Efficient Large Language Models: A Survey

ZHONGWEI WAN, The Ohio State University, USA
XIN WANG, The Ohio State University, USA
CHE LIU, Imperial College London, UK
SAMIUL ALAM, The Ohio State University, USA
YU ZHENG, Michigan State University, USA
JIACHEN LIU, University of Michigan, USA
ZHONGNAN QU*, Amazon AWS AI, USA
SHEN YAN, Google Research, USA
YI ZHU, Boson AI, USA
QUANLU ZHANG, Microsoft Research Asia, China
MOSHARAF CHOWDHURY, University of Michigan, USA
MI ZHANG, The Ohio State University, USA

Large Language Models (LLMs) have demonstrated remarkable capabilities in important tasks such as natural language understanding, language generation, and complex reasoning and have the potential to make a substantial impact on our society. Such capabilities, however, come with the considerable resources they demand, highlighting the strong need to develop effective techniques for addressing their efficiency challenges. In this survey, we provide a systematic and comprehensive review of efficient LLMs research. We organize the literature in a taxonomy consisting of three main categories, covering distinct yet interconnected efficient LLMs topics from model-centric, data-centric, and framework-centric perspective, respectively. We have also created a GitHub repository where we compile the papers featured in this survey at https://github.com/AIoT-MLSys-Lab/Efficient-LLMs-Survey, and will actively maintain this repository and incorporate new research as it emerges. We hope our survey can serve as a valuable resource to help researchers and practitioners gain a systematic understanding of the research developments in efficient LLMs and inspire them to contribute to this important and exciting field.

Additional Key Words and Phrases: Large Language Models; Efficient Methods; Machine Learning Systems

ACM Reference Format:
Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, and Mi Zhang. 2023. Efficient Large Language Models: A Survey. 1, 1, Article 101 (December 2023), 54 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*The work is done outside Amazon.

Authors' addresses: Zhongwei Wan, The Ohio State University, USA, wan.512@osu.edu; Xin Wang, The Ohio State University, USA, wang.15980@osu.edu; Che Liu, Imperial College London, UK, che.liu21@imperial.ac.uk; Samiul Alam, The Ohio State University, USA, alam.140@osu.edu; Yu Zheng, Michigan State University, USA, zhengy30@msu.edu; Jiachen Liu, University of Michigan, USA, amberljc@umich.edu; Zhongnan Qu, Amazon AWS AI, USA, ; Shen Yan, Google Research, USA, shenyan@google.com; Yi Zhu, Boson AI, USA, yi@boson.ai; Quanlu Zhang, Microsoft Research Asia, China, quzha@microsoft.com; Mosharaf Chowdhury, University of Michigan, USA, mosharaf@umich.edu; Mi Zhang, The Ohio State University, USA, mizhang.1@osu.edu.

XXXX-XXXX/2023/12-ART101 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Large Language Models (LLMs) are a type of advanced AI models designed to understand and generate human languages. Recently, we have witnessed a surge in LLMs include those developed by Open AI (GPT-3 [20] and GPT-4 [200]), Google (Gemini [270], GLaM [69], PaLM [51], PaLM-2 [7]), Meta (LLaMA-1 [276] and LLaMA-2 [277]), and other models such as BLOOM [239], PanGu-$\sum$ [233], and GLM [336]. These models have demonstrated remarkable performance across a variety of tasks such as natural language understanding (NLU), language generation, complex reasoning [321], and domain-specific tasks related to biomedicine [102, 282, 283], law [70] and code generation [35, 302]. Such performance breakthroughs can be attributed to their massive scales in model sizes and volumes of training data, as they contain billions or even trillions of parameters while being trained on a gigantic amount of data from diverse sources.

Although LLMs are leading the next wave of AI revolution, the remarkable capabilities of LLMs come at the cost of their substantial resource demands [51, 69, 200, 233]. Figure 1 illustrates the relationship between model performance and model training time in terms of GPU hours for LLaMA series, where the size of each circle is proportional to the number of model parameters. As shown, although larger models are able to achieve better performance, the amounts of GPU hours used for training them grow exponentially as model sizes scale up. In addition to training, inference also contributes quite significantly to the operational cost of LLMs. Figure 2 depicts the relationship between model performance and inference throughput. Similarly, scaling up the model size enables better performance but comes at the cost of lower inference throughput (higher inference latency), presenting challenges for these models in expanding their reach to a broader customer base and diverse applications in a cost-effective way.

The high resource demands of LLMs highlight the strong need to develop techniques to enhance the efficiency of LLMs. As shown in Figure 2, compared to LLaMA-1-33B, Mistral-7B [123], which uses grouped-query attention and sliding window attention to speed up inference, achieves comparable performance and much higher throughput. This superiority highlights the feasibility and significance of designing efficiency techniques for LLMs.

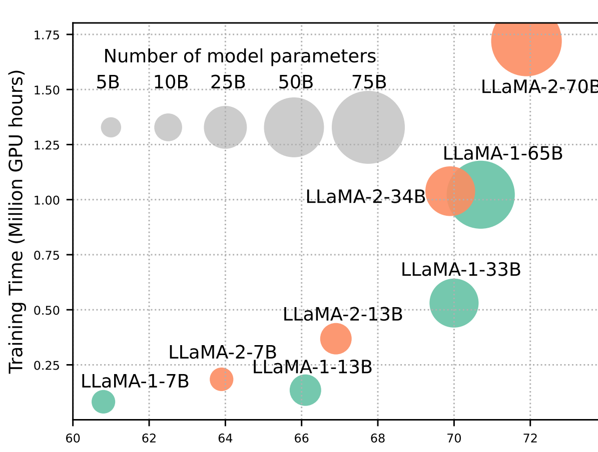


Fig. 1. Illustration of model performance and model training time in GPU hours of LLaMA models at different scales. The reported performance is the average score of several commonsense reasoning benchmarks. The training time is based on Nvidia A100 80GB GPU. The size of each circle corresponds to the number of model parameters. The original data can be found in [276, 277].

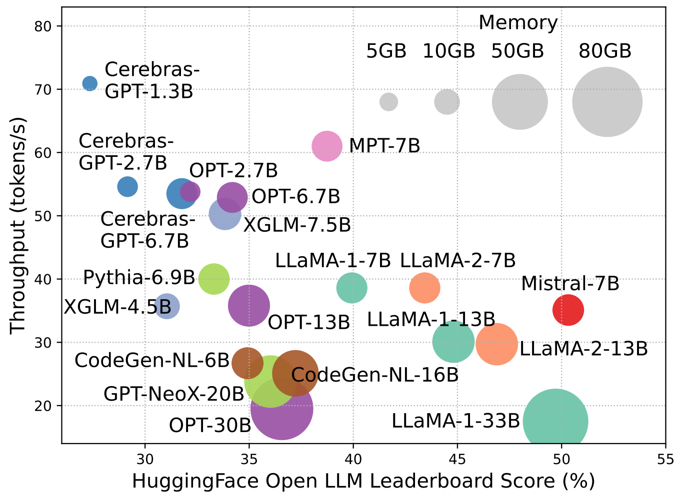


Fig. 2. Performance score *vs.* inference throughput for various LLMs. The throughputs are measured on Nvidia A100 80GB GPU with 16-bit floating point quantization. The size of each circle corresponds to the memory footprint (in Gigabytes) of each model when running with batch size of 1, prompt size of 256 and generating 1000 tokens. The original data can be found in [120].

The overarching goal of this survey is to provide a holistic view of the technological advances in efficient LLMs and summarize the existing research directions. As illustrated in Figure 3, we organize the literature in a taxonomy consisting of three main categories, covering efficient LLMs topics from **model-centric**, **data-centric**, and **framework-centric** perspective, respectively. These three categories cover distinct yet interconnected research topics, collectively providing a systematic and comprehensive review of efficient LLMs research. Specifically,

- **Model-Centric Methods:** Model-centric methods focus on both algorithm-level and system-level efficient techniques where the model itself is the focal point. With billions or even trillions of parameters, LLMs exhibit distinct characteristics [301] compared to smaller-scale models, necessitating the development of new techniques. In §2, we survey efficient techniques that cover research directions related to model compression, efficient pre-training, efficient fine-tuning, efficient inference, and efficient architecture design.
- **Data-Centric Methods:** In the realm of LLMs, the importance of data is as crucial as that of the model itself. Data-centric methods focus on the role of the quality and structure of data in enhancing the efficiency of LLMs. In §3, we survey efficient techniques that cover research directions related to data selection and prompt engineering.
- **LLM Frameworks**: The advent of LLMs has necessitated the development of specialized frameworks to efficiently handle their training, inference, and serving. While mainstream AI frameworks such as TensorFlow, PyTorch, and JAX provide the foundations, they lack built-in support for specific optimizations and features crucial for LLMs. In §4, we survey existing frameworks specifically designed for efficient LLMs, addressing their unique features, underlying libraries, and specializations.

In addition to the survey, we have established a GitHub repository where we compile the papers featured in the survey, organizing them with the same taxonomy: https://github.com/AIoT-MLSys-Lab/Efficient-LLMs-Survey. We will actively maintain it and incorporate new research as it emerges.

Although there are a few surveys on LLMs [26, 128, 299, 354], this survey provides a focused review and discussion on the literature related to the efficiency aspect of LLMs. There are also surveys on efficient Transformers [269] and their training methods [363]. In contrast, this survey specifically focuses on efficiency techniques designed for models of more than billions of parameters. We hope this survey together with the GitHub repo can help researchers and practitioners navigate through the literature and serve as a catalyst for inspiring further research on efficient LLMs.

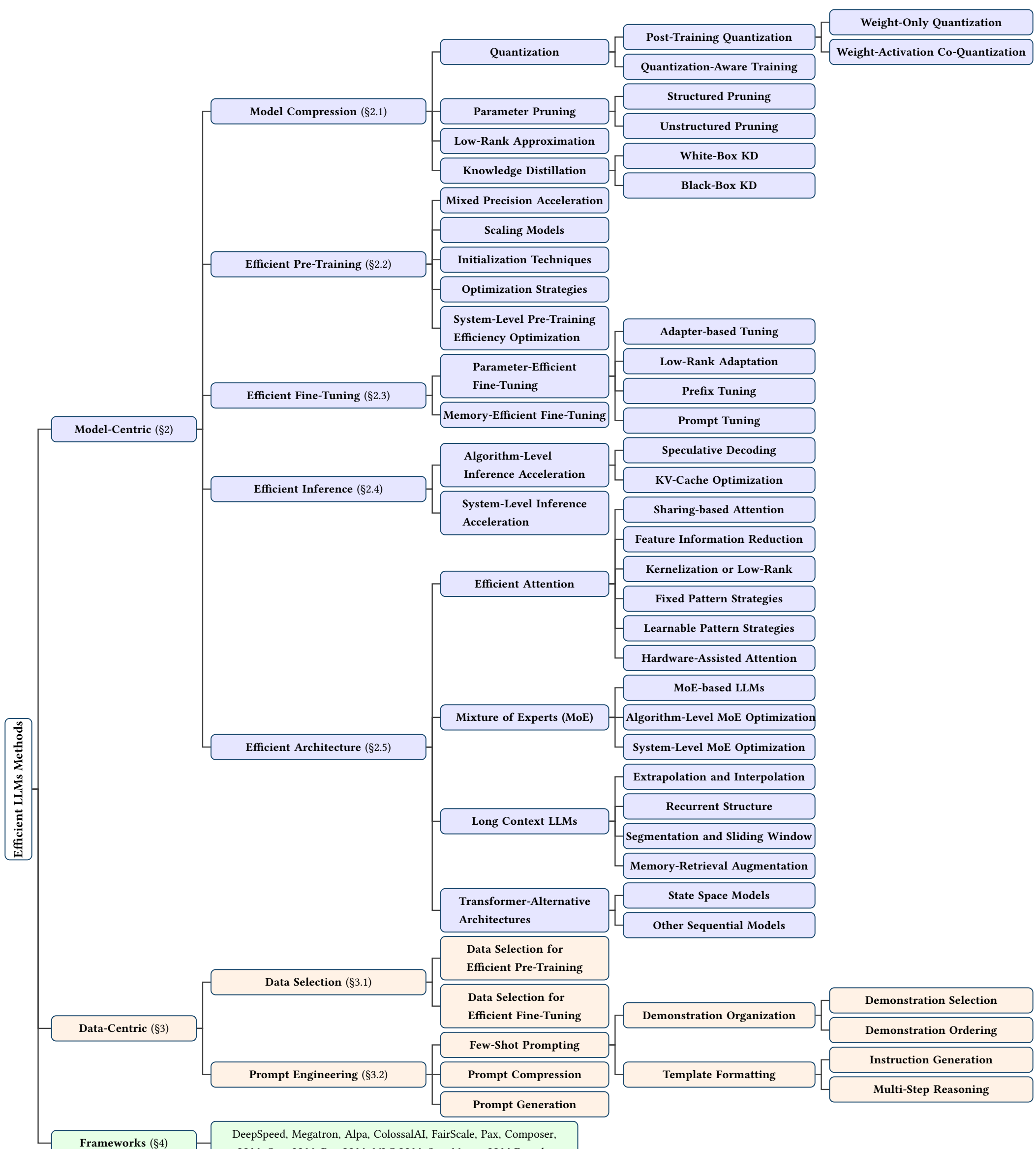


Fig. 3. Taxonomy of efficient large language models (LLMs) literature.

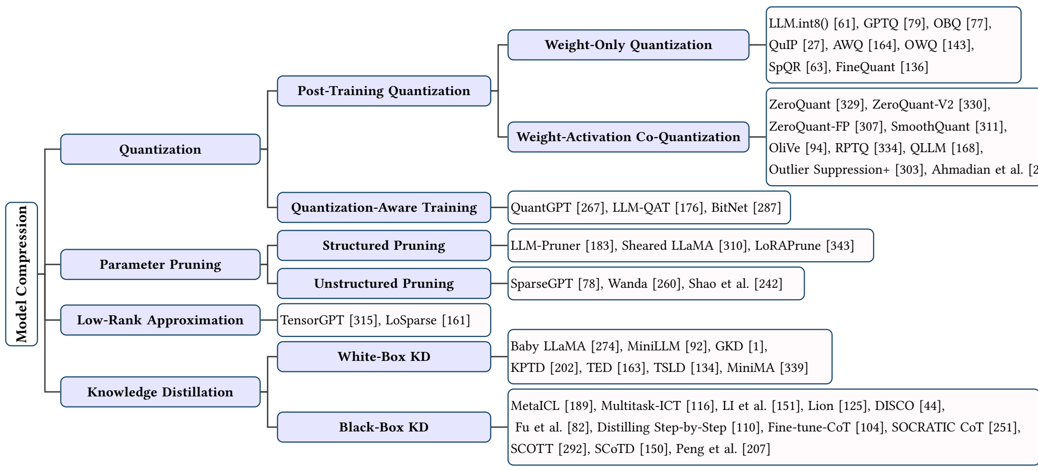


Fig. 4. Summary of model compression techniques for LLMs.

2 MODEL-CENTRIC METHODS

2.1 Model Compression

As summarized in Figure 4, model compression techniques for LLMs can be grouped into four categories: quantization, parameter pruning, low-rank approximation, and knowledge distillation.

2.1.1 ***Quantization.***

Quantization compresses LLMs by converting model weights and/or activations of high-precision data types $\mathbf{X}^{\mathrm{H}}$ such as 32-bit floating point into low-precision data types X^{L} such as 8-bit integer [61] or 4-bit integer [62]:

$$\mathbf{X}^{\mathrm{L}} = \mathrm{Round}\left(\frac{\mathrm{absmax}\left(\mathrm{X}^{\mathrm{L}}\right)}{\mathrm{absmax}\left(\mathrm{X}^{\mathrm{H}}\right)}\mathbf{H}^{\mathrm{H}}\right) = \mathrm{Round}\left(\mathcal{K}\cdot\mathbf{X}^{\mathrm{H}}\right), \text{and } \mathbf{X}^{\mathrm{H}} = \frac{\mathbf{X}^{\mathrm{L}}}{\mathcal{K}} \tag{1}$$

where Round denotes mapping a floating number into an approximate integer; absmax denotes the absolute maximum of the input elements; and $\mathcal{K}$ denotes the quantization constant.

Quantization techniques for LLMs can be classified as post-training quantization (PTQ) and quantization-aware training (QAT).

Post-Training Quantization (PTQ). PTQ quantizes LLMs after the model has been trained. To compensate for the accuracy drop, PTQ uses a small calibration dataset to update the quantized weights and/or activations. PTQ for LLMs can in general be grouped into two categories: weight-only quantization, and weight-activation co-quantization.

- ***Weight-Only Quantization*** focuses on quantizing model weights only for LLMs. For example, Dettmers et al. [61] introduces the first multibillion-scale Int8 weight quantization method named LLM.int8 () that significantly reduces memory usage during inference while being able to maintain the full precision model performance. Frantar et al. [79] push one step further and propose GPTQ, a post-training weight quantization method that compresses LLM weights to 3 or 4 bits instead of 8 bits. GPTQ employs layer-wise quantization with Optimal Brain Quantization (OBQ) [77] to update weights with inverse Hessian information. This technique enables quantizing GPT models with 175 billion parameters in roughly four GPU

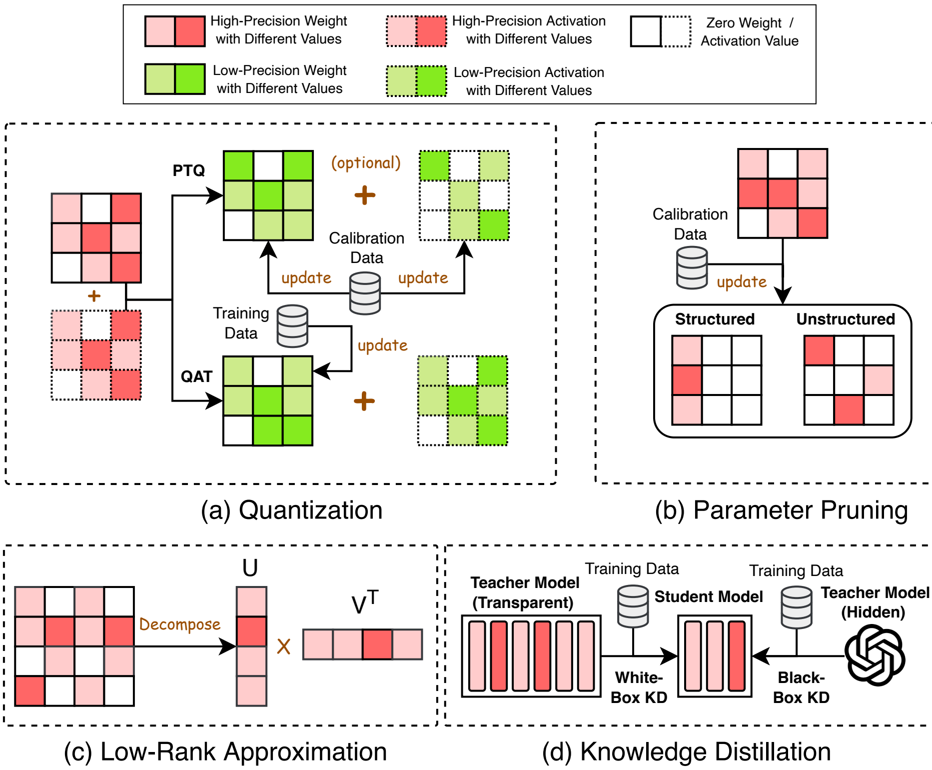


Fig. 5. Illustrations of model compression techniques for LLMs.

hours with minimal accuracy loss compared to the original model. Driven by the insights that quantization can be more effective when model weights and proxy Hessian matrices are incoherent, Chee et al. [27] propose QuIP, a post-training quantization method that applies incoherence processing to quantize LLMs to 2 bits per weight. Lin et al. [164] observe that there exists a small portion of model weights with larger activation magnitudes referred to as salient weights that determine the quantization loss. Based on this observation, they propose a weight quantization approach named activation-aware weight quantization (AWQ) to quantize LLMs while preserving the salient weights in high precision. Similarly, Lee et al. [143] also observe that activation outliers amplifies weight quantization loss. They propose outlier-aware weight quantization (OWQ) to identify those vulnerable weights with activation outliers and allocate high-precision to them. Dettmers et al. [63] propose Sparse-Quantized Representation (SpQR) to separate outlier weights that are prone to large quantization errors. These outlier weights are stored at higher precision levels, while the rest are compressed to 3-4 bits. They then propose a decoding scheme designed for the SpQR format, which accelerates the inference process on a token-by-token basis. Kim et al. [136] tackle the problem of outliers skewing the distribution of quantized weights, and propose FineQuant which employs an empirically crafted, heuristic-based approach to allocate varying levels of granularity to different weight matrices within the model.

- ***Weight-Activation Co-Quantization*** quantizes both model weights and activations. Due to the existence of outliers, activations are more difficult to quantize than model weights [19]. Yao et al. [329] propose ZeroQuant, which utilizes group-wise quantization for model weights and token-wise quantization for activations. However, ZeroQuant could not maintain accuracy for models with more than 175 billion parameters. To address this issue, Yao et al. [330] and Wu et al. [307] propose ZeroQuant-FP and ZeroQuant-V2 respectively which both utilize low-rank matrices to recover the accuracy drop. Xiao et al. [311] propose SmoothQuant which introduces a per-channel scaling transformation that migrates the quantization difficulty from activations to weights to achieve lossless quantization of weights and activations to 8 bits for LLMs up to 530 billion parameters. Guo et al. [94] pinpoint outliers are critical in weight and activation quantization but their nearby normal values are not. Based on this observation, they propose OliVe, which prunes normal values adjacent to the outliers so that the outliers can be encoded with low precision. Yuan et al. [334] identify the challenge of quantizing activations when different channels have disparate ranges. They propose RPTQ, which groups channels in activations that display similar value ranges and applies uniform quantization parameters to the values in each group. Liu et al. [168] propose QLLM, an adaptive channel reassembly method that efficiently tackles activation outliers and utilizes calibration data to offset the information loss incurred from quantization. Wei et al. [303] observe that the activation outliers in LLMs are asymmetric and tend to cluster in particular channels. Based on this observation, they propose Outlier Suppression+, which introduces operations that shift and scale channels individually to neutralize asymmetric outliers. Lastly, Ahmadian et al. [2] demonstrate that it is possible to suppress large activation outliers at scales as large as 52B. Given the right optimization choices during pre-training, they can quantize models ranging in size from 410M to 52B with minimal accuracy degradation.

Quantization-Aware Training (QAT). QAT quantizes LLMs during the training process itself so as to allow LLMs to learn quantization-friendly representations. Compared to PTQ, since QAT requires training using the complete training set to make up for its accuracy drop, it is much more expensive and time consuming. Tao et al. [267] aim to address quantization challenges in models like GPT-2 caused by uniform word embeddings, and propose QuantGPT, which combines contrastive distillation from a full-precision teacher model and logit distillation to a quantized student model during auto-regressive pretraining. LLM-QAT [176] uses data generated by LLMs itself to distill knowledge, with the aim of quantizing a student model. Specifically, it retains the original output distribution and is capable of quantizing any generative model, irrespective of its initial training data. Besides quantizing weights and activations, LLM-QAT also tackles the quantization of the key-value cache, a crucial step for enhancing throughput and accommodating long sequence dependencies in LLMs. BitNet [287] pioneers QAT for 1-bit LLMs, using low-precision binary weights and quantized activations, while keeping optimizer states and gradients high-precision during training, requiring only a replacement of the nn.Linear layer to train 1-bit weights from scratch.

2.1.2 *Parameter Pruning*.

Parameter pruning compresses LLMs by removing redundant model weights. Parameter pruning methods for LLMs can be categorized into structured pruning and unstructured pruning.

Structured Pruning. Structured pruning focuses on pruning structured patterns such as groups of consecutive parameters or hierarchical structures such as rows, columns, or sub-blocks of the LLM weight matrices. For example, LLM-Pruner [183] introduces a task-agnostic structured pruning strategy that selectively eliminates non-essential interconnected structures using gradient information. It utilizes a small amount of data to obtain the weight, parameter, and group importance of the coupled structure for LLaMA [276], and uses LoRA [111] to recover performance after pruning, showing competitive zero shot performance. Sheared LLaMA [310] proposes two techniques. The first technique is targeted structured pruning, which prunes a larger model to a designated target shape by eliminating layers, heads, and intermediate and hidden dimensions in a end-to-end fashion. The second technique is dynamic batch loading, which dynamically alters the components of the sampled data in each training batch based on losses in various domains. Through these two techniques, Sheared LLaMA is able to prune the LLaMA2-7B model down to 1.3B parameters. LoRAPrune [343] introduces a LoRA-based pruning criterion using LoRA's weights and gradients instead of pre-trained weights' gradients for importance estimation. By employing a structured iterative pruning process to eliminate excess channels and heads, LoRAPrune outperforms LLM-Pruner in efficiency at a 50% compression rate.

Unstructured Pruning. Unstructured pruning, on the other hand, focuses on pruning model weights individually, and thus has much more flexibility compared to structured pruning. Frantar and Alistarh [78] present SparseGPT, a one-shot LLM pruning approach that does not require retraining. It formulates pruning as a sparse regression problem and solves it by utilizing an approximate solver based on the inversion of the Hessian matrix. In doing so, SparseGPT reaches 60% unstructured sparsity even on models such as OPT-135B while experiencing only a slight reduction in perplexity. Sun et al. [260] propose Wanda which prunes weights based on the product values of weight magnitudes and their respective input activations. Compared to SparseGPT, Wanda neither relies on second-order information nor necessitates weight update, and performs competitively against SparseGPT. Shao et al. [242] propose to utilize Hessian sensitivity-aware mixed sparsity pruning to achieve a minimum of 50% sparsity in LLMs without retraining. This method adaptively assigns sparsity based on sensitivity to minimize the error induced by pruning while preserving the overall level of sparsity.

2.1.3 *Low-Rank Approximation*.

Low-rank approximation compresses LLMs by approximating the weight matrix $\mathbf{W}^{m\times n}$ of LLMs with low-rank matrices $\mathbf{U}$ and $\mathbf{V}$ such that $\mathbf{W} \approx \mathbf{U}\mathbf{V}^\top$, where $\mathbf{U} \in \mathbb{R}^{m\times r}$, $\mathbf{V} \in \mathbb{R}^{n\times r}$, and r is typically much smaller than m, n. In doing so, low-rank approximation reduces the number of parameters and enhances computational efficiency. In particular, Xu et al. [315] introduce TensorGPT which compresses the embedding layers of LLMs using Tensor-Train Decomposition (TTD). It transforms and breaks down each token embedding and creates an efficient embedding format named Matrix Product State (MPS) that can be efficiently computed in a distributed manner. LoSparse [161] aims to compress the coherent and expressive components within neurons through low-rank approximation while eliminating the incoherent and non-expressive elements via pruning the sparse matrix. It uses iteration training to calculate the important score of column neurons for pruning, outperforming conventional iterative pruning methods.

2.1.4 *Knowledge Distillation*.

Knowledge Distillation (KD) compresses LLMs by training a smaller student model to emulate the performance of the LLM as the teacher model such that the student model is computationally less expansive yet maintains a high level of performance similar to the teacher model. KD for LLMs can be categorized into white-box KD methods and black-box KD methods.

White-Box Knowledge Distillation. White-box KD refers to KD techniques where the parameters or logits of the teacher LLM are used in the distillation process [87]. For example, Baby LLaMA [274] trains an ensemble of GPT-2 and a collection of smaller LLaMA-1 models using the BabyLM dataset of 10M words. This ensemble is then distilled into a compact LLaMA model with 58 million parameters, which outperforms both its original teacher models as well as a comparable model that was trained without the use of distillation. Gu et al. [92] observe that conventional KD objectives, such as Kullback-Leibler divergence (KLD), may not be well suited for open text generation tasks due to their more complex output spaces compared to classification tasks. To address this issue, they propose MiniLLM that minimizes reverse KLD using the gradient of the objective function through policy gradient techniques [264]. This approach surpasses the performance of standard KD benchmarks on the 13-billion-parameter LLaMA-1 model [276]. Similarly, generalized knowledge distillation (GKD) [1] addresses the issue of distribution mismatch by drawing output sequences from the student model during training. GKD tackles the problem of model under-specification by optimizing different divergence measures, like reverse KL. This approach aims to produce samples from the student model that are probable within the teacher model's distribution. KPTD [202] demonstrates that KD methods can successfully transfer and disseminate knowledge from entity definitions into the parameters of a pre-trained language model. Specifically, it creates a transfer set by prompting the language model to generate text based on the definition of the entity. Then the models' parameters are updated to align the distribution of the student language model with that of the teacher model. TED [163] introduces a technique for layer-specific task distillation. It uses specially designed filters to align the internal states of both student and teacher models in each layer. These filters extract the relevant knowledge from the internal states that is beneficial for the specific task. TED shows considerable and steady gains in performance on both continual pre-training and fine-tuning. TSLD [134] leverages token-level distillation to enhance QAT, which overcomes the limitations of layer-to-layer KD in token prediction recovery by reforming intermediate representation and has successfully applied QAT to LLMs. Lastly, MiniMA [339] proposes a viewport towards the capacity gap in distilling LLMs, converting it into a principle through analysis and introducing a 3B Language Model that sets a new benchmark for compute-performance pareto frontier.

Black-Box Knowledge Distillation. Different from white-box KD, in black-box KD, only the outputs generated from the teacher LLM are used in the distillation process. Inspired by MetaICL and MetalICL [43, 189], where the language model is meta-trained in a wide range of tasks using in-context learning objectives and then fine-tuned for unseen tasks through in-context learning, Multitask-ICT [116] introduces a concept known as in-context learning distillation. This method aims to transfer the few-shot learning capabilities from the LLM teacher to the student model. Similarly, LI et al. [151] introduce a hybrid prompting technique that employs multi-task learning along with explanations generated by GPT-3 text-davinci-002 version [201]. This method is used to distill explanations into smaller models, achieving consistent and significant improvements over strong single-task fine-tuning benchmarks in different scenarios. Lion [125] introduces an adversarial distillation architecture aimed at enhancing the efficiency of knowledge transfer by incrementally improving the skill level of the student model. Specifically, it prompts LLMs to recognize challenging instructions and create new complex instructions for the student model, thereby establishing a three-phase adversarial cycle involving imitation, discrimination, and generation. DISCO [44] involves prompting a general LLM to produce phrasal perturbations. These generated perturbations are then filtered by a specialized teacher model to distill high-quality counterfactual data into smaller student models, allowing the smaller models to learn causal representations more reliably. Recently, some studies have shown that chain-of-thought (CoT) prompting can elicit language models to solve complex reasoning tasks step by step, with the aim of transfer this ability from LLMs into smaller models through black-box KD. For example, Fu et al. [82] aim to enhance the CoT math reasoning capabilities of smaller models. Specifically, they employ a method that involves instruct-tuning an student model (FlanT5) by distilling the reasoning pathways found in the GSM8K dataset from a LLM teacher (GPT-3.5 code-davinci-002 [35]). The small model is then selected based

Table 1. Pre-training costs of representative LLMs.

Model	Parameter Size	Data Scale	GPUs Cost	Training Time
GPT-3 [20]	175B	300B tokens	-	-
GPT-NeoX-20B [17]	20B	825GB corpus	96 A100-40G	-
OPT [346]	175B	180B tokens	992 A100-80G	-
BLOOM [239]	176B	366B tokens	384 A100-80G	105 days
GLM [336]	130B	400B tokens	786 A100-40G	60 days
LLaMA [276]	65B	1.4T tokens	2048 A100-80G	21 days
LLaMA-2 [277]	70B	2T tokens	A100-80G	71,680 GPU days
Gopher [223]	280B	300B tokens	1024 A100	13.4 days
LaMDA [273]	137B	768B tokens	1024 TPU-v3	57.7 days
GLaM [69]	1200B	280B tokens	1024 TPU-v4	574 hours
PanGu-α [337]	13B	1.1TB corpus	2048 Ascend 910	-
PanGu-$\sum$ [233]	1085B	329B tokens	512 Ascend 910	100 days
PaLM [51]	540B	780B tokens	6144 TPU-v4	-
PaLM-2 [7]	-	3.6T tokens	TPUv4	-
WeLM [258]	10B	300B tokens	128 A100-40G	24 days
Flan-PaLM [52]	540B	-	512 TPU-v4	37 hours
AlexaTM [254]	20B	1.3 tokens	128 A100	120 days
Codegeex [357]	13B	850 tokens	1536 Ascend 910	60 days
MPT-7B [272]	7B	1T tokens	-	-

on its average performance on three separate, withheld math reasoning datasets to confirm its ability to generalize well to new, out-of-distribution scenarios. Likewise, Distilling Step-by-Step [110] claims that to match the performance of LLMs, fine-tuning and distilling smaller models require substantial amounts of training data. To address this, it proposes a technique that uses CoT prompting to extract LLM rationales for extra guidance in training smaller models within a multi-task setting, achieving better performance compared to few shot prompted LLMs. Fine-tune-CoT [104] utilizes existing zero-shot CoT prompting techniques [139] to create rationales from LLMs. These rationales are then used to fine-tune smaller student models. The approach also introduces diverse reasoning, a method that employs stochastic sampling to generate a variety of reasoning solutions from teacher models, which serves to enrich the training data for the student models. SOCRATIC CoT [251] employs a method that breaks down the original problem into a series of smaller tasks and utilizes this decomposition to direct the intermediate steps of reasoning. This approach is used to train a pair of smaller, distilled models: one that specializes in dissecting the problem and another focused on solving these sub-problems. SCOTT [292] uses rationales generated by LLMs to train a student model under a counterfactual reasoning framework. This approach ensures that the student model does not overlook the provided rationales, thereby preventing it from making inconsistent predictions. SCoTD [150] presents a method called symbolic CoT distillation. It involves drawing CoT rationales from a LLM using unlabeled data instances. A smaller model is then trained to predict both the sampled rationales and the associated labels. Lastly, Peng et al. [207] utilize GPT-4 as a teacher model to generate English and Chinese instruction-based datasets to refine student LLMs such as LLaMA. Their results show that the 52K data points generated by GPT-4 are able to improve zero-shot performance compared to instruction-following data generated from previous state-of-the-art models.

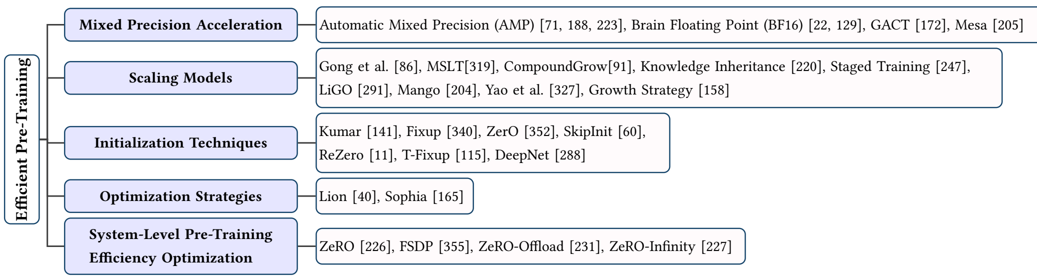


Fig. 6. Summary of efficient pre-training techniques for LLMs.

2.2 Efficient Pre-Training

As shown in Table 1, pre-training LLMs incurs high costs. Efficient pre-training aims to enhance the efficiency and reduce the cost of the LLM pre-training process. As summarized in Figure 6, efficient pre-training techniques can be grouped into four categories: mixed precision acceleration, scaling models, initialization techniques, and optimization strategies.

Mixed Precision Acceleration. Mixed precision acceleration enhances pre-training efficiency by using the low-precision model for forward and backward propagation and converting the calculated low-precision gradients to high-precision ones for updating the original high-precision weights. For example, Micikevicius et al. [188] propose Automatic Mixed Precision (AMP) to keep a master copy of weights in full-precision FP32 for updates, whereas weights, activations, and gradients are stored in FP16 for arithmetic operations. Notably, the improved version of AMP [71] optimizer has eliminated the copy of FP32 weights, but the optimizer (AdamW) still use FP32 internally. However, Rae et al. [223] demonstrate that FP16 results in accuracy loss. To counteract this performance drop, Brain Floating Point (BF16) was proposed [22, 129], which achieves better performance by assigning more bits to the exponent and fewer to the significant bits. Lastly, recent studies [172, 205] have shown that combining mixed-precision acceleration with activation compressed training (ACT) can further facilitate memory-efficient Transformer pre-training.

Scaling Models. Techniques based on scaling models accelerate pre-training convergence and reduce training costs by using the weights of a small model to scale up to a large model. For example, Gong et al. [86] introduce Progressive Stacking to transfer knowledge from a simpler model to a more complex one and then uses progressive stacking to enhance the model's training efficiency and convergence speed. Yang et al. [319] observe that as the depth of the model increases through progressive stacking, the training speed however decreases. To address this issue, they propose multi-stage layer training (MSLT), which only updates the output and newly introduced top encoder layers while keeping the previously trained layers unchanged. Once all the layers have been trained, MSLT fine-tunes the entire model by updating each layer in just 20% of the total steps, making it more time-efficient than the traditional progressive stacking approach. Gu et al. [91] introduce CompoundGrow, which begins with the training of a small model and incrementally expands it using a mix of model growth techniques, including increasing input length, model breadth, and depth, leading to an acceleration in the pre-training process by up to 82.2%. Qin et al. [220] propose Knowledge Inheritance which employs knowledge distillation as an auxiliary supervision during pre-training. This aids in effectively training a larger model from a smaller teacher model, thereby enhancing both the speed of pre-training and the generalization ability. Shen et al. [247] introduce Staged Training that begins with a small model and progressively increases its depth and breadth through a growth operator, which includes model parameters, the state of the optimizer,

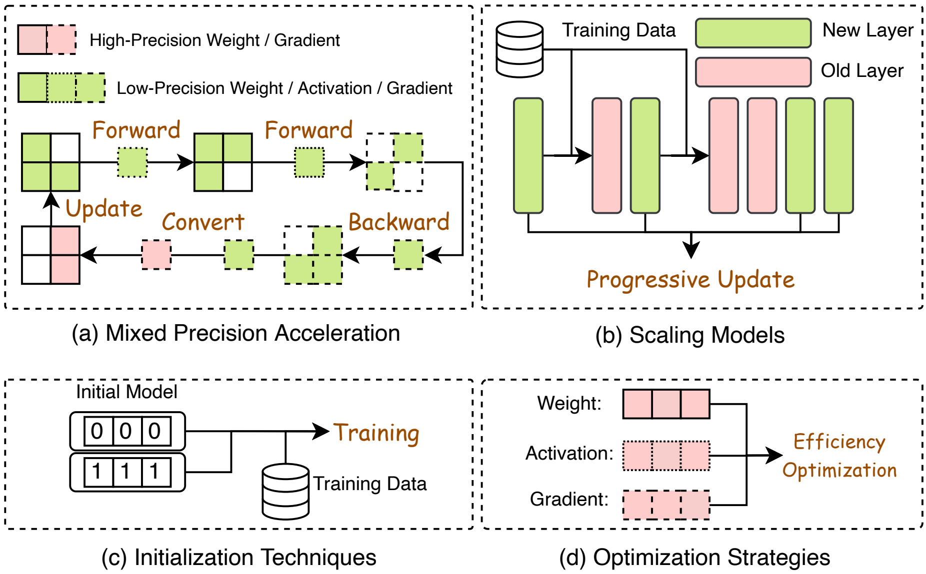


Fig. 7. Illustrations of efficient pre-training techniques for LLM.

and the learning rate schedule. By starting each phase with the results from the previous one, it effectively reuses computation, leading to a more efficient training process. Chen et al. [31] propose function-reserving initialization (FPI) and advanced knowledge initialization (AKI) to transfer the knowledge of a smaller pre-trained model to a large model to improve the pre-training efficiency of the large model. Specifically, FPI gives the larger model a behavior similar to that of the smaller model, laying a strong basis for optimization; and AKI promotes faster convergence by replicating weights from higher layers. Wang et al. [291] propose Linear Growth Operator (LiGO) that linearly maps the parameters of a smaller model to initiate a larger one, using a composition of width-and depth-growth operators, further enhanced with Kronecker factorization to capture architectural knowledge. Mango [204] introduces a technique that establishes a linear relationship between each weight of the target model and all weights of the pretrained model to boost acceleration capabilities. It also employs multi-linear operators to decrease computational and spatial complexity during pre-training. Drawing from these scaling techniques and the progressive pre-training [327], recent LLMs like FLM-101B [158] introduce a growth strategy to cut LLM training costs by expanding model structures offline and resuming from the previous stage's smaller model checkpoint.

Initialization Techniques. Initialization plays a key role in enhancing the efficiency of LLM pre-training since a good initialization can accelerate the convergence of the model. Most LLMs employ initialization techniques that were adopted in training smaller-scale models, such as conventional initialization techniques like [103, 141]. For example, initialization method introduced by Kumar [141] aims to balance input and output variances. Fixup [340] and ZerO [352] set the residual stem to zero, preserving signal identity. SkipInit [60] substitutes batch normalization with a zero-value multiplier. ReZero [11] adds zero-valued parameters to maintain identity, leading to faster convergence. T-Fixup [115] follows Fixup to adopt rescaling schemes for the initialization of residual blocks of Transformer models. DeepNet [288] adjusts the residual connection in deep Transformers using Post-LN-init, ensuring stable inputs to Layer-Normalization and mitigating gradient vanishing for stable optimization.

Optimization Strategies. Popular LLMs such as GPT-3 [20], OPT [346], BLOOM [239], and Chinchilla [105] are predominately pre-trained using Adam [137] or AdamW [178] as optimizers. However, both Adam and AdamW have a huge demand on memory and are computationally expensive. Some studies [40, 165] propose new optimizers to accelerate the pre-training of LLMs. Chen et al. [40] propose to leverage search techniques to traverse a large and sparse program space to discover optimizers for model training. The discovered optimizer, named Lion, is more memory-efficient than Adam as it only keeps track of the momentum. Liu et al. [165] propose Sophia as a lightweight second-order optimizer that outpaces Adam with doubling the pre-training speed. Sophia calculates the moving average of gradients and the estimated Hessian, dividing the former by the latter and applying element-wise clipping. It effectively moderates update sizes, addresses non-convexity and rapid hessian changes, enhancing both memory utilization and efficiency.

System-Level Pre-Training Efficiency Optimization. Due to the high demand on memory and compute resources, LLMs are usually pre-trained across multiple compute nodes in a distributed manner. Therefore, most techniques for improving pre-training efficiency at the system level focus on distributed training. Existing efficient distributed training methods that are used for general AI model training can also be applied to LLM pre-training. For example, data parallelism [155, 241] involves splitting the training dataset into multiple subsets on separate nodes. Each node computes gradients independently and then shares them with others to update the model parameters. Pipeline parallelism [117, 196] divides the input minibatch into several smaller batches, and then distributes the execution of these microbatches across multiple GPUs. Tensor parallelism [16, 197, 285, 317] splits the model's weight matrices across multiple nodes. Each node is responsible for executing the forward and backward passes with a segment of the model's weights and their computed results are then aggregated. Although these parallelism techniques tackle the computing and memory constraints for training LLMs, they are still limited in maintaining computation, communication and development efficiency when fitting all of the runtime states, including gradients, optimizer states and activation states into limited memory. To bridge this gap, Zero Redundancy Data Parallelism (ZeRO) [226] provides three stages of optimization to partition the intermediate states during pre-training across different nodes. Specifically, ZeRO-1 only partitions the optimizer states, and ZeRO-2 partitions both the optimizer states and the gradients. Both ZeRO-1 and ZeRO-2 reduce runtime memory compared to data parallelism, while only consuming the same communication volume as data parallelism. ZeRO-3 provides a more aggressive partitioning that also splits the model parameter across the nodes compared with ZeRO-1 and ZeRO-2. Although runtime memory is further reduced through ZeRO-3, there is a modest 50% increase in communication overhead under this stage. Therefore, it is recommended to use ZeRO-3 within a node to minimize the communication time while using ZeRO-1 and ZeRO-2 across nodes. Fully Sharded Data Parallel (FSDP) [355] shares a similar idea for optimization, and designs a hybrid sharding strategy to allow users to define which nodes or processes to partition the gradients, parameter, and optimizer states across different nodes. In the case when the weight memory exceeds the aggregated memory that can be provided by all of the compute nodes, ZeRO-Offload [231] enables offloading to CPU for any stage of ZeRO, and ZeRO-Infinity [227] provides a way to offload to NVMe drives in addition to CPU memory. However, it is quite difficult to maintain performance using these two alternatives, as the data movement between CPU and GPU is slow.

2.3 Efficient Fine-Tuning

Efficient fine-tuning aims to enhance the efficiency of the fine-tuning process for LLMs. As shown in Figure 8, efficient fine-tuning methods can be grouped into parameter-efficient fine-tuning (PEFT), and memory-efficient fine-tuning (MEFT).

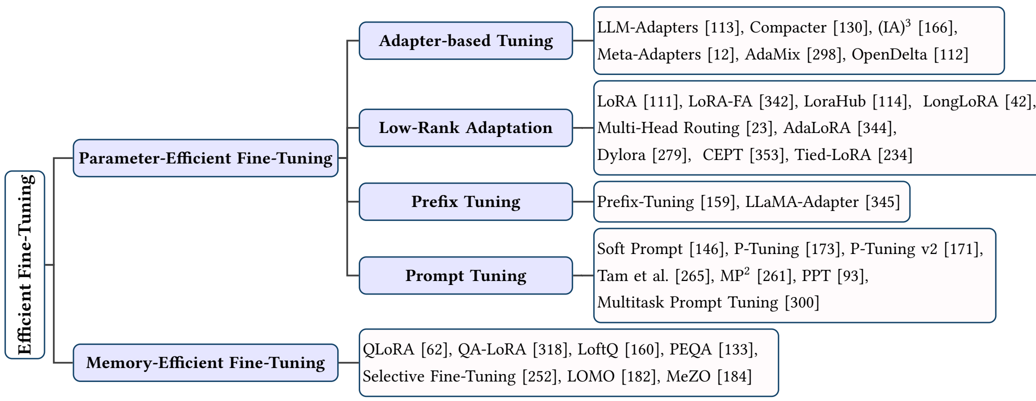


Fig. 8. Summary of efficient fine-tuning methods for LLMs.

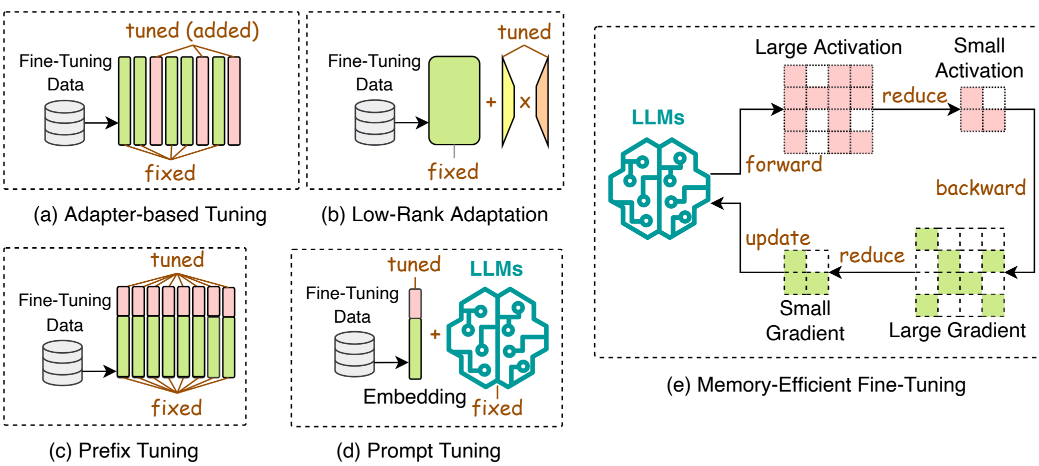


Fig. 9. Illustrations of Parameter-Efficient Fine-Tuning (a)-(d) and Memory-Efficient Fine-Tuning (e).

2.3.1 *Parameter-Efficient Fine-Tuning*.

Parameter-efficient fine-tuning (PEFT) aims to adapt an LLM to downstream tasks by freezing the whole LLM backbone and only updating a small set of extra parameters. In general, PEFT methods can be grouped into four categories: adapter-based tuning, low-rank adaptation, prefix tuning, and prompt tuning.

Adapter-based Tuning. Adapters are bottleneck-like trainable modules integrated into LLMs, which first down-project the input feature vector followed by a non-linear layer and then up-project back to the original size [108]. Adapter-based tuning includes both series adapters and parallel adapters. In series adapters, each LLM layer has two adapter modules added after its attention and feed-forward modules; parallel adapters position two adapter modules alongside the attention and feed-forward modules within each layer of the LLM. In particular, Hu et al. [113] propose LLM-Adapters, which integrates series or parallel adapters into LLMs for fine-tuning on different tasks. Karimi Mahabadi et al. [130] propose Compacter which unifies adapters, low-rank techniques, and the latest hyper-complex multiplication layers to achieve a balanced trade-off between the amount of trainable parameters and task performance. $(IA)^3$ [166] introduces a technique that scales activations using learned vectors, which outperforms few-shot in-context learning (ICL) in both accuracy and computational efficiency. Following meta-learning principles, Meta-Adapters [12] designs a resource-efficient fine-tuning technique for the few-shot scenario where it incorporates adapter layers that have been meta-learned into a pre-trained model, transforming the fixed pre-trained model into an efficient few-shot learning framework. AdaMix [298] takes inspiration from sparsely-activated mixture-of-experts (MoE) models [365] and proposes a mixture of adaptation modules to learn multiple views of the given task. Lastly, OpenDelta [112] is an open-source software library that offers a versatile and plug-and-play framework for implementing a range of adapter-based techniques, and is designed to be compatible with various LLMs architectures.

Low-Rank Adaptation. Low-Rank Adaptation (LoRA) [111] is a widely used PEFT approach for LLMs. Instead of directly adjusting the weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ as $\mathbf{W} \leftarrow \mathbf{W} + \Delta\mathbf{W}$, LoRA introduces two trainable low-rank matrices $\mathbf{A} \in \mathbb{R}^{m \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times n}$ and expresses $\Delta\mathbf{W}$ as $\Delta\mathbf{W} = \mathbf{A} \cdot \mathbf{B}$. As such, only the small matrices $\mathbf{A}$ and $\mathbf{B}$ are updated during fine-tuning, while the original large weight matrix remains frozen, making the fine-tuning process more efficient. Though effective, LoRA still requires the update of all the parameters of the low-rank matrices for all the layers of the LLM at every single fine-tuning iteration. To enhance the efficiency of LoRA, LoRA-FA [342] keeps the projection-down weights of $\mathbf{A}$ fixed while updating the projection-up weights of $\mathbf{B}$ in each LoRA adapter so that the weight modifications during fine-tuning are confined to a low-rank space, thereby eliminating the need to store the full-rank input activations. LoraHub [114] explores the composability of LoRA for the purpose of generalizing across different tasks. It combines LoRA modules that have been trained on various tasks with the goal of attaining good performance on tasks that have not been seen before. LongLoRA [42] extends LoRA to the long-context fine-tuning scenario. It introduces shift short attention (S^2-Attn), which effectively facilitates context expansion, showing that LoRA is effective for long context when utilizing trainable embedding and normalization. Multi-Head Routing (MHR) [23] extends LoRA to Mixture-of-Experts (MoE) architectures. It outperforms Polytropon [213] when operating with a similar parameter allocation. Notably, it achieves competitive performance while focusing on fine-tuning the routing function alone, without making adjustments to the adapters, demonstrating remarkable parameter efficiency. Zhang et al. [344] observe that many PEFT techniques neglect the differing significance of various weight parameters. To address this, they propose AdaLoRA which employs singular value decomposition to parameterize incremental updates and adaptively distributes the parameter budget based on the importance score of each weight matrix. Valipour et al. [279] identify that the rank in LoRA is static and cannot be adaptively adjusted during fine-tuning. To address this issue, they propose Dylora, which introduces a dynamic low-rank adaptation method that trains LoRA blocks across multiple ranks rather than just one by organizing the representations learned by the adapter module based on their ranks. Different from above-mentioned methods that mainly apply PEFT to full-size LLMs, CEPT [353] introduces a new framework that utilizes compressed LLMs. Specifically, it assesses how prevalent LLM compression methods affect PEFT performance and subsequently implements strategies for knowledge retention and recovery to counteract the loss of knowledge induced by such compression techniques. Furthermore, Tied-LoRA [234] uses weight tying and selective training to further increase parameter efficiency of LoRA.

Prefix Tuning. Prefix-Tuning [159] adds a series of trainable vectors, known as prefix tokens, to each layer in an LLM. These prefix tokens are tailored to specific tasks and can be treated as virtual word embeddings. LLaMA-Adapter [345] incorporates a set of trainable adaptation embeddings and attaches them to the word embeddings in the upper layers of the LLMs. A zero-initialized attention scheme with zero gating is also introduced. It dynamically incorporates new guiding signals into LLaMA-1 while retaining its pre-trained knowledge.

Prompt Tuning. Different from prefix tuning, prompt tuning incorporates trainable prompt tokens at the input layer. These tokens can be inserted either as a prefix or anywhere within the input tokens. Soft Prompt [146] keeps the entire pre-trained model fixed while adding an extra k trainable tokens at the beginning of the input text for each downstream task. It outperforms few-shot prompts and narrows the performance gap compared to full model fine-tuning. P-Tuning [173] utilizes a small number of parameters as prompts, which are processed by a prompt encoder before being used as input for pre-trained LLMs. Instead of searching for discrete prompts, P-Tuning fine-tunes these prompts through gradient descent and improves performance on a wide range of NLU task. Liu et al. [171] observe that earlier versions of prefix tuning struggle with complex sequence labeling tasks. To address this, they propose P-Tuning v2 which enhances prefix tuning by introducing continuous prompts at each layer of the pre-trained model, rather than at the input layer only. This modification has proven effective in boosting performance across various parameter sizes for tasks related to natural language understanding. Tam et al. [265] introduce efficient prompt tuning for text retrieval, updating just 0.1% of parameters and outperforming traditional full-parameter update methods in diverse domains. Sun et al. [261] claim that prompt tuning tends to struggle in few-shot learning scenarios, and thus propose MP^2 that pre-trains a collection of modular prompts using multitask learning. These prompts are then selectively triggered and assembled by a trainable routing mechanism for specific tasks. As a result, MP^2 can quickly adapt to downstream tasks by learning how to merge and reuse pretrained modular prompts. Different from MP^2, PPT [93] attributes the performance degradation of prompt tuning in few-shot learning to the poor initialization of soft prompt, and thus proposes to add the soft prompt into the pre-training stage for a better initialization. Multitask Prompt Tuning [300] harnesses the knowledge of the various tasks through the use of prompt vectors in a multitask learning setting. Specifically, it initially learns a single, transferable prompt by extracting knowledge from various task-specific source prompts, and then applies multiplicative low-rank updates to this prompt to effectively tailor it for each downstream task. By doing this, Multitask Prompt Tuning is able to attain performance levels that are competitive compared to full fine-tuning methods.

2.3.2 Memory-Efficient Fine-Tuning.

As the parameters of LLMs expand, the sizes of memory needed for fine-tuning also increase, making memory a significant hurdle in fine-tuning. Consequently, minimizing memory usage in fine-tuning for improving efficiency has also emerged as a critical topic. Dettmers et al. [62] propose QLoRA which first quantizes the model into a 4-bit NormalFloat data type, and then fine-tunes this quantized model with added low-rank adapter (LoRA) weights [111]. In doing so, QLoRA reduces memory usage during fine-tuning without performance degradation compared to standard full-model fine-tuning. QA-LoRA [318] improves QLoRA by introducing group-wise operators that improve quantization flexibility (each group is quantized separately) while reducing adaptation parameters (each group utilizes shared adaptation parameters). Similarly, LoftQ [160] combines model quantization with singular value decomposition (SVD) to approximate the original high-precision pre-trained weights. As a result, it offers a favorable initialization point for subsequent LoRA fine-tuning, leading to enhancements over QLoRA. PEQA [133] introduces a two-stage approach to quantization-aware fine-tuning. In the first stage, the parameter matrix for each fully connected layer is quantized into a matrix of low-bit integers along with a scalar vector. In the second stage, the low-bit matrix remains unchanged, while fine-tuning is focused solely on the scalar vector

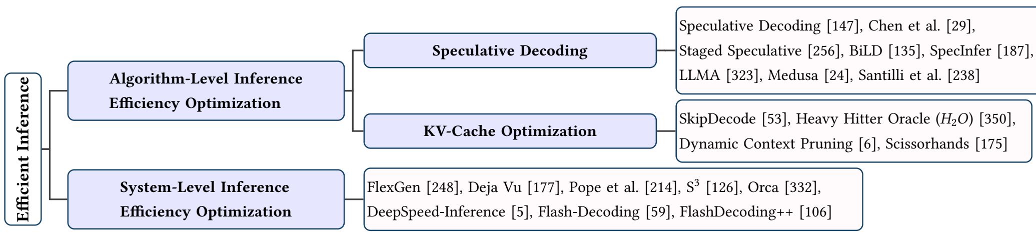
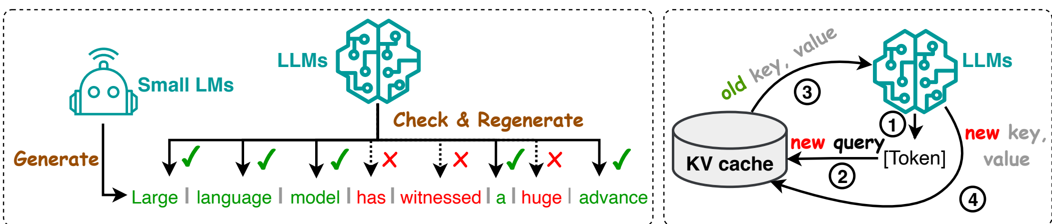


Fig. 10. Summary of efficient inference techniques for LLMs.



(a) Speculative Decoding (b) KV-cache Optimization

Fig. 11. Illustrations of algorithm-level efficiency optimization techniques for LLM inference.

for each specific downstream task. Employing this two-stage approach, PEQA not only minimizes memory usage during fine-tuning but also speeds up inference time by maintaining weights in a low-bit quantized form. Simoulin et al. [252] propose Selective Fine-Tuning which minimizes memory usage by specifically preserving a subset of intermediate activations from the forward pass for which the calculated gradients are nonzero. Notably, this approach delivers performance equivalent to full fine-tuning while using just up to one-third of the GPU memory required otherwise. Lv et al. [182] introduce LOMO, which minimizes memory consumption during fine-tuning by combining gradient calculation and parameter updating into a single step. As such, LOMO eliminates all components of the optimizer state, lowering the memory requirements for gradient tensors to $O(1)$. MeZO [184] improves the zeroth-order method [255] for gradient estimation using only two forward passes. This enables efficient fine-tuning of LLMs with memory requirements similar to inference and supports both full-parameter and PEFT methods like LoRA [111] and prefix tuning [159], enabling MeZO to train a 30-billion parameter model on a single A100 80GB GPU.

2.4 Efficient Inference

Efficient inference aims to enhance the efficiency of the inference process for LLMs. As summarized in Figure 10, efficient inference techniques can be grouped into techniques at the algorithm level and system level.

Algorithm-Level Inference Efficiency Optimization. Techniques that enhance LLM inference efficiency at the algorithm level include speculative decoding and KV-cache optimization.

- ***Speculative Decoding.*** Speculative decoding (i.e., speculative sampling) [147] is a decoding strategy for autoregressive language models that speed up sampling by parallel token computation through using smaller draft models to create speculative prefixes for the larger target model. Chen et al. [29] propose to run a faster autoregressive model K times and then evaluate the preliminary output with the large target LLM. A tailored rejection sampling strategy is employed to approve a selection of the draft tokens in a left-to-right order, thereby recapturing the distribution of the target model during the procedure. Staged Speculative [256] transforms the speculative batch into a tree structure representing potential token sequences. This restructuring aims to expedite the generation of larger and improved speculative batches. It introduces an additional phase for speculative decoding of the initial model, thereby enhancing overall performance. BiLD [135] optimizes speculative decoding through two innovative strategies: the fallback policy that permits the smaller draft model to waive control to the larger target model when it lacks sufficient confidence, and the rollback policy that enables the target model to revisit and rectify any inaccurate predictions made by the smaller draft model. SpecInfer [187] speeds up inference by employing speculative inference techniques and token tree validation. Its core idea involves merging a range of small speculative models that have been fine-tuned collectively to collaboratively forecast the output of the LLM, which is then used to validate all the predictions. LLMA [323] chooses a text segment from a closely related reference and duplicates its tokens into the decoder. It then concurrently assesses the suitability of these tokens as the decoding output within a single decoding step. This approach results in a speed increase of more than two times for LLMs while maintaining the same generated results as traditional greedy decoding. Medusa [24] involves freezing the LLM backbone, fine-tuning additional heads, and using a tree-based attention mechanism to process predictions in parallel to speed up the decoding process. Lastly, Santilli et al. [238] propose parallel decoding including the Jacobi and Gauss-Seidel fixed-point iteration methods for speculative decoding. Among these strategies, Jacobi decoding was extended into Lookahead decoding [81] to enhance the efficiency of LLMs.

- ***KV-Cache Optimization.*** Minimizing the repeated computation of Key-Value (KV) pairs during the inference process of LLMs is also key to enhancing the inference efficiency. Corro et al. [53] propose SkipDecode, a token-level early exit approach that utilizes a unique exit point for each token in a batch at every sequence position, and skips the lower and middle layers to accelerate the inference process. Zhang et al. [350] point out that KV-cache is scaling linearly with the sequence length and batch size. They propose a KV cache eviction strategy that formulates the KV cache eviction as a dynamic sub-modular problem and dynamically retains a balance between recent and important tokens, reducing the latency for LLMs inference. Dynamic Context Pruning [6] utilizes a learnable mechanism to identify and remove non-informative KV-cache tokens. In doing so, it not only enhances efficiency but also improves interpretability. Liu et al. [175] underscore the Persistence of Importance Hypothesis, suggesting that only tokens that were crucial at an earlier phase will have a significant impact on subsequent stages. Based on this theory, they propose Scissorhands that introduces a streamlined algorithm for LLM inference using a compact KV-cache.

System-Level Inference Efficiency Optimization. The efficiency of LLM inference can also be optimized at the system level. For example, FlexGen [248] is a high-throughput inference engine that enables the execution of LLMs on GPUs with limited memory. It uses a linear programming-based search approach to coordinate various hardware, combining the memory and computation from GPU, CPU, and disk. Furthermore, FlexGen quantizes the weights and attention cache to 4 bits, increasing the inference speed of OPT-175B [346] on a single 16GB GPU. Deja Vu [177] presents the notion of contextual sparsity, which is a collection of MLP and attention modules that produce the same result as a dense model, but with fewer components. This technique trains predictors to identify the sparsity and then uses kernel fusion and memory coalescing to speed up the inference process. Pope et al. [214] develop a simple analytical framework to select the best multi-dimensional partitioning methods optimized for TPU v4 slices based on the application

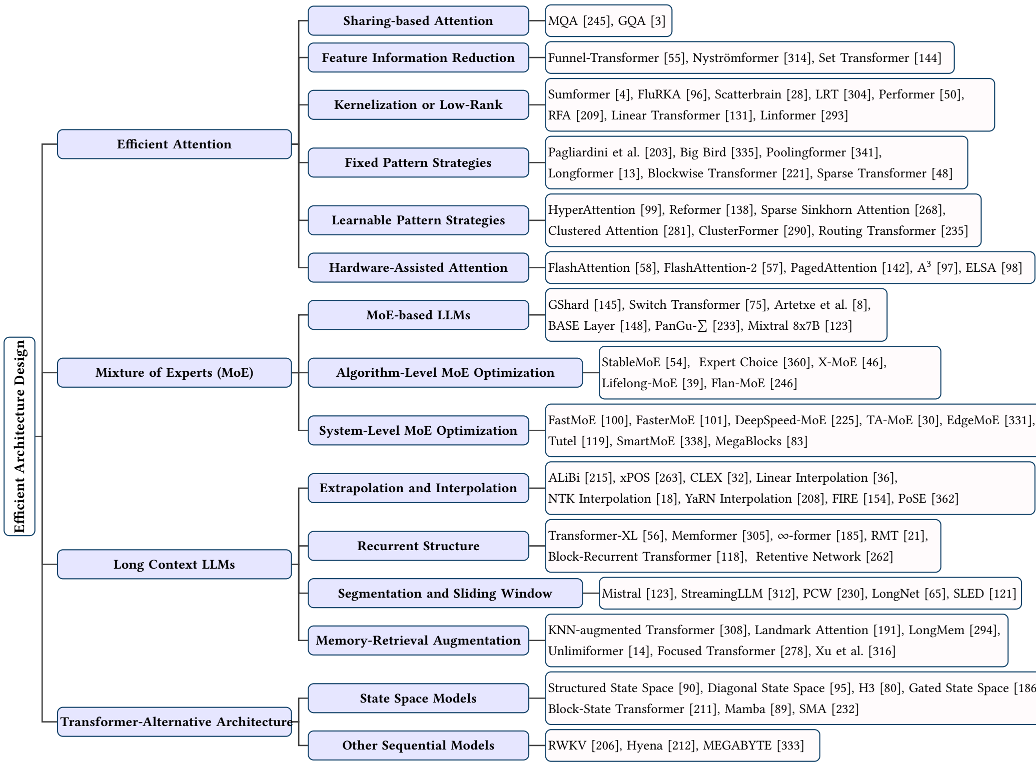


Fig. 12. Summary of efficient architecture designs for LLMs.

requirements. By combining this with some existing low-level optimizations, they have achieved greater efficiency on PaLM [51] in comparison to the FasterTransformer [199] standards. S^3 [126] has created a system that is aware of the output sequence beforehand. It can anticipate the length of the sequence and arrange generation requests accordingly, optimizing the utilization of device resources and increasing the rate of production. Orca [332] employs iteration-level scheduling to decide batch sizes. When a sequence in a batch is completed, it is substituted with a new one, resulting in improved GPU utilization compared to static batching. DeepSpeed-Inference [5] is a multi-GPU inference approach that is designed to enhance the efficiency of both dense and sparse Transformer models when they are contained within the collective GPU memory. Furthermore, it provides a mixed inference technique that utilizes CPU and NVMe memory, in addition to GPU memory and computation, guaranteeing high-throughput inference even for models that are too large to fit in the combined GPU memory. Flash-Decoding [59] is a technique that boosts the speed of long-context inference by breaking down keys/values into smaller pieces, computing attention on these pieces in parallel, and then combining them to generate the final output. FlashDecoding++ [106] supports mainstream language models and hardware backends through asynchronous softmax, double buffering for flat GEMM optimization, and heuristic dataflow, resulting in up to 4.86x and 2.18x acceleration on NVIDIA and AMD GPUs respectively compared to HuggingFace implementations.

2.5 Efficient Architecture Design

Efficient architecture design for LLMs refers to the strategic optimization of model architecture and computational processes to enhance performance and scalability while minimizing resource consumption. Figure 12 summarizes efficient architecture designs for LLMs.

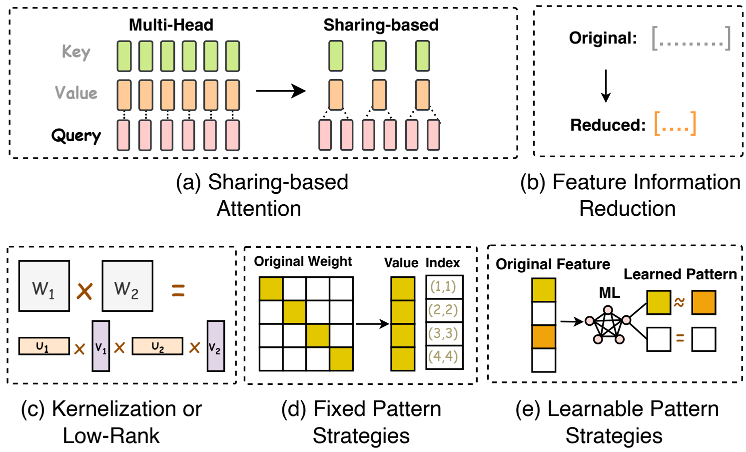


Fig. 13. Illustrations of attention optimizations.

2.5.1 *Efficient Attention*.

The quadratic time and space complexity of attention modules considerably slows down the pre-training, inference and fine-tuning of LLMs [132]. A lot of techniques have been proposed to make attention lightweight for more efficient execution. These techniques can be generally categorized as sharing-based attention, feature information reduction, kernelization or low-rank, fixed pattern strategies, learnable pattern strategies, and hardware-assisted attention.

Sharing-based Attention. Sharing-based attention aims to accelerate attention computation during inference through different KV heads sharing schemes. For example, LLaMA-2 [277] optimizes the autoregressive decoding process by using multi-query attention (MQA) [245] and grouped-query attention (GQA) [3]. In contrast to multi-head attention, which uses several attention layers (heads) simultaneously with distinct linear transformations for queries, keys, values, and outputs, MQA has all its heads sharing one set of keys and values. While MQA utilizes only one key-value head to speed up decoder inference, it might compromise quality. To address this, GQA offers a modified version of MQA by employing more than one key-value heads but fewer than the total number of query heads to enhance the inference quality.

Feature Information Reduction. The principle of feature information reduction, as evidenced by models such as Funnel-Transformer [55], Nyströmformer [314], and Set Transformer [144], is to cut computation demands by reducing feature information within a sequence, which leads to a proportionate reduction in required computation resources. For example, Funnel-Transformer [55] reduces the sequence length of hidden states to decrease computational costs, while its decoder can reconstruct deep representations for each token from this compressed sequence.

Kernelization or Low-Rank. Kernelization or low-rank techniques adopted by models such as Sumformer [4], FluRKA [96], Scatterbrain [28], Low-Rank Transformer (LRT) [304], Performer [50], Random Feature Attention (RFA) [209], Linear Transformer [131], and Linformer [293], , enhance computational efficacy by utilizing low-rank representations of the self-attention matrix or by adopting attention kernelization techniques. Specifically, low-rank methods focus on compacting the dimensions of attention keys and values. For example, Linformer [293] proposes to segment scaled dot-product attention into smaller units via linear projection. Kernelization, a variant of low-rank technique, focuses on approximating the attention matrix [49]. For example, Performer [50] condenses softmax attention-kernels using positive orthogonal random features. Sumformer [4] approximates the equivariant sequence-to-sequence function, offering a universal solution for both Linformer and Performer.

Fixed Pattern Strategies. Fixed pattern strategies adopted by models such as [203], Big Bird [335], Poolingformer [341], Longformer [13], Blockwise Transformer [221], and Sparse Transformer [48] improve efficiency by sparsifying the attention matrix. This is achieved by confining the attention scope to predetermined patterns, such as local windows or fixed-stride block patterns. For instance, Longformer [13]'s attention mechanism, designed as an alternative to conventional self-attention, merges local windowed attention with globally oriented attention tailored to specific tasks. Pagliardini et al. [203] have expanded FlashAttention [58] to support a broad spectrum of attention sparsity patterns, including key-query dropping and hashing-based attention techniques.

Learnable Pattern Strategies. Learnable pattern strategies adopted by models such as HyperAttention [99], Reformer [138], Sparse Sinkhorn Attention [268], Clustered Attention [281], ClusterFormer [290], and Routing Transformer [235] improve efficiency by learning token relevance and subsequently grouping tokens into buckets or clusters. As an example, HyperAttention [99] proposes a parameterization for spectral approximation and employs two key metrics: the maximal column norm in the normalized attention matrix and the row norm ratio in the unnormalized matrix after large entry removal. It also utilizes the learnable sort locality-sensitive hashing (sortLSH) technique and fast matrix multiplication via row norm sampling. Their experiment results show that HyperAttention enhances both inference and training speeds for LLMs with only minimal performance degradation.

Hardware-Assisted Attention. Besides algorithmic approaches that sparsify attentions and thereby streamline the computation of the attention matrix, several studies concentrate on realizing efficient and lightweight attention mechanisms from hardware aspects. For example, FlashAttention [58] and FlashAttention-2 [57] aim to reduce the communication times between GPU high-bandwidth memory (HBM) and GPU on-chip SRAM when calculating the attention module in LLMs. Instead of transmitting the values and results between HBM and SRAM multiple times as is done in the standard attention mechanism, FlashAttention combines all the attention operations into one kernel and tiles the weight matrices into smaller blocks to better fit the small SRAM. As a result, only one communication is required to process each attention block, significantly increasing the efficiency for processing the entire attention block. Inspired by virtual memory and paging techniques, PagedAttention [142] enables the storage of continuous keys and values in non-contiguous memory space. Specifically, PagedAttention divides the KV cache of each sequence into blocks, each containing the keys and values for a fixed number of tokens. During the attention computation, the PagedAttention kernel manages these blocks efficiently by maintaining a block table to reduce memory fragmentation. Specifically, the contiguous logical blocks of a sequence are mapped to non-contiguous physical blocks via the table and the table automatically allocates a new physical block for every newly generated token. This reduces the amount of memory wasted when generating new tokens, thus improving its efficiency. A^3 [97] introduces an innovative candidate selection process that reduces the number of keys and offers a custom hardware pipeline that taps into parallelism to speed up approximated attention techniques, further enhancing their efficiency. ELSA [98] uses Kronecker decomposition to approximate the attention module, which not only reduces its complexity, but also makes it more suitable for parallelization on hardware, thus making it more efficient when used for inference.

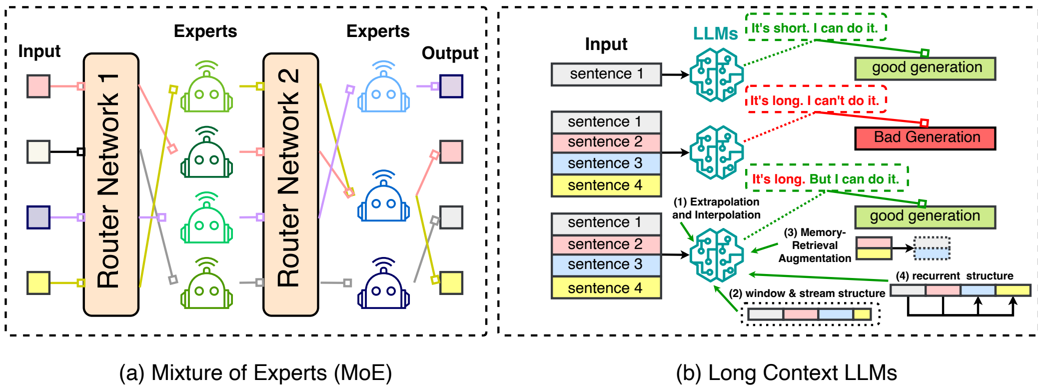


Fig. 14. Illustrations of Mixture of Experts (MoE) and Long Context LLMs.

2.5.2 *Mixture of Experts (MoE).*

Mixture of Experts (MoE) represents a sparse methodology utilized prominently in large-scale models like LLMs. It operates on the principle of segmenting a designated task into several sub-tasks, and then developing numerous smaller, specialized models, dubbed *experts*, with each honing in on a distinct sub-task. Subsequently, these experts collaborate to deliver a consolidated output. For pre-traning or fine-tuning, MoE helps to manage a huge number of parameters efficiently, enhancing the model's capacity and potentially its performance while keeping the computational and memory requirements relatively manageable. For inference, MoE decreases the inference time by not engaging all experts simultaneously, but rather activating only a select few. Additionally, MoE is capable of minimizing communication between devices in model-distributed scenarios by allocating each expert to an individual accelerator; communication is only necessary between the accelerators that host the router and the relevant expert model [128].

MoE-based LLMs. Several MoE-based LLMs have been proposed. For example, GShard [145] is a MoE-based LLM that offers a refined method to articulate a variety of parallel computation frameworks with minor modifications to the existing model code. It also amplifies a multilingual neural machine translation Transformer model with Sparsely-Gated MoE beyond 600 billion parameters through automatic sharding. Switch Transformer [75] brings forth a switch routing algorithm and crafts intuitively enhanced models, lowering communication and computational expenditures. It encompasses up to one trillion parameters, dividing tasks among up to 2,048 experts, thereby illustrating the scalability and efficacy of the MoE framework. Artetxe et al. [8] scale sparse language models to 1.1T parameters, discerning superior performance up to this scale in language modeling, zero-shot and few-shot learning in comparison to dense models. This suggests that sparse MoE models are a computationally efficient substitute for traditionally employed dense architectures. BASE Layer [148] defines token-to-expert allocation as a linear assignment problem, allowing an optimal assignment where each expert acquires an equal number of tokens. PanGu-Σ [233] is a MoE-based LLM with 1.085T parameters, transitioned from the dense Transformer model to a sparse one with Random Routed Experts (RRE), and effectively trains the model over 329B tokens utilizing Expert Computation and Storage Separation (ECSS). Lastly, Mixtral 8x7B [123] is a MoE with 46.7B total parameters. By leveraging the advantage of MoE architecture, Mixtral 8x7B outperforms LLaMA-2 70B on most benchmarks such as MMLU, MBPP, and GSM-8K with 6x faster inference by only using 12.9B parameters of the model per token for inference.

Algorithm-Level MoE Optimization. The efficiency of MoE-based LLMs can be improved at the algorithm level. The technique termed Expert Choice [360] allows experts to pick the top-k tokens instead of having tokens choose the top-k experts, implying that each token can be directed to a variable number of experts while each expert maintains a fixed bucket size. This method demonstrates higher performance in the GLUE and SuperGLUE benchmarks, and outperforms the T5 dense model in 7 out of the 11 tasks. StableMoE [54] identifies the issue of altering target experts for identical input during training and addresses this by creating two training phases. Initially, it cultivates a balanced routing strategy, which is then distilled into a decoupled lightweight router. In the following phase, this distilled router is used for a fixed token-to-expert assignment, ensuring a stable routing strategy. X-MoE [46] notes that earlier routing mechanisms foster token clustering around expert centroids, indicating a tendency toward representation collapse. It proposes to estimate the routing scores between tokens and experts on a low-dimensional hyper-sphere. Lifelong-MoE [39] finds that MoE increases the capacity of the model to adapt to different corpus distributions in online data streams without extra computational cost, simply by incorporating additional expert layers and suitable expert regularization. This facilitates continuous pre-training of a MoE-based LLM on sequential data distributions without losing previous knowledge. Lastly, Flan-MoE [246] promotes the amalgamation of MoE and instruction tuning, observing that MoE models gain more from instruction tuning compared to dense models. In particular, Flan-MoE effectively enlarges language models without demanding an increase in computational resources or memory requirements.

System-Level MoE Optimization. Several system-level optimization techniques have been developed to accelerate the training and inference of MoE-based LLMs. For example, FastMoE [100] is a distributed MoE training system built on PyTorch, compatible with common accelerators. This system offers a hierarchical interface that allows both flexible model design and easy adaptation to various applications, such as Transformer-XL and Megatron-LM. FasterMoE [101] introduces a performance model that predicts latency and analyzes end-to-end performance through a roofline-like methodology. Utilizing this model, it presents a dynamic shadowing technique for load balancing, a concurrent fine-grained schedule for operations, and a strategy to alleviate network congestion by adjusting expert selection for model training. DeepSpeed-MoE [225] has designed a Pyramid-Residual MoE (PR-MoE) to enhance both the training and the inference efficiency of the MoE model parameter. PR-MoE is a dense-MoE hybrid that employs residual connections to optimally utilize experts, managing to reduce the parameter size by up to 3x without sacrificing quality or compute requirements. Additionally, it proposes a distilled variant, Mixture-of-Students (MoS), which can trim model size by up to 3.7x while retaining quality. TA-MoE [30] highlights that current MoE dispatch patterns do not fully leverage the underlying heterogeneous network environment and thus introduces a topology-aware routing strategy for large-scale MoE training that dynamically modifies the MoE dispatch pattern based on the network topology, making it outperform FastMoE, FasterMoE, and DeepSpeed-MoE. EdgeMoE [331] presents an on-device inference engine tailored for MoE-based LLMs. It optimizes memory and computation for inference by distributing the model across different storage levels. Specifically, non-expert model weights are stored directly on the edge device, while expert weights are kept externally and only loaded into the device's memory when necessary. Tutel [119] is a scalable stack for MoE with adaptive parallelism and pipelining features to accelerate training and inference. It employs a consistent layout for MoE parameters and input data, supporting switchable parallelism and dynamic pipelining without any mathematical inconsistencies or tensor migration costs, thus enabling free run-time optimization. SmartMoE [338] focuses on distributed training for MoE. In the offline stage, SmartMoE constructs a search space of hybrid parallelism strategies. In the online stage, it incorporates light-weight algorithms to identify the optimal parallel strategy. Lastly, MegaBlocks [83] transforms MoE-oriented computation with block-sparse operations and creates block-sparse GPU kernels to optimize MoE computation on hardware. This leads to training time up to 40% faster compared to Tutel and 2.4x faster than dense DNNs trained with Megatron-LM.

2.5.3 *Long Context LLMs*.

In many real-world applications, such as multi-turn conversations and meeting summarization, existing LLMs are often required to comprehend or generate context sequences that are much longer than what they have been pre-trained with and may result in a degragation in accuracy due to the poor memorization for the long context. The most obvious and direct way to address this issue is to fine-tune LLMs with similar long-sequence data, which is time consuming and computation intensive. Recently, various new methods have been developed to enable LLMs to adapt to longer context lengths in a more efficient way, including extrapolation and interpolation, recurrent structure, window segment and sliding structure, and memory-retrieval augmentation.

Extrapolation and Interpolation. Standard positional encoding methods like absolute positional embeddings (APE) [280], learned positional embeddings (LPE) [286], relative positional embeddings (RPE) [244], relative positional bias [224], and rotary position embeddings (RoPE) [259] have advanced the integration of positional information in LLMs. For example, LPE has been used by GPT-3 [20] and OPT [346]; RPE was used by Gopher [223] and Chinchilla [105], whereas RoPE was used by LLaMA-1 and GLM-130B. However, it is still challenging to train LLMs on sequences with a limited maximum length while still ensuring them to generalize well on significantly longer sequences during inference. Given that, techniques based on positional extrapolation [32, 215, 263] and positional interpolation [36, 154, 208] have been proposed.

Positional extrapolation strategies extend the encoding of positional information beyond what the model has explicitly learned during training. For example, ALiBi [215] applies attention with linear biases to attain extrapolation for sequences that exceed the maximum length seen during training. Through applying negatively biased attention scores, with a linearly diminishing penalty based on the distance between the pertinent key and query, as opposed to using position embeddings, it can facilitate efficient length extrapolation. Different from ALiBi [215], xPOS [263] characterizes attention resolution as a marker for extrapolation and utilizes a relative position embedding to enhance attention resolution, thereby improving length extrapolation. However, these techniques have not been implemented in some of the recent LLMs such as GPT-4 [200], LLaMA [276], or LLaMA-2 [277]. CLEX [32] proposes to generalize position embedding scaling with ordinary differential equations to model continuous dynamics over length scaling factors. By doing so, CLEX gets rid of the limitations of existing positional extrapolation scaling methods to enable long-sequence generation.

Positional interpolation strategies, on the other hand, reduce the scale of input position indices and extend the context window sizes, allowing LLMs to maintain their performance over longer text sequences. For example, Chen et al. [36] highlight that extending beyond the trained context length might impair the self-attention mechanism. They suggest a method that reduces the position indices through linear interpolation, aligning the maximum position index with the prior context window limit encountered during the pre-training phase. NTK interpolation [18] modifies the base of the RoPE, effectively changing the rotational velocity of each RoPE dimension. YaRN interpolation [208] uses a ramp function to blend linear and NTK interpolation in varying proportions across dimensions and incorporates a temperature factor to counteract distribution shifts in the attention matrix due to long inputs. FIRE [154] proposes a functional relative position encoding using learnable mapping of input positions to biases and progressive interpolation, ensuring bounded input for encoding functions across all sequence lengths to enable length generalization. PoSE [362] proposes positional skip-wise training that smartly simulates long inputs using a fixed context window and design distinct skipping bias terms to manipulate the position indices of each chunk. This strategy reduces memory and time overhead compared with full-length fine-tuning.

Recurrent Structure. LLMs' ability to manage long sequences can also be enhanced through recurrence structure. For example, Transformer-XL [56] presents a segment-level recurrence mechanism and utilizes enhanced relative positional encoding to capture long-term dependencies and address the long-context fragmentation issue. Memformer [305] leverages an external dynamic memory for encoding and retrieving past information, achieving linear time and constant memory space complexity for long sequences. It also proposes Memory Replay Back-Propagation (MRBP) to facilitate long-range back-propagation through time with significantly lower memory requirements. ∞-former [185] presents a Transformer model augmented with unbounded long-term memory (LTM), employing a continuous space attention framework to balance the quantity of information units accommodated in memory against the granularity of their representations. Recurrent Memory Transformer (RMT) [21] uses a recurrence mechanism to retain information from the past segment level by incorporating special memory tokens into the input or output sequence, demonstrating superior performance compared to Transformer-XL in long context modeling. Block-Recurrent Transformers [118] utilize self-attention and cross-attention to execute a recurrent function across a broad set of state vectors and tokens so as to model long sequences through parallel computation. Lastly, Retentive Network [262] introduces a multi-scale retention mechanism as an alternative to multi-head attention. By encompassing parallel and chunk-wise recurrent representations, it results in effective scaling, allows for parallel training, and achieves training parallelization and constant inference cost, while offering linear long-sequence memory complexity compared to other Transformer models.

Segmentation and Sliding Window. Segmentation and sliding window techniques tackle the issue of long-context processing by dividing the input data into smaller segments, or applying a moving window to slide through the long sequence. For instance, Mistral [123] uses sliding window attention to effectively handle sequences of arbitrary length with a reduced inference cost. StreamingLLM [312] identifies an attention sink phenomenon, noting that retaining the Key-Value of initial tokens significantly restores the performance of window attention. Based on this observation, it suggests an efficient framework via merging window context and the first token, allowing LLMs trained with a finite length attention window, but have the ability to generalize to infinite sequence lengths without any fine-tuning. Parallel Context Windows (PCW) [230] segments a long context into chunks, limiting the attention mechanism to function only within each window, and then redeploys the positional embeddings across these windows. LongNet [65] proposes dilated attention, which exponentially expands the attentive field as the distance increases, enabling the handling of sequence lengths of more than 1 billion tokens. LongNet can be implemented by parallelizing training by partitioning the sequence dimension. SLED [121] is a straightforward method for handling long sequences that repurposes and capitalizes on well-validated short-text language models for use in LLMs.

Memory-Retrieval Augmentation. Several studies tackle the inference of extremely long text by employing memory-retrieval augmentation strategies. A notable example is the KNN-augmented Transformer [308], which extends the attention context size by utilizing k-nearest-neighbor (KNN) lookup to fetch previously similar context embeddings. Landmark Attention [191] employs a landmark token to represent each block of input and trains the attention mechanism to utilize it for choosing relevant blocks. This allows the direct retrieval of blocks through the attention mechanism while maintaining the random access flexibility of the previous context, demonstrating impressive performance on LLaMA-1 for long-context modeling. LongMem [294] proposes a decoupled network architecture with the original backbone LLM as a memory encoder and an adaptive residual side network as a memory retriever and reader, efficiently caching and updating long-term past contexts to prevent knowledge staleness. Unlimiformer [14] enhances the KNN-augmented Transformer by outputting attention dot-product scores as KNN distances, enabling the indexing of virtually unlimited input sequences. Focused Transformer (FoT) [278] highlights that the ratio of relevant keys to irrelevant ones diminishes as the context length increases and proposes an optimized solution through contrastive learning to refine the structure of the key-value space. Lastly, Xu et al. [316] discover that an LLM with a 4K context window, when augmented with simple retrieval during generation, can match the performance of a fine-tuned LLM with a 16K context window using positional interpolation [36] on long context tasks, while requiring significantly less computation.

2.5.4 *Transformer-Alternate Architectures*.

While Transformer-based architectures are now at the forefront of LLMs, some studies propose new architectures to supplant Transformer-based architectures.

State Space Models. A promising approach that aims to substitute the attention mechanism is state space models (SSMs). SSM is formulated as $x'(t) = Ax(t) + Bu(t)$, $y(t) = Cx(t) + Du(t)$, which maps a single-dimension input signal $u(t)$ to an N-dimension latent state $x(t)$ before projecting to a single-dimension output signal $y(t)$, where A, B, C, D are parameters learned by gradient descent [90]. Compared to attention that has quadratic complexity, SSMs provide near-linear computational complexity relative to the length of the sequence. Given such advantage, a series of techniques have been proposed to improve SSMs. For example, the Structured State Space sequence model (S4) [90] refines SSMs by conditioning matrix A with a low-rank correction. This enables stable diagonalization and simplifies the SSM to the well-studied computation of a Cauchy kernel. Diagonal State Space (DSS) [95] improves SSMs by proposing fully diagonal parameterization of state spaces instead of a diagonal plus low rank structure, demonstrating greater efficiency. To bridge the gap between SSMs and attention while adapting to modern hardware, H3 [80] stacks two SSMs to interact with their output and input projection, allowing it to log tokens and facilitate sequence-wide comparisons simultaneously. Mehta et al. [186] introduce a more efficient layer called Gated State Space (GSS), which has been empirically shown to be 2-3 times faster than the previous strategy [95] while maintaining the perplexity on multiple language modeling benchmarks. Block-State Transformer (BST) [211] designs a hybrid layer that combines an SSM sublayer for extended range contextualization with a Block Transformer sublayer for short-term sequence representation. Gu and Dao [89] propose Mamba to enhance SSMs by designing a selection mechanism to eliminate irrelevant data and developed a hardware-aware parallel algorithm for recurrent operation, achieving 5x higher throughput than Transformers. Ren et al. [232] propose a general modular activation mechanism, Sparse Modular Activation (SMA), that unifies previous works on MoE, adaptive computation, dynamic routing and sparse attention, and further applies SMA to develop a novel architecture, SeqBoat, to achieve state-of-the-art quality-efficiency trade-off.

Other Sequential Models. Lastly, some other architectures have been proposed to replace the Transformer layer. Receptance Weighted Key Value (RWKV) model [206] amalgamates the advantages of recurring neural networks (RNN) and Transformers. This combination is designed to utilize the effective parallelizable training feature of Transformers coupled with the efficient inference ability of RNNs, thereby forging a model adept at managing auto-regressive text generation and effectively tackling challenges associated with long sequence processing. Poli et al. [212] propose Hyena, a sub-quadratic alternative to the attention mechanism, mitigating the quadratic cost in

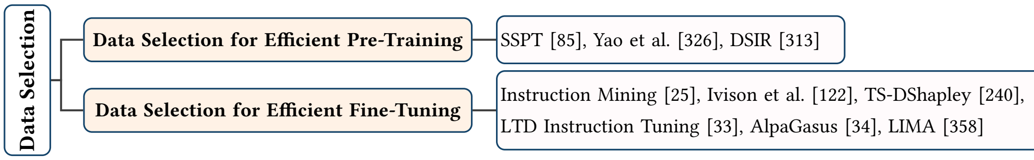


Fig. 15. Summary of data selection techniques for LLMs.

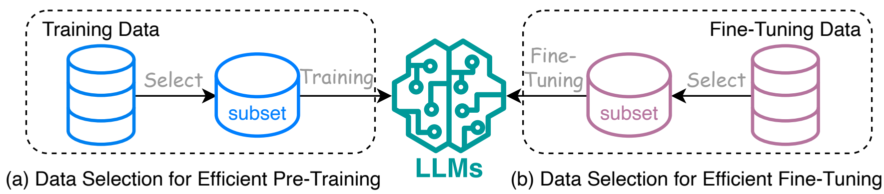


Fig. 16. Illustrations of data selection techniques for LLMs.

long sequences. This operator includes two efficient sub-quadratic primitives: an implicit long convolution and multiplicative element-wise gating of the input. Through this, Hyena facilitates the development of larger, more efficient convolutional language models for long sequences. Lastly, MEGABYTE [333] breaks down long byte sequences into fixed-sized patches akin to tokens, comprising a patch embedder for encoding, a global module acting as a large autoregressive Transformer for patch representations, and a local module for predicting bytes within a patch.

3 DATA-CENTRIC METHODS

3.1 Data Selection

Data selection for LLMs involves carefully selecting the most informative and diverse examples so that the model can efficiently capture essential patterns and features, accelerating the learning process [85, 237, 313, 326]. Figure 15 summarizes the latest data selection techniques for efficient LLM pre-training and fine-tuning.

*3.1.1 **Data Selection for Efficient Pre-Training**.*

Data selection enhances LLMs pre-training efficiency by allowing the model to focus on the most informative and relevant examples during training. By carefully curating a subset of representative data, the model can extract essential patterns and features, leading to a more efficient acquisition of generalized knowledge. For example, SSPT [85] is a pre-training task based on the principles of reading comprehension. It involves selecting answers from contextually relevant text passages, which has shown notable improvements in performance across various Machine Reading Comprehension (MRC) benchmarks. Yao et al. [326] propose a meta-learning-based method for the selection of linguistically informative sentences which significantly elevates the quality of machine-generated translations. Xie et al. [313] propose DSIR, a data selection method based on importance re-sampling for both general-purpose and specialized LLMs. It calculates how important different pieces of data are within a simpler set of features and chooses data based on these importance calculations.

*3.1.2 **Data Selection for Efficient Fine-Tuning**.*

Data selection can also boost fine-tuning efficiency since only a curated subset of examples is employed to refine the model. This approach ensures that the adaptation process is conducted

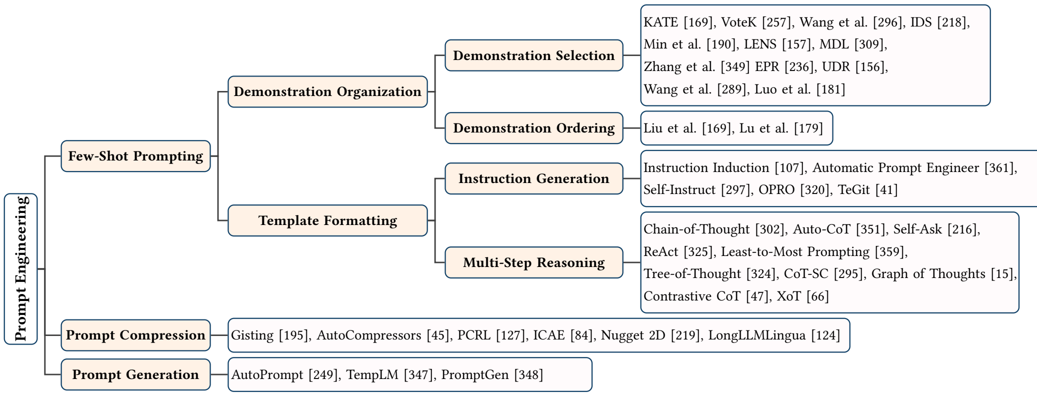


Fig. 17. Summary of prompt engineering techniques for LLMs.

with a focus on the specific nuances intrinsic to the target domain or task, making the fine-tuning process more efficient. For example, Instruction Mining [25] presents a linear evaluation method to assess data quality in instruction-following tasks. It highlights the importance of high-quality data, showing that models trained with Instruction Mining-curated datasets outperform those trained on generic datasets in 42.5% of cases. This underscores the significance of data quality and lays the groundwork for future improvements in instruction-following model efficacy. Ivison et al. [122] propose to use a few unlabeled examples to retrieve similar labeled ones from a larger multitask dataset, improving task-specific model training. This method outperforms standard multitask data sampling for fine-tuning and enhances few-shot fine-tuning, yielding a 2-23% relative improvement over current models. TS-DShapley [240] is introduced to address the computational challenges of applying Shapley-based data valuation to fine-tuning LLMs. It employs an efficient sampling-based method that aggregates Shapley values computed from subsets to evaluate the entire training set. Moreover, it incorporates a value transfer method that leverages information from a simple classifier trained using representations from the target language model. Low Training Data Instruction Tuning (LTD Instruction Tuning) [33] challenges the need for large datasets in fine-tuning, showing that less than 0.5% of the original dataset can effectively train task-specific models without compromising performance. This approach enables more resource-efficient practices in data-scarce environments, combining selective data strategies with tailored training protocols for optimal data efficiency. AlpaGasus [34] is a model fine-tuned on a mere 9k high-quality data points, which are meticulously filtered from a larger dataset of 52k. It outperforms the original model trained on the full dataset and reduces training time by 5.7x, demonstrating the power of high-quality data in instruction-fine-tuning. LIMA [358] fine-tunes LLMs with a small, selected set of examples, showing strong performance and challenging the need for extensive tuning. It generalizes well to new tasks and, in comparisons, matches or exceedes GPT-4 in 43% of cases, suggesting that LLMs gain most knowledge in pre-training, requiring minimal instruction tuning.

3.2 Prompt Engineering

Prompt engineering [170] focuses on designing effective inputs (i.e., prompts) to guide LLMs in generating desired outputs. It enhances inference efficiency by tailoring the input prompts or queries to better suit the capabilities and nuances of a specific language model. When used for some simple tasks, such as semantic classification, prompt engineering can even substitute fine-tuning to achieve high accuracy [167]. As summarized in Figure 17, prompt engineering techniques can be grouped into few-shot prompting, prompt compression, and prompt generation.

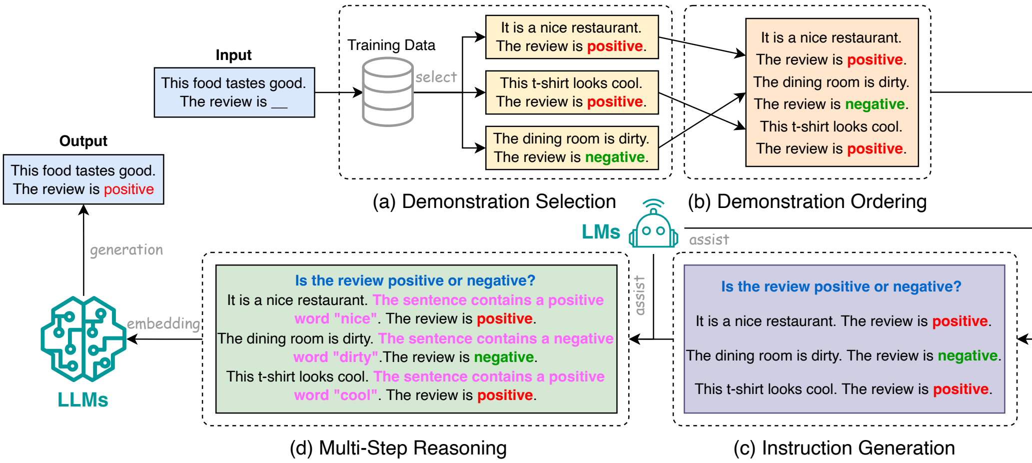


Fig. 18. Illustrations of few-shot prompting techniques for LLMs.

3.2.1 ***Few-Shot Prompting***.

Few-shot prompting involves providing a LLM with a limited set of examples (i.e., demonstrations) to steer its understanding to a task it is required to execute [301]. These demonstrations are selected from the LLM's training corpus based on their similarity to the test example, and the LLM is expected to use the knowledge gained from these similar demonstrations to make the correct prediction [67]. Few-shot prompting provides an efficient mechanism to use LLM by guiding the LLM to perform a wide variety of tasks without the need for additional training or fine-tuning. Furthermore, an effective few-shot prompting approach can make the created prompt concise enough to allow LLMs to quickly adjust to the task in high accuracy with only a slight increase of extra context, thus significantly improving inference speed. As illustrated in Figure 18, few-shot prompting techniques can generally be grouped into demonstration selection, demonstration ordering, instruction generation, and multi-step reasoning.

Demonstration Organization. Demonstration organization refers to organizing the demonstrations in an appropriate way so as to form a suitable prompt for inference. Demonstration organization has a significant impact on the inference speed. Improper organization may result in the processing of a considerable amount of unnecessary information, leading to significant slowdown. The main challenges of demonstration organization come from two perspectives: demonstration selection and demonstration ordering.

- ***Demonstration Selection.*** Demonstration selection aims to choose the good examples for few-shot prompting [67]. In order to generate a satisfactory result, a good selection of demonstrations may only require a few number of demonstrations to be used for the prompt, thus making the prompt concise and straightforward for a more efficient inference. Existing demonstration selection techniques can be grouped into unsupervised methods [157, 169, 190, 218, 257, 296, 309, 349] and supervised methods [156, 181, 236, 289]. Unsupervised methods aim to select the nearest examples from the training set using a predefined similarity function, such as L2 distance, cosine distance, and the minimum description length (MDL) [309]. For example, KATE [169] is an unsupervised selection method that directly uses the nearest neighbors of a given test sample as the corresponding demonstrations. VoteK [257] is an improved version of KATE to resolve its limitation that requires a large set of examples to achieve good performance. Unlike KATE, VoteK increases the diversity of the demonstrations by penalizing examples similar to those already selected. In comparison, supervised methods require training a domain-specific retriever from the training set and using it for demonstration selection. For example, EPR [236] is trained to select demonstrations from a small set of candidates initialized by the unsupervised retriever such as BM25 from the training corpse. UDR [156] further enhances EPR by adopting a unified demonstration retriever to unify the demonstration selection across different tasks. Compared to unsupervised methods, supervised methods often lead to a more satisfying generation result but require frequent adjustment of the retriever for handling the out-of-domain data, making them less efficient for inference.

- ***Demonstration Ordering.*** After selecting representative samples from the training set, the next step is ordering these samples in the prompt. The order of the demonstrations also has a significant impact on the performance of the model. Therefore, selecting the right order of demonstrations can help the model quickly reach a good generation quality with fewer samples, thus improving the inference efficiency. To date, only a few studies have delved into this area. For example, Liu et al. [169] suggest arranging demonstrations based on their distance from the input, placing the closest demonstration furthest to the right. Lu et al. [179] propose to develop both global and local entropy metrics and use the entropy metrics to set up the demonstration order.

Template Formatting. Template Formatting aims to design a suitable template to form the prompt. A good template typically compiles all the information needed by LLMs into a brief statement, making the prompt and the entire input context as succinct as possible, thus guaranteeing a higher inference efficiency. Template formatting design can be divided into two parts: instruction generation and multi-step reasoning.

- ***Instruction Generation.*** The instruction of the template refers to a short description of the task. By adding instructions to the prompt, LLMs can quickly understand the context and the task they are currently performing, and thus may require fewer demonstrations to create a desirable prompt. The performance of a given task is highly affected by the quality of the instructions. The instructions vary not only between different datasets for the same task but also between different models. Unlike demonstrations that are usually included in traditional datasets, the generation of instructions is heavily dependent on human efforts. To enhance the efficiency of instruction generation, automatic instruction generation techniques have been proposed. For example, Instruction Induction [107] and Automatic Prompt Engineer [361] have demonstrated that LLMs can generate task instructions. Wang et al. [297] propose Self-Instruct, an approach that allows LLMs to align with self-generated instructions, highlighting their inherent adaptability. Yang et al. [320] also discover that LLMs can be treated as an optimizer to iteratively generate better instructions for the target LLM and have applied this technique to various LLMs. Chen et al. [41] develop TeGit for training language models as task designers, which can automatically generate inputs and outputs together with high-quality instructions to better filter the noise based on a given human-written text for fine-tuning LLMs. Despite the promise of automatic instruction generation methods, their complexity is still a major bottleneck for their real-world adoption.
- ***Multi-Step Reasoning.*** Guiding the LLMs to produce a sequence of intermediate steps before outputting the final answer can greatly improve the quality of the generation. This technique is also referred to as Chain-of-Thought (CoT) prompting [302]. Rather than repeatedly choosing a few exemplary examples to make the context and task more understandable to the LLMs, CoT only concentrates on a limited number and adds the details for contemplation into the context, making the prompt more comprehensive and effective and guaranteeing

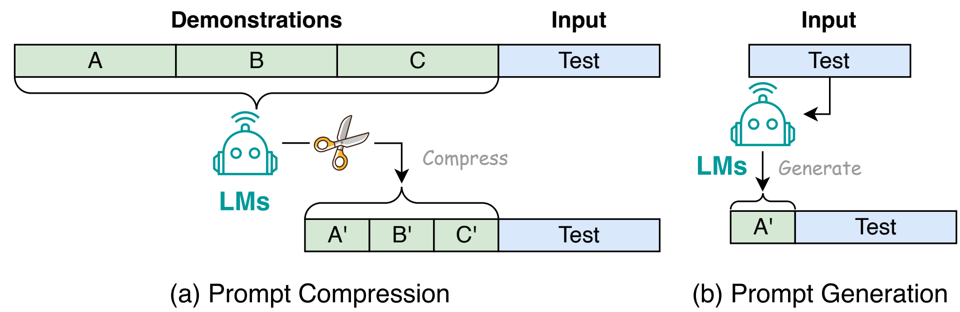


Fig. 19. Illustrations of Prompt Compression (a) and Prompt Generation (b) for LLMs.

a more efficient inference. However, despite the advantages of CoT, it is still difficult to ensure the accuracy of every intermediate step [67]. Given that, many techniques have been proposed to address this issue. For example, Auto-CoT [351] proposes to generate the CoT step by step from LLMs. Self-Ask [216] incorporates the self-generated question of each step into the CoT. ReAct [325] performs dynamic reasoning to create, maintain, and adjust high-level plans for acting, while interacting with external environments to incorporate additional information into reasoning. Least-to-Most Prompting [359] breaks down the complex question into smaller ones and answers them iteratively within the context of former questions and answers. Tree-of-Thought (ToT) [324] expends CoT to include exploration over coherent units of text and deliberates decision-making processes. CoT-SC [295] introduces a novel decoding approach called "self-consistency" to replace the simplistic greedy decoding in CoT prompting. It starts by sampling various reasoning paths instead of just the greedy one and then determines the most consistent answer by considering all the sampled paths. Graph of Thoughts (GoT) [15] represent information produced by an LLM as a generic graph, with "LLM thoughts" as vertices and edges indicating dependencies between these vertices. Contrastive CoT [47] proposes contrastive chain of thought to enhance language model reasoning by providing both valid and invalid reasoning demonstrations. Lastly, XoT [66] utilizes pretrained reinforcement learning and Monte Carlo Tree Search (MCTS) to integrate external domain knowledge into LLMs' thought processes, thereby boosting their ability to efficiently generalize to new, unseen problems.

3.2.2 ***Prompt Compression.***

Prompt compression (Figure 19(a)) accelerates the processing of LLM inputs through either condensing lengthy prompt inputs or learning compact prompt representations. Mu et al. [195] propose to train LLMs to distill prompts into a more concise set of tokens, referred to as gist tokens. These gist tokens encapsulate the knowledge of the original prompt and can be stored for future use. In doing so, it is able to compress prompts by up to 26 times, leading to a reduction in floating-point operations per second (FLOPs) by up to 40%. Chevalier et al. [45] propose AutoCompressors to condense long textual contexts into compact vectors, known as summary vectors, which can then be used as soft prompts for the language model. These summary vectors extend the model's context window, allowing it to handle longer documents with much less computational cost. Jung and Kim [127] propose Prompt Compression with Reinforcement Learning (PCRL) that employs a policy network to directly edit prompts, aiming to reduce token count while preserving performance. It achieves an average reduction of 24.6% in token count across various instruction prompts. Ge et al. [84] propose In-context Autoencoder (ICAE), which consists of a learnable encoder and a fixed decoder. The encoder compresses a long context into a limited number of memory slots, which the target language model can then condition on. With such design, ICAE is able to obtain 4x context compression. Nugget 2D [219] represents the historical context as compact "nuggets" that are trained to enable reconstruction. Furthermore, it has the flexibility to be initialized using readily available models like LLaMA. Lastly, LongLLMLingua [124] introduces a prompt compression technique containing question-aware coarse-to-fine compression, document reordering, dynamic compression ratios, and post-compression sub-sequence recovery to enhance LLMs' key information perception.

3.2.3 ***Prompt Generation***.

Prompt generation (Figure 19(b)) enhances the efficiency by automatically creating effective prompts that guide the model in generating specific and relevant responses instead of manual annotated data. AutoPrompt [249] proposes an automated method to generate prompts for a diverse set of tasks based on a gradient-guided search. It underscores the significance of human-written text in refining the quality and authenticity of data, emphasizing its pivotal role in optimizing LLM performance. TempLM [347] proposes to combine generative and template-based methodologies to distill LLMs into template-based generators, offering a harmonized solution for data-to-text tasks. PromptGen [348] is the first work considering dynamic prompt generation for knowledge probing, based on a pre-trained LLMs. It can automatically generate prompts conditional on the input sentence and outperforms AutoPrompt on the on the LAMA benchmark.

4 LLM FRAMEWORKS

DeepSpeed. Developed by Microsoft, DeepSpeed [229] is an integrated framework for both training and deploying LLMs. It has been used to train large models like Megatron-Turing NLG 530B [253] (in a joint effort with Nvidia Megatron framework) and BLOOM [239]. Within this framework, DeepSpeed-Inference is the foundational library. A pivotal feature of this module is ZeRO-Inference [226, 228], an optimization technique created to address GPU memory constraints for large model inference. ZeRO-Inference distributes model states across multiple GPUs and CPUs, providing an approach to managing the memory constraints of individual nodes. Another aspect of DeepSpeed-Inference is its deep fusion mechanism, which allows for the fusion of operations without the necessity for global synchronization by tiling computations across iteration space dimensions [149, 180, 231, 266]. Building on this, the DeepSpeed Model Implementations for Inference (DeepSpeed MII) module provides strategies for the deployment and management of popular deep learning models. Emphasizing performance, flexibility, and cost-efficiency, DeepSpeed MII incorporates advanced optimization techniques to improve model inference [228, 306, 329]. Furthermore, the introduction of DeepSpeed-Chat [328] adds chat support to the ecosystem. This module focuses on training chatbot models across different scales, integrating techniques from Reinforcement Learning from Human Feedback (RLHF) [88] with the DeepSpeed training system. Notably, its integration of the ZeRO-Offload optimizer [231] facilitates training on both CPUs and GPUs, irrespective of their memory capacities.

Megatron. Megatron [250] constitutes Nvidia's efforts to streamline training and deployment of LLMs such as GPT [222] and T5 [224]. It is the underlying framework used for Nvidia's Megatron models [140, 197, 250]. Megatron encompasses various specialized tools and frameworks for Nvidia GPUs. Central to Megatron's design is the strategic decomposition of the model's tensor operations, distributed across multiple GPUs, to optimize both processing speed and memory utilization, thus enhancing training throughput without compromising model fidelity [250]. Megatron also uses

Table 2. Comparison of LLM frameworks.

Framework	Training	Fine-Tuning	Inference	Features
DeepSpeed	✓	✓	✓	Data Parallelism, Model Parallelism, Pipeline Parallelism, Prompt Batching, Quantisation, Kernel Optimizations, Compression, Mixture of Experts.
Megatron	✓	✓	✓	Data Parallelism, Model Parallelism, Pipeline Parallelism, Prompt Batching, Automatic Mixed precision, Selective activation Recomputation
Alpa	✓	✓	✓	Data Parallelism, Model Parallelism, Pipeline Parallelism, Operator Parallelism, Automated Model-Parallel Training, Prompt Batching
Colossal AI	✓	✓	✓	Data Parallelism, Model Parallelism, Pipeline Parallelism, Mixed Precision Training, Gradient accumulation, heterogeneous Distributed Training, Prompt Batching, Quantization
FairScale	✓	✓	✓	Data Parallelism, Model Parallelism, Pipeline Parallelism, Activation Checkpointing, Model Offloading, Model scaling, Adascale Optimization
Pax	✓	✓	✓	Data Parallelism, Model Parallelism, Kernel Optimization
Composer	✓	✓	✓	Fully Sharded Data Parallelism, Elastic sharded checkpointing, Flash Attention
vLLM	✗	✗	✓	Data Parallelism, Model Parallelism, Tensor Paralellism, Efficient management via PagedAttention, Optimized CUDA kernels, Dynamic Batching, Quantization
OpenLLM	✗	✓	✓	Distributed Finetuning and Inference, Integration with BentoML, LangChain, and Transformers Agents, Prometheus Metrics, Token Streaming
Ray LLM	✗	✗	✓	Distributed Inference, Integration with Alpa, Prompt Batching, Quantization, Prometheus Metrics
MLC LLM	✗	✗	✓	Distributed Inference, Compiler Acceleration, Prompt Batching, Quantization
Sax	✗	✗	✓	Distribute Inference, Serves PaxML, JAX, and PyTorch models, Slice Serving, Prometheus Metrics
Mosec	✗	✗	✓	Distribute Inference, Dynamic Batching, Rust-based Task Coordinator, Prometheus Metrics
LLM Foundry	✗	✗	✓	Distribute Inference, Dynamic Batching, Prompt Batching

FasterTransformer [199] for optimizing the inference process for large Transformer models. Furthermore, FasterTransformer is used for handling varying precision modes like FP16 and INT8, catering to diverse operational needs. The system also incorporates algorithms tailored to specific GPU architectures like Turing and Volta, emphasizing performance optimization [199]. Finally, Megatron uses TensorRT-LLM which provides developers with advanced tools and optimizations specifically tailored for LLMs, aiming to significantly reduce latency and enhance throughput for real-time applications. Notably, TensorRT-LLM integrates optimized kernels from FasterTransformer [275] and employs tensor parallelism, facilitating efficient inference at scale across multiple GPUs and servers without necessitating developer intervention or model changes.

Alpa. Alpa [356] is a library for training and serving large-scale neural networks. Alpa strategically addresses both inter- and intra-operator parallelism, aiming for a holistic enhancement in distributed deep learning performance. It has example implementations of GPT-2 [222], BLOOM [239], OPT [346], CodeGen [198] among others. At the core of Alpa's methodology is its automatic parallelization. By deploying an auto-tuning framework, Alpa dynamically identifies the optimal parallelism strategy tailored to specific deep learning models and hardware configurations. Furthermore, Alpa showcases an integrated design that combines both data and model parallelism [162, 364]. By doing so, Alpa harnesses the collective benefits of these parallelism techniques, leading to optimized resource utilization and enhanced training throughput during serving.

ColossalAI. ColossalAI [152] is a framework tailored to address the challenges of large-scale distributed training [284]. ColossalAI provides a unified solution that harmonizes scalability, efficiency, and versatility. It has implementations for LLaMA [277], GPT-3 [20], GPT-2 [222], BERT [64], PaLM, OPT [346], ViT [68]. Central to Colossal-AI's design is its emphasis on holistic integration. By amalgamating various components of deep learning pipelines, from data preprocessing to model training and validation, ColossalAI provides a streamlined platform that reduces fragmentation and enhances workflow efficiency [16]. This integrated approach mitigates the complexities often associated with orchestrating large-scale training in distributed environments. Furthermore, recognizing the dynamic landscape of deep learning research and applications, the system is architected to be inherently modular [38]. In addition, the framework integrates several other advanced optimization techniques [16, 73, 74, 153, 174, 285] and features like quantization, gradient accumulation, and mixed precision. By leveraging state-of-the-art algorithms and methodologies, Colossal-AI seeks to optimize both computational and communication overheads inherent in parallel training, leading to reduced training times and enhanced model performance.

FairScale. Developed by Meta, FairScale [72] is an extension library to PyTorch, dedicated to high-performance and large-scale training initiatives. The ethos of FairScale is rooted in three fundamental principles: usability, which emphasizes the ease of understanding and utilization of FairScale's APIs aiming to minimize cognitive overhead for users; modularity, which endorses a seamless amalgamation of multiple FairScale APIs within the users' training loops, thus promoting flexibility; and performance, which is centered around delivering optimal scaling and efficiency through FairScale's APIs. Additionally, FairScale provides support for Fully Sharded Data Parallel (FSDP) as the preferred method for scaling the training operations of extensive neural networks. It is therefore a powerful tool for distributed training and inference. Additionally, it has key features for training in resource-constrained systems providing support for activation checkpointing, efficient model offloading, and scaling.

Pax. Developed by Google, Pax [9] is a JAX-based efficient distributed training framework. Pax has been used to train PaLM-2 [7] and Bard [109]. It targets scalability and has reference examples for large model training, including across modalities (such as text, vision, speech, etc.). It is heavily integrated with JAX and uses many libraries in the JAX ecosystem. Pax contains many key components, including SeqIO to handle sequential data processing, Optax for optimization, Fiddle for configuration, Orbax for checkpointing, PyGLove for automatic differentiation, and Flax for creating high-performance neural networks.

Composer. Designed by Mosaic ML, Composer [193] is aimed at making the training of neural networks faster and more efficient. It has been used to train Mosaic ML's MPT 7B and MPT 30B models and Replit's Code V-1.5 3B. The library is built on top of PyTorch and provides a collection of speedup methods that users can incorporate into their own training loops or use with the Composer trainer for a better experience. It supports FSDP for efficient parallelism, elastic shared checkpointing for robust intermittent training, and a dataset streaming implementation allowing to download datasets from cloud blob storage on the fly during training. Composer is therefore designed to be versatile with a Functional API for integrating methods directly into its training loops, as well as a Trainer API which automatically implements a PyTorch-based training loop, reducing the workload for ML developers.

vLLM. vLLM [142] represents a methodological shift in the approach to serving LLMs. Central to vLLM's design is PagedAttention, a mechanism that segments the attention key and value (KV) cache for a set number of tokens. Unlike contiguous space storage, PagedAttention's blocks for the KV cache are stored flexibly, akin to the virtual memory management. This facilitates memory sharing at a block level across various sequences tied to the same request or even different requests, thus enhancing memory management efficiency in handling attention mechanisms. It also allows on-demand buffer allocation, while also eliminating external fragmentation as the blocks are uniformly sized. Furthermore, vLLM incorporates an adaptive loading technique. This technique, rooted in heuristic methodologies, discerns the number of pages to be loaded into memory based on the input. Complementing this, vLLM integrates a parameter compression strategy as well. By storing model parameters in a compressed state and decompressing them during real-time serving, vLLM further optimizes memory usage. Additionally, vLLM supports state-of-the-art quantization techniques and optimized CUDA kernels supporting fast model execution. The library also added support for AMD's ROCm GPUs. vLLM is therefore, not only a useful tool for distributed training, it can also handle efficient high-throughput model serving workloads.

OpenLLM. OpenLLM [210] delineates a comprehensive approach to the deployment and operation of LLMs within production environments. Anchored within the BentoML ecosystem, OpenLLM is crafted to bridge the gap between the training of LLMs and their seamless integration into real-world applications. A defining characteristic of OpenLLM is its emphasis on modularity and scalability. Recognizing the diverse needs of production environments, OpenLLM promotes a component-based architecture. Further enhancing its value proposition, OpenLLM integrates advanced caching mechanisms. By leveraging these mechanisms, the system aims to optimize repetitive queries, leading to reduced operational costs and enhanced response times. Additionally, OpenLLM's design incorporates robust monitoring and logging tools, ensuring that operational insights are readily available for performance tuning and troubleshooting.

Ray-LLM. Ray-LLM [217] represents a strategic fusion of LLMs with the Ray ecosystem [192], aiming to optimize the deployment and operation of LLMs. Situated at the intersection of cutting-edge model architecture and scalable infrastructure, Ray-LLM seeks to redefine the paradigms of LLM utilization. At the core of Ray-LLM's approach is the leveraging of Ray's inherent distributed computing capabilities. Recognizing the computational demands of LLMs, Ray-LLM integrates Ray's distributed task scheduling and execution mechanisms, ensuring that LLM tasks are efficiently distributed across available resources. This seamless integration potentially leads to enhanced model performance, reduced latency, and optimized resource utilization. Since it is built on top of the Ray Ecosystem, Ray-LLM is a good library to quickly prototype, train and deploy large models on clusters. It also comes with advanced monitoring support as well, enabling its usage in serving.

MLC-LLM. MLC-LLM [271] aspires to empower individuals to develop, optimize, and deploy AI models on a diverse array of devices. Central to MLC-LLM's approach is the concept of device-native AI. Recognizing the vast spectrum of devices in use today, from high-end servers to smartphones, MLC-LLM compiles models and deploys them in a process that is inherently tailored to the specific capabilities and constraints of each device [37, 76, 243]. This device-native focus ensures that AI models are not only efficient but also highly optimized for the environments in which they operate. With its strong focus on compiling and optimizing models for prototyping on edge devices, MLC-LLM is a powerful tool for deploying on-device AI models and exhibits state-of-the-art performance in terms of throughput across a range of devices.

Sax. Sax [10] is a platform designed by Google for deploying Pax, JAX, and PyTorch models for inference tasks. Within Sax, there is a unit referred to as Sax cell (or Sax cluster) thatl is made up of an administrative server coupled with multiple model servers. The role of the admin server is multifaceted: it monitors the model servers, allocates published models to these servers for inference, and guides clients in finding the appropriate model server for specific published models. Sax is essentially complementary to the Pax framework and while Pax focuses on massively distributed workloads, Sax is geared toward model serving.

Mosec. Mosec [322] is designed for serving large deep learning models particularly in cloud environments. It is built to streamline the serving of machine learning models into backend services and microservices. Key features include high performance due to Rust-built web layer and task coordination, easy-to-use Python interface, dynamic batching, pipelined stages for handling mixed workloads, and cloud-friendliness with model warmup, graceful shutdown, and Prometheus monitoring metrics, making it easily manageable by Kubernetes or other container orchestration systems. Mosec is centered around cloud ecosystems and is well suited for serving models efficiently with its web layer, allowing developers to focus on model optimization and backend logic.

LLM Foundry. LLM Foundry [194] is a library for finetuning, evaluating, and deploying LLMs for inference with Composer and the MosaicML platform. It supports distributed inference, dynamic batching, and prompt batching for efficient deployment. Similar to its complimentary training framework Composer, LLM Foundry is designed to be easy to use, efficient, and flexible, aimed at enabling rapid experimentation with the latest techniques in LLMs. It also provides straightforward interfaces to Mosaic's Pretrained Transformers (MPT) (GPT-style models with built-in support for features like FlashAttention [58] and ALiBi [215]). It is complementary to MosaicML's Composer framework and while Composer focuses on distributed training, LLM Foundry provides support for deploying those models and enabling rapid experimentation with the latest techniques.

5 CONCLUDING REMARKS

In this survey, we provide a systematic review of efficient LLMs, an important area of research aimed at democratizing LLMs. We start with motivating the necessity for efficient LLMs. Guided by a taxonomy, we review algorithm-level and system-level efficient techniques for LLMs from model-centric and data-centric perspectives respectively. Furthermore, we review LLM frameworks with specific optimizations and features crucial for efficient LLMs. We believe that efficiency will play an increasingly important role in LLMs and LLMs-oriented systems. We hope this survey could enable researchers and practitioners to quickly get started in this field and act as a catalyst to inspire new research on efficient LLMs.

REFERENCES

[1] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. 2023. Generalized Knowledge Distillation for Auto-regressive Language Models. arXiv:2306.13649 [cs.LG]

[2] Arash Ahmadian, Saurabh Dash, Hongyu Chen, Bharat Venkitesh, Stephen Gou, Phil Blunsom, Ahmet Üstün, and Sara Hooker. 2023. Intriguing Properties of Quantization at Scale. arXiv:2305.19268 [cs.LG]

[3] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. arXiv:2305.13245 [cs.CL]

[4] Silas Alberti, Niclas Dern, Laura Thesing, and Gitta Kutyniok. 2023. Sumformer: Universal Approximation for Efficient Transformers. arXiv:2307.02301 [cs.LG]

[5] Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, and Yuxiong He. 2022. DeepSpeed-Inference: Enabling Efficient Inference of Transformer Models at Unprecedented Scale. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '22)*. IEEE Press, Dallas, Texas, Article 46, 15 pages.

[6] Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hofmann. 2023. Dynamic Context Pruning for Efficient and Interpretable Autoregressive Transformers. arXiv:2305.15805 [cs.CL]

[7] Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. PaLM 2 Technical Report. arXiv:2305.10403 [cs.CL]

[8] Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, Giri Anantharaman, Xian Li, Shuohui Chen, Halil Akin, Mandeep Baines, Louis Martin, Xing Zhou, Punit Singh Koura, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Mona Diab, Zornitsa Kozareva, and Ves Stoyanov. 2022. Efficient Large Scale Language Modeling with Mixtures of Experts. arXiv:2112.10684 [cs.CL]

[9] Pax Authors. 2023. Pax: A Jax-based Machine Learning Framework for Large Scale Models. https://github.com/google/paxml. https://github.com/google/paxml GitHub repository.

[10] Sax Authors. 2023. Sax. https://github.com/google/saxml. Accessed: 2023-10-07.

[11] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. 2021. ReZero is all you need: fast convergence at large depth. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (Proceedings of Machine Learning Research)*, Cassio de Campos and Marloes H. Maathuis (Eds.), Vol. 161. PMLR, online, 1352–1361. https://proceedings.mlr.press/v161/bachlechner21a.html

[12] Trapit Bansal, Salaheddin Alzubi, Tong Wang, Jay-Yoon Lee, and Andrew McCallum. 2022. Meta-Adapters: Parameter Efficient Few-shot Fine-tuning through Meta-Learning. In *First Conference on Automated Machine Learning (Main Track)*. PMLR, Baltimore, US, –. https://openreview.net/forum?id=BCGNf-prLg5

[13] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. arXiv:2004.05150 [cs.CL]

[14] Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley. 2023. Unlimiformer: Long-Range Transformers with Unlimited Length Input. arXiv:2305.01625 [cs.CL]

[15] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2023. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. arXiv:2308.09687 [cs.CL]

[16] Zhengda Bian, Qifan Xu, Boxiang Wang, and Yang You. 2021. Maximizing Parallelism in Distributed Training for Huge Neural Networks. arXiv:2105.14450 [cs.DC]

[17] Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An Open-Source Autoregressive Language Model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, Angela Fan, Suzana Ilic, Thomas Wolf, and Matthias Gallé (Eds.). Association for Computational Linguistics, virtual+Dublin, 95–136. https://doi.org/10.18653/v1/2022.bigscience-1.9

[18] bloc97. 2023. 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/. Accessed: 2023-12-19.

[19] Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. 2021. Understanding and Overcoming the Challenges of Efficient Transformer Quantization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 7947–7969. https://doi.org/10.18653/v1/2021.emnlp-main.627

[20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners.

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

[22] Neil Burgess, Jelena Milanovic, Nigel Stephens, Konstantinos Monachopoulos, and David Mansell. 2019. Bfloat16 Processing for Neural Networks. In *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*. IEEE Press, Kyoto, 88–91. https://doi.org/10.1109/ARITH.2019.00022

[23] Lucas Caccia, Edoardo Ponti, Zhan Su, Matheus Pereira, Nicolas Le Roux, and Alessandro Sordoni. 2023. Multi-Head Adapter Routing for Cross-Task Generalization. arXiv:2211.03831 [cs.AI]

[24] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, and Tri Dao. 2023. Medusa: Simple framework for accelerating llm generation with multiple decoding heads.

[25] Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. 2023. Instruction Mining: When Data Mining Meets Large Language Model Finetuning. arXiv:2307.06290 [cs.CL]

[26] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Kaijie Zhu, Hao Chen, Linyi Yang, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2023. A survey on evaluation of large language models. *arXiv preprint arXiv:2307.03109* (2023).

[27] Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. 2023. QuIP: 2-Bit Quantization of Large Language Models With Guarantees. arXiv:2307.13304 [cs.LG]

[28] Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. 2021. Scatterbrain: Unifying sparse and low-rank attention. *Advances in Neural Information Processing Systems* 34 (2021), 17413–17426.

[29] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating Large Language Model Decoding with Speculative Sampling. arXiv:2302.01318 [cs.CL]

[30] Chang Chen, Min Li, Zhihua Wu, Dianhai Yu, and Chao Yang. 2022. TA-MoE: Topology-Aware Large Scale Mixture-of-Expert Training. *Advances in Neural Information Processing Systems* 35 (2022), 22173–22186.

[31] Cheng Chen, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. 2021. bert2BERT: Towards Reusable Pretrained Language Models. arXiv:2110.07143 [cs.CL]

[32] Guanzheng Chen, Xin Li, Zaiqiao Meng, Shangsong Liang, and Lidong Bing. 2023. CLEX: Continuous Length Extrapolation for Large Language Models. arXiv:2310.16450 [cs.CL]

[33] Hao Chen, Yiming Zhang, Qi Zhang, Hantao Yang, Xiaomeng Hu, Xuetao Ma, Yifan Yanggong, and Junbo Zhao. 2023. Maybe Only 0.5Exploration of Low Training Data Instruction Tuning. arXiv:2305.09246 [cs.AI]

[34] Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. 2023. AlpaGasus: Training A Better Alpaca with Fewer Data. arXiv:2307.08701 [cs.CL]

[35] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba.

2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374 [cs.LG]
[36] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending Context Window of Large Language Models via Positional Interpolation. arXiv:2306.15595 [cs.CL]
[37] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. 2018. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. USENIX Association, Carlsbad, CA, 578–594. https://www.usenix.org/conference/osdi18/presentation/chen
[38] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training Deep Nets with Sublinear Memory Cost. arXiv:1604.06174 [cs.LG]
[39] Wuyang Chen, Yanqi Zhou, Nan Du, Yanping Huang, James Laudon, Zhifeng Chen, and Claire Cui. 2023. Lifelong Language Pretraining with Distribution-Specialized Experts. In *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*. JMLR.org, Honolulu, Hawaii, USA, Article 213, 13 pages.
[40] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. 2023. Symbolic Discovery of Optimization Algorithms. arXiv:2302.06675 [cs.LG]
[41] Yongrui Chen, Haiyun Jiang, Xinting Huang, Shuming Shi, and Guilin Qi. 2023. TeGit: Generating High-Quality Instruction-Tuning Data with Text-Grounded Task Design. arXiv:2309.05447 [cs.CL]
[42] Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023. LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models. arXiv:2309.12307 [cs.CL]
[43] Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. 2022. Meta-learning via Language Model In-context Tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 719–730. https://doi.org/10.18653/v1/2022.acl-long.53
[44] Zeming Chen, Qiyue Gao, Antoine Bosselut, Ashish Sabharwal, and Kyle Richardson. 2023. DISCO: Distilling Counterfactuals with Large Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 5514–5528. https://doi.org/10.18653/v1/2023.acl-long.302
[45] Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting Language Models to Compress Contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 3829–3846. https://doi.org/10.18653/v1/2023.emnlp-main.232
[46] Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, et al. 2022. On the representation collapse of sparse mixture of experts. *Advances in Neural Information Processing Systems* 35 (2022), 34600–34613.
[47] Yew Ken Chia, Guizhen Chen, Luu Anh Tuan, Soujanya Poria, and Lidong Bing. 2023. Contrastive Chain-of-Thought Prompting. arXiv:2311.09277 [cs.CL]
[48] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating Long Sequences with Sparse Transformers. arXiv:1904.10509 [cs.LG]
[49] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, David Belanger, Lucy Colwell, et al. 2020. Masked language modeling for proteins via linearly scalable long-context transformers. *arXiv preprint arXiv:2006.03555* (2020).
[50] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. 2021. Rethinking Attention with Performers. https://openreview.net/forum?id=Ua6zuk0WRH
[51] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. PaLM: Scaling Language Modeling with Pathways. arXiv:2204.02311 [cs.CL]
[52] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav

Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling Instruction-Finetuned Language Models. arXiv:2210.11416 [cs.LG]

[53] Luciano Del Corro, Allie Del Giorno, Sahaj Agarwal, Bin Yu, Ahmed Awadallah, and Subhabrata Mukherjee. 2023. SkipDecode: Autoregressive Skip Decoding with Batching and Caching for Efficient LLM Inference. arXiv:2307.02628 [cs.CL]

[54] Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. StableMoE: Stable Routing Strategy for Mixture of Experts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 7085–7095. https://doi.org/10.18653/v1/2022.acl-long.489

[55] Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing. *Advances in Neural Information Processing Systems* 33 (2020), 4271–4282.

[56] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Anna Korhonen, David Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, Florence, Italy, 2978–2988. https://doi.org/10.18653/v1/P19-1285

[57] Tri Dao. 2023. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. arXiv:2307.08691 [cs.LG]

[58] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *Advances in Neural Information Processing Systems* 35 (2022), 16344–16359.

[59] Tri Dao, Daniel Haziza, Francisco Massa, and Grigory Sizov. 2023. Flash-Decoding for long-context inference. https://pytorch.org/blog/flash-decoding/. Accessed: 2023-12-13.

[60] Soham De and Sam Smith. 2020. Batch Normalization Biases Residual Blocks Towards the Identity Function in Deep Networks. *Advances in Neural Information Processing Systems* 33 (2020), 19964–19975.

[61] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. arXiv:2208.07339 [cs.LG]

[62] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. arXiv:2305.14314 [cs.LG]

[63] Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2023. SpQR: A Sparse-Quantized Representation for Near-Lossless LLM Weight Compression. arXiv:2306.03078 [cs.CL]

[64] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[65] Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. 2023. LongNet: Scaling Transformers to 1,000,000,000 Tokens. arXiv:2307.02486 [cs.CL]

[66] Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. Everything of Thoughts: Defying the Law of Penrose Triangle for Thought Generation. arXiv:2311.04254 [cs.AI]

[67] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A Survey on In-context Learning. arXiv:2301.00234 [cs.CL]

[68] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*. ICLR, Addis Ababa, Ethiopia. https://openreview.net/forum?id=YicbFdNTTy

[69] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. GLaM: Efficient Scaling of Language Models with Mixture-of-Experts. In *International Conference on Machine Learning*. PMLR, Baltimore, Maryland, 5547–5569.

[70] Lance Eliot. 2021. Generative Pre-Trained Transformers (GPT-3) Pertain To AI In The Law. https://doi.org/10.2139/ssrn.3974887

[71] Facebook AI Research (FAIR). 2023. fairseq: FP16 Optimizer - Line 468. https://github.com/facebookresearch/fairseq/blob/main/fairseq/optim/fp16_optimizer.py. Accessed: 2023-12-13.

[72] FairScale authors. 2021. FairScale: A general purpose modular PyTorch library for high performance and large scale training. https://github.com/facebookresearch/fairscale.

[73] J. Fang et al. 2023. Parallel Training of Pre-Trained Models via Chunk-Based Dynamic Memory Management. *IEEE Transactions on Parallel and Distributed Systems* 34, 1 (2023), 304–315.

[74] Jiarui Fang, Geng Zhang, Jiatong Han, Shenggui Li, Zhengda Bian, Yongbin Li, Jin Liu, and Yang You. 2022. A Frequency-aware Software Cache for Large Recommendation System Embeddings. arXiv:2208.05321 [cs.IR]

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

[76] Siyuan Feng, Bohan Hou, Hongyi Jin, Wuwei Lin, Junru Shao, Ruihang Lai, Zihao Ye, Lianmin Zheng, Cody Hao Yu, Yong Yu, and Tianqi Chen. 2023. TensorIR: An Abstraction for Automatic Tensorized Program Optimization. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (Vancouver, BC, Canada) *(ASPLOS 2023)*. Association for Computing Machinery, New York, NY, USA, 804–817. https://doi.org/10.1145/3575693.3576933

[77] Elias Frantar and Dan Alistarh. 2022. Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). New Orleans, Louisiana. https://openreview.net/forum?id=ksVGCOlOEba

[78] Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot. *ArXiv* abs/2301.00774 (2023).

[79] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. OPTQ: Accurate Quantization for Generative Pre-trained Transformers. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=tcbBPnfwxS

[80] Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. 2023. Hungry Hungry Hippos: Towards Language Modeling with State Space Models. arXiv:2212.14052 [cs.LG]

[81] Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2023. Breaking the Sequential Dependency of LLM Inference Using Lookahead Decoding. https://lmsys.org/blog/2023-11-21-lookahead-decoding/

[82] Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing Smaller Language Models towards Multi-Step Reasoning. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.), Vol. 202. PMLR, Honolulu, Hawaii, 10421–10430. https://proceedings.mlr.press/v202/fu23d.html

[83] Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. 2023. MegaBlocks: Efficient Sparse Training with Mixture-of-Experts. *Proceedings of Machine Learning and Systems* 5 (2023).

[84] Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context Autoencoder for Context Compression in a Large Language Model. arXiv:2307.06945 [cs.CL]

[85] Michael Glass, Alfio Gliozzo, Rishav Chakravarti, Anthony Ferritto, Lin Pan, G P Shrivatsa Bhargav, Dinesh Garg, and Avi Sil. 2020. Span Selection Pre-training for Question Answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 2773–2782. https://doi.org/10.18653/v1/2020.acl-main.247

[86] Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. 2019. Efficient Training of BERT by Progressively Stacking. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, Long Beach, California, 2337–2346. https://proceedings.mlr.press/v97/gong19a.html

[87] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision* 129 (2021), 1789–1819.

[88] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. 2013. Policy Shaping: Integrating Human Feedback with Reinforcement Learning. In *Advances in Neural Information Processing Systems*, C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper%5Ffiles/paper/2013/file/e034fb6b66aacc1d48f445ddfb08da98-Paper.pdf

[89] Albert Gu and Tri Dao. 2023. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. arXiv:2312.00752 [cs.LG]

[90] Albert Gu, Karan Goel, and Christopher Re. 2022. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*. https://openreview.net/forum?id=uYLFoz1vlAC

[91] Xiaotao Gu, Liyuan Liu, Hongkun Yu, Jing Li, Chen Chen, and Jiawei Han. 2021. On the Transformer Growth for Progressive BERT Training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (Eds.). Association for Computational Linguistics, Online, 5174–5180. https://doi.org/10.18653/v1/2021.naacl-main.406

[92] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Knowledge Distillation of Large Language Models. arXiv:2306.08543 [cs.CL]

[93] Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. PPT: Pre-trained Prompt Tuning for Few-shot Learning. arXiv:2109.04332 [cs.CL]

[94] Cong Guo, Jiaming Tang, Weiming Hu, Jingwen Leng, Chen Zhang, Fan Yang, Yunxin Liu, Minyi Guo, and Yuhao Zhu. 2023. OliVe: Accelerating Large Language Models via Hardware-Friendly Outlier-Victim Pair Quantization. In *Proceedings of the 50th Annual International Symposium on Computer Architecture* (Orlando, FL, USA) *(ISCA '23)*. Association for Computing Machinery, New York, NY, USA, Article 3, 15 pages. https://doi.org/10.1145/3579371.3589038

[95] Ankit Gupta, Albert Gu, and Jonathan Berant. 2022. Diagonal State Spaces are as Effective as Structured State Spaces. *Advances in Neural Information Processing Systems* 35 (2022), 22982–22994.

[96] Ahan Gupta, Yueming Yuan, Yanqi Zhou, and Charith Mendis. 2023. FLuRKA: Fast fused Low-Rank & Kernel Attention. arXiv:2306.15799 [cs.LG]

[97] Tae Jun Ham, Sung Jun Jung, Seonghak Kim, Young H. Oh, Yeonhong Park, Yoonho Song, Jung-Hun Park, Sanghee Lee, Kyoung Park, Jae W. Lee, and Deog-Kyoon Jeong. 2020. A^3: Accelerating Attention Mechanisms in Neural Networks with Approximation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 328–341. https://doi.org/10.1109/HPCA47549.2020.00035

[98] Tae Jun Ham, Yejin Lee, Seong Hoon Seo, Soosung Kim, Hyunji Choi, Sung Jun Jung, and Jae W. Lee. 2021. ELSA: Hardware-Software Co-design for Efficient, Lightweight Self-Attention Mechanism in Neural Networks. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. 692–705. https://doi.org/10.1109/ISCA52012.2021.00060

[99] Insu Han, Rajesh Jayaram, Amin Karbasi, Vahab Mirrokni, David P. Woodruff, and Amir Zandieh. 2023. HyperAttention: Long-context Attention in Near-Linear Time. arXiv:2310.05869 [cs.LG]

[100] Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. 2021. FastMoE: A Fast Mixture-of-Expert Training System. arXiv:2103.13262 [cs.LG]

[101] Jiaao He, Jidong Zhai, Tiago Antunes, Haojie Wang, Fuwen Luo, Shangfeng Shi, and Qin Li. 2022. FasterMoE: modeling and optimizing training of large-scale dynamic pre-trained models. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. 120–134.

[102] Kai He, Rui Mao, Qika Lin, Yucheng Ruan, Xiang Lan, Mengling Feng, and Erik Cambria. 2023. A Survey of Large Language Models for Healthcare: from Data, Technology, and Applications to Accountability and Ethics. *arXiv preprint arXiv:2310.05694* (2023).

[103] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*. 1026–1034. https://doi.org/10.1109/ICCV.2015.123

[104] Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large Language Models Are Reasoning Teachers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 14852–14882. https://doi.org/10.18653/v1/2023.acl-long.830

[105] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. 2022. An Empirical Analysis of Compute-Optimal Large Language Model Training. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=iBBcRUlOAPR

[106] Ke Hong, Guohao Dai, Jiaming Xu, Qiuli Mao, Xiuhong Li, Jun Liu, Kangdi Chen, Yuhan Dong, and Yu Wang. 2023. FlashDecoding++: Faster Large Language Model Inference on GPUs. arXiv:2311.01282 [cs.LG]

[107] Or Honovich, Uri Shaham, Samuel R. Bowman, and Omer Levy. 2022. Instruction Induction: From Few Examples to Natural Language Task Descriptions. arXiv:2205.10782 [cs.CL]

[108] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, 2790–2799. https://proceedings.mlr.press/v97/houlsby19a.html

[109] Sissie Hsiao, Yury Pinsky, and Sundar Pichai. 2023. Bard: Google's Generative Language Model. https://blog.google/products/search/bard-updates/. Accessed: October 7, 2023.

[110] Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling Step-by-Step! Outperforming Larger Language Models with Less Training Data and Smaller Model Sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 8003–8017. https://doi.org/10.18653/v1/2023.findings-acl.507

[111] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*. https://openreview.net/forum?id=nZeVKeeFYf9

[112] Shengding Hu, Ning Ding, Weilin Zhao, Xingtai Lv, Zhen Zhang, Zhiyuan Liu, and Maosong Sun. 2023. OpenDelta: A Plug-and-play Library for Parameter-efficient Adaptation of Pre-trained Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Danushka Bollegala, Ruihong Huang, and Alan Ritter (Eds.). Association for Computational Linguistics, Toronto, Canada, 274–281. https://doi.org/10.18653/v1/2023.acl-demo.26

[113] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 5254–5276. https://aclanthology.org/2023.emnlp-main.319

[114] Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. LoraHub: Efficient Cross-Task Generalization via Dynamic LoRA Composition. arXiv:2307.13269 [cs.CL]

[115] Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. 2020. Improving Transformer Optimization Through Better Initialization. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Hal Daumé III and Aarti Singh (Eds.), Vol. 119. PMLR, 4475–4483. https://proceedings.mlr.press/v119/huang20f.html

[116] Yukun Huang, Yanda Chen, Zhou Yu, and Kathleen McKeown. 2022. In-context Learning Distillation: Transferring Few-shot Learning Ability of Pre-trained Language Models. arXiv:2212.10670 [cs.CL]

[117] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. 2019. GPipe: Efficient Training of Giant Neural Networks Using Pipeline Parallelism. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., Red Hook, NY, USA, Article 10, 10 pages.

[118] DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. 2022. Block-recurrent transformers. *Advances in Neural Information Processing Systems* 35 (2022), 33248–33261.

[119] Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, et al. 2023. Tutel: Adaptive mixture-of-experts at scale. *Proceedings of Machine Learning and Systems* 5 (2023).

[120] Régis Pierrard Ilyas Moutawwakil. 2023. LLM-Perf Leaderboard. https://huggingface.co/spaces/optimum/llm-perf-leaderboard.

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

[122] Hamish Ivison, Noah A. Smith, Hannaneh Hajishirzi, and Pradeep Dasigi. 2023. Data-Efficient Finetuning Using Cross-Task Nearest Neighbors. In *Findings of the Association for Computational Linguistics: ACL 2023*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 9036–9061. https://doi.org/10.18653/v1/2023.findings-acl.576

[123] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL]

[124] Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LongLLMLingua: Accelerating and Enhancing LLMs in Long Context Scenarios via Prompt Compression. arXiv:2310.06839 [cs.CL]

[125] Yuxin Jiang, Chunkit Chan, Mingyang Chen, and Wei Wang. 2023. Lion: Adversarial Distillation of Proprietary Large Language Models. arXiv:2305.12870 [cs.CL]

[126] Yunho Jin, Chun-Feng Wu, David Brooks, and Gu-Yeon Wei. 2023. S^3: Increasing GPU Utilization during Generative Inference for Higher Throughput. arXiv:2306.06000 [cs.AR]

[127] Hoyoun Jung and Kyung-Joong Kim. 2023. Discrete Prompt Compression with Reinforcement Learning. arXiv:2308.08758 [cs.CL]

[128] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and Applications of Large Language Models. arXiv:2307.10169 [cs.CL]

[129] Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. 2019. A Study of BFLOAT16 for Deep Learning Training. arXiv:1905.12322 [cs.LG]

[130] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient Low-Rank Hypercomplex Adapter Layers. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., New Orleans, Louisiana, 1022–1035.

https://proceedings.neurips.cc/paper%5Ffiles/paper/2021/file/081be9fdff07f3bc808f935906ef70c0-Paper.pdf

[131] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Hal Daumé III and Aarti Singh (Eds.), Vol. 119. PMLR, 5156–5165. https://proceedings.mlr.press/v119/katharopoulos20a.html

[132] Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. 2022. On The Computational Complexity of Self-Attention. arXiv:2209.04881 [cs.LG]

[133] Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. 2023. Memory-Efficient Fine-Tuning of Compressed Large Language Models via sub-4-bit Integer Quantization. arXiv:2305.14152 [cs.LG]

[134] Minsoo Kim, Sihwa Lee, Janghwan Lee, Sukjin Hong, Du-Seong Chang, Wonyong Sung, and Jungwook Choi. 2023. Token-Scaled Logit Distillation for Ternary Weight Generative Language Models. arXiv:2308.06744 [cs.CL]

[135] Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W. Mahoney, Amir Gholami, and Kurt Keutzer. 2023. Speculative Decoding with Big Little Decoder. arXiv:2302.07863 [cs.CL]

[136] Young Jin Kim, Rawn Henry, Raffy Fahim, and Hany Hassan Awadalla. 2023. FineQuant: Unlocking Efficiency with Fine-Grained Weight-Only Quantization for LLMs. arXiv:2308.09723 [cs.LG]

[137] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[138] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. arXiv:2001.04451 [cs.LG]

[139] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large Language Models are Zero-Shot Reasoners. arXiv:2205.11916 [cs.CL]

[140] Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. 2023. Reducing activation recomputation in large transformer models. *Proceedings of Machine Learning and Systems* 5.

[141] Siddharth Krishna Kumar. 2017. On weight initialization in deep neural networks. arXiv:1704.08863 [cs.LG]

[142] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles* (Koblenz, Germany) *(SOSP '23)*. Association for Computing Machinery, New York, NY, USA, 611–626. https://doi.org/10.1145/3600006.3613165

[143] Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. 2023. OWQ: Lessons learned from activation outliers for weight quantization in large language models. arXiv:2306.02272 [cs.CL]

[144] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, 3744–3753. https://proceedings.mlr.press/v97/lee19d.html

[145] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. In *International Conference on Learning Representations*. https://openreview.net/forum?id=qrwe7XHTmYb

[146] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 3045–3059. https://doi.org/10.18653/v1/2021.emnlp-main.243

[147] Yaniv Leviathan, Matan Kalman, and Y. Matias. 2022. Fast Inference from Transformers via Speculative Decoding. In *International Conference on Machine Learning*.

[148] Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*. PMLR, 6265–6274.

[149] Conglong Li, Ammar Ahmad Awan, Hanlin Tang, Samyam Rajbhandari, and Yuxiong He. 2021. 1-bit LAMB: Communication Efficient Large-Scale Large-Batch Training with LAMB's Convergence Speed. In *HiPC 2022*. arXiv:arXiv:2104.06069

[150] Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. 2023. Symbolic Chain-of-Thought Distillation: Small Models Can Also "Think" Step-by-Step. *ArXiv* abs/2306.14050 (2023).

[151] SHIYANG LI, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jingu Qian, Baolin Peng, Yi Mao, Wenhu Chen, and Xifeng Yan. 2022. Explanations from Large Language Models Make Small Reasoners Better. *ArXiv* abs/2210.06726 (2022).

[152] Shenggui Li, Hongxin Liu, Zhengda Bian, Jiarui Fang, Haichen Huang, Yuliang Liu, Boxiang Wang, and Yang You. 2023. Colossal-AI: A Unified Deep Learning System For Large-Scale Parallel Training. In *Proceedings of the 52nd International Conference on Parallel Processing* (<conf-loc>, <city>Salt Lake City</city>, <state>UT</state>,

<country>USA</country>, </conf-loc>) *(ICPP '23)*. Association for Computing Machinery, New York, NY, USA, 766–775. https://doi.org/10.1145/3605573.3605613

[153] S. Li, F. Xue, C. Baranwal, Y. Li, and Y. You. 2021. Sequence Parallelism: Long Sequence Training from System Perspective. *arXiv* (2021).

[154] Shanda Li, Chong You, Guru Guruganesh, Joshua Ainslie, Santiago Ontañón, Manzil Zaheer, Sumit Sanghai, Yiming Yang, Sanjiv Kumar, and Srinadh Bhojanapalli. 2023. Functional Interpolation for Relative Positions Improves Long Context Transformers. *CoRR* abs/2310.04418 (2023).

[155] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. 2020. PyTorch Distributed: Experiences on Accelerating Data Parallel Training. *CoRR* abs/2006.15704 (2020).

[156] Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023. Unified Demonstration Retriever for In-Context Learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Toronto, Canada, 4644–4668. https://doi.org/10.18653/v1/2023.acl-long.256

[157] Xiaonan Li and Xipeng Qiu. 2023. Finding supporting examples for in-context learning. *arXiv preprint arXiv:2302.13539* (2023).

[158] Xiang Li, Yiqun Yao, Xin Jiang, Xuezhi Fang, Xuying Meng, Siqi Fan, Peng Han, Jing Li, Li Du, Bowen Qin, et al. 2023. FLM-101B: An Open LLM and How to Train It with $100 K Budget. *arXiv preprint arXiv:2309.03852* (2023).

[159] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* abs/2101.00190 (2021).

[160] Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. 2023. LoftQ: LoRA-Fine-Tuning-Aware Quantization for Large Language Models. *arXiv preprint arXiv:2310.08659* (2023).

[161] Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2023. LoSparse: Structured Compression of Large Language Models based on Low-Rank and Sparse Approximation. In *International Conference on Machine Learning*. https://api.semanticscholar.org/CorpusID:259203385

[162] Zhuohan Li, Lianmin Zheng, Yinmin Zhong, Vincent Liu, Ying Sheng, Xin Jin, Yanping Huang, Zhifeng Chen, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. AlpaServe: Statistical Multiplexing with Model Parallelism for Deep Learning Serving. In *Proceedings of the 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.

[163] Chen Liang, Simiao Zuo, Qingru Zhang, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2023. Less is more: Task-aware layer-wise distillation for language model compression. In *International Conference on Machine Learning*. PMLR, 20852–20867.

[164] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. *arXiv preprint arXiv:2306.00978* (2023).

[165] Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. 2023. Sophia: A Scalable Stochastic Second-order Optimizer for Language Model Pre-training. *arXiv preprint arXiv:2305.14342* (2023).

[166] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning. *ArXiv* abs/2205.05638 (2022).

[167] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning. arXiv:2205.05638 [cs.LG]

[168] Jing Liu, Ruihao Gong, Xiuying Wei, Zhiwei Dong, Jianfei Cai, and Bohan Zhuang. 2023. QLLM: Accurate and Efficient Low-Bitwidth Quantization for Large Language Models. *arXiv preprint arXiv:2310.08041* (2023).

[169] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What Makes Good In-Context Examples for GPT-3?. In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*. Association for Computational Linguistics, Dublin, Ireland and Online, 100–114. https://doi.org/10.18653/v1/2022.deelio-1.10

[170] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Comput. Surveys* 55, 9 (2023), 1–35.

[171] Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks. *ArXiv* abs/2110.07602 (2021).

[172] Xiaoxuan Liu, Lianmin Zheng, Dequan Wang, Yukuo Cen, Weize Chen, Xu Han, Jianfei Chen, Zhiyuan Liu, Jie Tang, Joey Gonzalez, et al. 2022. GACT: Activation compressed training for generic network architectures. In *International Conference on Machine Learning*. PMLR, 14139–14152.

[173] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT Understands, Too. *ArXiv* abs/2103.10385 (2021).

[174] Y. Liu, S. Li, J. Fang, Y. Shao, B. Yao, and Y. You. 2023. Colossal-Auto: Unified Automation of Parallelization and Activation Checkpoint for Large-scale Models. *arXiv* (2023).

[175] Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023. Scissorhands: Exploiting the Persistence of Importance Hypothesis for LLM KV Cache Compression at Test Time. *arXiv preprint arXiv:2305.17118* (2023).

[176] Zechun Liu, Barlas Oğuz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023. LLM-QAT: Data-Free Quantization Aware Training for Large Language Models. *ArXiv* abs/2305.17888 (2023).

[177] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. 2023. Deja Vu: Contextual Sparsity for Efficient LLMs at Inference Time. In *International Conference on Machine Learning*. PMLR, 22137–22176.

[178] Ilya Loshchilov and Frank Hutter. 2017. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101* (2017).

[179] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, 8086–8098. https://doi.org/10.18653/v1/2022.acl-long.556

[180] Yucheng Lu, Conglong Li, Minjia Zhang, Christopher De Sa, and Yuxiong He. 2022. Maximizing Communication Efficiency for Large-scale Training via 0/1 Adam. (2022). arXiv:arXiv:2202.06009

[181] Man Luo, Xin Xu, Zhuyun Dai, Panupong Pasupat, Mehran Kazemi, Chitta Baral, Vaiva Imbrasaite, and Vincent Y Zhao. 2023. Dr. ICL: Demonstration-Retrieved In-context Learning. *arXiv preprint arXiv:2305.14128* (2023).

[182] Kai Lv, Yuqing Yang, Tengxiao Liu, Qi jie Gao, Qipeng Guo, and Xipeng Qiu. 2023. Full Parameter Fine-tuning for Large Language Models with Limited Resources. *ArXiv* abs/2306.09782 (2023).

[183] Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-Pruner: On the Structural Pruning of Large Language Models. *arXiv preprint arXiv:2305.11627* (2023).

[184] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. 2023. Fine-Tuning Language Models with Just Forward Passes. *arXiv preprint arXiv:2305.17333* (2023).

[185] Pedro Henrique Martins, Zita Marinho, and Andre Martins. 2022. ∞-former: Infinite Memory Transformer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 5468–5485. https://doi.org/10.18653/v1/2022.acl-long.375

[186] Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. 2022. Long range language modeling via gated state spaces. *arXiv preprint arXiv:2206.13947* (2022).

[187] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Rae Ying Yee Wong, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2023. SpecInfer: Accelerating Generative LLM Serving with Speculative Inference and Token Tree Verification. *ArXiv* abs/2305.09781 (2023).

[188] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2017. Mixed precision training. *arXiv preprint arXiv:1710.03740* (2017).

[189] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2021. MetaICL: Learning to Learn In Context. *ArXiv* abs/2110.15943 (2021).

[190] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the Role of Demonstrations: What makes In-context Learning Work?. In *EMNLP*.

[191] Amirkeivan Mohtashami and Martin Jaggi. 2023. Landmark Attention: Random-Access Infinite Context Length for Transformers. *arXiv preprint arXiv:2305.16300* (2023).

[192] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. 2018. Ray: A distributed framework for emerging *AI* applications. In *13th USENIX symposium on operating systems design and implementation (OSDI 18)*. 561–577.

[193] MosaicML. 2023. Composer. https://github.com/mosaicml/composer. GitHub repository.

[194] MosaicML. 2023. LLM Foundry. https://github.com/mosaicml/llm-foundry. GitHub repository.

[195] Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *arXiv preprint arXiv:2304.08467* (2023).

[196] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei Zaharia. 2019. PipeDream: Generalized Pipeline Parallelism for DNN Training. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles* (Huntsville, Ontario, Canada) *(SOSP '19)*. Association for

Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3341301.3359646
[197] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. 2021. Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (St. Louis, Missouri) *(SC '21)*. Association for Computing Machinery, New York, NY, USA, Article 58, 15 pages. https://doi.org/10.1145/3458817.3476209
[198] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474* (2022).
[199] NVIDIA. 2023. FasterTransformer: High Performance Transformer Kernels. https://github.com/NVIDIA/FasterTransformer. GitHub repository.
[200] OpenAI. 2023. GPT-4 Technical Report. *ArXiv* abs/2303.08774 (2023).
[201] OpenAI. 2023. GPT Base Model. https://platform.openai.com/docs/models/gpt-base. Accessed: 2023-12-13.
[202] Shankar Padmanabhan, Yasumasa Onoe, Michael JQ Zhang, Greg Durrett, and Eunsol Choi. 2023. Propagating Knowledge Updates to LMs Through Distillation. *arXiv preprint arXiv:2306.09306* (2023).
[203] Matteo Pagliardini, Daniele Paliotta, Martin Jaggi, and Franccois Fleuret. 2023. Faster Causal Attention Over Large Sequences Through Sparse Flash Attention. *ArXiv* abs/2306.01160 (2023).
[204] Yu Pan, Ye Yuan, Yichun Yin, Zenglin Xu, Lifeng Shang, Xin Jiang, and Qun Liu. 2023. Reusing Pretrained Models by Multi-linear Operators for Efficient Training. *CoRR* abs/2310.10699 (2023).
[205] Zizheng Pan, Peng Chen, Haoyu He, Jing Liu, Jianfei Cai, and Bohan Zhuang. 2021. Mesa: A Memory-saving Training Framework for Transformers. *arXiv preprint arXiv:2111.11124* (2021).
[206] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, et al. 2023. RWKV: Reinventing RNNs for the Transformer Era. *arXiv preprint arXiv:2305.13048* (2023).
[207] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction Tuning with GPT-4. *ArXiv* abs/2304.03277 (2023).
[208] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071* (2023).
[209] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. 2021. Random feature attention. *arXiv preprint arXiv:2103.02143* (2021).
[210] Aaron Pham, Chaoyu Yang, Sean Sheng, Shenyang Zhao, Sauyon Lee, Bo Jiang, Fog Dong, Xipeng Guan, and Frost Ming. 2023. *OpenLLM: Operating LLMs in production.* https://github.com/bentoml/OpenLLM
[211] Jonathan Pilault, Mahan Fathi, Orhan Firat, Christopher Pal, Pierre-Luc Bacon, and Ross Goroshin. 2023. Block-State Transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*. New Orleans, Louisiana. https://openreview.net/forum?id=XRTxIBs2eu
[212] Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. 2023. Hyena hierarchy: Towards larger convolutional language models. *arXiv preprint arXiv:2302.10866* (2023).
[213] Edoardo Maria Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. 2023. Combining Parameter-efficient Modules for Task-level Generalisation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. 687–702.
[214] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently Scaling Transformer Inference. *Proceedings of Machine Learning and Systems* 5 (2023).
[215] Ofir Press, Noah Smith, and Mike Lewis. 2022. Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation. In *International Conference on Learning Representations*. https://openreview.net/forum?id=R8sQPpGCv0
[216] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and Narrowing the Compositionality Gap in Language Models. arXiv:2210.03350 [cs.CL]
[217] Ray Project. 2023. *RayLLM - LLMs on Ray.* https://github.com/ray-project/ray-llm GitHub repository.
[218] Chengwei Qin, Aston Zhang, Anirudh Dagar, and Wenming Ye. 2023. In-Context Learning with Iterative Demonstration Selection. *arXiv preprint arXiv:2310.09881* (2023).
[219] Guanghui Qin, Corby Rosset, Ethan C. Chau, Nikhil Rao, and Benjamin Van Durme. 2023. Nugget 2D: Dynamic Contextual Compression for Scaling Decoder-only Language Models. https://api.semanticscholar.org/CorpusID:263620438
[220] Yujia Qin, Yankai Lin, Jing Yi, Jiajie Zhang, Xu Han, Zhengyan Zhang, Yusheng Su, Zhiyuan Liu, Peng Li, Maosong Sun, et al. 2021. Knowledge inheritance for pre-trained language models. *arXiv preprint arXiv:2105.13880* (2021).

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

[222] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. https://api.semanticscholar.org/CorpusID:160025533

[223] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, et al. 2021. Scaling Language Models: Methods, Analysis & Insights from Training Gopher. *ArXiv* abs/2112.11446 (2021). https://api.semanticscholar.org/CorpusID:245353475

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

[225] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. 2022. DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale. In *International Conference on Machine Learning*.

[226] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. ZeRO: Memory Optimizations toward Training Trillion Parameter Models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Atlanta, Georgia) *(SC '20)*. IEEE Press, Article 20, 16 pages.

[227] Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. 2021. ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (St. Louis, Missouri) *(SC '21)*. Association for Computing Machinery, New York, NY, USA, Article 59, 14 pages. https://doi.org/10.1145/3458817.3476205

[228] Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. 2021. ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning. In *SC 2021*. arXiv:arXiv:2104.07857

[229] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20, Tutorial)*.

[230] Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. Parallel context windows for large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 6383–6402.

[231] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. ZeRO-Offload: Democratizing Billion-Scale Model Training. In *USENIX ATC 2021*. arXiv:arXiv:2101.06840

[232] Liliang Ren, Yang Liu, Shuohang Wang, Yichong Xu, Chenguang Zhu, and ChengXiang Zhai. 2023. Sparse Modular Activation for Efficient Sequence Modeling. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=TfbzX6I14i

[233] Xiaozhe Ren, Pingyi Zhou, Xinfan Meng, Xinjing Huang, Yadao Wang, Weichao Wang, Pengfei Li, Xiaoda Zhang, A. V. Podolskiy, Grigory Arshinov, A. Bout, Irina Piontkovskaya, Jiansheng Wei, Xin Jiang, Teng Su, Qun Liu, and Jun Yao. 2023. PanGu-Σ: Towards Trillion Parameter Language Model with Sparse Heterogeneous Computing. *ArXiv* abs/2303.10845 (2023). https://api.semanticscholar.org/CorpusID:257666647

[234] Adithya Renduchintala, Tugrul Konuk, and Oleksii Kuchaiev. 2023. Tied-Lora: Enhacing parameter efficiency of LoRA with weight tying.

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

[236] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning To Retrieve Prompts for In-Context Learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Seattle, United States, 2655–2671. https://doi.org/10.18653/v1/2022.naacl-main.191

[237] Lucía Santamaría and Amittai Axelrod. 2019. Data selection with cluster-based language difference models and cynical selection. *arXiv preprint arXiv:1904.04900* (2019).

[238] Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodola. 2023. Accelerating Transformer Inference for Translation via Parallel Decoding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 12336–12355. https://doi.org/10.18653/v1/2023.acl-long.689

[239] Teven Le Scao, Angela Fan, Christopher Akiki, Elizabeth-Jane Pavlick, Suzana Ili'c, Daniel Hesslow, Roman Castagn'e, Alexandra Sasha Luccioni, Franccois Yvon, Matthias Gallé, et al. 2022. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. *ArXiv* abs/2211.05100 (2022). https://api.semanticscholar.org/CorpusID:253420279

[240] Stephanie Schoch, Ritwick Mishra, and Yangfeng Ji. 2023. Data Selection for Fine-tuning Large Language Models Using Transferred Shapley Values. *arXiv preprint arXiv:2306.10165* (2023).

[241] Christopher J. Shallue, Jaehoon Lee, Joseph M. Antognini, Jascha Narain Sohl-Dickstein, Roy Frostig, and George E. Dahl. 2018. Measuring the Effects of Data Parallelism on Neural Network Training. *ArXiv* abs/1811.03600 (2018). https://api.semanticscholar.org/CorpusID:53214190

[242] Hang Shao, Bei Liu, and Yanmin Qian. 2023. One-Shot Sensitivity-Aware Mixed Sparsity Pruning for Large Language Models. https://api.semanticscholar.org/CorpusID:264146174

[243] Junru Shao, Xiyou Zhou, Siyuan Feng, Bohan Hou, Ruihang Lai, Hongyi Jin, Wuwei Lin, Masahiro Masuda, Cody Hao Yu, and Tianqi Chen. 2022. Tensor Program Optimization with Probabilistic Programs. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 35783–35796. https://proceedings.neurips.cc/paper%5Ffiles/paper/2022/file/e894eafae43e68b4c8dfdacf742bcbf3-Paper-Conference.pdf

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

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

[246] Sheng Shen, Le Hou, Yan-Quan Zhou, Nan Du, S. Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, Tu Vu, Yuexin Wu, Wuyang Chen, Albert Webson, Yunxuan Li, Vincent Zhao, Hongkun Yu, Kurt Keutzer, Trevor Darrell, and Denny Zhou. 2023. Mixture-of-Experts Meets Instruction Tuning: A Winning Combination for Large Language Models.

[247] Sheng Shen, Pete Walsh, Kurt Keutzer, Jesse Dodge, Matthew Peters, and Iz Beltagy. 2022. Staged training for transformer language models. In *International Conference on Machine Learning*. PMLR, 19893–19908.

[248] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Daniel Y. Fu, Zhiqiang Xie, Beidi Chen, Clark W. Barrett, Joseph Gonzalez, Percy Liang, Christopher Ré, Ioan Cristian Stoica, and Ce Zhang. 2023. High-throughput Generative Inference of Large Language Models with a Single GPU. In *International Conference on Machine Learning*.

[249] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980* (2020).

[250] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053* (2019).

[251] Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2022. Distilling Reasoning Capabilities into Smaller Language Models. In *Annual Meeting of the Association for Computational Linguistics*.

[252] Antoine Simoulin, Namyong Park, Xiaoyi Liu, and Grey Yang. 2023. Memory-Efficient Selective Fine-Tuning. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*.

[253] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Anand Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. 2022. Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model. *ArXiv* abs/2201.11990 (2022).

[254] Saleh Soltan, Shankar Ananthakrishnan, Jack FitzGerald, Rahul Gupta, Wael Hamza, Haidar Khan, Charith Peris, Stephen Rawls, Andy Rosenbaum, Anna Rumshisky, et al. 2022. Alexatm 20b: Few-shot learning using a large-scale multilingual seq2seq model. *arXiv preprint arXiv:2208.01448* (2022).

[255] James C. Spall. 1992. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Trans. Automat. Control* 37 (1992), 332–341.

[256] Benjamin Spector and Chris Re. 2023. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623* (2023).

[257] Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2022. Selective Annotation Makes Language Models Better Few-Shot Learners. arXiv:2209.01975 [cs.CL]

[258] Hui Su, Xiao Zhou, Houjin Yu, Xiaoyu Shen, Yuwen Chen, Zilin Zhu, Yang Yu, and Jie Zhou. 2022. WeLM: A Well-Read Pre-trained Language Model for Chinese. *arXiv preprint arXiv:2209.10372* (2022).

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

[260] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2023. A Simple and Effective Pruning Approach for Large Language Models. *ArXiv* abs/2306.11695 (2023).

[261] Tianxiang Sun, Zhengfu He, Qinen Zhu, Xipeng Qiu, and Xuanjing Huang. 2022. Multitask Pre-training of Modular Prompt for Chinese Few-Shot Learning. In *Annual Meeting of the Association for Computational Linguistics*.

[262] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. [n.d.]. Retentive Network: A Successor to Transformer for Large Language Models (2023). *arXiv preprint ArXiv:2307.08621* ([n. d.]).

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

[264] Richard S. Sutton, David A. McAllester, Satinder Singh, and Y. Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NIPS*.

[265] Weng Lam Tam, Xiao Liu, Kaixuan Ji, Lilong Xue, Xingjian Zhang, Yuxiao Dong, Jiahua Liu, Maodi Hu, and Jie Tang. 2022. Parameter-Efficient Prompt Tuning Makes Generalized and Calibrated Neural Text Retrievers. *CoRR* abs/2207.07087 (2022). https://doi.org/10.48550/arXiv.2207.07087 arXiv:2207.07087

[266] Hanlin Tang, Shaoduo Gan, Ammar Ahmad Awan, Samyam Rajbhandari, Conglong Li, Xiangru Lian, Ji Liu, Ce Zhang, and Yuxiong He. 2021. 1-bit Adam: Communication Efficient Large-Scale Training with Adam's Convergence Speed. In *ICML 2021*. arXiv:arXiv:2102.02888

[267] Chaofan Tao, Lu Hou, Wei Zhang, Lifeng Shang, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. 2022. Compression of generative pre-trained language models via quantization. *arXiv preprint arXiv:2203.10705* (2022).

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

[269] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient Transformers: A Survey. *ACM Comput. Surv.* 55, 6, Article 109 (dec 2022), 28 pages. https://doi.org/10.1145/3530811

[270] Gemini Team and Google. 2023. Gemini: A Family of Highly Capable Multimodal Models. https://storage.googleapis.com/deepmind-media/gemini/gemini_1_report.pdf.

[271] MLC team. 2023. *MLC-LLM*. https://github.com/mlc-ai/mlc-llm

[272] The MosaicML NLP Team. 2023. Introducing MPT-7B: A New Standard for Open-Source, Commercially Usable LLMs. https://www.mosaicml.com/blog/mpt-7b. Accessed: 2023-12-13.

[273] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239* (2022).

[274] Inar Timiryasov and Jean-Loup Tastet. 2023. Baby Llama: knowledge distillation from an ensemble of teachers trained on a small dataset with no performance penalty. *arXiv preprint arXiv:2308.02019* (2023).

[275] Denis Timonin, Bo Yang Hsueh, and Vinh Nguyen. 2022. Accelerated Inference for Large Transformer Models using Nvidia Triton Inference Server. *NVIDIA blog* (2022).

[276] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *ArXiv* abs/2302.13971 (2023). https://api.semanticscholar.org/CorpusID:257219404

[277] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *ArXiv* abs/2307.09288 (2023). https://api.semanticscholar.org/CorpusID:259950998

[278] Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr MiłOś. 2023. Focused transformer: Contrastive training for context scaling. *arXiv preprint arXiv:2307.03170* (2023).

[279] Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2022. DyLoRA: Parameter-Efficient Tuning of Pre-trained Models using Dynamic Search-Free Low-Rank Adaptation. *ArXiv* abs/2210.07558 (2022).

[280] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. https://api.semanticscholar.org/CorpusID:13756489

[281] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. 2020. Fast transformers with clustered attention. *Advances in Neural Information Processing Systems* 33 (2020), 21665–21674.

[282] Zhongwei Wan, Che Liu, Mi Zhang, Jie Fu, Benyou Wang, Sibo Cheng, Lei Ma, César Quilodrán-Casas, and Rossella Arcucci. 2023. Med-UniC: Unifying Cross-Lingual Medical Vision-Language Pre-Training by Diminishing Bias. *arXiv preprint arXiv:2305.19894* (2023).

[283] Zhongwei Wan, Yichun Yin, Wei Zhang, Jiaxin Shi, Lifeng Shang, Guangyong Chen, Xin Jiang, and Qun Liu. 2022. G-MAP: General Memory-Augmented Pre-trained Language Model for Domain Tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 6585–6597. https://doi.org/10.18653/v1/2022.emnlp-main.441

[284] Boxiang Wang, Qifan Xu, Zhengda Bian, and Yang You. 2021. 2.5-dimensional distributed model training. *arXiv e-prints* (2021), arXiv–2105.

[285] B. Wang, Q. Xu, Z. Bian, and Y. You. 2022. Tesseract: Parallelize the Tensor Parallelism Efficiently. In *Proceedings of the 51th International Conference on Parallel Processing*.

[286] Guoxin Wang, Yijuan Lu, Lei Cui, Tengchao Lv, Dinei Florencio, and Cha Zhang. 2022. A Simple yet Effective Learnable Positional Encoding Method for Improving Document Transformer Model. In *Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022*. 453–463.

[287] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. 2023. BitNet: Scaling 1-bit Transformers for Large Language Models. https://api.semanticscholar.org/CorpusID:264172438

[288] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. 2022. Deepnet: Scaling transformers to 1,000 layers. *arXiv preprint arXiv:2203.00555* (2022).

[289] Liang Wang, Nan Yang, and Furu Wei. 2023. Learning to Retrieve In-Context Examples for Large Language Models. arXiv:2307.07164 [cs.CL]

[290] Ningning Wang, Guobing Gan, Peng Zhang, Shuai Zhang, Junqiu Wei, Qun Liu, and Xin Jiang. 2022. Clusterformer: Neural clustering attention for efficient and effective transformer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2390–2402.

[291] Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. 2023. Learning to grow pretrained models for efficient transformer training. *arXiv preprint arXiv:2303.00980* (2023).

[292] Peifeng Wang, Zhengyang Wang, Zheng Li, Yifan Gao, Bing Yin, and Xiang Ren. 2023. SCOTT: Self-Consistent Chain-of-Thought Distillation. In *Annual Meeting of the Association for Computational Linguistics*.

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

[294] Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Augmenting Language Models with Long-Term Memory. *arXiv preprint arXiv:2306.07174* (2023).

[295] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Huai hsin Chi, and Denny Zhou. 2022. Self-Consistency Improves Chain of Thought Reasoning in Language Models. *ArXiv* abs/2203.11171 (2022).

[296] Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang. 2023. Large Language Models Are Latent Variable Models: Explaining and Finding Good Demonstrations for In-Context Learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[297] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 13484–13508. https://doi.org/10.18653/v1/2023.acl-long.754

[298] Yaqing Wang, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, and Jianfeng Gao. 2022. AdaMix: Mixture-of-Adaptations for Parameter-efficient Model Tuning. *ArXiv* abs/2210.17451 (2022).

[299] Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. 2023. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966* (2023).

[300] Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogério Schmidt Feris, Huan Sun, and Yoon Kim. 2023. Multitask Prompt Tuning Enables Parameter-Efficient Transfer Learning. *ArXiv* abs/2303.02861 (2023).

[301] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed Huai hsin Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent Abilities of Large Language Models. *Trans. Mach. Learn. Res.* 2022 (2022).

[302] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35, 24824–24837.

[303] Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. 2023. Outlier Suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *ArXiv* abs/2304.09145 (2023).

[304] Genta Indra Winata, Samuel Cahyawijaya, Zhaojiang Lin, Zihan Liu, and Pascale Fung. 2020. Lightweight and efficient end-to-end speech recognition using low-rank transformer. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6144–6148.

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

[306] Xiaoxia Wu, Cheng Li, Reza Yazdani Aminabadi, Zhewei Yao, and Yuxiong He. 2023. Understanding INT4 Quantization for Transformer Models: Latency Speedup, Composability, and Failure Cases. (2023). arXiv:arXiv:2301.12017

[307] Xiaoxia Wu, Zhewei Yao, and Yuxiong He. 2023. ZeroQuant-FP: A Leap Forward in LLMs Post-Training W4A8 Quantization Using Floating-Point Formats. (2023). arXiv:arXiv:2307.09782

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

[309] Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. 2023. Self-Adaptive In-Context Learning: An Information Compression Perspective for In-Context Example Selection and Ordering. arXiv:2212.10375 [cs.CL]

[310] M. Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared LLaMA: Accelerating Language Model Pre-training via Structured Pruning.

[311] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*. PMLR, 38087–38099.

[312] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient Streaming Language Models with Attention Sinks. *arXiv preprint arXiv:2309.17453* (2023).

[313] Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. 2023. Data selection for language models via importance resampling. *arXiv preprint arXiv:2302.03169* (2023).

[314] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. 2021. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 14138–14148.

[315] Mingxue Xu, Yao Lei Xu, and Danilo P. Mandic. 2023. TensorGPT: Efficient Compression of the Embedding Layer in LLMs based on the Tensor-Train Decomposition. *ArXiv* abs/2307.00526 (2023).

[316] Peng Xu, Wei Ping, Xianchao Wu, Lawrence C. McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. 2023. Retrieval meets Long Context Large Language Models. https://api.semanticscholar.org/CorpusID:263620134

[317] Qifan Xu and Yang You. 2023. An Efficient 2D Method for Training Super-Large Deep Learning Models. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 222–232. https://doi.org/10.1109/IPDPS54959.2023.00031

[318] Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhensu Chen, Xiaopeng Zhang, and Qi Tian. 2023. QA-LoRA: Quantization-Aware Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2309.14717* (2023).

[319] Cheng Yang, Shengnan Wang, Chao Yang, Yuechuan Li, Ru He, and Jingqiao Zhang. 2020. Progressively stacking 2.0: A Multi-stage Layerwise Training Method for BERT Training Speedup. *arXiv preprint arXiv:2011.13635* (2020).

[320] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2023. Large Language Models as Optimizers. arXiv:2309.03409 [cs.LG]

[321] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. 2023. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. *ArXiv* abs/2304.13712 (2023).

[322] Keming Yang, Zichen Liu, and Philip Cheng. 2021. *MOSEC: Model Serving made Efficient in the Cloud.* https://github.com/mosecorg/mosec

[323] Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Inference with Reference: Lossless Acceleration of Large Language Models. *ArXiv* abs/2304.04487 (2023). https://api.semanticscholar.org/CorpusID:258048436

[324] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. arXiv:2305.10601 [cs.CL]

[325] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629 [cs.CL]

[326] Xingcheng Yao, Yanan Zheng, Xiaocong Yang, and Zhilin Yang. 2022. Nlp from scratch without large-scale pretraining: A simple and efficient framework. In *International Conference on Machine Learning*. PMLR, 25438–25451.

[327] Yiqun Yao, Zheng Zhang, Jing Li, and Yequan Wang. 2023. 2x Faster Language Model Pre-training via Masked Structural Growth. *arXiv preprint arXiv:2305.02869* (2023).

[328] Zhewei Yao, Reza Yazdani Aminabadi, Olatunji Ruwase, Samyam Rajbhandari, Xiaoxia Wu, Ammar Ahmad Awan, Jeff Rasley, Minjia Zhang, Conglong Li, Connor Holmes, Zhongzhu Zhou, Michael Wyatt, Molly Smith, Lev Kurilenko, Heyang Qin, Masahiro Tanaka, Shuai Che, Shuaiwen Leon Song, and Yuxiong He. 2023. DeepSpeed-Chat: Easy, Fast and Affordable RLHF Training of ChatGPT-like Models at All Scales. (2023). arXiv:arXiv:2308.01320

[329] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers. In *NeurIPS 2022*. arXiv:arXiv:2206.01861

[330] Zhewei Yao, Xiaoxia Wu, Cheng Li, Stephen Youn, and Yuxiong He. 2023. ZeroQuant-V2: Exploring Post-training Quantization in LLMs from Comprehensive Study to Low Rank Compensation.

[331] Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. 2023. EdgeMoE: Fast On-Device Inference of MoE-based Large Language Models. *arXiv preprint arXiv:2308.14352* (2023).

[332] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A distributed serving system for Transformer-Based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. 521–538.

[333] Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. 2023. Megabyte: Predicting million-byte sequences with multiscale transformers. *arXiv preprint arXiv:2305.07185* (2023).

[334] Zhihang Yuan, Lin Niu, Jia-Wen Liu, Wenyu Liu, Xinggang Wang, Yuzhang Shang, Guangyu Sun, Qiang Wu, Jiaxiang Wu, and Bingzhe Wu. 2023. RPTQ: Reorder-based Post-training Quantization for Large Language Models. *ArXiv* abs/2304.01089 (2023).

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

[336] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, P. Zhang, Yuxiao Dong, and Jie Tang. 2022. GLM-130B: An Open Bilingual Pre-trained Model. *ArXiv* abs/2210.02414 (2022). https://api.semanticscholar.org/CorpusID:252715691

[337] Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, ZhenZhang Yang, Kaisheng Wang, Xiaoda Zhang, et al. 2021. Pangu-α: Large-scale Autoregressive Pretrained Chinese Language Models with Auto-parallel Computation. *arXiv preprint arXiv:2104.12369* (2021).

[338] Mingshu Zhai, Jiaao He, Zixuan Ma, Zan Zong, Runqing Zhang, and Jidong Zhai. 2023. SmartMoE: Efficiently Training Sparsely-Activated Models through Combining Offline and Online Parallelization. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. 961–975.

[339] Chen Zhang, Dawei Song, Zheyu Ye, and Yan Gao. 2023. Towards the Law of Capacity Gap in Distilling Language Models. *arXiv preprint arXiv:2311.07052* (2023).

[340] Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. 2019. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321* (2019).

[341] Hang Zhang, Yeyun Gong, Yelong Shen, Weisheng Li, Jiancheng Lv, Nan Duan, and Weizhu Chen. 2021. Poolingformer: Long document modeling with pooling attention. In *International Conference on Machine Learning*. PMLR, 12437–12446.

[342] Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. 2023. LoRA-FA: Memory-efficient Low-rank Adaptation for Large Language Models Fine-tuning. *ArXiv* abs/2308.03303 (2023).

[343] Mingyang Zhang, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, Bohan Zhuang, et al. 2023. Pruning Meets Low-Rank Parameter-Efficient Fine-Tuning. *arXiv preprint arXiv:2305.18403* (2023).

[344] Qingru Zhang, Minshuo Chen, Alexander W. Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. *ArXiv* abs/2303.10512 (2023).

[345] Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Jiao Qiao. 2023. LLaMA-Adapter: Efficient Fine-tuning of Language Models with Zero-init Attention. *ArXiv* abs/2303.16199 (2023).

[346] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. OPT: Open Pre-trained Transformer Language Models. *arXiv preprint arXiv:2205.01068* (2022).

[347] Tianyi Zhang, Mina Lee, Lisa Li, Ende Shen, and Tatsunori B Hashimoto. 2022. TempLM: Distilling Language Models into Template-Based Generators. *arXiv preprint arXiv:2205.11055* (2022).

[348] Yue Zhang, Hongliang Fei, Dingcheng Li, and Ping Li. 2022. Promptgen: Automatically generate prompts using generative models. In *Findings of the Association for Computational Linguistics: NAACL 2022*. 30–37.

[349] Yiming Zhang, Shi Feng, and Chenhao Tan. 2022. Active Example Selection for In-Context Learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 9134–9148. https://aclanthology.org/2022.emnlp-main.622

[350] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. 2023. H_2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. *arXiv preprint arXiv:2306.14048* (2023).

[351] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic Chain of Thought Prompting in Large Language Models. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=5NTt8GFjUHkr

[352] Jiawei Zhao, Florian Schäfer, and Anima Anandkumar. 2021. ZerO Initialization: Initializing Neural Networks with only Zeros and Ones. *arXiv preprint arXiv:2110.12661* (2021).

[353] Weilin Zhao, Yuxiang Huang, Xu Han, Zhiyuan Liu, Zhengyan Zhang, and Maosong Sun. 2023. CPET: Effective Parameter-Efficient Tuning for Compressed Large Language Models. *ArXiv* abs/2307.07705 (2023).

[354] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Z. Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jianyun Nie, and Ji rong Wen. 2023. A Survey of Large Language Models. *ArXiv* abs/2303.18223 (2023).

[355] Yanli Zhao, Andrew Gu, Rohan Varma, Liangchen Luo, Chien chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, and Shen Li. 2023. PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel. *Proc. VLDB Endow.* 16 (2023), 3848–3860. https://api.semanticscholar.org/CorpusID:258297871

[356] Lianmin Zheng, Zhuohan Li, Hao Zhang, Yonghao Zhuang, Zhifeng Chen, Yanping Huang, Yida Wang, Yuanzhong Xu, Danyang Zhuo, Eric P. Xing, Joseph E. Gonzalez, and Ion Stoica. 2022. Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning. In *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.

[357] Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Zihan Wang, Lei Shen, Andi Wang, Yang Li, et al. 2023. Codegeex: A pre-trained model for code generation with multilingual evaluations on humaneval-x. *arXiv preprint arXiv:2303.17568* (2023).

[358] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. LIMA: Less Is More for Alignment. arXiv:2305.11206 [cs.CL]

[359] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. arXiv:2205.10625 [cs.AI]

[360] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. 2022. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems* 35 (2022), 7103–7114.

[361] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large Language Models are Human-Level Prompt Engineers. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=92gvk82DE-

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

[363] Bohan Zhuang, Jing Liu, Zizheng Pan, Haoyu He, Yuetian Weng, and Chunhua Shen. 2023. A survey on efficient training of transformers. *arXiv preprint arXiv:2302.01107* (2023).

[364] Yonghao Zhuang, Hexu Zhao, Lianmin Zheng, Zhuohan Li, Eric P. Xing, Qirong Ho, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. 2023. On Optimizing the Communication of Model Parallelism. In *Proceedings of Machine Learning and Systems (MLSys)*.

[365] Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Tuo Zhao, and Jianfeng Gao. 2021. Taming Sparsely Activated Transformer with Stochastic Experts. *ArXiv* abs/2110.04260 (2021).