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
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.

CCS Concepts: • **Computing methodologies → Machine learning**; **Natural language processing**.

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

ACM Reference Format:
Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, 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), 53 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

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; 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 the surge of LLMs include GPT-series (GPT-3 [21] and GPT-4 [197]), Google-series (Gemini [266], GLaM [71], PaLM [50], PaLM-2 [8]), Meta-series (LLaMA 1&2 [272, 273]), BLOOM [233], PanGu-$\sum$ [227], and GLM [339], as well as the remarkable performance they have achieved in a variety of tasks such as natural language understanding (NLU), language generation, complex reasoning [320], and domain-specific tasks related to biomedicine [278, 280], law [72] and code generation [34, 300]. Such performance breakthroughs can be attributed to their massive scales, as they contain billions or even trillions of parameters while being trained on a gigantic amount of data from a diverse range of 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 [50, 71, 197, 227]. Figure 1 illustrates the relationship between model performance and the carbon emissions during training for LLaMA series. As shown, the amount of carbon emitted grows exponentially as the number of model parameter scales up. In addition to training, inference also contributes quite significantly to the operational cost of LLMs. As depicted in Figure 2, more advanced LLMs exhibit higher memory usage and energy consumption during inference, presenting challenges for these models in expanding their reach to a broader customer base and diverse applications in a cost-effective way. With the rapid expansion of applications and the customer base for LLMs, the operational cost during inference in terms of energy consumption and memory usage would increase and exceed the training cost and become the dominant factor for the overall environmental impact.

The high resource consumption of LLMs drives the demand of developing techniques to enhance the efficiency of LLMs. 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,

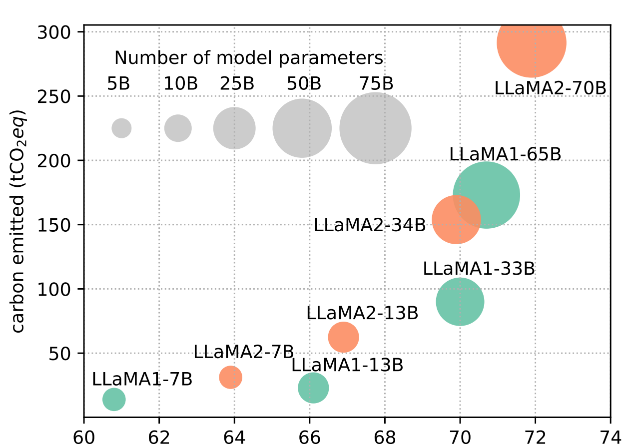

Fig. 1. Illustrations of the peformance and carbon emissions for training LLMs of different scales. The reported performance is the average score of several commonsense reasoning benchmarks. The carbon emissions is estimated based on the GPU and time used for training the model. The size of each bubble corresponds to the number of model parameters. The original data can be found in [272, 273].

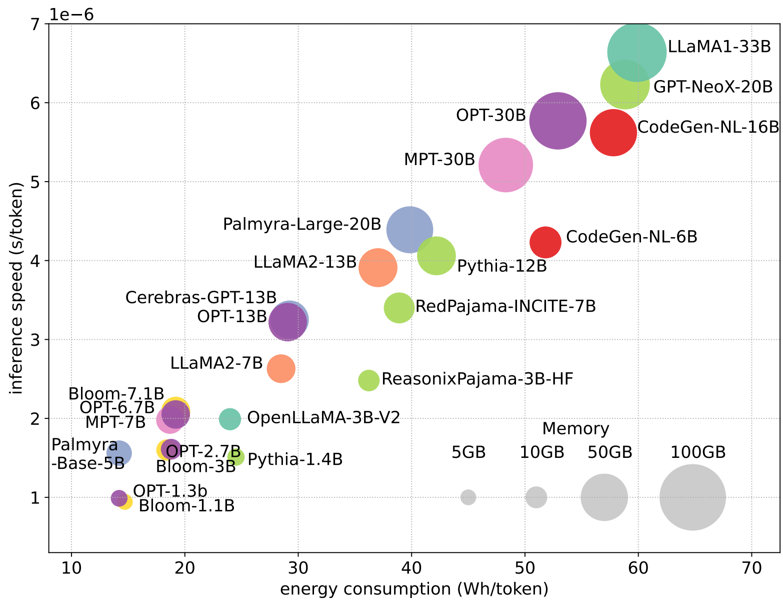

Fig. 2. Energy consumption *vs.* inference speed for various LLMs. The energy consumption and inference speed are measured on Nvidia A100-80GB GPUs with 16-bit floating point quantization. The size of each bubble corresponds to the memory (in Gigabytes) of each model. The original data can be found in [120].

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 [299] 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.

Finally, we have established a GitHub repository where we compile the papers featured in this survey, organizing them within the same taxonomy: https://github.com/AIoT-MLSys-Lab/Efficient-LLMs-Survey. We will actively maintain it and incorporate new research as it emerges. We hope this survey together with the GitHub repository can help researchers and practitioners navigate through the literature and serve as a catalyst for inspiring further research on efficient LLMs.

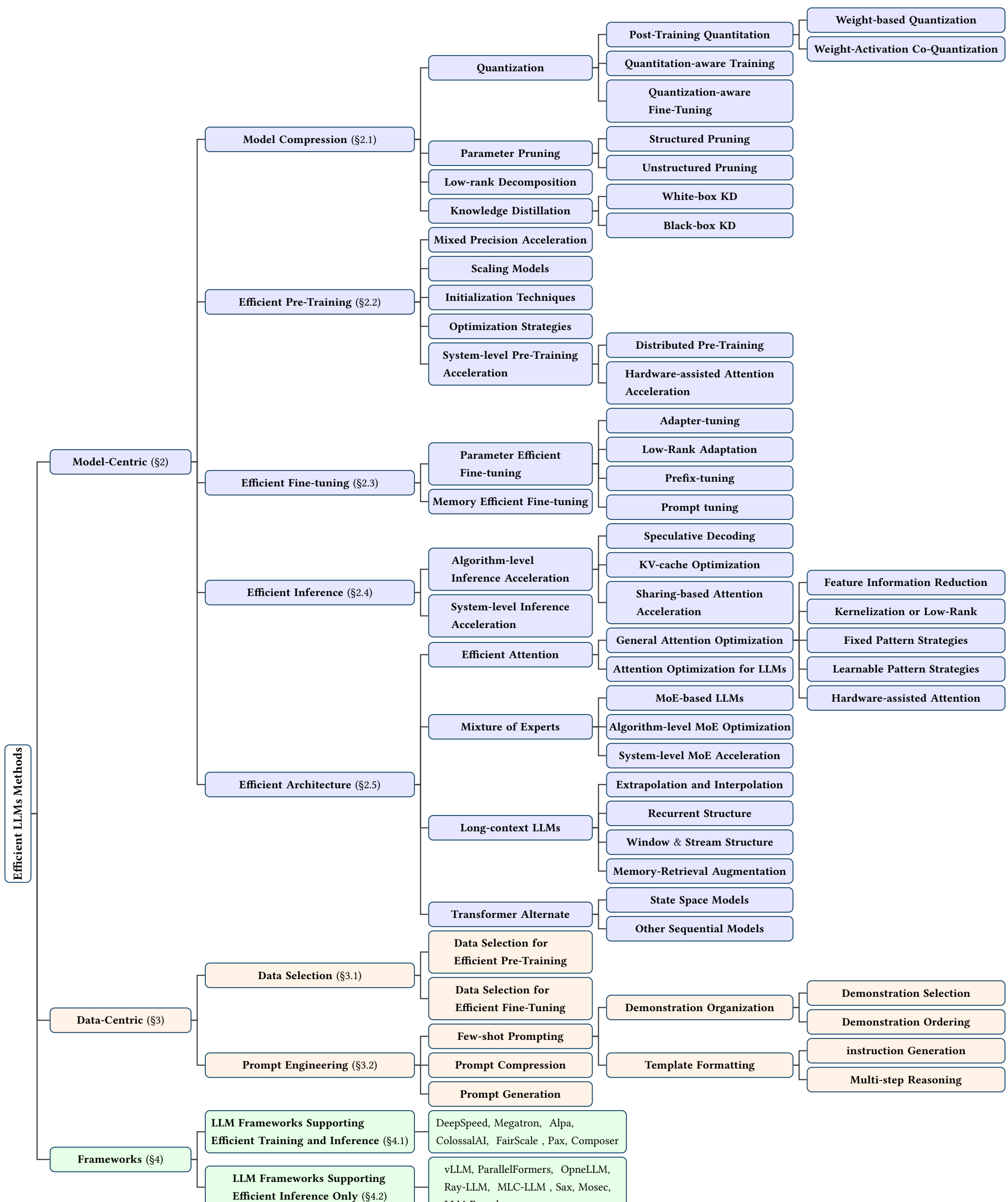

Fig. 3. Taxonomy of efficient large language models (LLMs) literature.

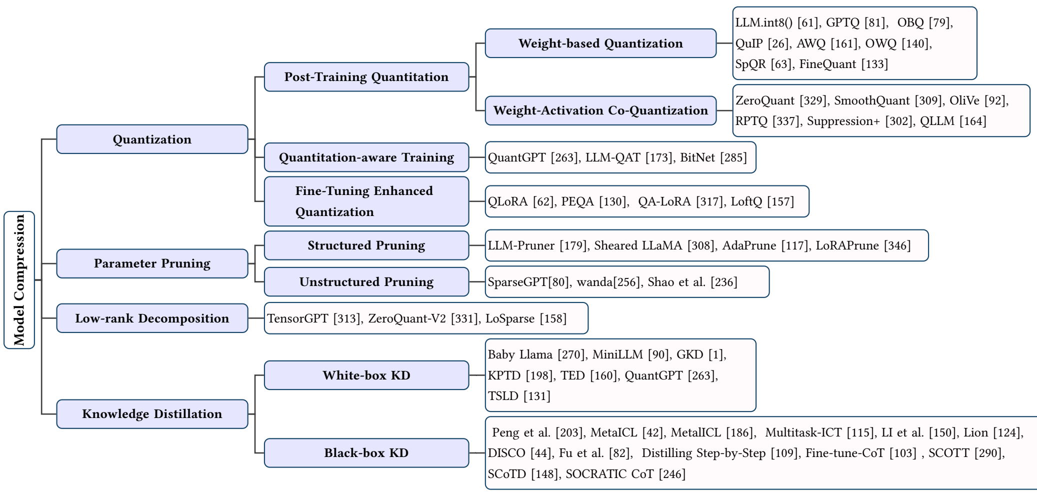

Fig. 4. Summary of model compression techniques for LLMs.

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.

Depending on the stage at which quantization is performed, quantization techniques for LLMs can be classified as post-training quantification (PTQ), quantitatively aware training (QAT) and quantitatively aware fine tuning (QAFT).

Post-Training Quantitation (PTQ). PTQ quantizes LLMs after the model has been trained. PTQ for LLMs can in general be grouped into two categories: weight-based quantization, and weight-activation co-quantization.

- ***Weight-based Quantization*** focuses on quantizing model weights only for LLMs. For examples, Dettmers et al. [61] introduce 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. [81] 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) [79], to update weights with inverse Hessian information. This

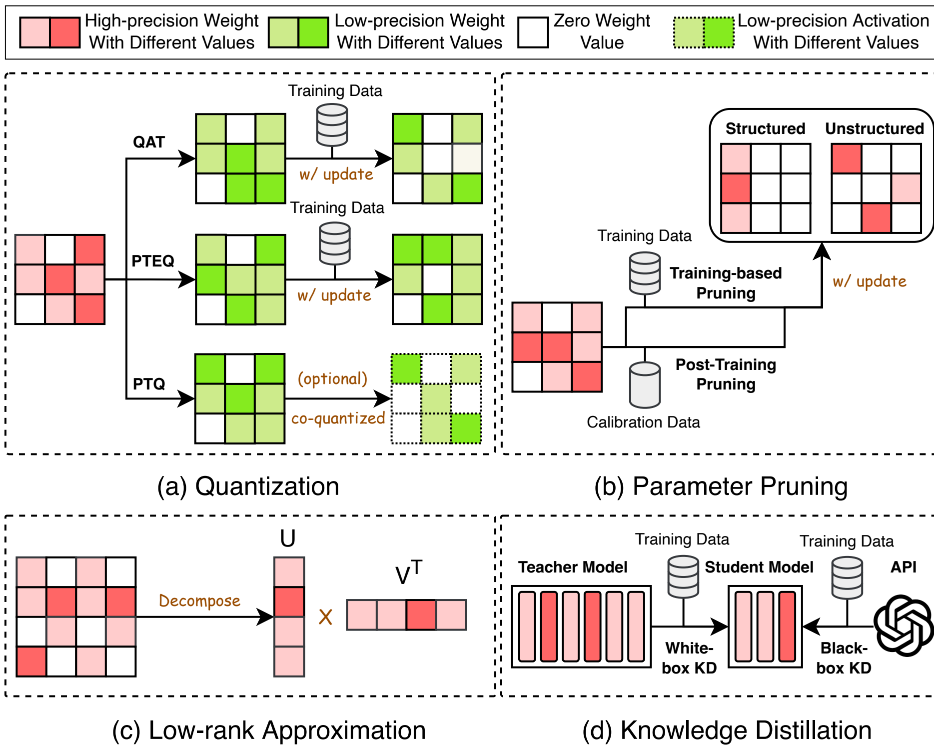

Fig. 5. Illustrations of model compression techniques for LLMs.

technique enables quantizing GPT models with 175 billion parameters in roughly four GPU 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. [26] propose QuIP, a post-training quantization method that applies incoherence processing to quantize LLMs to 2 bits per weight. Lin et al. [161] 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. [140] 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. [133] 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 [20]. 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. Xiao et al. [309] 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. [92] 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. [337] 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. [164] 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. [302] 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.

Quantization-aware Training (QAT). Different from PTQ, QAT quantizes LLMs during the training process itself, and thus is much more expensive and resource consuming. LLM-QAT [173] uses data generated by LLMs itself to distill knowledge, with the aim of quantifying 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. Tao et al. [263] 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. BitNet [285] pioneers quantization-aware training 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.

Quantization-aware Fine-Tuning (QAFT). QAFT quantizes LLMs in the fine-tuning stage with the objective to reduce memory usage of fine-tuning. 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 [110]. In doing so, QLoRA significantly reduces memory usage during finetuning without performance degradation compared to standard full-model finetuning. QA-LoRA [317] 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 [157] 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. Lastly, PEQA [130] 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 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.

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, for instance, rows, columns, or sub-blocks of a weight matrix of LLMs. For example, LLM-Pruner [179] introduces a task-agnostic structured pruning strategy that selectively eliminates non-essential interconnected structures using gradient information. It adopts a small amount of data to obtain the weight, parameter, and group importance of the coupled structure for LLaMA [272], and uses LoRA [110], an efficient low-rank fine-tuning strategy to recover performance after pruning, showing an acceptable result for zero shot. Sheared LLaMA [308] encompasses two tactics. The first 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 tactic 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 tactics, Sheared LLaMA is able to prune the LLaMA2-7B model down to the 1.3B and 2.7B parameters. AdaPrune [117] applies a novel transposable fine-grained sparsity mask to $N : M$ fine-grained block sparsity, in which for each block of M weights, it has at least N zeros. Specifically, AdaPrune has shown good performance via magnitude-based weight selection. LoRAPrune [346] introduces a LoRA-based pruning criterion using LoRA's weights and gradients instead of pre-trained weights' gradients for importance estimation. It employs a structured iterative pruning process to eliminate excess channels and heads, surpassing LLM-Prune in efficiency at a 50

Unstructured Pruning. Unstructured pruning focuses on pruning model weights individually without considering the model's internal structure. Frantar and Alistarh [80] 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. [256] 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. [236] 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. [313] 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. Yao et al. [331] propose ZeroQuant-V2 which combines low-rank approximation and post-training quantization.

Specifically, it adopts a low-rank compensation techniques, which uses low-rank matrices to improve model quality recovery with a minimal increase in model size. LoSparse [158] 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. Depending on whether the parameters of LLMs are needed during the distillation process, 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 of the teacher model are transparent [86]. For example, Baby Llama [270] trains an ensemble of GPT-2 and a collection of smaller LLaMA 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. [90] observe that conventional KD objectives, such as Kullback-Leibler forward 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 [260]. This approach surpasses the performance of standard KD benchmarks on the 13-billion-parameter LLaMA model [272]. 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 [198] 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 [160] 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.

Black-box Knowledge Distillation. Different from white-box KD, black-box KD refers to KD techniques where the parameters of LLMs are not available during distillation and can only be accessed through the API interface. Inspired by MetaICL and MetalICL [42, 186], 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 [115] 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. [150] introduce a new hybrid prompting technique that employs multi-task learning along with explanations generated by GPT-3 *text-davinci-002* [1] version. This method is used to distill explanations into smaller models,

[1]https://platform.openai.com/docs/models/gpt-base

achieving consistent and significant improvements over strong single-task fine-tuning benchmarks in different scenarios. Lion [124] introduces an innovative 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] aims 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* [34]). The small model is then selected based 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 [109] 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 [103] utilizes existing zero-shot CoT prompting techniques [137] 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 [246] 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 [290] 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 [148] presents a method called symbolic CoT distillation. It involves drawing CoT rationales from a LLM using unlabeled data instances. Then A smaller model is trained to predict both the sampled rationales and the associated labels.

2.2 Efficient Pre-Training

As shown in Table 1, the cost of pre-training LLMs is extremely expensive. Efficient pre-training aims to enhance the efficiency and reduce the cost of the LLM pre-training process. As summarized in Figure 7, 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 calculating gradients, weights, and activations with low-precision weights and converting them back to low-precision ones before applying them to update the original weights. Specifically, Micikevicius et al. [185] 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 [2] optimizer has eliminated the copy

[2]https://github.com/facebookresearch/fairseq/blob/main/fairseq/optim/fp16_optimizer.py#L468

Table 1. Pre-training cost of some representative LLMs.

Model	Parameter Size	Data Scale	GPUs Cost	Training Time
GPT-3 [21]	175B	300B tokens	-	-
GPT-NeoX-20B [19]	20B	825GB corpus	96 A100-40G	-
OPT [350]	175B	180B tokens	992 A100-80G	-
BLOOM [233]	176B	366B tokens	384 A100-80G	105 days
GLM [339]	130B	400B tokens	786 A100-40G	60 days
LLaMA-1 [272]	65B	1.4T tokens	2048 A100-80G	21 days
LLaMA-2 [273]	70B	2T tokens	A100-80G	71,680 GPU days
Gopher [219]	280B	300B tokens	1024 A100	13.4 days
LaMDA [269]	137B	768B tokens	1024 TPU-v3	57.7 days
GLaM [71]	1200B	280B tokens	1024 TPU-v4	574 hours
PanGu-α [340]	13B	1.1TB corpus	2048 Ascend 910	-
PanGu-$\sum$ [227]	1085B	329B tokens	512 Ascend 910	100 days
PaLM [50]	540B	780B tokens	6144 TPU-v4	-
PaLM-2 [8]	-	3.6T tokens	TPUv4	-
WeLM [253]	10B	300B tokens	128 A100-40G	24 days
Flan-PaLM [51]	540B	-	512 TPU-v4	37 hours
AlexaTM [249]	20B	1.3 tokens	128 A100	120 days
Codegeex [359]	13B	850 tokens	1536 Ascend 910	60 days
MPT-7B [268]	7B	1T tokens	-	-

of FP32 weights, but the optimizer (adamw) still use fp32 internally. However, some studies [219] suggest that FP16 could result in accuracy loss. To counteract this performance drop, Brain Floating Point (BF16) was proposed [23, 128], which achieves better performance by assigning more bits to the exponent and fewer to the significant bits. Lastly, recent studies [169, 201] have shown that combining mixed-precision acceleration with activation compressed training (ACT) can further facilitate memory-efficient transformer pre-training.

Scaling Models. Scaling models accelerate pre-training convergence and reduce training costs by using the parameters of a small model to scale up to a large model, which can inspire efficient pre-training designs for LLMs. For example, Gong et al. [85] introduce Progressive Stacking to transfer insights from a simpler model to a more complex one and then uses progressive stacking to enhance the model's training efficiency and speed of convergence. Yang et al. [318] 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. [89] introduce Compound-Grow, 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. [216] 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. [242] 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, and the learning rate schedule. By starting each phase with the

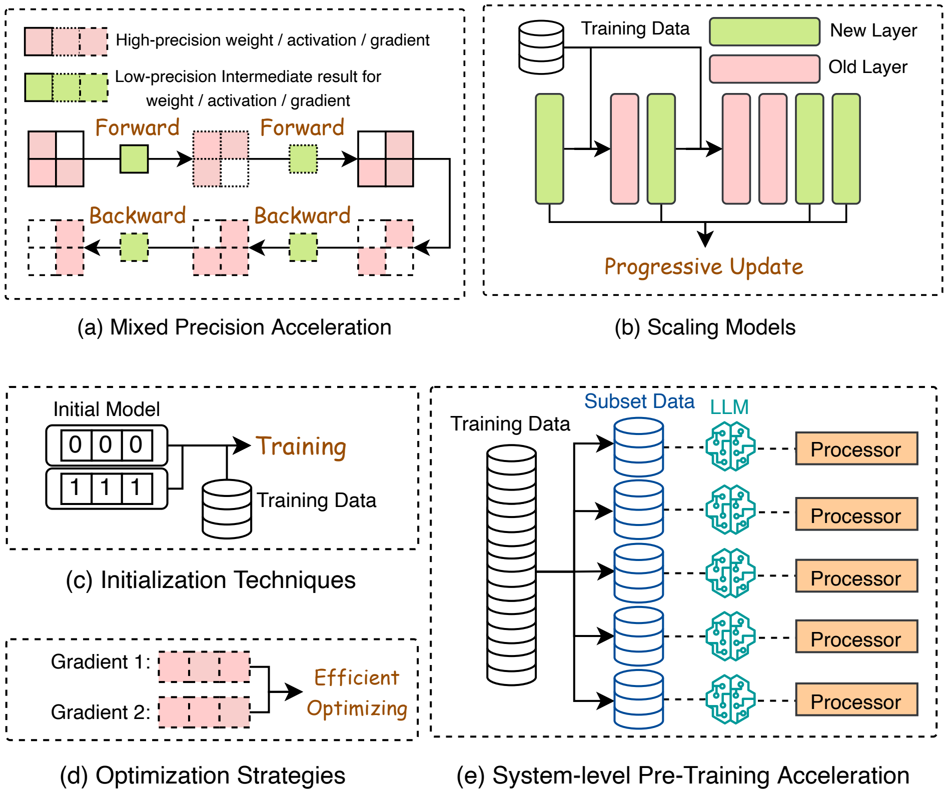

Fig. 6. Illustrations of efficient pre-training techniques for LLM.

results from the previous one, it effectively reuses computation, leading to a more efficient training process. Chen et al. [30] propose function-reserving initialization (FPI) and advanced knowledge initialization (AKI) to transfer the knowledge of an a smaller pre-trained model to a large model so as 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. [289] 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 [200] 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 [155] 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 LLM pre-training. Most LLMs employ initialization techniques that were adopted in training smaller-scale models, such as conventional initialization techniques like [102, 138]. For example, initialization methods introduced by Krishna Kumar [138] and He et al. [102] aim to balance input and output variances. Fixup [343] and

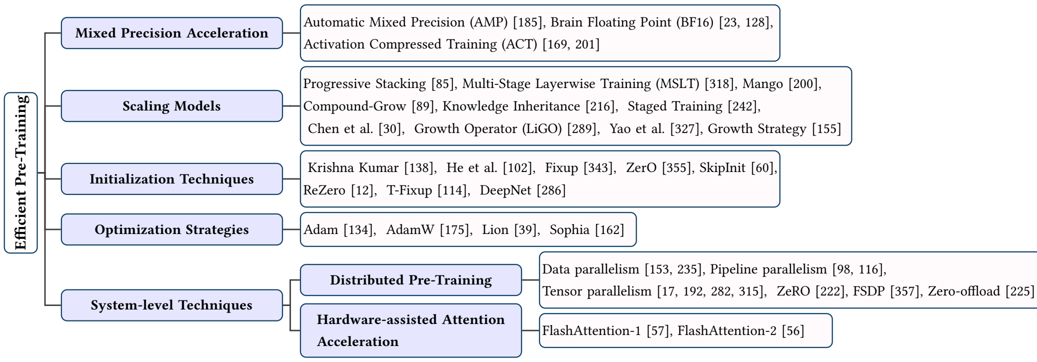

Fig. 7. Summary of efficient pre-training techniques for LLMs.

ZerO [355] set the residual stem to zero, preserving signal identity. SkipInit [60] substitutes batch normalization with a zero-value multiplier. ReZero [12] adds zero-valued parameters to maintain identity, leading to faster convergence. T-Fixup [114] follows Fixup to adopt rescaling schemes for the initialization of residual blocks of Transformer models. DeepNet [286] 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. LLMs including GPT-3 [21], OPT [350], BLOOM [233], and Chinchilla [104] are predominately pre-trained using Adam [134] or AdamW [175] as optimizers. Recently, some studies [39, 162] propose new optimizers to accelerate the pre-training of LLMs. Chen et al. [39] 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. [162] propose Sophia as a lightweight second-order optimizer that outpaces Adam with doubling the pre-training speed. It calculates the moving average of gradients and the estimated Hessian, dividing the former by the latter and applying element-wise clipping. Sophia effectively moderates update sizes, addresses non-convexity and rapid Hessian changes, enhancing both memory utilization and efficiency.

System-level Pre-Training Acceleration.

- **Distributed Pre-Training.** Distributed pre-training techniques refer to methods to leverage distributed processing capabilities of hardware to train LLMs more efficiently. Existing methods that are applied for general AI model training can also be applied for LLM pre-training. Data parallelism [153, 235] involves splitting the training dataset into multiple subsets and processing each subset simultaneously on separate machines (e.g., GPUs or TPUs). Each machine computes gradients independently and then shares these gradients with others to update the model parameters. Pipeline parallelism [98, 116] divides the neural network into stages, with each stage being assigned to a different device. Data flow sequentially through these stages. Each device is responsible for computing the forward and backward passes of its stage. Tensor parallelism splits the neural network model's parameter matrices across multiple devices. Each device is responsible for computing a portion of the parameters of the model's forward and backward passes. Recent work such as Megatron-LM [192] and Colossal-AI [17, 282, 315] has further exploited the combination of tensor parallelism and the above two parallelism techniques to acquire the best acceleration. ZeRO [222] attempts to keep only a portion of the data on each GPU, accessing the remaining data from other GPUs as

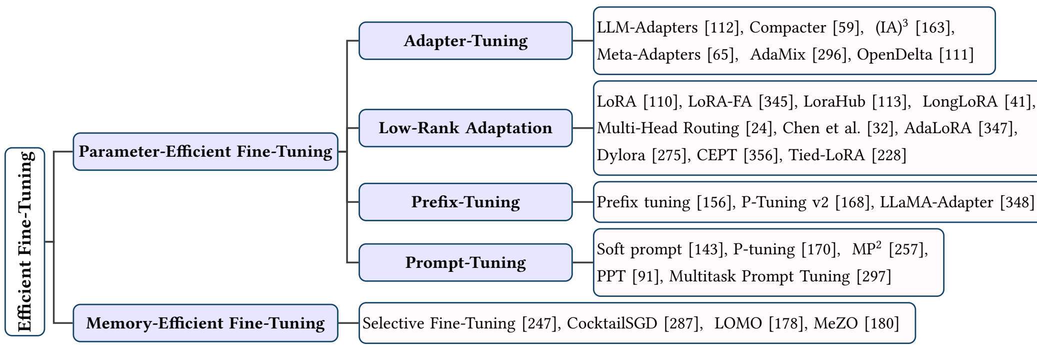

Fig. 8. Summary of efficient fine-tuning methods for LLMs.

needed. FSDP [357] also facilitates the state partition of the optimizer, the gradient partition and the parameter partition for the LLM training. In addition, CPU offload techniques, such as Zero-offload [225], is another perspective to accelerate the training of LLMs, which uses the external memory of a less powerful processing unit such as the CPU to increase the computing memory capacity through communication. The goal of offloading is to fully exploit CPU memory as an extension to further improve the parallizing capability of powerful processing units while keeping the communication overhead as low as possible.

- **Hardware-assisted Attention Acceleration.** FlashAttention [56, 57] is an algorithm developed to accelerate the training of Transformer models, particularly when dealing with long sequences. Traditional attention layer within Transformers poses a computational challenge as the sequence length increases, causing a quadratic increase in runtime and memory requirements. FlashAttention [3] addresses this by reorganizing the attention computation and employing techniques like tiling and re-computation to significantly speed up the process and transition memory usage from being quadratic to linear with respect to sequence length. This is particularly beneficial for training models on long sequences, which is a common scenario when dealing with modern parallelism techniques like data parallel, pipeline parallel, or tensor parallel, especially as they distribute data and model across multiple GPUs. Specifically, FlashAttention-1 [57] designs an IO-aware exact attention algorithm that aims to reduce the number of memory reads/writes between GPU high-bandwidth memory (HBM) and GPU on-chip SRAM. Furthermore, it is also extended to block-sparse attention, yielding an approximate attention algorithm that is faster than any existing approximate attention method. On the basis of observation that FlashAttention-1 isn't as quick as optimized matrix-multiply (GEMM) operations yet, FlashAttention-2 [56] proposes a improved work partitioning by tweaking the algorithm to reduce the number of non-matmul FLOPs, parallelizing the attention computation, and distributing the work between warps to reduce communication through shared memory.

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).

[3] https://github.com/Dao-AILab/flash-attention

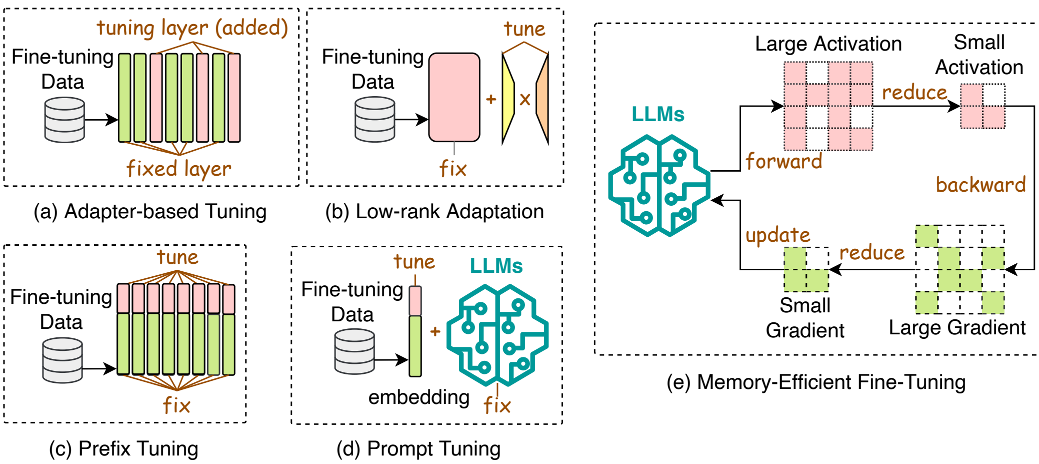

Fig. 9. Illustrations of Parameter-Efficient Fine-Tuning (a)-(d) and Memory-Efficient Fine-Tuning (e).

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 [107]. Adapter-based tuning includes both series adapters [21, 107, 272, 279] and parallel adapters [101, 206]. 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. [112] propose LLM-Adapters, which integrates series or parallel adapters into LLMs for fine-tuning on different tasks. Davison [59] 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. $(IA)^3$ [163] 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 [65] 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 [296] takes inspiration from sparsely-activated mixture-of-experts (MoE) models [366] and proposes a mixture of adaptation modules to learn multiple views of the given task. Lastly, OpenDelta [111] 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) [110] is one of the most popular PEFT methods for LLMs. Instead of updating the parameter matrix as $\mathbf{W} \leftarrow \mathbf{W} + \Delta\mathbf{W}$ where $\Delta\mathbf{W}$ is the gradient matrix, LoRA incorporates two smaller trainable low-rank matrices $\mathbf{A} \in \mathbb{R}^{m \times r}$ and

$\mathbf{B} \in \mathbb{R}^{r \times n}$ where $\Delta\mathbf{W} = \mathbf{A} \cdot \mathbf{B}$ into all the layers of the original LLM, allowing the model to adapt to new information while maintaining the original LLM unchanged to preserve prior knowledge. 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 [345] 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 [113] 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 [41] 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) [24] extends LoRA to Mixture-of-Experts (MoE) architectures. It outperforms Polytropon [209] 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. [347] 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. [275] 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 [356] 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 [228] uses weight tying and selective training to further increase parameter efficiency of LoRA.

Prefix Tuning. Prefix tuning [156] 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. Liu et al. [168] 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. LLaMA-Adapter [348] 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 while retaining its pre-trained knowledge.

Prompt 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 [143] 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 [170] 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

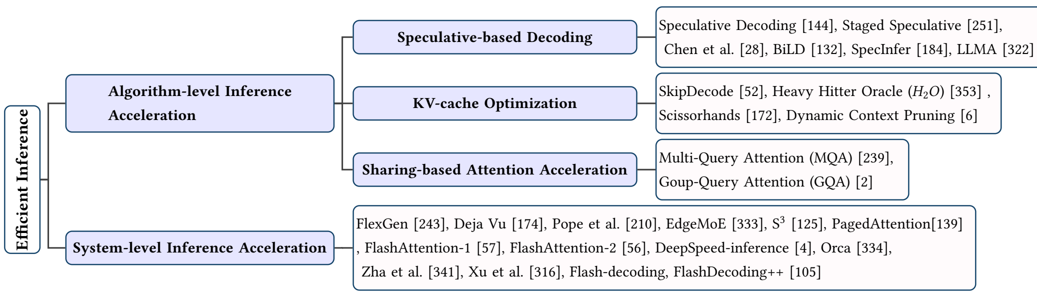

Fig. 10. Summary of efficient inference techniques for LLMs.

through gradient descent and improves performance on a wide range of NLU task. Sun et al. [257] 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 [91] 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 [297] 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 while maintaining training stability, convergence, and high accuracy in fine-tuning has emerged as a critical research topic. Simoulin et al. [247] 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. [178] 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 [180] improves the zeroth-order method [250] 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 [110] and prefix tuning [156], 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 acceleration techniques at the algorithm level and system level.

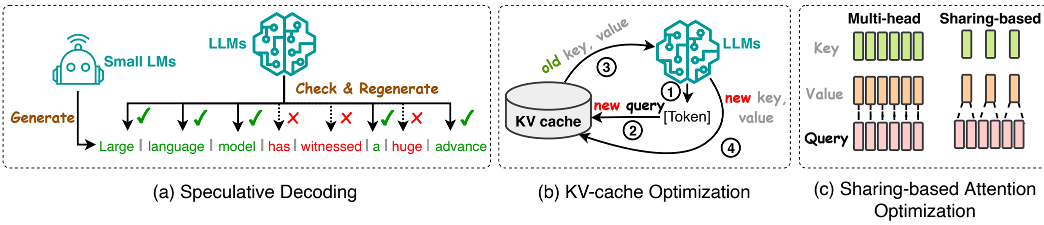

Fig. 11. Illustrations of Algorithm-level acceleration techniques for LLM inference.

Algorithm-level Inference Acceleration. Techniques that enhance LLM inference efficiency at the algorithm level can be in general grouped into three categories: speculative decoding, KV-cache optimization, and sharing-based attention acceleration.

- ***Speculative Decoding.*** Speculative decoding (i.e., speculative sampling) [144] 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. [28] 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 [251] 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 [132] 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 [184] 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 [322] 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.
- ***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. [52] 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. [353] 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. [172] 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.

- ***Sharing-based Attention Acceleration.*** Sharing-based attention acceleration aims to accelerate attention computation during inference through different KV heads sharing schemes. For instance, LLaMA2 [273] optimizes the autoregressive decoding process by using multi-query attention (MQA) [239] and groupedquery attention (GQA) [2]. 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.

System-level Inference Acceleration. Speeding up LLM inference can also be achieved at the system level. For example, FlexGen [243] introduces a high-throughput inference engine to run LLMs on memory-constrained GPUs. It employs a linear programming-based search strategy to manage different hardware components adeptly, integrating computational resources from the GPU, CPU, and disk. Additionally, FlexGen reduces the weights and attention cache to 4 bits, enhancing the inference speed of OPT-175B [350] on a 16GB GPU. Deja Vu [174] defines the concept of contextual sparsity, which are small groups of input-agnostic MLP and attention modules that yield the same output of the dense model. Deja Vu trains predictors to predict contextual sparsity and utilizes kernel fusion and memory coalescing to achieve speedup of contextual sparsity. Pope et al. [210] have created a framework to choose the most efficient ways to divide data specifically for TPU v4, depending on what the application needs. By merging this with a set of detailed improvements, they achieve better speed and efficiency in using the model's resources on PaLM [50] compared to the FasterTransformer[4] benchmarks. EdgeMoE [333] introduces the first on-device processing system specifically designed for LLMs based on mixture-of-expert (MoE) structure. It manages memory and computation for inference by smartly dividing the model across different storage levels. In particular, non-expert model weights are saved directly on the device, but expert weights are stored externally and only brought into the device's memory when they are needed. S^3 [125] develops a system that knows the a priori of the output sequence. It predicts the length of this sequence and plans generation requests based on this prediction, optimizing device resource use and improving throughput, managing any incorrect predictions. `Orca` [335] uses iteration-level scheduling to determine batch sizes. Once a sequence in a batch finishes, it's replaced by a new one, leading to better GPU utilization than static batching. FasterTransformer [271] employs a variety of strategies to accelerate inference, such as layer fusion which amalgamates multiple layers of neural networks into a single layer to be computed with a single kernel, along with activations caching that designates a buffer for storing preceding keys and values at every step. It also incorporates memory optimization to repurpose the memory buffer of activations/outputs across different decoder layers, as well as MatMul kernel auto-tuning and conducting inference with lower precisions. PagedAttention [139] develops an attention method influenced by traditional virtual memory and paging methods used in operating systems. It creates a vLLM[5], a system for LLMs, which allows efficient sharing of KV-cache both within and between requests. This helps minimize memory consumption and speed up high-throughput inference. In addition to pre-training acceleration, FlashAttention-1 [57] accomplishes efficient inference by fusing the matrix multiplications and

[4]https://github.com/NVIDIA/FasterTransformer
[5]https://github.com/vllm-project/vllm

softmax operations of the attention computation into one kernel. Additionally, techniques like tiling are employed to minimize the total amount of memory required to compute attention at one time. Next, in addition to the extended support for larger head dimensions, FlashAttention-2 [56] introduces support for MQA [239] and GQA [2], which are likely to further optimize the attention computation process, accommodating various models like GPT-J [279], CodeGen [195], and CodeGen2 [194]. This extension allows these models to exploit FlashAttention-2 for enhanced speed and memory efficiency during inference. DeepSpeed-inference [4] presents a multi-GPU inference approach designed to reduce latency and boost the throughput of both dense and sparse transformer models when contained within the collective GPU memory. Additionally, it offers a mixed inference method that employs CPU and NVMe memory, along with GPU memory and computation, ensuring high-throughput inference even with models too large to be accommodated by the combined GPU memory. Flash-decoding [6], an advanced method based on FlashAttention, enhances long-context inference speed by dividing keys/values into smaller chunks, computing attention on these chunks in parallel, and then aggregating them to generate the final output to better optimize inference speed. FlashDecoding++ [105] supports mainstream LLMs and hardware back-end via asynchronized softmax, double buffering for flat GEMM optimization, and heuristic dataflow, achieving up to 4.86× and 2.18× acceleration on GPUs compared to Hugging Face implementations.

2.5 Efficient Architecture Design

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

*2.5.1 **Efficient Attention**.*

General Attention Optimization. Even though LLMs have impressive performance, the quadratic time and space complexity of transformer architecture with respect to sequence length pose significant limitations during training. Previous works [265] focus primarily on improving the memory complexity of the self-attention mechanism and improving the general efficiency of the Transformer architecture. These approaches include:

- **Feature Information Reduction.** The principle of feature information reduction, as evidenced by models such as Funnel-Transformer [54], Nyströmformer [312], and Set Transformer [141], endeavors to curtail computational demands by reducing feature information within a sequence, which subsequently leads to a proportionate reduction in computational resources required.
- **Kernelization or Low-Rank.** Kernelization or low-rank methods encompassing models such as Sumformer [3], FluRKA [94], Scatterbrain [27], Linformer [291], Low-Rank Transformer [303], Performer [49], Random Feature Attention [205], and Linear Transformer [129], enhances computational efficacy by utilizing low-rank representations of the self-attention matrix or by adopting attention kernelization techniques.
- **Fixed Pattern Strategies.** Fixed pattern strategies include models such as Paliotta et al. [199], Big Bird [338], Poolingformer [344], Longformer [13], Blockwise Transformer [217], and Sparse Transformer [48], which 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.

[6]https://pytorch.org/blog/flash-decoding/

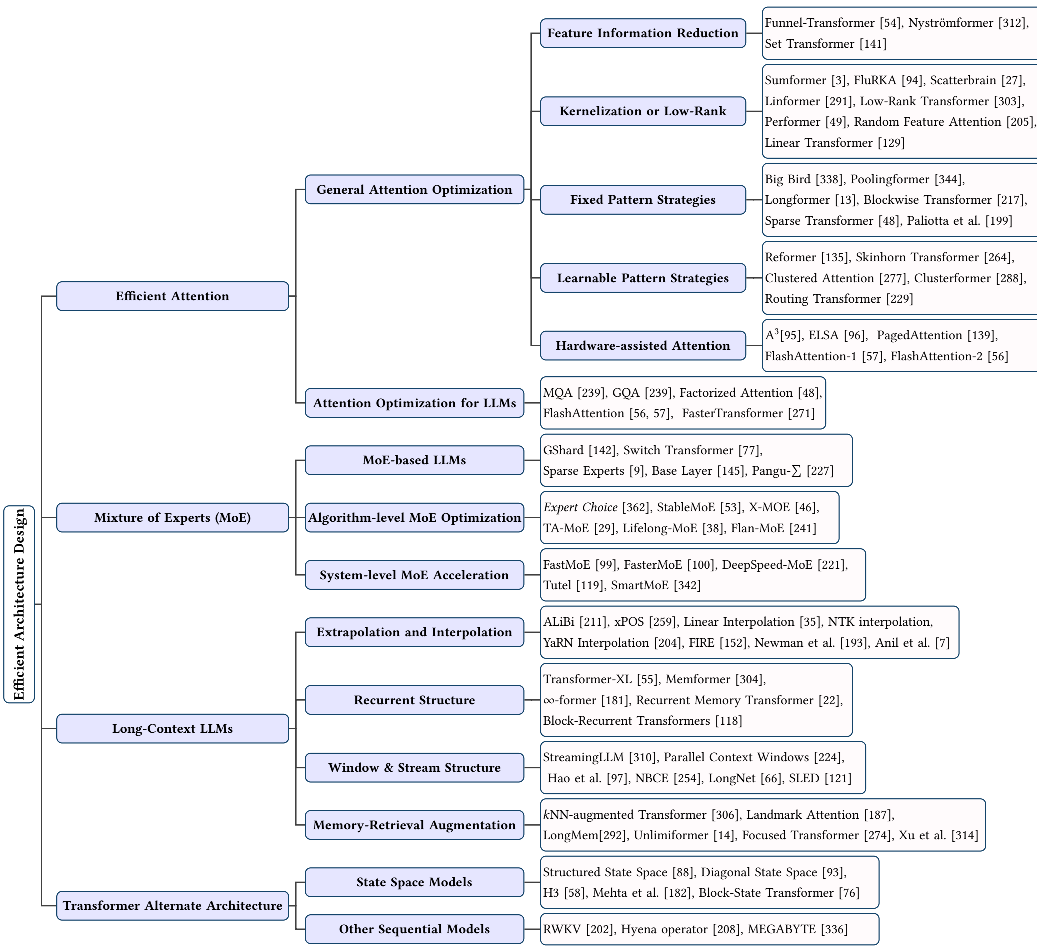

Fig. 12. Summary of efficient architecture designs for LLMs.

- **Learnable Pattern Strategies.** Represented by models such as Reformer [135], Skinhorn Transformer [264], Clustered Attention [277], Clusterformer [288], and Routing Transformer [229], strives to categorize or organize the input sequences by learning techniques. This approach facilitates a more comprehensive perspective of the sequence, all while preserving the computational advantages offered by fixed pattern methodologies.
- **Hardware-assisted Attention.** Beyond the algorithmic approaches that sparsify attentions and thereby streamline the computation of the attention matrix, several pioneering studies [56, 57, 95, 96, 139] concentrate on realizing efficient and lightweight attention mechanisms in hardware. For example, A^3[95] 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 [96] brings forth a solution co-designed for hardware and software, which markedly cuts down computational needs by adeptly excluding relations with minimal impact on the end result. More recently, PagedAttention [139] unveils an efficient attention methodology through the

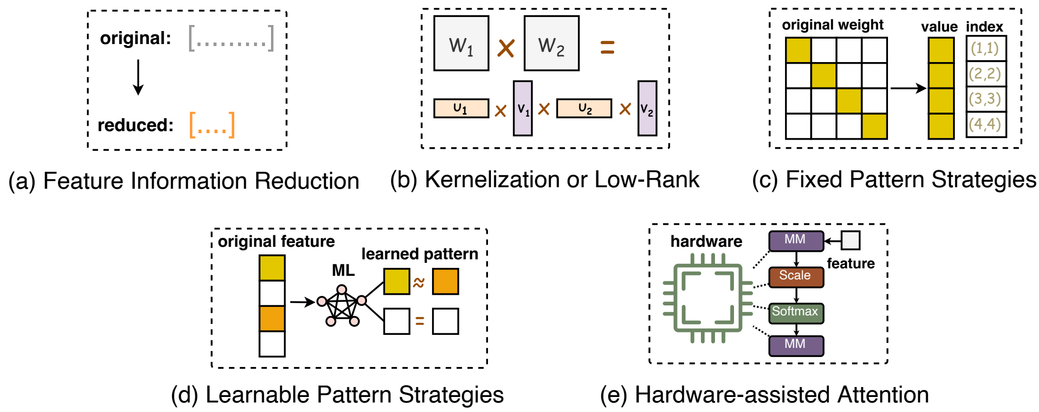

Fig. 13. Illustrations of attention optimizations.

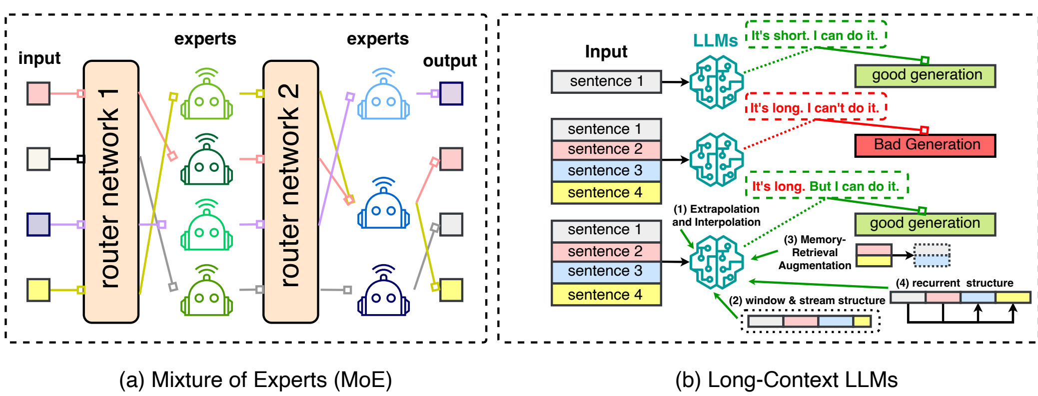

Fig. 14. Illustrations of Mixture of Experts (MoE) and Long-Context LLMs.

communal use of a KV-cache. Both FlashAttention-1 [57] and FlashAttention-2 [56] advocate for IO-aware precise attention methodologies.

Attention Optimization for LLMs. Some existing LLMs employ sparse or hardware-assisted attention methods to enhance both training and inference efficiency. For example, GPT-3 [21] utilizes Factorized Attention [48] over traditional full attention, reducing computational demands. Additionally, LLMs such as PaLM [50], StarCoder [149], and LLaMA2 [273] adopt Sharing-based Attention Optimization techniques like Multi-query attention (MQA) [239] and Group-Query Attention (GQA) [2], as discussed in Sec 2.4, rather than conventional multi-head attention (MHA). By sharing identical linear transformation matrices for keys and values, these weight-sharing methods notably reduce computational overheads with only a slight decline in model performance. Differently, rather than compromising model quality for computational efficiency, certain LLMs such as Falcon-180B [7] and GPT-NeoX-20B [19] harness FlashAttention [56, 57]. This approach refines the speed and memory usage of GPU-based attention modules with an emphasis on IO efficiency.

2.5.2 *Mixture of Experts (MoE).*

[7]https://huggingface.co/tiiuae/falcon-180B

Mixture of Experts (MoE) [9, 43, 70, 214, 240] represents a sparse methodology utilized within machine learning, prominently in the training of 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 collaborated to deliver a consolidated output. For pre-traning or fine-tuning, MoE framework 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. In the context of inference, MoE models decrease the inference time by not engaging all experts simultaneously, but rather activating only a select few. Additionally, they are 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 [127].

MoE-based LLMs. Several studies [9, 77, 142, 227] have employed MoE, a proficient sparse framework, for training LLMs with a vast number of parameters. Earlier on, Google unveiled GShard [142], which augments the MoE idea to adeptly manage billions of parameters. GShard offers a refined method to articulate a variety of parallel computation frameworks with minor modifications to the existing model code. It can also amplify a multilingual neural machine translation Transformer model with Sparsely-Gated Mixture-of-Experts beyond 600 billion parameters through automatic sharding. Subsequently, Switch Transformer [77] 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. Sparse Experts [9] 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, which reduces energy use. Base Layer [145] 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 [227] is an 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).

Algorithm-level MoE Optimization. Recently, various studies [38, 53, 362] have suggested approaches to improve efficient training of MoE-based LLMs. The strategy termed *Expert Choice* [362] 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 refined strategy has the potential to more than double the speed of pre-training convergence. On the other hand, StableMoE [53] identifies the issue of altering target experts for identical input during training and tackles 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 then suggests estimating the routing scores between tokens and experts on a low-dimensional hyper-sphere. TA-MoE [29] highlights that current MoE dispatch patterns are not fully leveraging the underlying heterogeneous network environment, and introduces a topology-aware routing strategy for large-scale MoE training from a model-system co-design standpoint, which can dynamically modify the MoE dispatch pattern based on the network topology. Lifelong-MoE [38] 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 LLM on sequential data distributions without losing previous knowledge. Lastly, Flan-MoE [241] promotes the amalgamation of MoE methodology and instruction tuning, observing that MoE models gain more from instruction tuning compared to dense models. Specifically, FLAN-MOE effectively enlarges language models without demanding an increase in computational resources or memory requirements.

System-level MoE Acceleration. Moreover, several MoE-centric acceleration frameworks have been developed to facilitate the efficient training of MoE-based LLMs with multi-billion to trillion parameters, including FastMoE [99], FasterMoE [100], DeepSpeed-MoE [221], Tutel [119], and SmartMoE [342]. Specifically, FastMoE [99] 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. Then FasterMoE [100] 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. DeepSpeed-MoE [221] has designed a Pyramid-Residual MoE (PR-MoE) to enhance the 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. Tutel [119] is a scalable stack for MoE with adaptive parallelism and pipelining features. 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. Furthermore, SmartMoE [342] conducts distributed automated training for sparsely activated models. By investigating a wider scope of hybrid parallelism suited for data-sensitive models, it breaks down the space into offline static pools with online selection. SmartMoE also implements a data-sensitive method for pre-training performance prediction and employs an efficient algorithm for dynamic run-time strategy optimization.

2.5.3 ***Long-Context LLMs.***

The constraint of limited context lengths hinders the ability of LLMs to effectively manage long inputs, as the quadratic computational expenses in terms of time and memory are associated with context lengths. Meanwhile, long context processing is critical for LLMs like GPT-3 (up to 16K tokens) [21], GPT-4 (up to 32K tokens) [197], LLaMA-1 (up to 2048 tokens) [272], and LLaMA-2 (from 4096 to 32K [8]) [273] and others, for several reasons: (1) Understanding Context: LLMs need to understand the broader context of a discussion or text to provide coherent, relevant, and accurate responses, which entails recognizing prior information, discerning relationships between different pieces of information, and maintaining a consistent narrative or argument. (2) Handling Multi-turn Conversations: like ChatGPT [332], understanding the context of previous exchanges is crucial for generating sensible and relevant responses. Without long context processing, LLMs might give inaccurate or nonsensical answers as they won't be able to reference prior conversation turns effectively. (3) Coherence and Consistency of Responses: Long context processing helps in maintaining thematic consistency [261], avoiding contradictions, and ensuring that the generated

[8]https://together.ai/blog/llama-2-7b-32k

text flows logically from one sentence to the next. (4) Enhanced Problem-Solving and Reasoning: Long context processing is fundamental for complex problem-solving and reasoning tasks [197, 301] which require considering multiple pieces of information, often spread out over a large text or several turns of a conversation. Moreover, [147, 166] highlight that decoder-only LLMs such as GPT-3.5 handle information at the beginning or end of the input context effectively, but struggle with accessing information in the middle, leading to a U-shaped performance curve. Hence, it's vital to develop efficient long-context processing architectures for LLMs, and we next present several of the current state-of-the-art approaches.

Extrapolation and Interpolation. Standard positional encoding methods like absolute positional embeddings (APE) [276], learned positional embeddings (LPE) [284], relative positional embedding (RPE) [238], relative positional bias [220], and rotary position embeddings (RoPE) [255] have advanced the integration of positional information in Transformer-based models. Among a range of advanced LLMs, GPT-3 [21] and OPT [350] adopt LPE, Gopher [219] and Chinchilla [104] employ RPE, while LLaMA and GLM-130B utilize RoPE. However, it's still challengable to train LLMs on sequences with a limited maximum length while ensuring they generalize well to significantly longer sequences during inference. For long-context extension based on optimized sequential encoding strategies, some recent studies has successfully proposed enhanced positional extrapolation [211, 259] and interpolation strategies [35, 152, 204] for LLMs. Specifically, ALiBi [211], suggests employing attention with linear biases to attain extrapolation during inference for sequences exceeding the maximum length seen during training. The method is achieved by 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. Thereby, it can facilitate efficient length extrapolation. Different from ALiBi [211], xPOS [259] characterizes attention resolution as a marker for extrapolation and utilizes a relative position embedding to deliberately enhance attention resolution, thereby improving length extrapolation. However, these techniques have not been implemented in some of the recent LLMs such as GPT4 [197], or LLaMA-1,2 [272, 273]. For Positional Interpolation strategy, Chen et al. [35] 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, achievable with just a few fine-tuning steps. Besides, NTK interpolation[9] modifies RoPE's base, effectively changing the rotational velocity of each RoPE dimension. YaRN Interpolation [204] 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 [152] proposes an innovative 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, enabling length generalization.

Apart from positional extrapolation and Interpolation, Newman et al. [193] find that avoiding EOS prediction leads to better length extrapolation performance in certain tasks. Anil et al. [7] views the length extrapolation as an out-of-distribution generalization issue concerning long-context reasoning. Following this notion, they introduce a technique that merges the in-context learning capabilities of LLMs with several internal reasoning steps prior to reaching the final outcomes, leading to a significant enhancement in length generalization.

Recurrent Structure. Some researches [22, 55, 118, 181, 304] focus on enhancing language models with memory features through recurrence, which augments the language models' ability to manage

[9]https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have/

extremely long sequences. Previously, Transformer-XL [55] presents a new segment-level recurrence mechanism and utilizes enhanced relative positional encoding to capture longer-term dependencies, while also addressing the long-context fragmentation issue. Next, Memformer [304] leverages an external dynamic memory for encoding and retrieving past information, achieving linear time and constant memory space complexity for long sequences. And it also proposes Memory Replay Back-Propagation (MRBP) to facilitate long-range back-propagation through time with significantly lower memory requirements. ∞-former [181] presents a transformer model augmented with an 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. Recently, the Recurrent Memory Transformer (RMT) [22] uses a recurrence mechanism to retain past segment-level information by incorporating special memory tokens into the input or output sequence, demonstrating superior performance compared to Transformer-XL in long-context modeling. Furthermore, Block-Recurrent Transformers [118] utilize self-attention and cross-attention to proficiently execute a recurrent function across a broad set of state vectors and tokens, and effectively model long sequences through parallel computation. Retentive Network [258] introduces a multiscale retention mechanism as an alternative to multihead attention, encompassing three computational paradigms: parallel, recurrent, and chunkwise recurrent representations. It results in effective scaling, allows for parallel training, and offers cost-effective deployment and efficient inference, outperforming the Transformer model in these aspects.

Window & Stream Structure. To mitigate the constraint of a fixed attention window during the pre-training phase, several studies [66, 97, 121, 191, 224, 310] tackle the issue of long-context processing or inference in Large Language Models (LLMs) by developing new windowing mechanisms and streaming architectures. Importantly, StreamingLLM [310] identifies an *attention sink* phenomenon, noting that retaining the Key-Value (KV) 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. Besides, Parallel Context Windows (PCW) [224] 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. Hao et al. [97] introduce Structured Prompting, where grouped demonstration examples are individually encoded with carefully crafted position embeddings. These examples are then collectively attended to by the test example through a re-scaled attention mechanism. NBCE [254] proposes a naive Bayes-based method to extend context length. LongNet [66] proposes dilated attention, which exponentially expands the attentive field as the distance increases, enabling the handling of sequence lengths of over 1 billion tokens. LongNet can be implemented by parallelizing the training through 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 [14, 187, 274, 292, 306, 314] the inference of extremely long text by employing memory-based retrieval strategies. A notable example is the kNN-augmented Transformer [306], which extends the attention context size by utilizing k-nearest-neighbor (kNN) lookup to fetch previously similar context embeddings. Recently, Landmark Attention [187] employs a landmark token to represent each block of input and trains the attention mechanism to utilize it for choosing relevant blocks. This allows for the direct retrieval of blocks through the attention mechanism while maintaining the random access flexibility of the previous context, demonstrating impressive performance on LLaMA for long-context modeling. LongMem[292] 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. Likewise, Focused Transformer [274] 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. Furthermore, Xu et al. [314] 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[35] 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 [58, 76, 88, 182, 202, 208, 336] propose new architectures to supplant transformer architecture.

State Space Models. A particular approach aims to substitute the attention mechanism with state space models (SSMs) [58, 76, 88, 93, 182] formulated as $x'(t) = Ax(t) + Bu(t)$, $y(t) = Cx(t) + Du(t)$, and A is state matrix. SSMs provides near-linear computational complexity relative to the length of the sequence. Specifically, Structured State Space (S4) [88] is a novel sequence model that refines State Space Models (SSMs) by conditioning the A matrix 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) [93] improved SSMs by proposing fully diagonal parameterization of state spaces instead of a diagonal plus low rank structure, demonstrating greater efficiency. Furthermore, to bridge the gap between SSMs and attention while adapting to modern hardware, H3 [58] stacks two SSMs to interact with their outputs and input projection, allowing H3 to log tokens and facilitate sequence-wide comparisons simultaneously. Subsequently, Mehta et al. [182] 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 [93], while maintaining the perplexity on multiple language modeling benchmarks. Then the Block-State Transformer (BST) [76] designs a hybrid layer that combines an SSM sublayer for extended range contextualization with a Block Transformer sublayer for short-term sequence representation, enhancing performance.

Other Sequential Models. Besides, other new architectures [202, 208, 336] are proposes to replace the transformer layer. Receptance Weighted Key Value (RWKV) model [202] amalgamates the advantages of Recurrent Neural Networks (RNNs) 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. Additionally, Hyena operator [208] is a sub-quadratic alternative to the attention mechanism, mitigating the quadratic cost in long sequences. This operator includes two efficient subquadratic primitives: an implicit long convolution and multiplicative element-wise gating of the input. Through this, the Hyena Hierarchy facilitates the development of larger, more efficient convolutional language models for long sequences. MEGABYTE [336] 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.

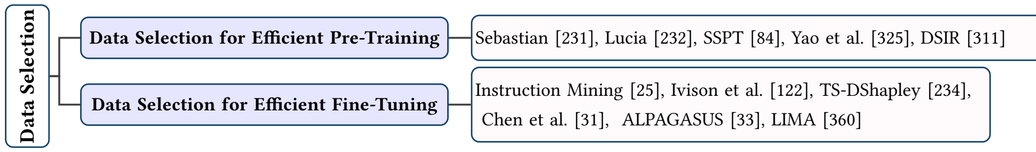

Fig. 15. Summary of data selection techniques for LLMs.

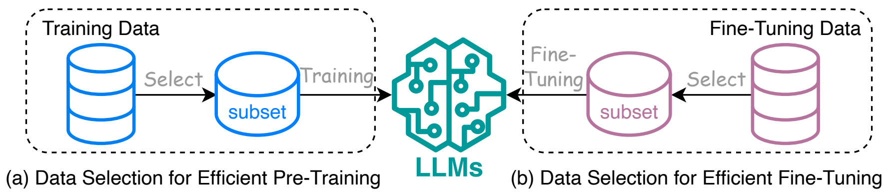

Fig. 16. Illustrations of Data Selection Methods for LLMs.

3 DATA-CENTRIC METHODS

3.1 Data Selection

Data selection for LLMs involves careful consideration of the source, quality, and pre-processing of data. Ensuring high-quality data is fundamental to the development of efficient and reliable LLMs, as it affects their ability to learn, generalize, and perform accurately on various tasks. [84, 232, 311, 325]. This process is critical in avoiding the propagation of biases and inaccuracies in the models, enabling LLMs training to converge. Researchers are developing strategies like optimized data selection, data compression, and instruction tuning to improve performance while using fewer resources. Figure 15 summarizes the latest data selection techniques for efficient pre-training and fine-tuning.

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

Ruder et al. [231] introduce a method to explore the nuanced complexities of data selection in multi-domain sentiment analysis. It proposes an original data selection strategy that combines both domain-specific and domain-agnostic data, thereby enhancing the predictive accuracy of sentiment classification algorithms. Santamaría and Axelrod [232] propose to employ a sophisticated cluster-based method that utilizes Brown clusters to refine the corpus vocabulary. This approach offers an optimized mechanism for task-specific corpus modeling, outperforming traditional Moore-Lewis methods in achieving superior perplexity and out-of-vocabulary (OOV) rates on domain-specific data sets. SSPT [84] 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. [325] propose a meta-learning-based method for the meticulous selection of linguistically informative sentences which significantly elevates the quality of machine-generated translations. Xie et al. [311] 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.***

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.

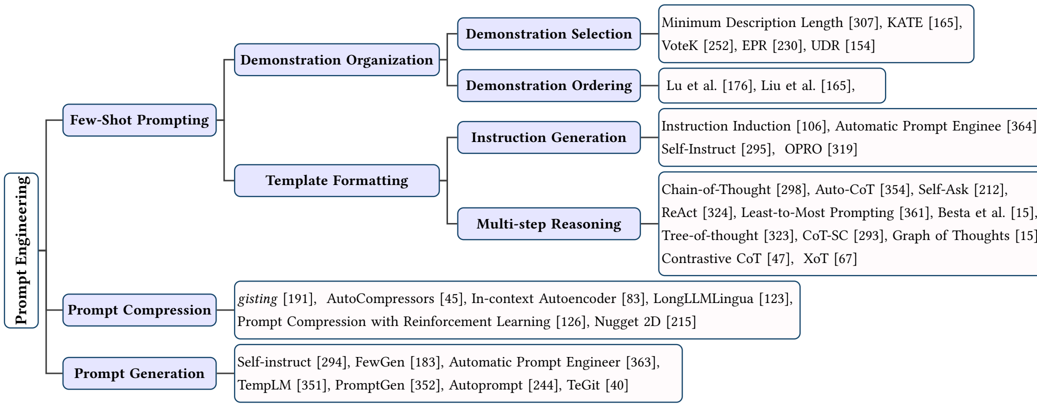

Fig. 17. Summary of Prompt Engineering techniques for LLMs.

This underscores the significance of data quality and lays the groundwork for future improvements in instruction-following model efficacy. Ivison et al. [122] propose using 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 [234] 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 [31] 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 [33] is a model that is 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 nearly 5.7 times, demonstrating the power of high-quality data in instruction-fine-tuning. LIMA [360] fine-tunes LLMs with a small, select set of examples, showing strong performance and challenging the need for extensive tuning. It generalizes well to new tasks and, in comparisons, matched or exceeded 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 [167] is about designing effective inputs (prompts) to guide LLMs in generating desired outputs. It is crucial for LLMs since prompt engineering enables the customization of LLMs for specialized tasks without requiring large amounts of labeled data. Efficient techniques enable these models to process information and response accurately with less computation overhead. The computational cost associated with prompt-based language models has been a subject of ongoing research, especially in the context of task-specific applications. As summarized in Figure 17, prompt engineering techniques can be grouped into few-shot prompting, prompt compression, and prompt generation.

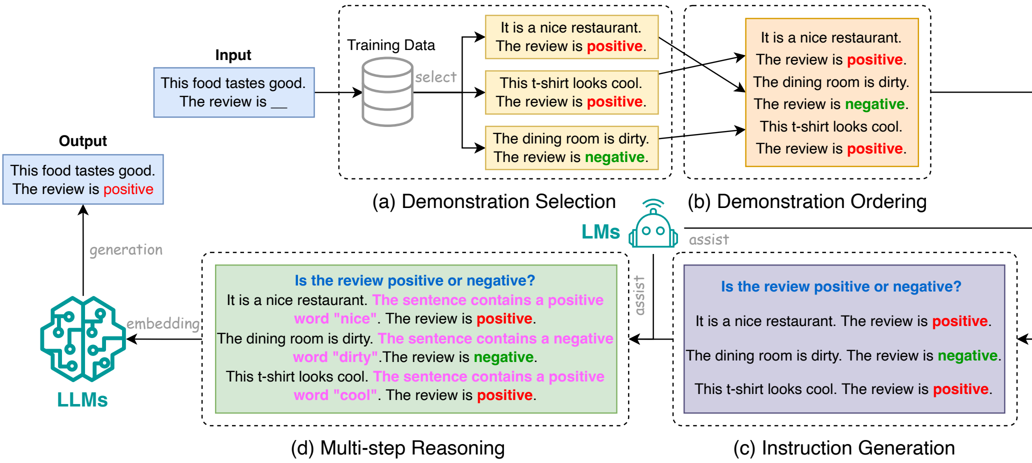

Fig. 18. Illustrations of Few-Shot Prompting Methods for LLMs.

3.2.1 *Few-Shot Prompting*.

Few-shot prompting involves giving a LLM a limited set of examples (referred to as demonstrations) to steer its understanding of a task it is required to execute [298]. 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 [68]. 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. As illustrated in Figure 18, few-shot prompting techniques can generally be grouped into demonstration selection, demonstration ordering, instruction generation, and multistep reasoning.

Demonstration Organization. Demonstration organization refers to organize the demonstrations in an appropriate way to form a suitable prompt for inferencing. The main challenges range from two perspectives, including selection and ordering.

- ***Demonstration Selection.*** Demonstration selection aims to choose the good examples for in-context learning [68]. Existing demonstration selection techniques can be grouped into supervised and unsupervised methods. 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) [307]. For example, KATE [165] is an unsupervised selection method that directly uses the nearest neighbors of a given test sample as the corresponding in-context examples. VoteK [252] 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 [230] is trained to select demonstrations from a small set of candidates initialized by the unsupervised retriever such as BM25 from the training corpse. UDR [154] further enhances EPR by adopting a unified demonstration retriever to unify the demonstration selection across different tasks. Compared to the unsupervised method, the supervised method often leads to a more satisfying generation result but requires frequent adjustment of the retriever for handling the out-of-domain data, making it less efficient for inference.

- ***Demonstration Ordering.*** After selecting representative samples from the training set, another major challenge is the organization of these samples in the prompt. The performance of the model is sensitive to the order of these demonstrations, as indicated in [176]. To date, only a few studies have delved into this area. For example, Liu et al. [165] suggest arranging demonstrations based on their distance from the input, placing the closest demonstration furthest to the right. Lu et al. [176] propose to develop both global and local entropy metrics and use the entropy metric to set up the demonstration order.

Template Formatting. Template Formatting aims to design a suitable template to form the prompt is also critical to guarantee a good generation quality, which is apart from demonstration organization. Template formatting design can be divided into two main parts, including instruction generation and multi-step reasoning.

- ***Instruction Generation.*** The instruction of the template refers to a short description of the task. The prediction accuracy for 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, [106, 364] have demonstrated that LLMs can generate task instructions, and some of them have also proposed their unique automatic instruction generation methods through the interactions with LLMs [295, 364]. Yang et al. [319] 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. Despite the promise of automatic instruction generation methods, their complexity is still a major bottleneck for their real-world adoption.
- ***Multi-step Reasoning.*** Recent advances in few-shot prompting show that guiding the LLMs to produce a sequence of intermediate steps before outputting the final answer can greatly improve the quality of the generation. This augmentation technique is also referred to as chain-of-thought (CoT) prompting [298]. Despite the advantages of CoT, it is still difficult to ensure the accuracy of every intermediate step [68]. Given that, many techniques have been proposed to address this issue. For example, Auto-CoT [354] proposes to generate the CoTs step by step from LLMs. Self-Ask [212] incorporates the self-generated question of each step into the CoT. ReAct [324] 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 [361] breaks down the complex question into smaller ones and answers them iteratively within the context of former questions and answers. Tree-of-thought (ToT) [323] expends CoT to include exploration over coherent units of text and deliberates decision-making processes. CoT-SC [293] 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) [326] depicts human thought processes as a graph instead of a chain. GoT represents thought units as nodes and their connections as edges to capture the non-linear aspect of human thinking for a more advanced model. Similarly, Besta et al. [15] represent information produced by an LLM as a generic graph, with "LLM thoughts" as vertices and edges indicating dependencies between these vertices. Furthermore, Contrastive CoT [47] proposes contrastive chain of thought to enhance language model reasoning by providing both valid and invalid reasoning demonstrations. XoT [67] 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 aims to accelerate the processing the inputs of LLMs through either condensing lengthy prompt inputs or learning and employing prompt representations. Mu et al. [191] 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 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 [126] propose Prompt Compression with Reinforcement Learning (PCRL) which 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. [83] 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 [215] 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. Recently, LongLLMLingua [123] 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 aims to automatically create effective prompts that guide the model in generating specific and relevant responses instead of manual annotated data. Wang et al. [294] propose Self-instruct, an approach that allows LLMs to align with self-generated instructions, highlighting their inherent adaptability. This notion is further expanded upon by [183], which demonstrated the potential of LLMs as robust data generators for augmentation-enhanced few-shot learning. Zhou et al. [363] posited a compelling argument that LLMs can function as intuitive prompt engineers. In a fusion of generative and template-based methodologies, TempLM [351] distills LLMs into template-based generators, offering a harmonized solution for data-to-text tasks. This is further enriched by PromptGen [352], an innovative system that automates the generation of intelligent prompts. Emphasizing the criticality of high-quality instruction-tuning data, both Autoprompt [244] and TeGit [40] underscored the significance of human-written text in refining the quality and authenticity of data. These collective advancements underscore the evolving landscape of prompt generation, emphasizing its pivotal role in optimizing LLM performance.

4 LLM FRAMEWORKS

4.1 LLM Frameworks Supporting Efficient Training and Inference

DeepSpeed. Developed by Microsoft, DeepSpeed is an integrated framework for both training and deploying LLMs. It has been used to train large models like Megatron-Turing NLG 530B [248] (in a joint effort with Nvidia Megatron framework) and BLOOM [233]. Within this framework, `DeepSpeed Inference` is the foundational library. A pivotal feature of this module is *ZeRO-Inference* [222, 223], an optimization technique created to address GPU memory constraints for large model

Table 2. Comparison of LLM frameworks.

Category	Framework	Features
Supports Training and Inference	`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
	`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
Supports Inference only	`vLLM`	Data Parallelism, Model Parallelism, Efficient management via PagedAttention, Optimized CUDA kernels, Dynamic Batching
	`ParallelFormers`	Distributed Finetuning and Inference, Prompt Batching, Quantization, Automatic Mixed Precision, Token Streaming, Prometheus Metrics
	`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

inference. ZeRO-Inference distributes model states across multiple GPUs and CPUs, providing an approach to managing the memory constraints of individual devices. 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 [146, 177, 226, 262]. 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 [5, 223, 305, 330]. 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) [87] with the DeepSpeed training system. Notably, its integration of the *ZeRO-Offload* optimizer [226] facilitates training on both CPUs and GPUs, irrespective of their memory capacities.

Megatron. Megatron [245], structured for the training and deployment of Large Language Models (LLMs), encompasses various specialized tools and frameworks, collaboratively contributing to the effective handling of LLMs. It is a robust framework tailored for training large-scale language models, with a particular emphasis on the effective deployment of GPU model parallelism. It facilitates the training of multi-billion parameter language models, a task that historically posed computational and memory challenges. Megatron, as of writing, supports BERT [64], GPT [218], and T5 [220] models. Central to Megatron-LM'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 potentially enhancing training throughput without compromising model fidelity [245]. Integral to Megatron is *Faster Transformer* [196], which is geared towards optimizing the inference process for large transformer models. It encapsulates two core components: a library designed for transforming a trained Transformer model into an optimized format ready for distributed inference, and a backend utilized by Triton to run the model across multiple GPUs, employing both tensor and pipeline parallelism. Furthermore, Faster Transformer is capable of handling varying precision modes such as 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 [196]. Finally, *TensorRT-LLM* represents a concerted effort to enhance the performance and deployment of LLMs. As part of Nvidia's renowned TensorRT suite, TensorRT-LLM 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 and employs tensor parallelism, facilitating efficient inference at scale across multiple GPUs and servers without necessitating developer intervention or model changes.

Alpa. Alpa [358] 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 GPT2 [218], BLOOM [233], Open Pre-Trained Transformer (OPT) [349], Codegen [195] among others. At the crux 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 [159, 365]. By doing so, the system seeks to harness the collective benefits of these parallelism techniques, potentially leading to optimized resource utilization and enhanced training throughput.

ColossalAI. ColossalAI [16] is an integrative deep-learning system tailored to address the challenges of large-scale parallel training [281]. Rooted in the recognition of the computational demands and intricacies associated with expansive models, the system seeks to provide a unified solution that harmonizes scalability, efficiency, and versatility. It has implementations for LLaMA 1/2 [273], GPT-3[21], GPT-2 [218], BERT [64], PaLM, OPT [349], ViT [69]. 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, the system aims to offer a streamlined platform that reduces fragmentation and enhances workflow efficiency [18]. This integrated approach potentially mitigates the complexities often associated with orchestrating large-scale training across distributed environments. Furthermore, recognizing the dynamic landscape of deep learning research and applications, the system is architected to be inherently modular [37]. In addition, the system integrates a number of other advanced optimization techniques [18, 74, 75, 151, 171, 283]. By leveraging state-of-the-art algorithms and methodologies, Colossal-AI seeks to optimize both computational and communication overheads inherent in parallel training, potentially leading to reduced training times and enhanced model performance.

FairScale. FairScale [73], a creation of Meta Research, serves as 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 with the aim of minimizing 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 FullyShardedDataParallel (FSDP), promoted by Meta as the preferred method for scaling the training operations of extensive neural networks.

Pax. Pax [10], created by Google is a Jax-based framework to configure and run distributed machine learning experiments. The framework has been used to train PaLM-2 [8] and Bard[108]. 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 is essentially a combination of key components. These include 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[189] is aimed at making the training of neural networks faster and more efficient at the algorithmic level. 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. Composer is designed to be versatile with a Functional API for integrating methods directly into training loops, as well as a Trainer API which automatically implements a PyTorch-based training loop, reducing the workload for ML developers.

4.2 LLM Frameworks Supporting Efficient Inference Only

vLLM. vLLM [139] represents a methodological shift in the approach to serving LLMs. Central to vLLM's design is the Paged Attention, a mechanism that segments the attention key and value cache for a set number of tokens. Unlike contiguous space storage, Paged Attention's blocks for the KV cache are stored flexibly, akin to an Operating System's virtual memory management. This facilitates memory sharing at a block level, across various sequences tied to the same request or even different requests, 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 mechanism, 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, the system seeks to further optimize memory usage.

Parallelformers. Parallelformers [136], draws its foundation from Megatron-LM and is engineered to simplify the process of model parallelization in the HuggingFace Transformers library ecosystem. This design enables the parallelization of a diverse range of models within the HuggingFace Transformers library across multiple GPUs. The framework mirrors Huggingface's popular transformers framework and is easily adaptable.

OpneLLM. OpneLLM [207] 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, potentially 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 [213] represents a strategic fusion of LLMs with the Ray ecosystem [188], aiming to optimize the deployment and operation of these expansive models. Situated at the intersection of cutting-edge model architecture and scalable infrastructure, RayLLM seeks to redefine the paradigms of LLM utilization. At the core of RayLLM's approach is the leveraging of Ray's inherent distributed computing capabilities. Recognizing the computational demands of LLMs, RayLLM 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.

MLC-LLM. MLC-LLM [267] 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 everyday smartphones, MLC-LLM compiles models and deploys them in a process that is inherently tailored to the specific capabilities and constraints of each device [36, 78, 237]. This device-native focus ensures that AI models are not only efficient but also highly optimized for the environments in which they operate. Furthermore, MLC-LLM champions a user-centric design philosophy. By providing intuitive tools and frameworks, the system seeks to lower the barriers to AI model development, making it accessible even to those without deep technical expertise. This democratization is further complemented by advanced optimization tools that automate many of the intricate processes associated with model refinement, ensuring that models are both robust and efficient.

Sax. Sax [11] is a cutting-edge platform designed by Google for deploying Pax, JAX, and PyTorch models for inference tasks. Within Sax, there is a unit known as the Sax cell (or Sax cluster). This cell 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.

Mosec. Mosec [321] is designed for serving large deep learning models efficiently, particularly in cloud environments. It's built to streamline the deployment 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

LLM Foundry. LLM Foundry [190], is a library for finetuning, evaluating, and deploying LLMs for inference with Composer and the MosaicML platform. 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 the field of large language models.

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, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. 2023. GKD: Generalized Knowledge Distillation for Auto-regressive Sequence Models. *arXiv preprint arXiv:2306.13649* (2023).

[2] 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 preprint arXiv:2305.13245* (2023).

[3] Silas Alberti, Niclas Dern, Laura Thesing, and Gitta Kutyniok. 2023. Sumformer: Universal Approximation for Efficient Transformers. In *Topological, Algebraic and Geometric Learning Workshops 2023*. PMLR, 72–86.

[4] Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, et al. 2022. DeepSpeed-inference: enabling efficient inference of transformer models at unprecedented scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–15.

[5] Reza Yazdani Aminabadi, Samyam Rajbhandari, Minjia Zhang, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Jeff Rasley, Shaden Smith, Olatunji Ruwase, and Yuxiong He. 2022. DeepSpeed Inference: Enabling Efficient Inference of Transformer Models at Unprecedented Scale. In *SC 2022*. arXiv:arXiv:2207.00032

[6] Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hoffmann. 2023. Dynamic Context Pruning for Efficient and Interpretable Autoregressive Transformers. *arXiv preprint arXiv:2305.15805* (2023).

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

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

[9] Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, et al. 2021. Efficient large scale language modeling with mixtures of experts. *arXiv preprint arXiv:2112.10684* (2021).

[10] 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.

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

[12] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. 2021. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*. PMLR, 1352–1361.

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

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

[15] Maciej Besta, Nils Blach, Alevs Kubivcek, 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* abs/2308.09687 (2023).

[16] Zhengda Bian, Hongxin Liu, Boxiang Wang, Haichen Huang, Yongbin Li, Chuanrui Wang, Fan Cui, and Yang You. 2021. Colossal-AI: A Unified Deep Learning System For Large-Scale Parallel Training. *arXiv preprint arXiv:2110.14883* (2021).

[17] Zhengda Bian, Qifan Xu, Boxiang Wang, and Yang You. 2021. Maximizing Parallelism in Distributed Training for Huge Neural Networks. *ArXiv* abs/2105.14450 (2021). https://api.semanticscholar.org/CorpusID:235254604

[18] Zhengda Bian, Qifan Xu, Boxiang Wang, and Yang You. 2021. Maximizing parallelism in distributed training for huge neural networks. *arXiv preprint arXiv:2105.14450* (2021).

[19] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745* (2022).

[20] Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. 2021. Understanding and overcoming the challenges of efficient transformer quantization. *arXiv preprint arXiv:2109.12948* (2021).

[21] 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. *ArXiv* abs/2005.14165 (2020).

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

[23] 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, 88–91.

[24] Lucas Caccia, E. Ponti, Zhan Su, Matheus Pereira, Nicolas Le Roux, and Alessandro Sordoni. 2022. Multi-Head Adapter Routing for Cross-Task Generalization. https://api.semanticscholar.org/CorpusID:259262344

[25] Yihan Cao, Yanbin Kang, and Lichao Sun. 2023. Instruction mining: High-quality instruction data selection for large language models. *arXiv preprint arXiv:2307.06290* (2023).

[26] Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Chris De Sa. 2023. QuIP: 2-Bit Quantization of Large Language Models With Guarantees. *ArXiv* abs/2307.13304 (2023). https://api.semanticscholar.org/CorpusID:260154775

[27] 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.

[28] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, L. Sifre, and John M. Jumper. 2023. Accelerating Large Language Model Decoding with Speculative Sampling. *ArXiv* abs/2302.01318 (2023).

[29] 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.

[30] 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 preprint arXiv:2110.07143* (2021).

[31] Hao Chen, Yiming Zhang, Qi Zhang, Hantao Yang, Xiaomeng Hu, Xuetao Ma, Yifan Yanggong, and Junbo Zhao. 2023. Maybe Only 0.5% Data is Needed: A Preliminary Exploration of Low Training Data Instruction Tuning. *arXiv preprint arXiv:2305.09246* (2023).

[32] Jiaao Chen, Aston Zhang, Xingjian Shi, Mu Li, Alexander J. Smola, and Diyi Yang. 2023. Parameter-Efficient Fine-Tuning Design Spaces. *ArXiv* abs/2301.01821 (2023).

[33] Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. 2023. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701* (2023).

[34] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison Edwards, Yura 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, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Arun Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. 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* abs/2107.03374 (2021). https://api.semanticscholar.org/CorpusID:235755472

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

[36] 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

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

[38] Wuyang Chen, Yanqi Zhou, Nan Du, Yanping Huang, James Laudon, Zhifeng Chen, and Claire Cui. 2023. Lifelong language pretraining with distribution-specialized experts. In *International Conference on Machine Learning*. PMLR, 5383–5395.

[39] X Chen, C Liang, D Huang, E Real, K Wang, Y Liu, H Pham, X Dong, T Luong, CJ Hsieh, et al. [n.d.]. Symbolic Discovery of Optimization Algorithms. *arXiv preprint arXiv:2302.06675* ([n. d.]).

[40] 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 preprint arXiv:2309.05447* (2023).

[41] 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* abs/2309.12307 (2023). https://api.semanticscholar.org/CorpusID:262084134

[42] Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. 2021. Meta-learning via Language Model In-context Tuning. *ArXiv* abs/2110.07814 (2021).

[43] Zixiang Chen, Yihe Deng, Yue Wu, Quanquan Gu, and Yuanzhi Li. 2022. Towards understanding mixture of experts in deep learning. *arXiv preprint arXiv:2208.02813* (2022).

[44] Zeming Chen, Qiyue Gao, Kyle Richardson, Antoine Bosselut, and Ashish Sabharwal. 2022. DISCO: Distilling Counterfactuals with Large Language Models. In *Annual Meeting of the Association for Computational Linguistics*. https://api.semanticscholar.org/CorpusID:254877039

[45] Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting Language Models to Compress Contexts. *arXiv preprint arXiv:2305.14788* (2023).

[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, Anh Tuan Luu, Soujanya Poria, and Lidong Bing. 2023. Contrastive Chain-of-Thought Prompting.

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

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

[50] 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, et al. 2022. PaLM: Scaling Language Modeling with Pathways. *ArXiv* abs/2204.02311 (2022).

[51] Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling Instruction-Finetuned Language Models. *ArXiv* abs/2210.11416 (2022). https://api.semanticscholar.org/CorpusID:253018554

[52] Luciano Del Corro, Allison Del Giorno, Sahaj Agarwal, Ting Yu, Ahmed Hassan Awadallah, and Subhabrata Mukherjee. 2023. SkipDecode: Autoregressive Skip Decoding with Batching and Caching for Efficient LLM Inference. *ArXiv* abs/2307.02628 (2023).

[53] Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. StableMoE: Stable routing strategy for mixture of experts. *arXiv preprint arXiv:2204.08396* (2022).

[54] 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.

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

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

[57] 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.

[58] Tri Dao, Daniel Y Fu, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. 2022. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052* (2022).

[59] Joe Davison. 2021. Compacter: Efficient Low-Rank Hypercomplex Adapter Layers. In *Neural Information Processing Systems*.

[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* abs/2208.07339 (2022).

[62] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. *ArXiv* abs/2305.14314 (2023).

[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* abs/2306.03078 (2023).

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

[65] Prafulla Dhariwal. 2022. Meta-Adapters: Parameter Efficient Few-shot Fine-tuning through Meta-Learning.

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

[67] 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 preprint arXiv:2311.04254* (2023).

[68] 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]

[69] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[70] 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, 5547–5569.

[71] Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen S. Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V. Le, Yonghui Wu, Z. Chen, and Claire Cui. 2021. GLaM: Efficient Scaling of Language Models with Mixture-of-Experts. *ArXiv* abs/2112.06905 (2021). https://api.semanticscholar.org/CorpusID:245124124

[72] Lance B. Eliot. 2021. Generative Pre-Trained Transformers (GPT-3) Pertain To AI In The Law. *SSRN Electronic Journal* (2021).

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

[74] J. Fang et al. 2022. A Frequency-aware Software Cache for Large Recommendation System Embeddings. *arXiv* (2022).

[75] 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.

[76] Mahan Fathi, Jonathan Pilault, Pierre-Luc Bacon, Christopher Pal, Orhan Firat, and Ross Goroshin. 2023. Block-State Transformer. *arXiv preprint arXiv:2306.09539* (2023).

[77] 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.

[78] 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

[79] Elias Frantar and Dan Alistarh. 2022. Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning. *ArXiv* abs/2208.11580 (2022). https://api.semanticscholar.org/CorpusID:251765570

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

[81] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. *ArXiv* abs/2210.17323 (2022).

[82] Yao Fu, Hao-Chun Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing Smaller Language Models towards Multi-Step Reasoning. In *International Conference on Machine Learning*.

[83] Tao Ge, Jing Hu, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context Autoencoder for Context Compression in a Large Language Model. *arXiv preprint arXiv:2307.06945* (2023).

[84] Michael Glass, Alfio Gliozzo, Rishav Chakravarti, Anthony Ferritto, Lin Pan, GP Bhargav, Dinesh Garg, and Avirup Sil. 2019. Span selection pre-training for question answering. *arXiv preprint arXiv:1909.04120* (2019).

[85] 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, 2337–2346. https://proceedings.mlr.press/v97/gong19a.html

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

[87] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. 2013. Policy shaping: Integrating human feedback with reinforcement learning. *Advances in neural information processing systems* 26 (2013).

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

[89] Xiaotao Gu, Liyuan Liu, Hongkun Yu, Jing Li, Chen Chen, and Jiawei Han. 2020. On the transformer growth for progressive bert training. *arXiv preprint arXiv:2010.12562* (2020).

[90] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Knowledge Distillation of Large Language Models. *arXiv preprint arXiv:2306.08543* (2023).

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

[92] Cong Guo, Jiaming Tang, Weiming Hu, Jingwen Leng, Chen Zhang, Fan Yang, Yun-Bo Liu, Minyi Guo, and Yuhao Zhu. 2023. OliVe: Accelerating Large Language Models via Hardware-friendly Outlier-Victim Pair Quantization. *Proceedings of the 50th Annual International Symposium on Computer Architecture* (2023).

[93] 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.

[94] Ahan Gupta, Yueming Yuan, Yanqi Zhou, and Charith Mendis. 2023. FLuRKA: Fast fused Low-Rank & Kernel Attention. *arXiv preprint arXiv:2306.15799* (2023).

[95] 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, et al. 2020. Aˆ 3: Accelerating attention mechanisms in neural networks with approximation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 328–341.

[96] 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)*. IEEE, 692–705.

[97] Yaru Hao, Yutao Sun, Li Dong, Zhixiong Han, Yuxian Gu, and Furu Wei. 2022. Structured prompting: Scaling in-context learning to 1,000 examples. *arXiv preprint arXiv:2212.06713* (2022).

[98] Aaron Harlap, Deepak Narayanan, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, and Phillip B. Gibbons. 2018. PipeDream: Fast and Efficient Pipeline Parallel DNN Training. *ArXiv* abs/1806.03377 (2018). https://api.semanticscholar.org/CorpusID:47016772

[99] Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. 2021. Fastmoe: A fast mixture-of-expert training system. *arXiv preprint arXiv:2103.13262* (2021).

[100] 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.

[101] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a Unified View of Parameter-Efficient Transfer Learning. *ArXiv* abs/2110.04366 (2021).

[102] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*. 1026–1034.

[103] Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large Language Models Are Reasoning Teachers. In *Annual Meeting of the Association for Computational Linguistics*.

[104] 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, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and L. Sifre. 2022. Training Compute-Optimal Large Language Models. *ArXiv* abs/2203.15556 (2022). https://api.semanticscholar.org/CorpusID:247778764

[105] Ke Hong, Guohao Dai, Jiaming Xu, Qiuli Mao, Xiuhong Li, Jun Liu, Kangdi Chen, Hanyu Dong, and Yu Wang. 2023. FlashDecoding++: Faster Large Language Model Inference on GPUs. *arXiv preprint arXiv:2311.01282* (2023).

[106] 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]

[107] 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 *International Conference on Machine Learning*.

[108] 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.

[109] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander 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. *arXiv preprint arXiv:2305.02301* (2023).

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

[111] 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. *ArXiv* abs/2307.03084 (2023).

[112] Zhiqiang Hu, Yihuai Lan, Lei Wang, Wanyu Xu, Ee-Peng Lim, Roy Ka-Wei Lee, Lidong Bing, and Soujanya Poria. 2023. LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models. *ArXiv* abs/2304.01933 (2023).

[113] 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* abs/2307.13269 (2023).

[114] Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. 2020. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*. PMLR, 4475–4483.

[115] 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* abs/2212.10670 (2022).

[116] Yanping Huang, Yonglong Cheng, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, and Z. Chen. 2018. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. In *Neural Information Processing Systems*. https://api.semanticscholar.org/CorpusID:53670168

[117] Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Joseph Naor, and Daniel Soudry. 2021. Accelerated sparse neural training: A provable and efficient method to find n: m transposable masks. *Advances in neural information processing systems* 34 (2021), 21099–21111.

[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. 2022. Data-Efficient Finetuning Using Cross-Task Nearest Neighbors. *arXiv preprint arXiv:2212.00196* (2022).

[123] 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 preprint arXiv:2310.06839* (2023).

[124] Yuxin Jiang, Chunkit Chan, Mingyang Chen, and Wei Wang. 2023. Lion: Adversarial Distillation of Closed-Source Large Language Model. *ArXiv* abs/2305.12870 (2023).

[125] Yunho Jin, Chun-Feng Wu, David Brooks, and Gu-Yeon Wei. 2023. S3: Increasing GPU Utilization during Generative Inference for Higher Throughput. *arXiv preprint arXiv:2306.06000* (2023).

[126] Hoyoun Jung and Kyung-Joong Kim. 2023. Discrete Prompt Compression with Reinforcement Learning. *arXiv preprint arXiv:2308.08758* (2023).

[127] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169* (2023).

[128] Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, et al. 2019. A study of BFLOAT16 for deep learning training. *arXiv preprint arXiv:1905.12322* (2019).

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

[130] 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* abs/2305.14152 (2023).

[131] 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 preprint arXiv:2308.06744* (2023).

[132] Sehoon Kim, Karttikeya Mangalam, Suhong Moon, John Canny, Jitendra Malik, Michael W. Mahoney, Amir Gholami, and Kurt Keutzer. 2023. Speculative Decoding with Big Little Decoder.

[133] Young Jin Kim, Rawn Henry, Raffy Fahim, and Hany Hassan Awadalla. 2023. FineQuant: Unlocking Efficiency with Fine-Grained Weight-Only Quantization for LLMs. *ArXiv* abs/2308.09723 (2023).

[134] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

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

[136] Hyunwoong Ko. 2021. Parallelformers: An Efficient Model Parallelization Toolkit for Deployment. https://github.com/tunib-ai/parallelformers.

[137] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. *ArXiv* abs/2205.11916 (2022).

[138] Siddharth Krishna Kumar. 2017. On weight initialization in deep neural networks. *arXiv e-prints* (2017), arXiv–1704.

[139] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. *arXiv preprint arXiv:2309.06180* (2023).

[140] 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* abs/2306.02272 (2023).

[141] 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 *International conference on machine learning*. PMLR, 3744–3753.

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

[143] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Conference on Empirical Methods in Natural Language Processing*.

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

[145] 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.

[146] 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

[147] Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph E Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. [n.d.]. How long can open-source llms truly promise on context length?, June 2023. *URL https://lmsys.org/blog/2023-06-29-longchat* ([n. d.]).

[148] 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).

[149] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. StarCoder: may the source be with you! *arXiv preprint*

arXiv:2305.06161 (2023).
[150] 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).
[151] S. Li, F. Xue, C. Baranwal, Y. Li, and Y. You. 2021. Sequence Parallelism: Long Sequence Training from System Perspective. *arXiv* (2021).
[152] 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).
[153] 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).
[154] 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
[155] 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).
[156] 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).
[157] 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).
[158] 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
[159] 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)*.
[160] 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.
[161] 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* abs/2306.00978 (2023).
[162] 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).
[163] 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).
[164] 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).
[165] 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
[166] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172* (2023).
[167] 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.
[168] 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).
[169] 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.

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

[171] 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).

[172] 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).

[173] 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).

[174] 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.

[175] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[176] 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

[177] 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

[178] Kai Lv, Yuqing Yang, Tengxiao Liu, Qinghui Gao, Qipeng Guo, and Xipeng Qiu. 2023. Full Parameter Fine-tuning for Large Language Models with Limited Resources. *arXiv preprint arXiv:2306.09782* (2023).

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

[180] 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).

[181] Pedro Henrique Martins, Zita Marinho, and André FT Martins. 2021. ∞-former: Infinite Memory Transformer. *arXiv preprint arXiv:2109.00301* (2021).

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

[183] Yu Meng, Martin Michalski, Jiaxin Huang, Yu Zhang, Tarek Abdelzaher, and Jiawei Han. 2023. Tuning language models as training data generators for augmentation-enhanced few-shot learning. In *International Conference on Machine Learning*. PMLR, 24457–24477.

[184] 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).

[185] 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).

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

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

[188] 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.

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

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

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

[192] 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

[193] Benjamin Newman, John Hewitt, Percy Liang, and Christopher D Manning. 2020. The EOS decision and length extrapolation. *arXiv preprint arXiv:2010.07174* (2020).
[194] Erik Nijkamp, Hiroaki Hayashi, Caiming Xiong, Silvio Savarese, and Yingbo Zhou. 2023. Codegen2: Lessons for training llms on programming and natural languages. *arXiv preprint arXiv:2305.02309* (2023).
[195] 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).
[196] NVIDIA. the year the version you used was released or accessed. FasterTransformer: High Performance Transformer Kernels. https://github.com/NVIDIA/FasterTransformer. GitHub repository.
[197] OpenAI. 2023. GPT-4 Technical Report. *ArXiv* abs/2303.08774 (2023).
[198] 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).
[199] Daniele Paliotta, Matteo Pagliardini, Martin Jaggi, and François Fleuret. 2023. Fast Causal Attention with Dynamic Sparsity. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*.
[200] 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).
[201] 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).
[202] 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).
[203] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction Tuning with GPT-4. *ArXiv* abs/2304.03277 (2023).
[204] 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).
[205] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. 2021. Random feature attention. *arXiv preprint arXiv:2103.02143* (2021).
[206] Jonas Pfeiffer, Ivan Vulic, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-based Framework for Multi-task Cross-lingual Transfer. In *Conference on Empirical Methods in Natural Language Processing*.
[207] 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
[208] 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).
[209] 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.
[210] 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).
[211] Ofir Press, Noah A Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409* (2021).
[212] 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]
[213] Ray Project. 2023. *RayLLM - LLMs on Ray.* https://github.com/ray-project/ray-llm GitHub repository.
[214] Joan Puigcerver, Carlos Riquelme, Basil Mustafa, and Neil Houlsby. 2023. From Sparse to Soft Mixtures of Experts. *arXiv preprint arXiv:2308.00951* (2023).
[215] 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
[216] 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).
[217] 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).
[218] 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

[219] 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

[220] 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.

[221] 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*.

[222] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2019. ZeRO: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '20)*. arXiv:arXiv:1910.02054

[223] 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

[224] 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.

[225] 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. arXiv:2101.06840 [cs.DC]

[226] 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

[227] 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

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

[229] 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.

[230] 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

[231] Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. 2017. Data Selection Strategies for Multi-Domain Sentiment Analysis. *ArXiv* abs/1702.02426 (2017).

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

[233] 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

[234] 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).

[235] 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

[236] 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

[237] 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_files/paper/2022/file/e894eafae43e68b4c8dfdacf742bcbf3-Paper-Conference.pdf

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

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

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

[241] 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.

[242] 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.

[243] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Daniel Y Fu, Zhiqiang Xie, Beidi Chen, Clark Barrett, Joseph E Gonzalez, et al. 2023. High-throughput generative inference of large language models with a single gpu. *arXiv preprint arXiv:2303.06865* (2023).

[244] 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).

[245] 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).

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

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

[248] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay 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. (2022). arXiv:arXiv:2201.11990

[249] 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).

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

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

[252] 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]

[253] 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).

[254] Jianlin Su. 2023. Naive Bayes-based Context Extension. https://github.com/bojone/NBCE.

[255] 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).

[256] 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).

[257] 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*.

[258] 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.]).

[259] 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).

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

[261] Derek Tam, Anisha Mascarenhas, Shiyue Zhang, Sarah Kwan, Mohit Bansal, and Colin Raffel. 2022. Evaluating the factual consistency of large language models through summarization. *arXiv preprint arXiv:2211.08412* (2022).

[262] 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

[263] 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).

[264] 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.

[265] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient Transformers: A Survey. *Comput. Surveys* (2020).

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

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

[268] MN Team et al. 2023. Introducing MPT-7B: A New Standard for Open-Source, Commercially Usable LLMs.

[269] 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).

[270] 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).

[271] Denis Timonin, Bo Yang Hsueh, and Vinh Nguyen. 2022. Accelerated inference for large transformer models using nvidia triton inference server. *NVIDIA blog* (2022).

[272] 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

[273] 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

[274] 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).

[275] 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).

[276] 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

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

[278] 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).

[279] Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 billion parameter autoregressive language model.

[280] Benyou Wang, Qianqian Xie, Jiahuan Pei, Prayag Tiwari, Zhao Li, and Jie Fu. 2021. Pre-trained Language Models in Biomedical Domain: A Systematic Survey. *Comput. Surveys* (2021).

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

[282] Boxiang Wang, Qifan Xu, Zhengda Bian, and Yang You. 2021. Tesseract: Parallelize the Tensor Parallelism Efficiently. *Proceedings of the 51st International Conference on Parallel Processing* (2021). https://api.semanticscholar.org/CorpusID:251979875

[283] 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*.

[284] 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.

[285] 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

[286] 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).

[287] Jue Wang, Yucheng Lu, Binhang Yuan, Beidi Chen, Percy Liang, Christopher De Sa, Christopher Re, and Ce Zhang. 2023. CocktailSGD: Fine-tuning Foundation Models over 500Mbps Networks. In *International Conference on Machine Learning*. PMLR, 36058–36076.

[288] 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.

[289] 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).

[290] 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*.

[291] 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).

[292] 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).

[293] 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).

[294] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560* (2022).

[295] 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. arXiv:2212.10560 [cs.CL]

[296] 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).

[297] 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).

[298] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent Abilities of Large Language Models. arXiv:2206.07682 [cs.CL]

[299] 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).

[300] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. *ArXiv* abs/2201.11903 (2022). https://api.semanticscholar.org/CorpusID:246411621

[301] 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 (2022), 24824–24837.

[302] 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).

[303] 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.

[304] 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).

[305] 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

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

[307] 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]

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

[309] Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. 2022. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. *ArXiv* abs/2211.10438 (2022).

[310] 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).

[311] 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).

[312] 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.

[313] 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).

[314] 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

[315] Qifan Xu, Shenggui Li, Chaoyu Gong, and Yang You. 2023. An Efficient 2D Method for Training Super-Large Deep Learning Models. *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (2023), 222–232. https://api.semanticscholar.org/CorpusID:233210609

[316] Yaodan Xu, Jingzhou Sun, Sheng Zhou, and Zhisheng Niu. 2023. SMDP-Based Dynamic Batching for Efficient Inference on GPU-Based Platforms. *ArXiv* abs/2301.12865 (2023). https://api.semanticscholar.org/CorpusID:256390102

[317] 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).

[318] 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).

[319] 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]

[320] 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).

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

[322] 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

[323] 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]

[324] 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]

[325] 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.

[326] Yao Yao, Zuchao Li, and Hai Zhao. 2023. Beyond Chain-of-Thought, Effective Graph-of-Thought Reasoning in Large Language Models. *arXiv preprint arXiv:2305.16582* (2023).

[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. *ArXiv* abs/2206.01861 (2022).

[330] 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

[331] 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.

[332] Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, et al. 2023. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models. *arXiv preprint arXiv:2303.10420* (2023).

[333] 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).

[334] Gyeong-In Yu and Joo Seong Jeong. 2022. Orca: A Distributed Serving System for Transformer-Based Generative Models. In *USENIX Symposium on Operating Systems Design and Implementation*. https://api.semanticscholar.org/CorpusID:251734964

[335] 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.

[336] 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).

[337] 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).

[338] 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.

[339] 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

[340] Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, ZhenZhang Yang, Kaisheng Wang, Xiaoda Zhang, et al. 2021. Pangu-$alpha$: Large-scale autoregressive pretrained Chinese language models with auto-parallel computation. *arXiv preprint arXiv:2104.12369* (2021).

[341] Sheng Zha, Ziheng Jiang, Haibin Lin, and Zhi Zhang. 2019. Just-in-Time Dynamic-Batching. *ArXiv* abs/1904.07421 (2019). https://api.semanticscholar.org/CorpusID:119108574

[342] 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.

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

[344] 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.

[345] 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).

[346] 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).

[347] 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).

[348] 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).

[349] 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).

[350] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open Pre-trained Transformer Language Models. *ArXiv* abs/2205.01068 (2022). https://api.semanticscholar.org/CorpusID:248496292

[351] 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).

[352] 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.

[353] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. 2023. H _2 O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. *arXiv preprint arXiv:2306.14048* (2023).

[354] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic Chain of Thought Prompting in Large Language Models. arXiv:2210.03493 [cs.CL]

[355] Jiawei Zhao, Florian Schäfer, and Anima Anandkumar. 2021. ZerO initialization: Initializing neural networks with only zeros and ones. *arXiv preprint arXiv:2110.12661* (2021).

[356] 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).

[357] 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

[358] 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)*.

[359] 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).

[360] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206* (2023).

[361] 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]

[362] 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.

[363] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910* (2022).

[364] 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. arXiv:2211.01910 [cs.LG]

[365] 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)*.

[366] 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).