Beyond the Limits: A Survey of Techniques to Extend the Context Length in Large Language Models

Xindi Wang[1,2,3] , **Mahsa Salmani**[1] , **Parsa Omidi**[1] , **Xiangyu Ren**[1] ,
Mehdi Rezagholizadeh[1] and **Armaghan Eshaghi**[1*]
[1]Huawei Technologies Canada, Canada
[2]University of Western Ontario, Canada
[3]Vector Institute for Artificial Intelligence, Canada
xwang842@uwo.ca, {mahsa.salmani1, parsa.omidi, xiangyu.ren1, mehdi.rezagholizadeh, armaghan.eshaghi}@huawei.com

Abstract

Recently, large language models (LLMs) have shown remarkable capabilities including understanding context, engaging in logical reasoning, and generating responses. However, this is achieved at the expense of stringent computational and memory requirements, hindering their ability to effectively support long input sequences. This survey provides an inclusive review of the recent techniques and methods devised to extend the sequence length in LLMs, thereby enhancing their capacity for long-context understanding. In particular, we review and categorize a wide range of techniques including architectural modifications, such as modified positional encoding and altered attention mechanisms, which are designed to enhance the processing of longer sequences while avoiding a proportional increase in computational requirements. The diverse methodologies investigated in this study can be leveraged across different phases of LLMs, i.e., training, fine-tuning and inference. This enables LLMs to efficiently process extended sequences. The limitations of the current methodologies is discussed in the last section along with the suggestions for future research directions, underscoring the importance of sequence length in the continued advancement of LLMs.

1 Introduction

In the rapidly evolving domain of natural language processing (NLP), large language models (LLMs), such as GPT-3, PaLM and LLaMA, emerged as pivotal tools that have proved proficiency in understanding and generating human language including tasks such as language understanding, language generation, complex reasoning and other domains such as computer vision and autonomous driving [Brown *et al.*, 2020; Touvron *et al.*, 2023; Chowdhery *et al.*, 2024; Wang *et al.*, 2023]. In many real-world scenarios, such as multi-turn conversations and document summarization, LLMs are required to comprehend and produce long sequences in order to perform the task accurately during the inference phase. These context sequences are often substantially longer than those the LLMs were trained with, emphasising the fact that LLMs must have the capability to deal with lengthy sequences.

Processing long sequences by LLMs is a non-trivial task, which involves computational, structural, and practical challenges. Notably, increased sequence lengths can exponentially escalate processing requirements, particularly in transformer-based models with self-attention mechanisms. This not only increases the computational cost but also, the memory demands often surpass the capacity of advanced GPUs and thus, impeding efficient training [Dao *et al.*, 2022]. Hence, the efficiency of attention mechanisms, pivotal in addressing longer sequences, remains a key area of research, aiming to balance computational efficiency with model performance [Gu and Dao, 2023]. Moreover, maintaining contextual understanding and coherence over extended input spans further complicates the scenario, as it requires advanced methods to capture and utilize long-range dependencies. Finally, the evaluation and benchmarking of LLMs on long-sequence tasks also pose a significant challenge, demanding novel metrics and datasets for effective assessment [Kwan *et al.*, 2023]. Altogether, the aforementioned challenges highlight the intricacy and importance of advancing LLMs to proficiently support and utilize long sequences for various tasks.

In this survey, we provide a concise review of various approaches that have been developed to enable LLMs to handle long sequences. The overarching goal of the survey is to provide a detailed insight into those methods, as well as to highlight possible directions for future research. The techniques include architectural modifications, such as positional encoding modification, modified attention mechanisms and model compression techniques, which aim to optimize the processing of longer sequences without exponentially increasing computational and memory demands. Additionally, we explore the methods that can be adopted in different phases (training, fine-tuning, and inference), and have been pivotal in enabling LLMs to handle longer sequences, efficiently. The taxonomy of our literature review is shown in Figure 1. While there are existing surveys addressing LLMs with

*Corresponding author

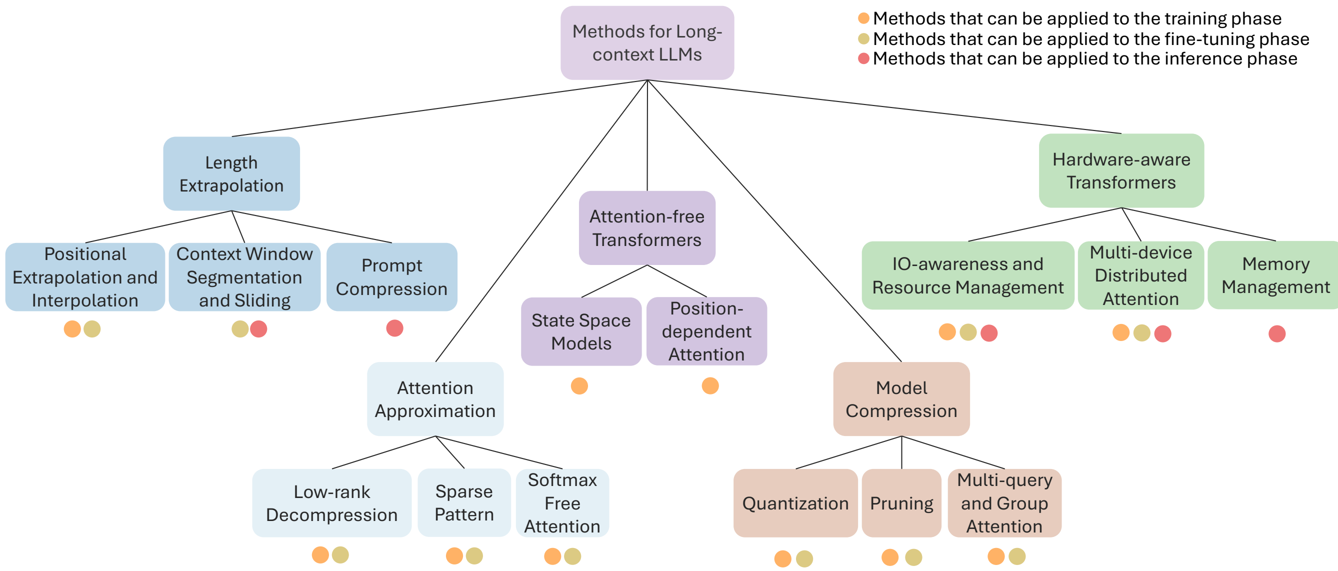

Figure 1: Taxonomy of Long-context LLM literature, which includes five distinct sections: length extrapolation, attention approximation, attention-free transformers, model compression, and hardware-aware transformers. We also establish connections between the methodologies and their related applicability scenarios. Some entail training a new model from scratch, others involve fine-tuning pre-trained models, and some implement over inference without any updates of hyper-parameters.

a more general scope [Zhao *et al.*, 2023; Naveed *et al.*, 2023; Wan *et al.*, 2023], this survey is particularly focused on evaluating the articles dealing with long sequences in LLMs. Moreover, there are other reviews on efficient Transformers and their training methodologies [Zhuang *et al.*, 2023; Huang *et al.*, 2023], but this survey specifically focuses on models and strategies that aim at enhancing the management of longer input sequences.

2 Length Extrapolation

In this section, we focus on methods whose primary objective is to enable LLMs to effectively support longer sequences. Among these methods, positional extrapolation and interpolation emerge as pivotal methods for extending the model's capacity to handle sequences longer than those on which the LLMs have been originally trained. Furthermore, we explore context window segmentation and sliding, a crucial technique that manipulates input sequences into smaller segments or moves the context window to enable processing of the longer sequence. Lastly, we review the strategy of prompt compression, an innovative approach to condense input prompts efficiently while retaining the essential information.

Positional Extrapolation and Interpolation. Position extrapolation and interpolation refer to the techniques that adjust the positional embedding (PE) associated with input tokens, which modify how these tokens are positioned and interpreted within the model's architecture. PEs play a pivotal role in the architecture of transformer models since they impart a crucial sense to the input tokens, enabling the model to discern the specific position of each token within the sequence. This ensures that the model can effectively capture and utilize the sequential information inherent in the input data. The vanilla transformer [Vaswani *et al.*, 2017] presents a novel Sinusoidal PE (SinPE) that uses sinusoidal functions to represent the absolute positions of the tokens. SinPE has become a widely used method, yet it has prompted further research into alternative approaches for handling positional information in transformer models. One alternative approach is trainable PEs, as explored by Chen *et al.* [2021], which learn an embedding mapping specific to the task. Another approach focuses on relative PEs, introduced by Shaw *et al.* [2018], which encodes the relative positions of tokens rather than their absolute positions, allowing for more flexible handling of varying sequence lengths. Additionally, the concept of Rotary PEs (RoPE) [Su *et al.*, 2024], involves rotating the query and key representations at an angle corresponding to the absolute positions of the tokens within the input sequence. This method provides a unique way of integrating positional information that can enhance the model's ability to capture complex dependencies. To further improve efficiency and support longer sequences, recent studies have investigated methods for positional extrapolation and interpolation.

Positional extrapolation refers to the model's ability to handle input sequences that exceed the length of those it was trained on, enabling the preservation of context and coherence over extended sequences. This capability is important for models tasked with understanding and generating lengthy documents or conversations. For example, Attention with Linear Biases (ALiBi) [Press *et al.*, 2022] introduces a heuristic of negative causal attention bias, which dispenses with PEs for tokens in the transformer model. ALiBi encodes position information by biasing the query-key attention scores proportionally to the distance between each pair of tokens. As compared to other PE schemes, ALiBi demonstrates superior extrapolation capabilities to unseen sequence lengths. Different from ALiBi, xPOS [Sun *et al.*, 2023b] extends causal

RoPE, which incorporates a unique exponential decay factor at each dimension of the rotation angle vector, thereby improving length extrapolation. Another approach CLEX [Chen *et al.*, 2024] uses ordinary differential equations to generalize PE scaling. By modeling continuous dynamics with length scaling factors, CLEX effectively overcomes the constraints of traditional positional extrapolation techniques.

On the other hand, positional interpolation deals with the model's proficiency in inserting or integrating new information within existing sequences. For example, positional interpolation proposed by Chen *et al.* [2023] applies linear scaling on the position indices, effectively aligning the maximum position index to correspond with the context window limit previously established during the pre-training phase. Experimental observations indicate that this strategy exhibits greater stability and necessitates fewer fine-tuning steps compared to direct extrapolation methods. Additionally, YaRN [Peng *et al.*, 2023b] extends RoPE by adopting an uneven interpolation of frequencies, specifically preserving the high-frequency components. This approach avoids losing important positional details that enhances the ability of the model to maintain critical positional information.

Context Window Segmentation and Sliding. LLMs based on transformers are inherently constrained by limited context windows, rendering them incapable of directly integrating or utilizing the entirety of information in long sequences. To mitigate this limitation, various methodologies have been developed to divide the input into segments and apply a sliding window approach to manage the context. One such approach is structured prompting [Hao *et al.*, 2022], which groups demonstration examples and encodes them individually with well-designed position encoding. These encoded examples are then collectively attended to by the test example through a re-scaled attention mechanism, ensuring that each segment receives adequate focus and relevance. Building on the idea of segmenting input, Ratner *et al.* [2023] introduces a parallel context window (PCW), which segments the long-context into chunks and restricts the attention mechanism to operate exclusively within each window. By redeploying positional encoding across these windows, this method ensures efficient processing of long sequences without overwhelming the attention mechanism. Another innovative approach is StreamingLLM [Xiao *et al.*, 2024], which addresses the "attention sink" phenomenon. This phenomenon occurs when a significant portion of the attention score is allocated to the initial tokens, regardless of their relevance. StreamingLLM merges window context with the first token, which enables the LLMs trained with a finite-length attention window to be effectively generalized to infinite sequence lengths without requiring additional fine-tuning.

Prompt Compression. Prompt compression refers to methods that shorten original prompts while keeping the important information. This process involves either condensing extensive prompt inputs or learning concise representations of prompts. LLMLingua [Jiang *et al.*, 2023a] employs streamlined and proficient language models, such as GPT-2 small or LLaMA-7B, to identify and eliminate extraneous tokens within prompts. This method facilitates the efficient execution of inferences with expansive language models, achieving a compression ratio of up to 20 times while maintaining performance with minimal decline. Building on this approach, LongLLMLingua [Jiang *et al.*, 2023b] addresses the inherent "lost in the middle" issue observed in LLMs, enhancing the processing of long-context information. This method not only reduces costs but also improves efficiency through prompt compression, resulting in a significant improvement of up to 21.4% in retrieval-augmented generation performance while using only a quarter of the tokens. Further advancing the field, Li *et al.* [2023b] introduce a novel method called "Selective Context". This approach systematically identifies and prunes redundancy within the input context to streamline the input, making it more compact and optimizing the overall efficiency of language model inferences. MemGPT [Packer *et al.*, 2023] is then proposed to overcome the limitations of fixed-length context windows in traditional LLMs. The primary goal is to simulate an infinite context while still effectively utilizing fixed-context models. MemGPT achieves this by autonomously managing its own memory through "function calls" allowing for dynamic context modifications during a task. It establishes a memory hierarchy, akin to traditional operation systems, and treats context windows as constrained memory resources. By enabling the LLM to control its context, MemGPT provides an illusion of longer context length.

3 Attention Approximation

The foundation of attention approximation lies in the ambition to reduce the computation and memory complexities of vanilla self-attention [Vaswani *et al.*, 2017], which increases quadratically with respect to the sequence length n, i.e., $O(n^2)$. This can be achieved by approximating the full-rank attention map with a low-rank counterpart, exploiting the sparse patterns in the attention layers, or simplifying the softmax-related complexity of vanilla attention. These techniques aim to provide efficient approximations that maintain the effectiveness of the attention mechanism while managing long sequences more efficiently.

Low-rank Decomposition. The transformer architecture utilizes a self-attention mechanism that involves three matrices, namely, Query ($\mathbf{Q}$), Key ($\mathbf{K}$), and Value ($\mathbf{V}$). The attention mechanism works by computing the similarity between the $\mathbf{Q}$ and $\mathbf{K}$ and the result is used to weight the $\mathbf{V}$, emphasizing the the most relevant information. The low-rank decomposition method can make the attention computation more efficient by reducing the number of parameters in the matrices. One such approach is Linear Encoder-Decoder (LED) [Winata *et al.*, 2020], which is proposed to decompose each of the three matrices into smaller matrices by adding an encoder and decoder before and after the self-dot-product to reduce the matrix size for approximation of linear parameter efficiency. Different from LED, Linformer [Wang *et al.*, 2020] introduces another linear projection mechanism that adds two smaller matrices before $\mathbf{K}$ and $\mathbf{V}$ to project them to a smaller size while leaving $\mathbf{Q}$ unchanged. Both methods optimize matrix computation through linear approximation. Autoformer [Wu *et al.*, 2021] further improves the ability of capturing long-term dependency by introducing an auto-correlation mechanism that leverages the Fast Fourier Transform (FFT) for time series decomposition. The decomposed matrix is then utilized for time series analysis, which enables the model to better capture and improve forecasting accuracy for long-term contexts. Deep neural networks (DNNs) have also been utilized for tensor decomposition in transformers. In particular, unlike traditional methods such as singular value decomposition, Deeptensor [Saragadam *et al.*, 2022] uses a DNN to learn an optimal regularizer for tensor decomposition when the distributions of the tensor is non-Gaussian.

Sparse Pattern. An alternative strategy to address the computation and memory challenges of the self-attention module in transformers involves leveraging sparse patterns to handle long contexts effectively. These patterns use a sparse attention matrix, where each token attends to a limited set of other tokens. Various methods have been proposed to introduce the sparsity, which, while not specifically designed for long contexts, can effectively help manage long sequences.

Among the most straightforward yet practical instances of sparse patterns, Block-wise Self Attention [Qiu *et al.*, 2020], stands out as an illustrative demonstrations. This method reduces the computation and memory cost by chunking the input sequence into fixed blocks. An alternative strategy involves having an individual token attend to tokens at regular, fixed internals. For instance, Longformer [Beltagy *et al.*, 2020] is a sparsifying mechanism that utilizes dilated windows of tokens to construct the attention matrix. LogSparse [Li *et al.*, 2019] is another method that sparsifies the attention matrix by restricting consideration to a limited window of tokens, where the window is defined by exponential steps from the token itself. This approach ensures a targeted focus range for each individual token. By employing LogSparse, it is guaranteed that any pair of tokens can exchange attention information with each other, while the memory usage of the transformer can be reduced to $O(n(Log\ n)^2)$. LongNet [Ding *et al.*, 2023] introduces dilated attention, in which attention allocation decreases exponentially as the distance between tokens increases. This approach exploits mixed dilated rates to accommodate both local and global dependencies between different tokens. It has been shown that by utilizing LongNet, a linear computation complexity, $O(n)$, and a logarithm dependency between tokens can be achieved.

Some other sparse transformers consider adaptive sparse patterns which are not dependent on the location of the tokens, but rather they rely on other dynamic factors such as embedding values or task-specific parameters. For instance, Routing Transformer [Roy *et al.*, 2021] exploits dynamic key-value pairs to infer sparsity patterns and hence, it removes the computation and memory requirements of attending to content unrelated to the query of interest. In particular, Routing Transformer utilizes k-means clustering to define the k most relevant columns in $\mathbf{Q}$ and $\mathbf{K}$, and assigns each query to the keys within the same cluster. Routing Transformer results in computation complexity of the order $O(n^{1.5})$. Reformer [Kitaev *et al.*, 2020] is another sparse approach which clusters the tokens prior to implementing attention, and it does so according to a hash-based similarity.

Softmax-free Attention. The efficacy of vanilla attention [Vaswani *et al.*, 2017] is often attributed to the softmax operation, which is important for capturing long dependencies. However, this operation causes quadratic complexity in both time and space, impeding the scalability of transformers for long sequences. Replacing the softmax operation can reduce computational complexity, enhancing the efficiency of processing long sequences. This category of approaches is called softmax-free attention.

CosFormer [Qin *et al.*, 2022] emulates softmax behaviors through a linear operatorthat re-weights the cosine-based distance. SOFT [Lu *et al.*, 2021] employs a Gaussian kernel function to replace the softmax, while SIMA [Koohpayegani and Pirsiavash, 2024] opts for normalizing query and key matrices using a simple L1-norm. Another set of approaches replaces softmax with the ReLU function for normalization, demonstrating that this substitution maintains performance, while preserving linear scalability [Shen *et al.*, 2023]. An alternative class of architectures centers around generalized kernelziable attention, wherein the conventional attention mechanism is formulated as a specific kernel function. For instance, Performer [Choromanski *et al.*, 2021] is an approach leveraging positive orthogonal random features to effectively model the attention mechanism into simplified softmax-free architecture with linear space and time complexity.

Another recently-developed transformer architecture that can be studied under this category (to varying degrees) is RetNet [Sun *et al.*, 2023c], which replaces the softmax operation with a D-matrix followed by group normalization (GroupNorm). The D-matrix introduces exponential decay weighting of previous tokens, diminishing the impact of distant tokens. The incorporation of GroupNorm adds non-linearity, a characteristic once inherent in softmax. A distinguished feature of RetNet is that it can be implemented in both parallel and sequential manners. Accordingly, it can exploit the accelerated token generation during inference, similar to Recurrent Neural Networks (RNN), and exploit the efficiency of parallelization during training.

4 Attention-free Transformers

Attention-free transformers refer to the computational approaches that provide dependency information between tokens without relying on the conventional attention mechanism. These mechanisms offer a different perspective on dependency calculation, while maintaining sub-quadratic memory complexity. In this study, we consider two distinct subcategories of this domain, namely, State Space Model (SSM) and positional-dependency attention—that enhance the handling of long contexts in LLMs.

State Space Model. SSM is a statistical sequence-to-sequence (seq2seq) model that employs linear projections of hidden states to compute the output sequence based on an input sequence. SSM introduces an RNN-like seq2seq model without non-linearity, which empowers parallel trainingand optimize the inference efficiency. The seq2seq operation based on the states can be analytically unrolled, resembling a convolutional operation with a parametrized kernel. Theoretically, similar to RNN, this convolution operation can extend to infinite length, enabling the computation of outputs without calculating individual states. During the training phase with the entire input sequence, this convolution process can be exceptionally rapid and parallel, setting it apart from the traditional RNN. However, the computational expense of the convolution kernel limits SSM's application in deep learning until the advent of Structured State Space (S4) [Gu *et al.*, 2022]. S4 integrates SSM, HIPPO [Gu *et al.*,], and structured matrices to solve the complexity of the convolution kernel. HIPPOO [Gu *et al.*,] is a particular representation of the original SSM, which takes input states and maps them to higher-dimensional states that can be seen as an online compression of history. However, with a finite size of states, this method cannot remember the entire input. This necessitates the introduction of exponential decay, particularly beneficial for recent past accuracy. Some approaches employ decay techniques, showcasing performance improvements [Orvieto *et al.*, 2023; Ma *et al.*, 2023b]. Hungry Hungry Hippos (H3) [Fu *et al.*, 2023] incorporates two SSMs, enabling local token attention and global token recall through a multiplicative gate mechanism, akin to LSTM gating. Hyena [Poli *et al.*, 2023], similar to H3, replaces the attention layer by interleaving implicitly parametrized long convolutions and data-controlled gating, effectively narrowing the quality gap with the vanilla attention mechanism at scale and achieving comparable perplexity with a reduced computational cost. Mamba [Gu and Dao, 2023] enhances SSMs by incorporating H3 with multi-layer perceptrons (MLP), refining reasoning capabilities through a strategic reorganization of the gating mechanism.

Position-dependent Attention. Within this distinct category, a unique form of dependency calculation emerges, where dependencies rely on the position of tokens rather than interactions between them. The Attention-free Transformer (AFT) [Zhai *et al.*, 2021], inspired by attention-based transformers, exclusively employs $\mathbf{K}$ and $\mathbf{V}$ while eliminating $\mathbf{Q}$ and its dot product with $\mathbf{K}$. Instead, AFT introduces a novel learnable matrix $\mathbf{W}$, which acts as a fixed attention map (static routing) consistent across all input sequences. Unlike adaptive weighting in vanilla attention, $\mathbf{W}$ considers only pairwise token positions, disregarding semantic dependencies. To enhance customization based on current input data, $\mathbf{K}$ accompanies $\mathbf{W}$.

Building upon the principles of AFT, Receptance Weighted Key Value (RWKV) architecture [Peng *et al.*, 2023a] adopts a similar approach with modifications to interaction weights for simplicity, and redefines $\mathbf{W}$ as a linear time decay of a vector with a much smaller size. RWKV provides the flexibility to formulate a seq2seq model as either a transformer or an RNN, similar to what we observe in RetNet. This proves advantageous for parallelizing computations during training using the transformer form of RWKV while maintaining consistent computational and memory complexity during inference through the RNN form, without limitations on sequence length. Although both AFT and RWKV imply trade-offs between performance and complexity, RWKV emerges as a practical alternative for dot-product transformers with the ability to scale up to very large models.

5 Model Compression

An alternative approach that can enable LLMs to support longer sequences is model compression. Various model compression approaches have distinct focal points. Some concentrate on minimizing the size of the LLM architecture by eliminating redundant weights, thereby reducing computational and memory requirements. Some others prioritize decreasing computation precision to alleviate computational complexity. Furthermore, certain approaches emphasize enhancing memory efficiency and optimizing data storage methods. In this section, we explore methods that exert a more significant impact on accommodating longer input sequences.

Quantization. Quantization has been considered as a promising approach for improving the computational time and energy efficiency of generic neural networks. Moreover, neural networks are robust enough to be quantized to lower bit-widths with a relatively small impact on the accuracy of the network [Gholami *et al.*, 2022]. That provides an insight into utilizing quantization to reduce the complexity of LLMs, and accordingly, enabling them to support longer input sequence [Zhu *et al.*, 2023]. Depending on the stage at which quantization is implemented, quantization techniques for LLMs can be classified as Quantization-Aware Training (QAT) and Post-Training Quantification (PTQ).

In QAT approach, the quantization is integrated into the training phase such that the network can be adapted to quantization effects. This adaptation helps mitigate the potential loss of accuracy that might occur as a result of quantization during the inference phase. However, applying QAT to LLMs can be challenging due to the computational cost and the latency, as QAT requires training over the whole training dataset to avoid significant accuracy degradation. LLM-QAT [Liu *et al.*, 2023b] addresses this issue by proposing data-free knowledge-distillation, in which the data generated by the LLM itself is used for knowledge distillation. As the proposed approach can retain the distribution of the non-quantized (original) output, it can be applied to any generative model, independent of the original training dataset.

On the other hand, PTQ involves reducing the precision of the weights and activations of a neural network after the completion of the training phase. The primary goal of PTQ is to reduce the memory and computational requirements of the model, making it more suitable for deployment on resource-limited devices. PTQ is simple and efficient, however, it can impose performance degradation due to the low precision. With the existing trade-off between the model size, computation speed and accuracy, this method can be used to improve the efficiency of LLMs without extensive training efforts.

The PTQ approaches can be categorized into weight-only quantization, which only focuses on quantizing the weights and weight-activation quantization, which quantizes both weights and activations. LLM.int8() [Dettmers *et al.*, 2022] is the first multi-billion-scale INT8 quantization procedure that reduces memory usage by half during inference, while it maintains the performance the same as that in the full-precision model. OPTQ [Frantar *et al.*, 2023] proposes a layer-wise quantization technique, which can further reduce the precision to 3 or 4 bits per weight element, with negligible accuracy degradation. Furthermore, Lin *et al.* [2024] find that weights do not carry equal importance for the performance, and accordingly, the quantization error can be significantly reduced by maintaining only 1% of salient weights in full-precision. They propose Activation-aware Quantization (AWQ) method, which retains the weights corresponding to large activations in full-precision. In order to address the significant quantization error resulted from the outliers in activations distribution, Lee *et al.* [2024] propose a mixed-precision quantization approach, namely, outlier-aware weight quantization (OWQ), which applies higher precision to the weights associated with outlier activations.

Pruning. Pruning refers to reducing the size of LLMs by removing redundant parameters that are less crucial for the models. Pruning can help optimize the model for deployment and make the model more efficient in terms of computation complexity and memory usage. Accordingly, pruning can be considered as an approach to enable a language model to support longer sequence length, while maintaining the desirable complexity and performance. In general, pruning a model can be categorized into structured and unstructured pruning.

Structured pruning aims at removing higher-granularity structures, such as entire neurons, layers, or rows/columns of weight matrices, which can result in a model that retains its original structure but with fewer parameters. LLM-Pruner [Ma *et al.*, 2023a] is a structural task-agnostic pruning approach that selectively removes non-critical connection structures by considering both first-order information and an approximated Hessian information gradient information. Alternatively, Sheared LLaMA [Xia *et al.*, 2024] uses a two-stage approach for pruning an LLM. In the first stage, it applies targeted structured pruning to shape the model by removing layers, heads, and intermediate connections. In the second stage, the batches of data are loaded dynamically and the model structure is modified in each training iteration based on losses in various domains. As a result, Sheared LLaMA achieves a compressed model that can outperform the LLMs, with the same size but trained from scratch.

Unstructured pruning involves with pruning individual parameters of a model independently based on their magnitudes or importance, resulting in an irregular sparse structure. Due to the irregularity in the structure and in the memory access patterns, unstructured pruning hinders the efficiency gain that might be achieved through structured pruning, and it requires specialized software and/or hardware for efficient deployment. SparseGPT [Frantar and Alistarh, 2023] compresses LLMs with billions of parameter by as much as 60%, almost without affecting the performance of the models. However, SparseGPT heavily relies on weight updates. To address this issue, Sun *et al.* [2023a] propose Wanda that prunes the weights according to novel criterion, which is mainly based on product value of the weights and their input activations.

Multi-query and Group Attention. While multi-head attention has demonstrated its effectiveness in characterizing the correlations among tokens, it suffers from the incremental memory bandwidth cost and longer latency during inference due to repeatedly loading the large $\mathbf{KV}$ tensors as the input sequence length increases. Multi-query attention (MQA) [Shazeer, 2019] is one of the approaches that address the aforementioned issue. In particular, MQA essentially reuses the same $\mathbf{KV}$ tensors across all attention heads of each query to reduce the memory bandwidth requirements during decoding and thus, allows longer sequences and faster decoding. Given its demonstrated performance with minor quality degradation, MQA has been adopted in several works. Google [Chowdhery *et al.*, 2024] trains a LLM named Pathways Language Model (PaLM) with the adoption of MQA to improve decoding speed and later PaLM-2 [Anil *et al.*, 2023] is released with improved computation efficiency. Pope *et al.* [2023] propose a partition-optimized model that enables up to $32\times$ larger context lengths with the help of MQA on LLMs. de Jong *et al.* [2023] adopts MQA to reduce the cross-attention computation at the decoders in Fusion-in-Decoder models with faster inference. More recently, Li *et al.* [2023a] introduce StarCoder, a Code LLM, with fast large-batch inference enabled by MQA. The shared $\mathbf{KV}$ tensors idea in MQA also inspired the emergence of other attention computation schemes. A grouped-query attention (GQA) [Ainslie *et al.*, 2023] mechanism is proposed to trade-off performance degradation and speed by sharing a subset of $\mathbf{KV}$ tensors.

6 Hardware-aware Transformers

A viable solution to challenges posed by long sequence in LLMs involves adapting algorithms to be hardware-aware, enhancing efficiency and enabling the processing of longer sequences. Our exploration encompasses a spectrum of innovations, each tailored to address distinct aspects of IO-awareness, resource management, multi-device distributed attention, and memory management.

IO-awareness and Resource Management. A critical concern in deep neural network models like transformers is the constant need for Read/Write operations from/to memory. FlashAttention [Dao *et al.*, 2022] addresses this challenge by making attention algorithms IO-aware, effectively managing reads and writes between different levels of GPU memory. This approach capitalizes on the insight that the softmax matrix in attention can be computed without materializing the entire matrix, utilizing tiling techniques. FlashAttention introduces parallelization over sequence length, processing different portions (blocks) of the sequence to compute attention in a more manageable block-wise operation within fast memory Static Random Access Memory (SRAM) in GPUs. Moreover, FlashAttention highlights the efficiency gains of recomputing attention during the backward pass of optimization. It utilizes blocks already present in SRAM instead of storing attention results in high bandwidth memory (HBM) and transferring them to SRAM again. Building on FlashAttention foundations, Block-wise Parallel Transformer (BPT) [Liu and Abbeel, 2023] fuses the feedforward layer with self-attention to further minimize IO usage, enabling the model to handle sequences up to four times longer than FlashAttention.

This IO-aware approach is not exclusive to attention-based transformers; similar techniques have been applied to expedite SSMs. SSMs, emerging as alternatives to transformers due to linear scalability and convolution implementation feasibility, present challenges in convolution-dominated computation time during training. FlashConv [Fu *et al.*, 2023] addresses this by leveraging the Cooley-Tukey decomposition of the FFT into a series of diagonal matrix multiplication, to take advantage of fast tensor cores. To accommodate longer sequences, FlashConv utilizes the recurrent properties of SSMs, allowing convolution to occur in different portions sequentially. Mamba [Gu and Dao, 2023] further enhances SSMs through innovative techniques such as kernel fusion, parallel scan, and recomputation, leveraging modern accelerators like GPUs for efficient memory hierarchy utilization.

Multi-device Distributed Attention. Both FlashAttention and BPT leverage distinct streaming multiprocessors in GPUs for parallel processing of different blocks. However, the limited size of SRAM imposes constraints on sequence length. Encouragingly, this concept can be expanded to accommodate very long sequences by distributing attention computation across multiple GPUs, as proposed by Ring Attention [Liu *et al.*, 2023a]. This innovative approach enables block-wise self-attention computation, seamlessly overlapping communication of key-value blocks among devices with the computation of each block on devices. As a result, it facilitates the processing of sequences several times longer than BPT, showcasing scalability across the device count. Another example of distributing attention computation across multiple devices is demonstrated by LongNet [Ding *et al.*, 2023]. LongNet possesses the capability to compute multiple attentions, each with a distinct dilated sparsity pattern. These computations operate independently and can be distributed over multiple devices, with each device corresponding to a single dilated pattern. This collective approach facilitates the processing of longer sequences.

Memory Management. Effective memory management is vital in LLMs, especially during the autoregressive inference phase. The sequential generation of tokens, repeated for each request, leads to a memory-bound workload, limiting GPU utilization and serving throughput. To enhance throughput, batching multiple requests requires efficient memory management, specifically for Key-Value ($\mathbf{KV}$) caches. The dynamic nature of $\mathbf{KV}$ cache growth and its unpredictable lifetime necessitate adaptive strategies for optimal memory utilization in varying context lengths.

PagedAttention [Kwon *et al.*, 2023] employs a virtual memory-inspired strategy during the inference phase to tackle the memory-bound challenges inherent in sequential generation. By segmenting $\mathbf{KV}$ caches into blocks, this approach achieves flexible memory management, effectively mitigating both internal and external fragmentation issues. In the pursuit of attention acceleration during inference, Flash-Decoding [Dao *et al.*, 2023] builds upon FlashAttention principles. Introducing a new parallelization dimension for keys/values sequence length, it ensures optimal GPU utilization even with small batch sizes and large context lengths. This approach proves instrumental in achieving up to $8\times$ faster generation for very long sequences. Additionally, other methods enhance memory management efficiency. FlashDecoding++ [Hong *et al.*, 2023], for instance, goes one step further by eliminating the need for synchronization in handling partial softmax computations, effectively addressing a limitation observed in prior works.

Another notable memory management technique is LLM-in-Flash [Alizadeh *et al.*, 2023], which leverages the larger size of flash memory compared to Dynamic Random Access Memory (DRAM). This method runs an LLM during inference efficiently by storing model parameters in flash memory and bringing them to DRAM on demand. To balance the lower bandwidth of Flash memory, the paper introduces an inference cost model considering flash memory characteristics. The technique incorporates sparsity awareness in feedforward layers and context-adaptive loading of the model. Although not specifically used to increase sequence length, this method has the potential to load a model up to twice the size of the available DRAM. This capacity could be harnessed to handle longer sequences while ensuring practical data transfer between DRAM and flash memory.

7 Conclusion and Future Directions

In this survey, a systematic review of different approaches for efficiently extending the sequence length in LLMs is provided. We start with the motivation of the work and the necessity of handling long sequences by LLMs. We then discuss the methods that encompass architectural changes, such as positional encoding modification, and attention mechanism modification,designed to improve long sequences handling without significantly increasing the computational cost. We further explore the methods that can be applied to different phases, such as training, fine-tuning and inference, which efficiently improve the LLM's capability of processing long sequences. These techniques not only address the immediate challenges posed by sequence length limitations but also pave the way for more complex and contextually aware LLMs.

Despite these advancements, challenges related to computational cost, model interpretability, and the ability to integrate external knowledge remain prevalent. The trade-offs between model complexity, processing speed, and accuracy continue to be a pivotal consideration in the design and implementation of LLMs for long sequences. Future research could focus on further optimizing the architecture of LLMs to enhance their efficiency in processing long sequences. Innovations could involve developing attention mechanisms or network structures that can handle long sequences more effectively while not increasing the computational cost. In addition, integrating LLMs with external knowledge could improve their ability in understanding and generating longer coherent and contextually accurate sequences. Exploring methods for effective knowledge integration and retrieval during the language generation process would be beneficial, too. Moreover, new training methodologies can be investigated to improve the ability of the model to understand and retain information over longer sequences. Techniques such as curriculum learning, where models are gradually exposed to increasingly longer sequences during training, could be one direction to explore. Last but not least, there is also a need for comprehensive benchmarking and evaluation frameworks that could accurately examine the capabilities of LLMs in handling long sequences. This includes creating datasets that specifically challenge the long-context processing capabilities of LLMs.

References

[Ainslie *et al.*, 2023] Joshua Ainslie, James Lee-Thorp, et al. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *EMNLP*, 2023.

[Alizadeh *et al.*, 2023] Keivan Alizadeh, Iman Mirzadeh, et al. LLM in a flash: Efficient large language model inference with limited memory. *arXiv:2312.11514*, 2023.

[Anil *et al.*, 2023] Rohan Anil, Andrew M Dai, et al. PaLM 2 technical report. *arXiv:2305.10403*, 2023.

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

[Brown *et al.*, 2020] Tom B. Brown, Benjamin Mann, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

[Chen *et al.*, 2021] Pu-Chin Chen, Henry Tsai, et al. A simple and effective positional encoding for transformers. In *EMNLP*, 2021.

[Chen *et al.*, 2023] Shouyuan Chen, Sherman Wong, et al. Extending context window of large language models via positional interpolation. *arXiv:2306.15595*, 2023.

[Chen *et al.*, 2024] Guanzheng Chen, Xin Li, et al. CLEX: Continuous length extrapolation for large language models. In *ICLR*, 2024.

[Choromanski *et al.*, 2021] Krzysztof M. Choromanski, Valerii Likhosherstov, et al. Rethinking attention with performers. In *ICLR*, 2021.

[Chowdhery *et al.*, 2024] Aakanksha Chowdhery, Sharan Narang, et al. PaLM: scaling language modeling with pathways. *JMLR*, 2024.

[Dao *et al.*, 2022] Tri Dao, Dan Fu, et al. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022.

[Dao *et al.*, 2023] Tri Dao, Daniel Haziza, et al. Flash-Decoding for long-context inference, 2023.

[de Jong *et al.*, 2023] Michiel de Jong, Yury Zemlyanskiy, et al. FiDO: Fusion-in-decoder optimized for stronger performance and faster inference. In *Findings of ACL*, 2023.

[Dettmers *et al.*, 2022] Tim Dettmers, Mike Lewis, et al. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *NeurIPS*, 2022.

[Ding *et al.*, 2023] Jiayu Ding, Shuming Ma, et al. LongNet: Scaling transformers to 1,000,000,000 tokens. *arXiv:2307.02486*, 2023.

[Frantar and Alistarh, 2023] Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *ICML*, 2023.

[Frantar *et al.*, 2023] Elias Frantar, Saleh Ashkboos, et al. OPTQ: Accurate quantization for generative pre-trained transformers. In *ICLR*, 2023.

[Fu *et al.*, 2023] Daniel Y Fu, Tri Dao, et al. Hungry Hungry Hippos: Towards language modeling with state space models. In *ICLR*, 2023.

[Gholami *et al.*, 2022] Amir Gholami, Sehoon Kim, et al. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, 2022.

[Gu and Dao, 2023] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv:2312.00752*, 2023.

[Gu *et al.*,] A. Gu, T. Dao, et al. HiPPO: Recurrent memory with optimal polynomial projections. In *NeurIPS,2020*.

[Gu *et al.*, 2022] Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *ICLR*, 2022.

[Hao *et al.*, 2022] Yaru Hao, Yutao Sun, et al. Structured prompting: Scaling in-context learning to 1,000 examples. *arXiv:2212.06713*, 2022.

[Hong *et al.*, 2023] Ke Hong, Guohao Dai, et al. FlashDecoding++: Faster large language model inference on gpus. *arXiv:2311.01282*, 2023.

[Huang *et al.*, 2023] Yunpeng Huang, Jingwei Xu, et al. Advancing transformer architecture in long-context large language models: A comprehensive survey. *arXiv:2311.12351*, 2023.

[Jiang *et al.*, 2023a] Huiqiang Jiang, Qianhui Wu, et al. LLMLingua: Compressing prompts for accelerated inference of large language models. In *EMNLP*, 2023.

[Jiang *et al.*, 2023b] Huiqiang Jiang, Qianhui Wu, et al. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. *arXiv:2310.06839*, 2023.

[Kitaev *et al.*, 2020] Nikita Kitaev, Lukasz Kaiser, et al. Reformer: The efficient transformer. In *ICLR*, 2020.

[Koohpayegani and Pirsiavash, 2024] Soroush A. Koohpayegani and Hamed Pirsiavash. SimA: Simple softmax-free attention for vision transformers. In *WACV*, 2024.

[Kwan *et al.*, 2023] Wai-Chung Kwan, Xingshan Zeng, et al. M4LE: A multi-ability multi-range multi-task multi-domain long-context evaluation benchmark for large language models. *arXiv:2310.19240*, 2023.

[Kwon *et al.*, 2023] Woosuk Kwon, Zhuohan Li, et al. Efficient memory management for large language model serving with pagedattention. In *SOSP*, 2023.

[Lee *et al.*, 2024] Changhun Lee, Jungyu Jin, et al. OWQ: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. In *AAAI*, 2024.

[Li *et al.*, 2019] Shiyang Li, Xiaoyong Jin, et al. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *NeurIPS*, 2019.

[Li *et al.*, 2023a] Raymond Li, Loubna Ben allal, et al. StarCoder: may the source be with you! *TMLR*, 2023.

[Li *et al.*, 2023b] Yucheng Li, Bo Dong, et al. Compressing context to enhance inference efficiency of large language models. In *EMNLP*, 2023.

[Lin *et al.*, 2024] Ji Lin, Jiaming Tang, et al. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In *MLSys*, 2024.

[Liu and Abbeel, 2023] Hao Liu and Pieter Abbeel. Blockwise parallel transformers for large context models. In *NeurIPS*, 2023.

[Liu *et al.*, 2023a] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. In *NeurIPS FMDM Workshop*, 2023.

[Liu *et al.*, 2023b] Zechun Liu, Barlas Oguz, et al. LLM-QAT: Data-free quantization aware training for large language models. *arXiv:2305.17888*, 2023.

[Lu *et al.*, 2021] Jiachen Lu, Jinghan Yao, et al. SOFT: Softmax-free transformer with linear complexity. In *NeurIPS*, 2021.

[Ma *et al.*, 2023a] Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the structural pruning of large language models. In *NeurIPS*, 2023.

[Ma *et al.*, 2023b] Xuezhe Ma, C. Zhou, et al. MEGA: Moving average equipped gated attention. In *ICLR*, 2023.

[Naveed *et al.*, 2023] Humza Naveed, Asad Ullah Khan, et al. A comprehensive overview of large language models. *arXiv:2307.06435*, 2023.

[Orvieto *et al.*, 2023] Antonio Orvieto, Samuel L Smith, et al. Resurrecting recurrent neural networks for long sequences. In *ICML*, 2023.

[Packer *et al.*, 2023] Charles Packer, Vivian Fang, et al. MemGPT: Towards LLMs as operating systems. *arXiv:2310.08560*, 2023.

[Peng *et al.*, 2023a] Bo Peng, Eric Alcaide, et al. RWKV: Reinventing RNNs for the transformer era. In *Findings of EMNLP*, 2023.

[Peng *et al.*, 2023b] Bowen Peng, Jeffrey Quesnelle, et al. YaRN: Efficient context window extension of large language models. In *ICLR*, 2023.

[Poli *et al.*, 2023] Michael Poli, Stefano Massaroli, et al. Hyena Hierarchy: towards larger convolutional language models. In *ICML*, 2023.

[Pope *et al.*, 2023] Reiner Pope, Sholto Douglas, et al. Efficiently scaling transformer inference. *MLSys*, 2023.

[Press *et al.*, 2022] Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *ICLR*, 2022.

[Qin *et al.*, 2022] Zhen Qin, Weixuan Sun, et al. cosFormer: Rethinking softmax in attention. In *ICLR*, 2022.

[Qiu *et al.*, 2020] Jiezhong Qiu, Hao Ma, et al. Blockwise self-attention for long document understanding. In *Findings of EMNLP*, 2020.

[Ratner *et al.*, 2023] Nir Ratner, Yoav Levine, et al. Parallel context windows for large language models. In *ACL*, 2023.

[Roy *et al.*, 2021] Aurko Roy, Mohammad Saffar, et al. Efficient content-based sparse attention with routing transformers. *TACL*, 2021.

[Saragadam *et al.*, 2022] Vishwanath Saragadam, Randall Balestriero, et al. DeepTensor: Low-rank tensor decomposition with deep network priors. *arXiv:2204.03145*, 2022.

[Shaw *et al.*, 2018] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *NAACL*, 2018.

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

[Shen *et al.*, 2023] Kai Shen, J. Guo, et al. A study on ReLU and softmax in transformer. *arXiv:2302.06461*, 2023.

[Su *et al.*, 2024] Jianlin Su, Murtadha Ahmed, et al. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024.

[Sun *et al.*, 2023a] Mingjie Sun, Zhuang Liu, et al. A simple and effective pruning approach for large language models. In *Workshop on ES-FoMo @ ICML2023*, 2023.

[Sun *et al.*, 2023b] Yutao Sun, Li Dong, et al. A length-extrapolatable transformer. In *ACL*, 2023.

[Sun *et al.*, 2023c] Yutao Sun, Li Dong, et al. Retentive Network: A successor to transformer for large language models. *arXiv:2307.08621*, 2023.

[Touvron *et al.*, 2023] Hugo Touvron, Thibaut Lavril, et al. LLaMA: Open and efficient foundation language models. *ArXiv*, abs/2302.13971, 2023.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, et al. Attention is all you need. In *NeurIPS*, 2017.

[Wan *et al.*, 2023] Z. Wan, X. Wang, et al. Efficient large language models: A survey. *arXiv:2312.03863*, 2023.

[Wang *et al.*, 2020] Sinong Wang, Belinda Z Li, et al. Linformer: Self-attention with linear complexity. *arXiv:2006.04768*, 2020.

[Wang *et al.*, 2023] Xindi Wang, Yufei Wang, et al. Investigating the learning behaviour of in-context learning: A comparison with supervised learning. *ECAI*, 2023.

[Winata *et al.*, 2020] Genta Indra Winata, Samuel Cahyawijaya, et al. Lightweight and efficient end-to-end speech recognition using low-rank transformer. In *ICASSP*, 2020.

[Wu *et al.*, 2021] Haixu Wu, Jiehui Xu, et al. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *NeurIPS*, 2021.

[Xia *et al.*, 2024] Mengzhou Xia, Tianyu Gao, et al. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *ICLR*, 2024.

[Xiao *et al.*, 2024] G. Xiao, Y. Tian, et al. Efficient streaming language models with attention sinks. In *ICLR*, 2024.

[Zhai *et al.*, 2021] Shuangfei Zhai, Walter Talbott, et al. An attention free transformer. *arXiv:2105.14103*, 2021.

[Zhao *et al.*, 2023] Wayne Xin Zhao, Kun Zhou, et al. A survey of large language models, 2023.

[Zhu *et al.*, 2023] Xunyu Zhu, Jian Li, et al. A survey on model compression for large language models. *arXiv:2308.07633*, 2023.

[Zhuang *et al.*, 2023] Bohan Zhuang, Jing Liu, et al. A survey on efficient training of transformers. In *IJCAI*, 2023.