	0.5	8.7	✓	✗	train/test/dev	423/123/61	11M	json
WikiText103	en	1	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	3.4	0	26.7	✓	✗	train/test/val	29.5K/62/60	517M	txt
DuLeMon	zh	1	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.6	0.3	1.4	✓	✗	train/test/dev	25.4K/1.0K/1.0K	40M	jsonl

Note that: the presence of common dirty data may result in extremely short samples, thus many datasets in the table containing samples with a minimum length approaching zero.

B Metrics

Table 2: The common evaluation metrics corresponding to each specific NLP task type in Tab.1.

Task Types	Metric Types								
	CE/PPL	BPC/BPW	*Acc/F1*	*EM*	*ROUGE-1/-2/-L*	*BLEU/METEOR/TER*	*EntMent*	*Pass@k*	*Human/Model Judge*
LM	✓	✓	✓	✗	✗	✗	✗	✗	✓
MCQA	✗	✗	✓	✗	✗	✗	✗	✓	✗
ExtQA	✗	✗	✓	✓	✓	✓	✓	✗	✓
Summ	✗	✗	✗	✗	✓	✓	✓	✗	✓
Class	✗	✗	✓	✗	✗	✗	✗	✗	✗
Match	✗	✗	✓	✗	✗	✗	✗	✓	✗
Math	✗	✗	✓	✓	✗	✗	✗	✓	✓
Cloze	✗	✗	✓	✓	✓	✓	✓	✓	✓
Gen	✗	✗	✓	✓	✓	✓	✓	✓	✓
MT	✗	✗	✗	✗	✗	✓	✓	✓	✓

Note that: the ✗ in the table does not imply that a specific metric cannot be applied to a task. Rather, it suggests that the metric might be less commonly used or that there could be more suitable alternatives.

C Baselines

Table 3: The basic information and publication links about baseline models when evaluating LLMs on long-context capabilities.

Model Name	Base Model	L_{max} (k)	Parameters (B)	Corpus/Task/Techs		Availability			
				Pre-Training	*Fine-Tuning*	*huggingface*	*github*	*homepage / blog*	paper
GPT-3.5-Turbo-16k	GPT3	16	-	-	-	-	-	here	-
LongChat-v1.5-7B-32k	Llama2	32	7	Condensing RoPE	Curated Conversation Data	here	here	here	-
CodeLLaMA-7B-16k	Llama2	16	7	-	LCFT IFT	here	-	-	here
ChatGLM2-6B-32k	GLM	32	6	Blank Infilling Multi-Task PT	Flash Attention PI MQA	here	here	-	here
ChatGLM3-6B-32k	GLM	32	6	Diverse training dataset More sufficient training steps More reasonable training strategy	New prompt format Function Call Code Interpreter Agent tasks	here	here	-	here
Giraffe-v1-13B-16k	Llama	16	13	-	PI IFT LoRA	here	here	-	here
Giraffe-v2-13B-32k	Llama2	32	13	-	Zero-Shot Linear Scaling	here	here	-	here
Llama-2-7B-32K-Instruct	Llama2	32	7	RedPajama UL2 Oscar 2K Exclusion	IFT BookSum Multi-Doc QA	here			
Vicuna-v1.5-7B-16k	Llama2	16	7	-	ShareGPT	here	here	here	here
MPT-7B-Storywriter	MPT-7B	65	7	ALiBi	BookSum	here	-	here	-
MPT-30B-chat	MPT-30B	8	30	-	Multi-Doc QA	here	-	here	-

D Toolkits

Table 4: The toolkits summary for enhancing LLMs efficiency and effectiveness across different stages.

Toolkit	Base / Org	Type	Utilities					Availability		
			Efficient PreTrain	*Efficient FineTune*	*Efficient Inference*	*Model Hub*	*Dataset Hub*	*paper*	*github*	*homepage*
HuggingFace	PyTorch/TensorFlow/Jax	framework	✓	✓	✓	✓	✓	-	here	here
DeepSpeed	PyTorch	framework	✓	✓	✓	✗	✗	here	here	here
xformers	Facebook	package	✓	✓	✓	✗	✗	-	here	-
Megatron-LM	NVIDIA	framework	✓	✓	✗	✗	✗	here	here	-
Faster Transformer	TensorFlow/PyTorch/Triton	package	✓	✓	✓	✗	✗	-	here	-
TensorRT	NVIDIA	library	✓	✓	✓	✗	✗	-	here	here
Triton	Openai	compiler	✓	✓	✓	✗	✗	-	here	-
vLLM	PyTorch	library	✗	✗	✓	✗	✗	here	here	here