Resource-efficient Algorithms and Systems of Foundation Models: A Survey

MENGWEI XU, Beijing University of Posts and Telecommunications, Beijing, China
DONGQI CAI, Beijing University of Posts and Telecommunications, Beijing, China
WANGSONG YIN, Peking University, Beijing, China
SHANGGUANG WANG, Beijing University of Posts and Telecommunications, Beijing, China
XIN JIN, Peking University, Beijing, China
XUANZHE LIU, Peking University, Beijing, China

Large foundation models, including large language models, vision transformers, diffusion, and large language model based multimodal models, are revolutionizing the entire machine learning lifecycle, from training to deployment. However, the substantial advancements in versatility and performance these models offer come at a significant cost in terms of hardware resources. To support the growth of these large models in a scalable and environmentally sustainable way, there has been a considerable focus on developing resource-efficient strategies. This survey delves into the critical importance of such research, examining both algorithmic and systemic aspects. It offers a comprehensive analysis and valuable insights gleaned from existing literature, encompassing a broad array of topics from cutting-edge model architectures and training/serving algorithms to practical system designs and implementations. The goal of this survey is to provide an overarching understanding of how current approaches are tackling the resource challenges posed by large foundation models and to potentially inspire future breakthroughs in this field.

CCS Concepts: • **Computing methodologies** → **Natural language processing**; **Computer vision;**

Additional Key Words and Phrases: Resource efficiency, foundation models, algorithm and system optimization

ACM Reference Format:
Mengwei Xu, Dongqi Cai, Wangsong Yin, Shangguang Wang, Xin Jin, and Xuanzhe Liu. 2025. Resource-efficient Algorithms and Systems of Foundation Models: A Survey. *ACM Comput. Surv.* 57, 5, Article 110 (January 2025), 39 pages. https://doi.org/10.1145/3706418

M. Xu, D. Cai, and W. Yin contributed equally to this survey.

M. Xu was supported by NSFC 62102045.

Authors' Contact Information: Mengwei Xu, Beijing University of Posts and Telecommunications, Beijing, Beijing, China; e-mail: mwx@bupt.edu.cn; Dongqi Cai, Beijing University of Posts and Telecommunications, Beijing, China; e-mail: cdq@bupt.edu.cn; Wangsong Yin, Peking University, Beijing, China; e-mail: yws@stu.pku.edu.cn; Shangguang Wang, Beijing University of Posts and Telecommunications, Beijing, China; e-mail: sgwang@bupt.edu.cn; Xin Jin, Peking University, Beijing, Beijing, China; e-mail: xinjinpku@pku.edu.cn; Xuanzhe Liu, Peking University, Beijing, Beijing, China; e-mail: liuxuanzhe@pku.edu.cn.

ACM 0360-0300/2025/01-ART110
https://doi.org/10.1145/3706418

1 Introduction

In the rapidly evolving field of **artificial intelligence (AI)**, a paradigm shift is under way. We are witnessing the transition from specialized, fragmented deep learning models to versatile, one-size-fits-all **foundation models (FMs)**. These advanced AI systems are capable of operating in an open-world context, interacting with open vocabularies and image pixels for unseen AI tasks (i.e., zero-shot abilities). They are exemplified by (1) **large language models (LLMs)** such as GPTs [25] that can ingest almost every NLP task in the form as a prompt; (2) **vision transformers models (ViTs)** such as **Masked Autoencoder (MAE)** [94] that can handle various downstream vision tasks; (3) **latent diffusion models (LDMs)** such as Stable Diffusion [218] that generate high-quality images with arbitrary text-based prompts; (4) multimodal models such as CLIP [212] and ImageBind [77] that map different modal data into the same latent space and are widely used as backbone for cross-modality tasks like image retrieval/search and visual-question answering. Such flexibility and generality mark a significant departure from the earlier era of AI, setting a new standard for how AI interfaces with the world.

The success of these FMs is deeply rooted in their scalability: unlike their predecessors, these models' accuracy and generalization ability can continuously expand with more data or parameters, without altering the underlying algorithms and architectures [279]. An impressive evidence is the scaling law [117]: it describes how the performance of transformer-based models can predictably improve with more model size and data volume; until today, the scaling law stands still. This scalability is not just a matter of model size; it extends to their ability to tackle increasingly complex tasks, making them a cornerstone in the journey toward AGI (artificial general intelligence).

However, the scalability comes at a cost of huge resource demand. FMs, by their very nature, are resource-hungry for training and deployment. These resources encompass not only the computing processors like GPUs and TPUs but also the memory, energy, and network bandwidth. For example, the pre-training of LLaMa-2-70B takes 1.7× millions of GPU hours and consumes 2.5×10^{12} joules of energy. The estimated total emissions were 291 tons of CO_2 equivalent. Beyond training, the data processing, experimentation, and inference stages consume comparable or even more electricity according to Meta AI [265]. A recent analysis [48] reveals that, to satisfy the continuation of the current trends in AI capacity and adoption, NVIDIA needs to ship 1.5 million AI server units per year by 2027. These servers, running at full capacity, would consume at least 85.4 terawatt-hours of electricity annually—more than what many countries like New Zealand and Austria use in a whole year. As FMs increase in size and complexity, their resource requirements escalate, posing a significant challenge in their development and deployment.

The huge resource footprint of a large FM also hinders its democratization. Up to the end of 2023, there were only a few major players capable of training and deploying the state-of-the-art FMs, who thereby have powerful control over the public and can potentially manipulate them in a way they prefer. The models are served on clouds instead of devices as many lightweight DNNs do [275, 302]; it makes data privacy preservation almost impossible. However, recently, smartphone vendors have been boasting about running large FMs locally and some pioneering engines were developed for on-device LLMs [6, 7, 76, 169], but the models demonstrated are limited to relatively small scale (e.g., <10 billion) and have not yet seen real-world deployment.

Therefore, a significant amount of research has been dedicated to enhance the efficiency of these FMs. These efforts span a wide range of approaches, from optimizing algorithms to system-level innovations, focusing on reducing the resource footprint of these models without compromising their performance. This survey aims to delve into these research efforts, exploring the diverse strategies employed to make FMs more resource-efficient. We will examine advancements in algorithmic efficiency, system optimizations, and the development of novel architectures that are less resource-intensive. The survey also spans from clouds to edge and devices, where the large FMs gain dramatic attention as well. Through this exploration, we aim to provide a comprehensive understanding of the current state and future directions of resource-efficient algorithms and systems in the realm of FMs.

Scope and Rationales. First, we survey only algorithm and system innovations; we exclude a huge body of work at hardware design. Second, the definition of *resource* in this survey is limited to mainly physical ones, including computing, memory, storage, bandwidth, and so forth; we exclude training data (labels) and privacy that can also be regarded as resources. Third, we mainly survey papers published by top-tier CS conferences (i.e., those included in CSRankings). We also manually pick related and potentially high-impact papers from arXiv. Fourth, we mainly survey papers published after the year 2020, since the innovation of AI is going fast, with old knowledge and methods being overturned frequently.

Organization. Section 2 overviews the classical FMs and their runtime cost. Section 3 investigates the architectural innovations that revise or replace the existing FM architectures. Section 4 and Section 5 examine the algorithm-level and system-level literature toward more resource-efficient FMs. Section 6 concludes the survey and presents potential future directions.

Comparison to Relevant Surveys. Concurrent to this work, there are a few (not yet peer-reviewed) surveys about efficient LLMs, spanning from compression [316], algorithms [54], system-algorithm [182, 246], and hardware [123]. As comparison, this work is the first comprehensive survey toward resource-efficient FMs, including not only LLMs but also multimodal ones that are equally important, such as diffusion models and ViTs. An extended version of this survey is available elsewhere [277].

2 FM Overview

Figure 1 illustrates the evolutionary trace of popular FMs up to January 2024. In general, there are three types of FMs: language-based, vision-based, and multimodal FMs.

Language FMs. Language FMs typically employ attention-based transformer architecture [244]. The process initiates by converting input words into high-dimensional vectors through an embedding layer. During processing, attention mechanisms assign varying weights to different segments of these input vectors. Following attention, layer normalization is applied to the output, ensuring stabilization and standardization of the activations. Subsequently, each position-wise vector undergoes transformation through a **feedforward network (FFN)**, introducing non-linearity and enabling the model to capture complex data patterns. Through multiple layers that incorporate these components, the transformer learns hierarchical representations of the input data. In the final stage, the output from the last transformer layer is directed into a linear layer, culminating in the final prediction.

Vision FMs. In this article, we use the term *vision FMs* to refer to the FMs that only involve the pure vision modality in their main pipeline. Vision FMs (e.g., SAM, seggpt [125, 254]) typically employ ViT architecture [56], a transformer-based visual information processing block. As such, efficient vision FMs (as well as those multimodal ones that rely on ViT) often benefit from the efficient ViT designs. Given an input image, ViT first splits an image into fixed-size patches (i.e., tokens) by a convolutional embedding layer. For instance, a standard size RGB image input (i.e., $3\times224\times224$) will be split to 14×14 patches with 16×16 pixels. This embedding overhead is almost negligible compared to the following compute-intensive transformer encoder (e.g., $<5\%$). Besides, an extra learnable classification token (CLS) is added to the token sequence to perform classification. After that, positional embeddings are added into each token, and tokens are fed to a standard transformer encoder. Depending on the specific downstream tasks, the hidden states generated by the transformer encoder are finally fed into different heads, such as classification and detection.

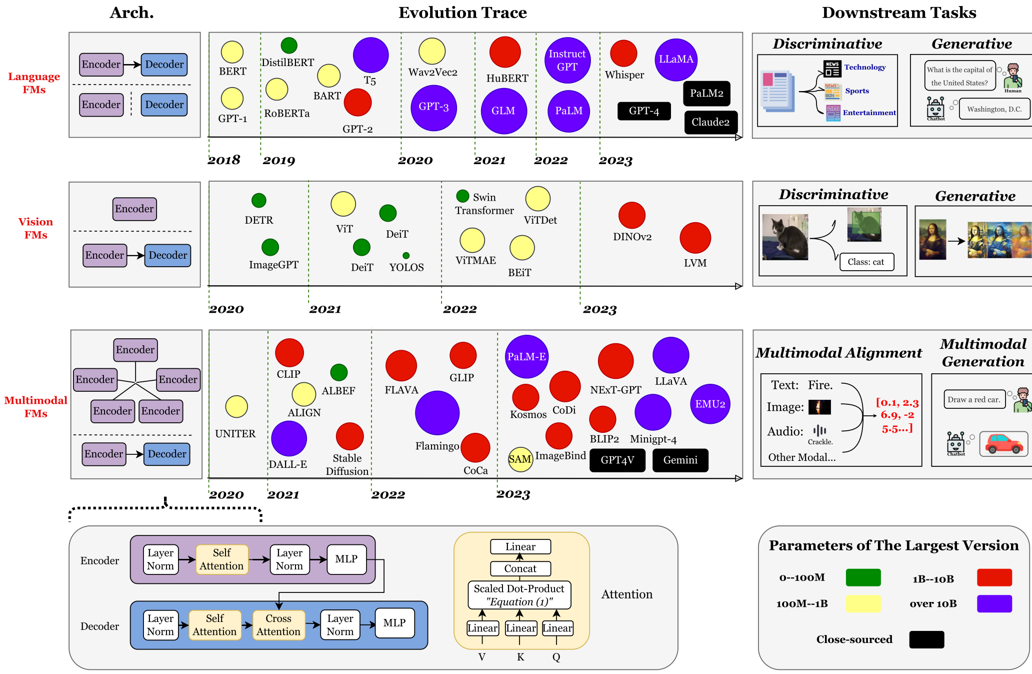


Fig. 1. The evolutionary trace of FMs.

Multimodal FMs. Multimodal FMs are used in two specific goals: encoding input data in different modalities into the same latent space, or generating output data in different modalities. The two lines of research have convergence—for example, multimodal-to-multimodal (or even any-to-any) generation. To ingest and align multimodal input data, existing model architectures like CLIP [212] typically consist of multiple transformer encoders, with each modality having its own set of transformer encoders. Notably, these encoders are generally trained from scratch, utilizing paired data with the aligned modalities and current modality. To generate multimodal data, FMs can either (1) reuse the LLM to generate text or (2) diffusion models [218] to generate high-quality image pixels. The diffusion module primarily consists of two components: an image encoder/decoder and a denoising network. There are also variants of diffusion model that replace the convolution with the transformer (e.g., DiTs) [204], as well as FMs [292] that involve richer modalities like IMU or audio. Yet such modalities are mainly embedded with only a dedicated embedding layer and reuse the same transformer architecture. Thereby, we do not discuss these models in isolation.

Applications of FMs. In real-world applications, language FMs like GPT-4 [196] have transformed tasks such as content generation [101], code assistance [142], and natural language understanding across multiple industries. These advancements enable chatbots and personal agents to better understand user queries and provide more meaningful responses. In the case of vision FMs, models such as SAM [125] are widely applied in medical imaging, allowing healthcare professionals to accurately segment and analyze images with minimal manual intervention, significantly improving diagnostic accuracy. Multimodal FMs, including CLIP [212] and Stable Diffusion [218], are transforming the creative industries by enabling artists to generate artwork from simple text prompts, thereby expanding creative possibilities while reducing manual effort.

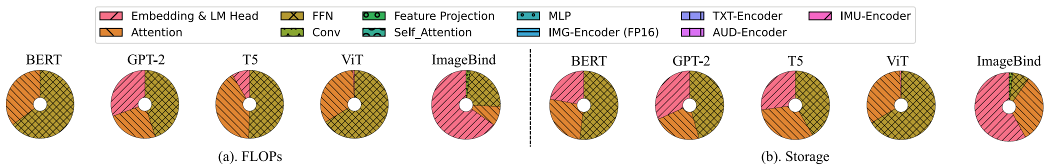


Fig. 2. Empirical computation and storage comparison across different FMs.

Cost Analysis of Transformer. Since most FMs are based on transformer architecture, we briefly analyze the resource cost of it. The attention mechanism in large FMs faces significant computational bottlenecks primarily due to its quadratic complexity. This complexity stems from calculating attention scores for every pair of positions within the input sequence, posing challenges in managing long sequences and impacting both training and inference efficiency. Additionally, beyond the attention mechanism, the computation complexity of the FFN scales linearly with input length but quadratically with the model's dimension. An increase in the length of the input sequence causes a substantial rise in computational demand, attributable to the quadratic nature of the attention mechanism. In quantitative terms, the computation complexity of attention is $O(T^2D)$, whereas that of the FFN is $O(TD^2)$, where T represents the sequence length and D the hidden state dimension of the model [159]. The decoder's attention mechanism, similar to that in the encoder, also experiences quadratic scaling with token length. This aspect becomes particularly significant in autoregressive decoding tasks, where each token's generation depends on the preceding ones, intensifying computational requirements. The implementation of a **key-value (KV)** cache in the decoder can substantially mitigate computational costs by reusing key and value vectors across various positions [132].

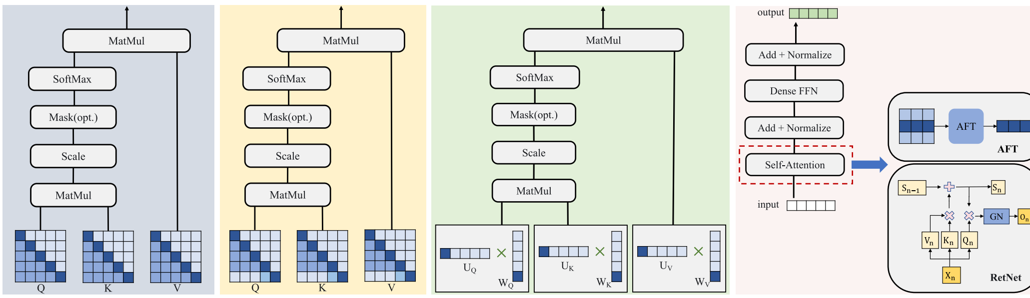
We empirically analyze the resource costs of different FMs by comparing their demands in terms of FLOPs and storage, as shown in Figure 2. For language models such as BERT, GPT-2, and T5, the embedding layer and LM head contribute significantly to storage. However, these components require minimal computational FLOPs. The FFN layer is the most computationally intensive component. Similar trends are observed in vision and speech models, such as Wav2Vec2 and ViTs, where convolution is not dominant. Instead, MLP and self-attention layers consume the most resources. In multimodal models like ImageBind, the IMG-Encoder is the most resource-demanding, whereas other encoders require significantly fewer resources.

3 Resource-Efficient Architectures

3.1 Efficient Attention

As summarized in Figure 3, numerous efforts have been invested to mitigate the huge resource cost of attention-based transformer architecture. The time and space complexity comparison is shown in Table 1.

3.1.1 Sparse Attention. Motivated by graph sparsification, sparse attention aims to build a sparse attention matrix. This approach aims to retain the empirical advantages of a fully quadratic self-attention scheme while employing a reduced number of inner products. For instance, Longformer [66], ETC [159], and BIGBIRD [294] decompose conventional attention into local windowed attention and task-specific global attention, effectively reducing self-attention complexity to linear. HEPOS [102] introduces head-wise positional strides, allowing each attention head to concentrate on a specific subset of the input sequence. MATE [58] transforms attention into a multi-view format, efficiently addressing either rows or columns in a table. TDANet [143] emulates



(a) Self-attention (b) Sparse attention (c) Approximate attention (d) Attention-free

Fig. 3. Illustrations of efficient attention architectures.

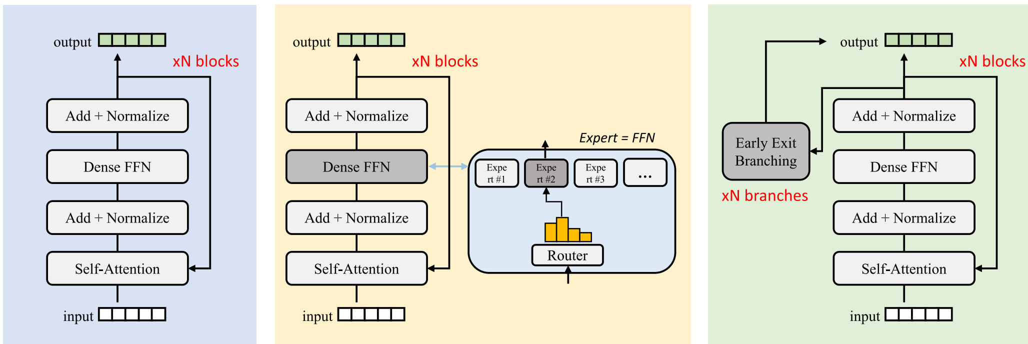
Table 1. Time and Space Complexity Comparison, Where T Represents Sequence Length and d Represents Hidden Dimension

Model	Time	Space	Model	Time	Space
Transformer [244]	$O(T^2d)$	$O(T^2 + Td)$	AFT [298]	$O(T^2d)$	$O(Td)$
Reformer [126]	$O(T \log Td)$	$O(T \log T + Td)$	Hyena [208]	$O(T \log Td)$	$O(Td)$
SSM [82]	$O(T \log Td)$	$O(Td)$	Linear Transformers [118]	$O(Td^2)$	$O(Td + d^2)$
RetNet [229]	$O(Td)$	$O(Td)$	RWKV [205]	$O(Td)$	$O(d)$

the human brain's top-down attention mechanism to selectively focus on the most relevant information, thereby enhancing speech separation efficiency.

3.1.2 Approximate Attention. Approximate attention mainly includes low-rank approximations of the self-attention matrix and innovative reformulations of the self-attention. Linformer [250] effectively decomposes the attention matrix into a low-rank matrix. It involves projecting the length dimensions of keys and values into a lower-dimensional space, resulting in a significant reduction in memory complexity. Reformer [126] utilizes locality-sensitive hashing to replace the conventional dot-product attention. Katharopoulos et al. [118] introduced a kernel-based alternative to self-attention, leveraging the associative property of matrix multiplication for computing self-attention weights. PolySketchFormer [116] employs polynomial functions and sketching techniques to approximate softmax attention outputs. Mega [178], featuring a single-head gated attention mechanism, incorporates exponential moving average. Deformable Attention [268] proposes a data-aware, deformable attention mechanism, contributing to improved performance within the ViT architecture. CrossViT [35] introduces linear cross-attention, empowering the ViT architecture to efficiently handle variably sized input tokens while mitigating computational costs.

3.1.3 Attention-Free Approaches. Despite the dominance of attention-based transformer architectures in large FMs, several works have put forth innovative architectures that hold the potential to replace the traditional transformer model. For instance, Hyena [208] introduces an architecture that interleaves implicitly parameterized long convolutions with data-controlled gating. This design provides a subquadratic alternative to attention in large-scale language models, thereby enhancing efficiency in processing long sequences. Another notable trend is the substitution of the attention mechanism with **state space models (SSMs)**, as explored in other works [44, 82, 197]. Mamba [81] seamlessly integrates selective SSMs into a streamlined neural network architecture, eliminating attention and MLP blocks. This model achieves a notable 5× speed increase over traditional transformers and exhibits linear scaling with sequence length. Recurrent-style transformers [26, 27] adopt a **recurrent neural network (RNN)**-based architecture, replacing attention



(a) The traditional Transformer (b) A typical MoE-version Transformer (c) A typical Early-Exiting Transformer

Fig. 4. Traditional and typical dynamic transformers.

with an RNN to achieve linear complexity. RWKV [205] combines the efficient parallelizable training of transformers with the effective inference capabilities of RNNs. RetNet [229] introduces an architecture that replaces multi-head attention with a multi-scale retention mechanism. During training, RetNet demonstrates 25% to 50% memory savings and a 7× acceleration compared to the standard transformer.

3.2 Dynamic Neural Network

3.2.1 Mixture of Experts. **Mixture-of-Experts (MoE)**, as illustrated in Figure 4(b), represents an efficient and sparse approach for training and deploying large FMs with extensive parameter sets. This model utilizes routed sparse parameters during inference. Switch Transformer [63] introduces a switch routing algorithm, leading to models with improved efficiency and reduced computational and communication costs. Switch Transformer demonstrates the scalability and effectiveness of the MoE framework by managing up to 1 trillion parameters, with as many as 2,048 experts. GLaM [57], a family of decoder-only language models, leverages a sparsely activated MoE design. V-MoE [217] presents a sparse adaptation of the ViT, scaling to 15 billion parameters, and achieves performance matching dense models while requiring less training time. LIMoE [188] represents the first multimodal model to incorporate sparse MoE, significantly outperforming CLIP in various tasks. Mistral AI introduces Mistral,[1] an MoE model comprising 8 experts, each with 7 billion parameters. This model outperforms the performance of the LLaMA2-70B model [238]. MoEfication [305] converts a model into its MoE variant with equivalent parameters. Sparse upcycling [127] initializes sparsely activated MoE from dense checkpoints, reducing about 50% of the original dense pre-training costs. FFF [23] divides the feed-forward layer into separate leaves instead of copying the entire feed-forward layer as an expert, being up to 220× faster than the original feed-forward layer with about 5% accuracy loss. Section 5.1 will detail systematic optimizations applied to MoE models.

3.2.2 Early Exiting. As illustrated in Figure 4(c), early-exiting optimization is a strategy that allows a model to terminate its computational process prematurely when it attains high confidence in the prediction or encounters resource constraints. He and Hofmann [93] investigate modifications to the standard transformer block, aiming for simpler yet efficient architectures without sacrificing performance. M4 [292] introduces a multi-path task execution framework, enabling elastic fine-tuning and execution of foundational model blocks for different training and inference

[1] https://mistral.ai/

tasks. FREE [20] proposes a shallow-deep module that synchronizes the decoding of the current token with previously processed early exit tokens. SkipDecode [49] is designed for batch inferencing and KV caching, overcoming previous limitations by establishing a unique exit point for each token in a batch at every sequence position. PABEE [315] enhances the efficiency of pre-trained language models by integrating internal classifiers at each layer. The inference process halts when predictions stabilize for a set number of steps, facilitating quicker predictions with reduced layer usage. DeeBERT [271] augments BERT's inference efficiency by incorporating early exit points. DeeBERT allows instances to terminate at intermediate layers based on confidence levels, effectively reducing computational demands and accelerating inference. Bakhtiarnia et al. [22] propose seven distinct architectural designs for early exit branches suitable for dynamic inference in ViT backbones. LGViT [274] presents an early-exiting framework tailored for general ViTs, featuring diverse exiting heads, such as local perception and global aggregation heads, to balance efficiency and accuracy. This approach achieves competitive performance with an approximate 1.8× speedup.

3.3 Diffusion-Specific Optimization

Generating images through diffusion models typically involves an iterative process with numerous denoising steps. Recent research has focused on accelerating the denoising process and reducing the resource requirements during image generation, which fall into three main categories: (1) efficient sampling, (2) diffusion in latent space, and (3) diffusion architecture variants.

3.3.1 Efficient Sampling. To enhance the denoising process of a diffusion model while maintaining or improving sample quality, many efforts have been made to improve the sampling process. These works emphasize resource and time efficiency in their architectures. Nichol and Dhariwal [192] made strides in enhancing the traditional DDPM by focusing on resource efficiency. Their improved model not only competes in log-likelihoods but also enhances sample quality. This efficiency is achieved by learning the variances of the reverse diffusion process and employing a hybrid training objective. This methodology requires fewer forward passes and shows improved scalability in terms of model capacity and computational power. DDIM [225] represents a significant improvement in time efficiency for diffusion models. By introducing a non-Markovian, deterministic approach to sampling, DDIM accelerates the generation process, allowing for faster sampling without compromising sample quality. PNDM [164] enhances the efficiency of DDPM in generating high-quality samples. The approach treats the diffusion process as solving differential equations on manifolds, greatly accelerating the inference process. DPM-Solver [175] utilizes a high-order solver that exploits the semi-linear structure of diffusion ODEs, facilitating fast and high-quality sample generation. Remarkably, DPM-Solver achieves this with as few as 10 to 20 denoising steps, highlighting the latency efficiency in sample generation.

3.3.2 Diffusion in Latent Space. In traditional diffusion models, operations are usually performed within the pixel space of images. However, this approach proves to be inefficient for high-resolution images because of the considerable computational demands and significant memory requirements. In response to these challenges, researchers proposed a shift toward conducting diffusion processes in latent space through VAEs. This paradigm results in substantial memory-efficient advancements, allowing for the generation of high-resolution images with reduced computational resources. LDM [218], also known as Stable Diffusion, serves as a notable example of memory-efficient image generation. By performing diffusion processes within a latent space derived from pixel data through a VAE, LDM effectively tackles scalability issues present in earlier diffusion models. LD-ZNet [207] leverages the memory-efficient properties of LDM for image segmentation tasks. This approach capitalizes on the deep semantic understanding inherent in LDM's internal features, providing a nuanced bridge between real and AI-generated

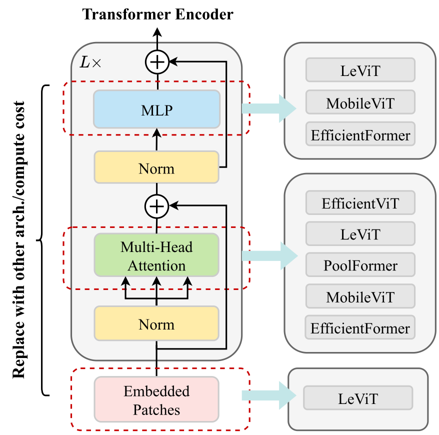


Fig. 5. A summary of resource-efficient VIT variants.

imagery. SALAD [130] introduces a memory-efficient methodology for 3D shape generation and manipulation with a cascaded diffusion model.

3.3.3 Diffusion Architecture Variants. Another method for enhancing diffusion models involves the adoption of more efficient model architectures. This strategy focuses on refining the structural framework of diffusion models to optimize their performance. SnapFusion [148] introduces an optimized text-to-image diffusion model for mobile devices, featuring a resource-efficient network architecture. This model overcomes the computational and latency limitations of existing models through a redesigned network architecture and improved step distillation. It generates high-quality 512×512 images in under 2 seconds with fewer denoising steps. ScaleCrafter [96] addresses the generation of ultra-high-resolution images using pre-trained diffusion models with an innovative and resource-efficient network design. ScaleCrafter incorporates techniques like "re-dilation," "dispersed convolution," and "noise-damped classifier-free guidance" to dynamically adjust convolutional perception fields during inference. ERNIE-ViLG [65] introduces a novel text-to-image diffusion model that integrates fine-grained textual and visual knowledge into a highly efficient network architecture. With a mixture-of-denoising-experts mechanism and scaling up to 24 billion parameters, ERNIE-ViLG outperforms the existing models on MS-COCO with a remarkable zero-shot FID-30k score of 6.75. Mobile diffusion [311] conducts a comprehensive examination of model architecture design to minimize model size and FLOPs. The authors also optimize the sampling steps, making one-step sampling compatible to downstream applications.

3.4 ViT-Specific Optimizations

As a transformer variant, ViT benefits from general optimizations aforementioned; yet, there also exist ViT-specific architecture optimizations as summarized in Figure 5. LeViT [80] is a hybrid neural network designed for efficient image classification. Its main backbone features a pyramid architecture, progressively reducing the dimensionality of features while concurrently increasing the number of attention heads. MobileViT [179] adheres to the idea of utilizing CNNs to construct a more lightweight transformer architecture. Through the design of a convolution-like MobileViT block, the model achieves a lightweight and low-latency implementation, specifically tailored for practical hardware platforms. EfficientFormer [153] designs a lightweight CNN-Transformer hybrid architecture, achieving more efficient on-device inference. EfficientViT [28] introduces a linear attention mechanism to alleviate the computational cost linked with the high overhead of softmax in non-linear attention. In the domain of super-resolution, EfficientViT achieves a speedup of up to 6.4 × compared to Restormer [295]. FastViT [242] introduces a token mixing operator that uses structural re-parameterization to lower the memory access cost by removing the skip-connections in the network. EfficientViT [167] identifies that the speed of existing transformer models is commonly bounded by memory-inefficient operations, especially the tensor reshaping and element-wise functions in MHSA. In response, the authors reduce the MHSA by a sandwiched structure. LightViT [103] presents several learning-based optimizations of pure convolution-free ViT architecture. EdgeViT [199] enables attention-based vision models to compete with the best lightweight CNNs in the tradeoff between accuracy and on-device efficiency.

4 Resource-Efficient Algorithms

This section focuses on resource-efficient large FMs techniques at the algorithm level. Compared to traditional DNNs, large FMs exhibit new characteristics such as their huge parameter set and autoregressive inference. This disparity has led to the emergence of numerous resource-efficient algorithms, which are categorized based on the lifecycle of FMs: pre-training, fine-tuning, serving algorithms, and model compression.

4.1 Pre-Training Algorithms

Pre-training for large FMs relies on a substantial amount of computation resources. For instance, GPT-3-175B consumes 3.14×10^{23} FLOPs and LLaMA-70B takes 1.7×10^{6} GPU hours. Consequently, optimizing the utilization of computational resources is crucial for the efficient pre-training of FMs. Resource-efficient algorithms can be categorized into training data deduction, neural architecture search, progressive learning, and mixed precision training.

4.1.1 Training Data Quality Control. A portion of work focus on controlling the quality of training data. DataComp [73] proposes a novel paradigm of locking the model/hyperparameters and refining the pre-training data. DFN [62] uses a proxy network as a modeling of the pre-training dataset. It recognizes that a better performance of the proxy network does not necessarily translate to the higher performance of the to-be-trained network. DataCompDR [243] of MobileCLIP leverages knowledge transfer from an image captioning model and an ensemble of strong CLIP encoders to improve the accuracy of efficient models.

4.1.2 Training Data Reduction. Pre-training for large FMs needs a dataset at the trillion scale, exemplified by 0.3 trillion tokens for GPT-3-175B [25] and 2 trillion tokens for LLaMa-2-70B [238]. More data indicates more resource expenditure. Thereby, prior literature resorts to reduce vast training data through two aspects: deduplicating text datasets and image patch removal.

Deduplicating text datasets [137] shows that training data has redundancy caused by near-duplicate examples and long repetitive substrings. The reduction of repetitions can lead to fewer training steps without compromising performance.

Image patch removal is achieved by either reducing the number of patch inputs to the model or reorganizing image tokens based on modified model architectures. For instance, TRIPS [112] employs a patch selection layer to reduce image patches. This layer computes attentive image tokens through text guidance, resulting in a 40% reduction in computation resources, compared to previous pre-training vision-language models. MAEs [94] mask image patches in the pre-training phrase, but the large masking ratio brings significant computation resource wastage. MixMAE [162] introduces a method for mixing multiple images at the patch level, thereby avoiding the need for introducing "[MASK]" symbols. COPA [113] introduces an auxiliary pre-training task called *patch-text alignment*. This patch-level alignment strategy aims to decrease redundancy in image patches. PatchDropout [170] introduces the concept of patch dropout to enhance both computation and memory efficiency. This method involves the random sampling of a subset of original image patches to effectively shorten the length of token sequences.

4.1.3 Progressive Learning. Progressive learning is a training strategy that begins by training a small model and then gradually increases the model size, throughout the training process. This approach optimizes computational resource usage by reusing the computations from the previous stage. Inspired by the insight that knowledge can be shared across models of different depths, StackingBERT [78] introduces a progressive stacking algorithm. This algorithm cost-effectively trains a large model with no performance degradation by sequentially stacking attention layers from smaller models. CompoundGrow [83] identifies the similarity between progressive training algorithms and NAS. Staged training [221] adopts a strategy where a small model is pre-trained initially, and subsequently the depth and width of the model are increased, continuing the training process. Knowledge inheritance [211] suggests employing existing pre-trained language models as teacher models to provide guidance during the training of larger models. The supplementary auxiliary supervision offered by the teacher model can effectively enhance the training speed of the larger model. The progressive training algorithm in AutoProg [141] is for the ViT. AutoProg automatically adjusts the growth schedule to achieve lossless performance and make training resource consumption minimal. LiGO [249] introduces small model parameters to initialize the large model through a trainable parameter linear map. LiGO achieves this by factorizing the growing transformation into a composition of linear operators at width and depth dimensions.

4.1.4 Mixed Precision Training. Mixed precision training often utilizes half-precision floating-point data representation instead of single precision. This approach significantly reduces memory requirements, approximately halving the storage space needed for weights, activations, and gradients. Mesa [201] proposes the combination of activation compressed training [31] with mixed precision training to further reduce the memory used by activations. The method quantifies activation based on the distribution of multi-head self-attention layers to minimize the approximation error. GACT [168] introduces a dynamically adjusted compression ratio based on the importance of each gradient.

4.2 Fine-Tuning Algorithms

Efficient fine-tuning algorithms are designed to reduce the workload to adapt a pre-trained FM to downstream tasks. As summarized in Figure 6, these techniques can be categorized into three groups: additive tuning, selective tuning, and re-parameter tuning.

4.2.1 Additive Tuning. Large FMs can achieve high performance with low costs by incorporating additional parameters and fine-tuning them for new tasks. In particular, this additive tuning process in large FMs can be categorized into three main classes: adapter tuning, prompt tuning, and prefix tuning.

Adapter tuning aims to reduce training costs by introducing adapter modules to specific layers (or all layers) of pre-trained large FMs. During tuning, the backbone of the pre-trained model remains frozen, and adapter modules are utilized to acquire task-specific knowledge. Some works [60, 200, 234] focus on designing adapters for multi-task or multimodal extensions. ADA [60] and MetaTroll [234] concentrate on incrementally extending pre-trained transformers' capabilities across multiple tasks. This approach helps alleviate catastrophic forgetting during learning while simultaneously reducing computational expenses. ST-Adapter [200] introduces built-in spatiotemporal reasoning abilities, allowing pre-trained models to significantly reduce the number of parameters that need to be updated in cross-modal tasks. HiWi [156] improves inference speed by applying adapters to pre-trained parameters rather than hidden representations. AdaMix [255]

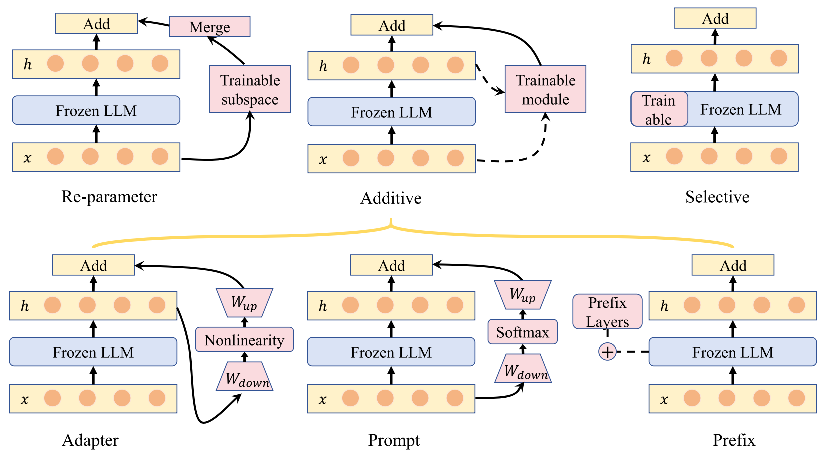


Fig. 6. A summary of various fine-tuning algorithms.

designs a combined mechanism that merges the weights of different adapters into a single adapter at each transformer layer. This innovation significantly reduces the additional storage cost introduced by multiple adapters. MEFT [157] designs a method for inserting adapters into the LLM by modifying the LLM to its reversible variant, reducing activation memory and thus improving the memory efficiency of fine-tuning. Residual Adapters [236] utilizes personalized residual adapters to address the issue of performance degradation in automatic speech recognition caused by non-standard speech. AutoProg [141] achieves lossless acceleration by automatically increasing the training overload on-the-fly. Such a procedure is done by progressively growth of subnets.

Prompt tuning involves designing a task-specific prompt for each task, with the aim of replacing the traditional fine-tuning of pre-trained large FMs parameters. By tuning the input prompts instead, this method significantly reduces the resources and time required for the fine-tuning. Some works [17, 138, 239] focus on improving the efficient scalability of prompts in multi-task settings. For example, PromptTuning [138], ATTEMPT [17], and BioInstruct [239] investigate how the utilization of mixed soft prompts can efficiently transfer knowledge across different tasks. These approaches help mitigate parameter update costs by reusing the frozen pre-trained large model. Furthermore, some works [36, 284] focus on minimizing prompt fine-tuning costs for specific tasks. For instance, DualPL [284] designs two prompts and separately captures the relevant knowledge of both tasks. This approach addresses the high cost associated with collecting state labels for slots and values in dialogue state tracking systems. In machine reading comprehension tasks, MPrompt [36] introduces task-specific multi-level prompt tuning to enhance the understanding of input semantics at different granularities while reducing the number of parameter updates.

Prefix tuning introduces a trainable, task-specific prefix part to each layer of large FMs. This technique aims to reduce the tuning cost by limiting the updates to the parameters in this prefix. Some works [160, 189, 245, 252, 308] focus on enhancing the performance of prefix tuning in specific domains. For example, UAPT [252] and Prefix-diffusion [160] address the issue of limited diversity in generating captions for images. These approaches extract image features from large FMs and design prefixes to enhance performance while reducing additional overhead. DOP [308] and DAPA [189] concentrate on domain-generalization problems in abstract summarization. These approaches design prefixes for each source domain to improve the model's generalization capabilities. PIP [245] focuses on syntactic control in paraphrase generation and reduces training costs by designing parsing-indicating prefixes.

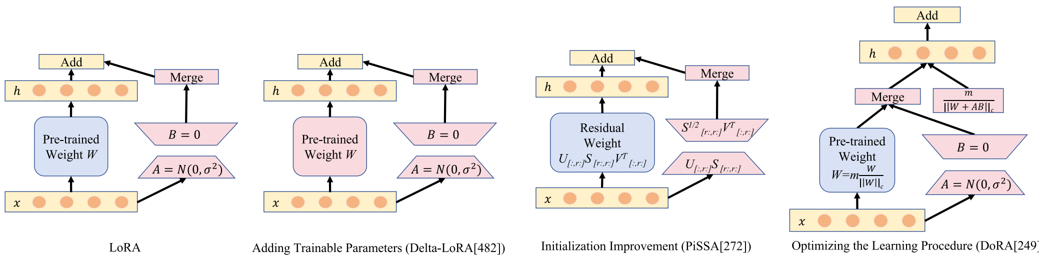


Fig. 7. LoRA and its optimization methods.

4.2.2 Selective Tuning. Selective tuning aims to maintain high performance on new tasks with low training costs by freezing the majority of parameters in large FMs and selectively updating only a small portion of the parameters. Some works focus on optimizing the performance of selective tuning. For example, SAM [72] explores how the choice of tunable parameters affects tuning. By proposing a second-order approximation method, it tunes fewer parameters to achieve better model performance. SmartFRZ [146] focuses on improving the efficiency of layer freezing by introducing an adaptive layer freezing technique based on different network structures. This innovation enhances system accuracy and training speed. FiSH-DiP [46] explores the effectiveness of tuning with limited data by introducing a sample-aware dynamic sparse tuning strategy. This approach selectively tunes partial parameters using sample feedback to enhance the model's generalization in resource-constrained situations. Token Mixing [171] and VL-PET [100] enhance fine-tuning efficiency of visual-language tasks by adjusting and selecting a subset of trainable parameters.

4.2.3 Re-parameter Tuning. Re-parameter tuning adapts large FMs by targeting a significantly smaller subspace than the original, expansive training space. This approach involves fine-tuning low-rank matrix parameters, a technique that effectively reduces the overall training cost. The majority of existing research centers on re-parameterization tuning through the implementation of the low-rank adapter design. For example, EfficientDM [95], QLoRA [51], PEQA [121], QALoRA [278], and LoftQ [151] incorporate quantization techniques, building upon the foundation of LoRA. GLoRA [32] enhances LoRA's generality, improving model transferability, few-shot capabilities, and domain generalization. PELA [87] derives inspiration from LoRA and devises a low-rank approximation compression method. LongLoRA [38] extends the capabilities of LoRA by incorporating context expansion through shift short attention. For ViT's linear layers, LBP-WHT [283] diminishes the computational costs of matrix multiplication by employing low-rank backward propagation based on the Walsh-Hadamard transform. Additionally, DSEE [37] investigates the application of sparse-aware low-rank updates on pre-trained model weights. Dynamic-Pooling [191] mechanisms are designed to predict inference boundaries through autoregressive prediction.

LoRA, as the most popular parameter-efficient fine-tuning method, still exhibits performance gaps when compared to full fine-tuning. To address this, various methods have been developed to enhance LoRA's performance, as shown in Figure 7. Delta-LoRA [318] aims to bridge the performance gap by updating the pre-trained weights through the product of low-rank matrices A and B, thus adding trainable parameters without incurring additional memory overhead. However, PiSSA [180] identifies an issue where LoRA initializes low-rank matrices with Gaussian random values and zeros, resulting in very small initial gradient values and slow convergence. Last, DoRA [166] and LoRA+ [92] focus on enhancing the learning process itself to further improve efficiency and effectiveness. DoRA decomposes the pre-trained weights into their magnitude and directional components, and fine-tunes the directional matrix. LoRA+ sets the unbalanced learning rate for different blocks, accelerating convergence and improving fine-tuning performance.

4.3 Inference Algorithms

4.3.1 Opportunistic Decoding. The autoregressive mechanism significantly hinders the inference efficiency of large FMs. To address this, various approaches aim to replace autoregressive decoding with more efficient non-autoregressive techniques. Speculative decoding has been widely acknowledged as an effective method to accelerate autoregressive decoding. It involves generating sequences autoregressively with a cost-efficient small model, followed by parallel token verification using a larger model. Leviathan et al. [139] report a 2 to 3× improvement in performance using speculative decoding on the T5X model, whereas a concurrent study [34] demonstrates similar speedups on a 70B Chinchilla model. SpecTr [230] further enhances speculative decoding by increasing the number of candidate tokens and improving the draft selection process, resulting in a 2.13× improvement in wall clock speed and an additional 1.37× speedup on standard benchmarks. ProphetNet [280] introduces a sequence modeling architecture that predicts future tokens, partially reducing the reliance on autoregression. In the draft stage, Draft & Verify [300] skips certain intermediate layers, achieving a 1.73× speedup when tested on Llama-2. Medusa [29] offers another non-autoregressive decoding architecture that requires no auxiliary model, predicting multiple tokens by pre-training heads for different timesteps and verifying them concurrently. Look-ahead decoding [70] accelerates inference in large FMs without relying on a draft model or data store, reducing decoding steps in proportion to *log*(FLOPs). Additionally, speculative decoding is the foundation for various inference systems, such as SpecInfer [183], which uses multiple draft models in the cloud, and LLMCad [272], deployed at the edge.

4.3.2 Input Filtering and Compression. This method includes directly filtering raw data (i.e., prompt filtering) or filtering hidden activations of FMs (i.e., token pruning).

Prompt Compression. Computations can be effectively reduced by compressing the prompt to the model. LLMLingua [114] introduces a prompt compression approach from a coarse-to-fine perspective. Wingate et al. [262] investigate the feasibility, applicability, and potential of compressing natural language for large FMs while preserving semantics. EntropyRank [240] presents an unsupervised approach for extracting keywords and keyphrases from textual data. This method leverages a pre-trained language large FM and incorporates Shannon's information maximization. LLMZip [241] employs LLaMA-7B for compressing natural language. Experimental results demonstrate that LLMZip outperforms cutting-edge text compression methods, including BSC, ZPAQ, and paq8h. AutoCompressors [40] utilizes large FMs to compress natural language into compact summary vectors. These vectors can then serve as soft prompts for large FM usage. ICAE [75] utilizes the capabilities of large FMs to condense an extensive context into concise memory slots. These memory slots are directly adaptable by the large FMs for diverse purposes. Nugget 2D [210] introduces a prompt compression method specifically designed to handle long contexts. CoT-Max [104] is a context pruner, aiming to enhance the **Chain-of-Thought (CoT)** ability of large FMs.

Token Pruning. Research has also explored the pruning of input sequences for transformers, often involving the incremental removal of less important tokens during inference. PoWER-BERT [79] proposes the direct learning of token pruning configurations. Length-Adaptive Transformer [120] extends this idea by introducing LengthDrop, a technique that entails training the model with various token pruning configurations, followed by an evolutionary search. TR-BERT [287] formulates token pruning as a multi-step token selection problem and addresses it through reinforcement learning. DynamicViT [214] hierarchically prunes redundant tokens based on their importance scores. AdaViT [181] and A-Vit [290] employ adaptive token reduction mechanisms and select different tokens for different images. AdaViT dynamically determines the usage of patches, self-attention heads, and transformer blocks based on the input. A-ViT discards tokens in ViTs during inference, adapting the token retention based on the complexity of the input images. SPViT [129] devises an adaptive instance-wise token selector and introduces a soft pruning technique. PuMer [30] combines similar textual and visual tokens during inference for large-scale vision-language models.

4.3.3 KV Cache. Optimizing memory for the KV cache is a crucial aspect of the autoregressive decoder-based model inference process.

Memory-Efficient Sparse Attention. An alternative approach involves leveraging sparse attention. However, it is noteworthy that most sparse attention designs, which primarily target the reduction of computational complexity [24, 294], do not necessarily lead to a reduction in KV cache memory consumption. This is because achieving a reduced memory footprint for the KV cache necessitates a more stringent sparsity pattern. Specifically, tokens that are sparsified should not be dynamically accessed in subsequent steps. To address this, H2O [306] introduces a KV cache eviction strategy designed for optimal memory efficiency. This strategy employs attention scores to identify and select the least important KV cache tokens in the current state for eviction. When compared to robust baselines, H2O demonstrates the capability to reduce latency by up to 1.9× and increase throughput by 29×. Dynamic Context Pruning [16] learns a memory-efficient KV cache eviction strategy during the pre-training phase. This approach has demonstrated the ability to achieve up to a 2× increase in inference throughput and even greater memory savings. Scissorhands [172] utilizes an innovative compact KV cache and results in a notable reduction in KV cache inference memory usage, achieving up to a 5× reduction while maintaining model quality. By employing a landmark token to demarcate a token block, Landmark Attention [187] optimizes KV cache storage. This approach enables the storage of most KV caches in a slower but larger capacity memory, resulting in reduced memory requirements without compromising performance.

4.3.4 Long Context. To effectively process long sequences, transformers need to adapt their positional encoding to enhance their capability to capture long-range information. Due to the quadratic computational cost associated with attention mechanisms, various resource-efficient optimizations have been proposed to handle long inputs. LM-Infinite [89] introduces a Λ-shaped attention mechanism to handle long contexts efficiently. Characterized by computational efficiency with O(n) time and space complexity, LM-Infinite consistently demonstrates fluency and quality in text generation for sequences as long as 128k tokens on arXiv and OpenWebText2 datasets. StreamingLLM [270] facilitates large FMs trained with a finite-length attention window to generalize to infinite stream decoding without the need for any fine-tuning. PCW [215] segments a long context into chunks or "windows," constrains the attention mechanism to operate solely within each window, and reuses positional embeddings across the windows. LongNet [53] introduces dilated attention, expanding the attentive field exponentially as the distance increases. This innovation allows LongNet to scale transformers efficiently, enabling them to handle sequences of up to 1 billion tokens. SLED [108], short for SLiding-Encoder and Decoder, repurposes and capitalizes on well-validated short-text pre-trained language models. Despite competing effectively with specialized models that are up to 50× larger, SLED does not require a dedicated and expensive pre-training step.

4.4 Model Compression

As summarized in Figure 8, model compression refers to a set of techniques aimed at reducing the model size without significant performance degradation, categorized into pruning, **knowledge**

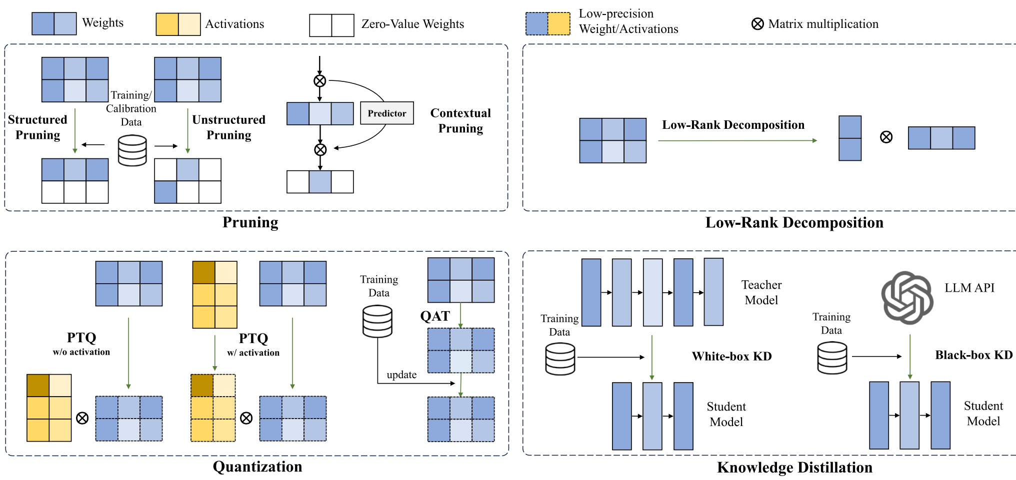


Fig. 8. Model compression techniques for LLMs.

Table 2. Model Compression Methods and Their Unique Challenges

Method	Categories	Unique Challenge
Pruning	Structured Pruning [177, 267, 301], Unstructured Pruning [68, 224, 228], Contextual Pruning [174, 227]	Massive re-pre-training, Unique transformer structures
Knowledge Distillation	White-Box KD [39, 186], Black-Box KD [84, 235]	Massive re-pre-training
Quantization	Quantization-Aware Training [173, 232], Post-Training Quantization [50, 67]	Quantization outliers, Per-tensor quantization
Low-Rank Decomposition	[119, 152, 276]	/

distillation (KD), quantization, and **low-rank decomposition (LoRD)**. While compression has been extensively studied in pre-LLM era [90, 91], compressing FMs faces unique challenges such as weight outliers and extensive training efforts, as presented in Table 2.

4.4.1 Pruning. The pruning technique removes redundant or non-essential connections, neurons, or layers from a neural network. The primary objective is to reduce the model size, subsequently decreasing computational and storage costs, while maintaining model accuracy. Structured pruning and unstructured pruning target weight reduction without modifying sparsity during inference. In contrast, contextual pruning dynamically selects activated neurons or layers during inference based on the sparsity of the model.

Structured pruning compresses large foundational models by eliminating entire structural components, such as groups of consecutive parameters or hierarchical structures. Examples of these structural components include channels or blocks of the model's weights. It is often combined with fine-tuning to mitigate accuracy loss. LLM-Pruner [177] is a task-agnostic structured pruning algorithm that utilizes a small amount of data to assess the importance of coupled structure weights. The method selectively removes non-essential model structures based on gradient information. LLM-Pruner incorporates LoRA to recover the model's accuracy after pruning. LoRAPrune [301] is another structured pruning approach based on LoRA, leveraging LoRA's weights and gradients for importance estimation. This method iteratively eliminates excess channels and attention heads, achieving superior results compared to LLM-Pruner. Lagunas et al. [133] improved structured pruning techniques by incorporating blocks of variable sizes. This integration is applied within the movement pruning framework during fine-tuning, resulting in the removal of entire model components, such as attention heads. It achieves a 2.4× speedup and is 74% smaller compared to the original BERT.

Structured pruning is also employed in the training of large foundational models as well. Sheared LLaMA [267] adopts an end-to-end approach to remove channels, encompassing layers, attention heads, intermediate layers, and hidden layers. Sheared LLaMA demonstrates the capability to prune the LLaMA2-7B model down to 1.3 billion parameters. AdaPrune[106] accelerates neural network training using transposable masks, resulting in a 2× speedup in matrix multiplications during both inference and training. GUM [220] considers neuron specificity and introduces pruning through network component-based global mobility and local uniqueness scores. This approach aims to simultaneously maximize sensitivity and uniqueness, effectively reducing redundant parameters in large FM weights. PLATON [303] tackles the uncertainty in importance scores during model pruning by employing the upper confidence bound of importance estimation. This approach ensures stability in training and leads to improved generalization.

Unstructured pruning does not consider the inherent structure of the model. Typically, it removes neurons with weights below a threshold, thereby compressing the model. When deploying unstructured pruning, specialized techniques are required to implement model storage compression. SparseGPT [68] treats the pruning framework as a generalized sparse regression problem and employs an approximate sparse regression solver, achieving 60% unstructured pruning on large GPT models like 175B. Wanda [228] leverages the observation of emergent large-magnitude features in large FMs. Wanda introduces sparsity by pruning weights with the smallest magnitudes multiplied by corresponding input activations, on a per-output basis. UPop [224] serves as a universal vision-language transformer compression framework, which incorporates unifiedly multimodal subnets and progressively searching/retraining. SIGE [224] is proposed to convert computation reduction into latency reduction on standard hardware, achieving notable accelerations for models like DDPM, Stable Diffusion, and GauGAN with minimal edits.

Contextual pruning selects the sparse state of each layer, making it hardware-optimization friendly. Deja Vu [174] dynamically predicts the sparsity of the next layer using the activations of the previous layer. It determines which neurons of MLP blocks and the heads of attention blocks need to be retained. To mitigate the overhead of this predictor, Deja Vu asynchronously predicts the next layer. PowerInfer [227] utilizes the sparsity of activation to dynamically predict the hot-activated neurons of the next layer and computes them on the GPU, whereas other cold-activated neurons are computed on the CPU. In comparison to llama.cpp [76], PowerInfer achieves up to 11× acceleration, enabling the 40B model to output 10 tokens per second on a personal computer.

4.4.2 Knowledge Distillation. KD transfers knowledge from a complex, heavy model (i.e., teacher model) to a simpler corresponding model (i.e., student model) for model compression. In general, there are two ways to apply KD to large FMs based on whether the internal structure of the teacher model is considered: white-box KD and black-box KD.

Black-Box KD. Assuming that the internal structure of the teacher's large base model is not visible, this approach fine-tunes the student model using prompt-response pairs generated by large FMs' API. The goal is to imbue the student model with the capabilities of the teacher model. For large FMs, the insights gained due to the increased parameter count contribute to strong generalization abilities. Therefore, techniques such as **In-Context Learning (ICL)** [55] and CoT [257] can be utilized to enable the student model to thoroughly learn the capabilities of the large FMs. ICL distillation transfers few-shot learning and language model capabilities from the teacher model to the student model by integrating ICL objectives with traditional language modeling objectives. In Meta-ICL [186] and Metal-ICL [39], language models undergo meta-training on diverse tasks using ICL objectives. This process enables them to fine-tune for unseen tasks through ICL. Multitask-ICT [105] introduces the concept of ICL distillation, fine-tuning models with ICL objectives and examples from target tasks. CoT introduces intermediate reasoning steps in prompts, guiding language models to solve complex reasoning tasks step by step. Fu et al. [71] enhance the mathematical reasoning capabilities of smaller models by instructing them through CoT distilled from LLM teachers. Distilling step-by-step [99] extracts rationales from large FMs using CoT in a multi-task framework, providing additional guidance for training smaller models in a multi-task environment. Fine-tune-CoT [97] uses zero-shot CoT prompting techniques, employing random sampling to generate multiple reasoning solutions from large FMs to guide the training of student models.

White-Box KD. In contrast to black-box KD, white-box KD not only has access to the output results of the teacher model but also to its structure and intermediate results. Therefore, white-box KD can better leverage the structure of the teacher model, enabling smaller student models to replicate and learn the capabilities of larger teacher models.

Timiryasov Tastet [235] train an ensemble consisting of a GPT-2 and small LLaMA models on the developmentally plausible BabyLM dataset. Subsequently, they distilled it into a small LLaMA model with 58 million parameters, surpassing in performance both of its teachers as well as a similar model trained without distillation. MiniLLM [84] distills smaller language models from generative larger language models. This approach replaces the forward KLD (Kullback-Leibler Divergence) objective in the standard KD approaches with reverse KLD, which is more suitable for KD on generative language models, to prevent the student model from overestimating the low-probability regions of the teacher distribution. Instead of solely relying on a fixed set of output sequences, GKD [11] trains the student model using self-generated output sequences. TED [155] employs task-aware filters to align the hidden representations of the student and the teacher at each layer. These filters are designed to select task-relevant knowledge from the hidden representations.

4.4.3 Quantization. Quantization is a well-established model compression method to mitigate the storage and computational demands. Compared to traditional DNNs, LLMs exhibit a higher frequency of activation outliers, which are crucial for maintaining model accuracy. Standard quantization often removes these outliers, leading to a significant performance drop.

Quantization-aware training (QAT) involves training a quantized model in such a way that it adapts its parameters to the lower precision introduced by quantization. The primary objective of this process is to mitigate the accuracy loss that occurs as a result of quantization. LLM-QAT tackles the issue of obtaining training data for LLMs by leveraging pre-trained models to generate samples through data-free distillation. Concurrently, it quantizes weights, activations, and KV cache, thereby improving training throughput. QuantGPT [232] achieves this by incorporating contrastive distillation from a full-precision teacher model and distilling logit information to a quantized student model during autoregressive pre-training. BitNet [247] pioneers QAT for 1-bit language models, training the language model with 1-bit weights and activations. Due to the substantial parameter count in large models often reaching tens or hundreds of billions, the training cost of QAT remains considerable. On the one hand, QAT for large FMs is often combined with KD to reduce the training cost, as seen in approaches such as LLM-QAT and QuantGPT. On the other hand, quantization is frequently employed in the fine-tuning process of large models, such as in PEQA [266] and QLoRA [51].

Post-training quantization (PTQ) converts a trained full-precision model to a low-precision model without retraining. The advantage of PTQ lies in compressing models without altering the model structure or necessitating retraining, thereby reducing the storage and computational costs of models. Due to its low deployment cost, PTQ is also the most easily deployable and widely applicable technique in model compression. However, unlike QAT and distillation, PTQ lacks the feedback loop for adjusting precision through training. Research related to PTQ often focuses on efficiently preserving relevant information in weights/activations while compressing models. PTQ can be categorized into two groups: weight-only quantization and weight-activation co-quantization.

Weight-only Quantization. Weight-only quantization only quantizes the model weights. There are two primary methods for mitigating quantization errors in the weight quantization of large FMs.

The first category involves identifying outliers and important weights in weights that significantly contribute to accuracy and treating these outliers specially. For instance, SpQR [52] identifies outlier weights and maintains them with high precision while quantizing the rest of the weights. LLM.int8() [50] employs vectorized quantization and mixed-precision decomposition to handle outlier values for efficient inference. LLM.int8() utilizes 8-bit quantization for matrix multiplication, effectively reducing GPU memory usage during inference. AWQ [158] reduces quantization error by protecting the top 1% important weights in the model, utilizing per-channel scaling to determine the optimal scaling factor. OWQ [135] analysis suggests that abnormal activations amplify quantization errors, and it employs a mixed-precision scheme, applying higher-precision quantization to weights with a significant impact from activated outlier values. SqueezeLLM [122] observes that sensitive weights determine the final model's quantization performance and proposes a non-uniform quantization approach to minimize quantization errors in these sensitive weights.

The second category of quantization reduction methods is based on the second-order information updated weights. GPTQ [69] employs layer-wise quantization with OBQ [67], utilizing inverse Hessian information to update weights. GPTQ reduces the bit-width of each weight to 3 or 4 bits, allowing quantization of GPT models with 175 billion parameters with minimal accuracy loss. QuIP [33] uses an adaptive rounding process, minimizing a second-order proxy objective for quantization.

Weights-Activation Co-quantization. Quantizing both weights and activation facilitates deployment on hardware accelerators. SmoothQuant [269] takes advantage of the similarity in the channel-wise activations of different tokens and performs quantization on both weight and activation using per-channel scaling transforms. RPTQ [293] recognizes the substantial range differences across different channels, reordering the channels for quantization and integrating them into layer normalization and linear layer weights. OliVe [85] adopts outlier-victim pair quantization and locally processes outliers. Outlier Suppression+ [258] builds upon Outlier Suppression [259], discovering that harmful outliers exhibit an asymmetric distribution mainly concentrated in specific channels. Considering the asymmetry of outliers and quantization errors from the weights of the next layer, this approach performs channel-level translation and scaling operations. QLLM [161] addresses the issue of activation outliers through an adaptive channel reassembly method and mitigates the information loss caused by quantization using calibration data. LLM-FP4 [285] quantizes weights into 4-bit float points, proposes per-channel activation quantization, and reparameters additional scaling factors as exponential biases of weights. ZeroQuant [286] combines layer-wise KD and optimized quantization support to achieve 8-bit quantization. FlexRound [136] updates the quantization scale of weights and activations by minimizing the error between the quantized values and the full-precision values. ATOM [310] significantly boosts serving throughput by using low-bit operators and considerably reduces memory consumption via low-bit quantization.

There is also extensive quantization research for backbone networks in FMs like ViT and BERT. For instance, BinaryBERT [195] and I-BERT [21] have achieved higher accuracy for BERT under low-precision quantization. Wang et al. [253] exploit the operator fusion [194], PTQ techniques, and structured pruning [133] to reduce the memory cost. They also reduce the number of computation operations of DeiT-Tiny [237]. Q-ViT [150], I-ViT [154], and OFQ [165] also achieve high

Table 3. Popular Open Source Tools for Training and Deploying Large FMs

Name	Descriptions	Tags	Name	Descriptions	Tags
DeepSpeed [1]	*An open-sourced Python library proposed by Microsoft. Supports MoE, long-sequence training, RLHF, ZeRO optimizations,and model compression.*	C/T/I	Megatron [190]	*The first cloud training system that introduces tensor parallelism to distributed training models like GPT, BERT, and T5. It is proposed by NVIDIA.*	C/T
Alpa [312]	*An automatic FM parallelization engine from UCB.*	C/T/I	FairScale [61]	*A new scaling library from Meta.*	C/T/I
Colossal AI [144]	*From HPC-AI Tech. Supports common parallelism strategies and heterogeneous memory management.*	C/T/I	FlexFlow [183]	*A cloud FM training and serving compiler from CMU and Stanford University. Automatic parallelization.*	C/T/I
PyTorch FSDP [309]	*A cloud large-scale training system atop PyTorch. It shards parameters, optimizer states, and gradients.*	C/T/I	HF PEFT [2]	*An efficient fine-tuning system from HuggingFace. It supports a set of PEFT methods like LoRA and p-tuning.*	C/T
MII [1]	*A library from DeepSpeed. Supports FastGen.*	C/I	vLLM [132]	*A serving engine from UC Berkeley. PagedAttention.*	C/I
LightLLM [4]	*A framework for token-wise's KV cache management.*	C/I	Ray LLM [9]	*A multiple LLMs serving solution from Anyscale.*	C/I
TGI [3]	*A high-performance serving engines from HuggingFace. It supports tensor parallelism, quantization with bitsandbytes and GPT-Q, and PagedAttention.*	C/I	TRT-LLM [8]	*A TensorRT toolbox for optimized LLM inference. It supports AWQ, GPTQ, SmoothQuant, speculative decoding, pipeline/tensor parallelism, and PagedAttention.*	C/I
llama.cpp [76]	*A popular on-device LLM serving engine supporting mixed F16/F32 precision and 2/3/4/5/6/8-bits int quantization. Mainly for LLaMA-based LLMs.*	C/E/I	MNN-LLM [7]	*An edge LLMs serving engine proposed by Alibaba and inherited from MNN. It optimizes the inference procedure separately in the prefill/decoding phase.*	E/I
mllm [6]	*A versatile and efficient on-device multimodal engine.*	E/I	MLC-LLM [5]	*Natively deploy LLMs with compiler-accelerated APIs.*	C/E/I

C, cloud; E, edge; T, training; I, inference.

accuracy for ViT under low-precision quantization. Q-Diffusion [147] compresses the noise estimation network to expedite the generation process of diffusion models.

4.4.4 Low-Rank Decomposition (LoRD). LoRD approximates the weight matrix in large FMs by decomposing a given weight matrix into two or more smaller matrices. LoRD has been widely applied in large FM fine-tuning methods like LoRA. LoRD has also shown substantial compression capabilities with minimal impact on performance, highlighting its potential for large FM compression [119]. To reduce the dimensionality of high-dimensional token embeddings underpinning large FMs, TensorGPT [276] proposes an approach based on the tensor-train decomposition, where each token embedding is treated as a matrix product state that can be efficiently computed in a distributed manner. Through TensorGPT, the embedding layer can be compressed by a factor of up to 38.40×. LoSparse [152] employs low-rank approximation to compress the coherent and expressive elements. The method uses iterative training to assess the significance scores of column neurons for the pruning process, showcasing superior performance compared to traditional iterative pruning techniques. Saha et al. [219] compress matrices through randomized low-rank and low-precision factorization, achieving compression ratios as aggressive as 1 bit per matrix coordinate while surpassing or maintaining the performance of traditional compression techniques. ViTALiTy [47] is an algorithm-hardware co-designed framework to enhance the inference efficiency of ViTs. It achieves approximation of the dot-product softmax operation with first-order Taylor attention, utilizing row-mean centering as the low-rank component to linearize the cost of attention blocks.

5 Resource-Efficient Systems

Training and serving systems are key to practical large FMs. This section investigates the system research to enable resource-efficient large FMs, notable in four aspects: (1) distributed training, (2) hardware-aware optimizations, (3) serving in cloud, and (4) serving in edge. Table 3 summarizes widely used open source frameworks in this domain.

5.1 Distributed Training

Distributed training systems serve as the foundation for training large FMs, encompassing pre-training and fine-tuning phases. Pre-training, involving intensive computation and communication, demands substantial resources compared to other large FM processes. Fine-tuning is widely used to transform a general-purpose model into a specialized model for particular use cases.

Considering the large scale and new execution pattern of large FMs, designing resource-efficient systems for FMs has drawn great attention from the community. We categorize techniques for optimizing distributed training systems, covering aspects such as resilience, parallelism, communication, storage, and heterogeneous GPUs. Additionally, MoE has emerged as a trend in training extremely large models, for which several approaches are tailored. These specialized methods are detailed at the end of this subsection.

Resilience. The increasing size and duration of training for large FMs have led to a rise in failures, emphasizing the importance of resilient training [261]. Fault tolerance approaches for large FMs primarily manifest in four forms. First, Varuna and Gemini [18, 256] facilitate resilient training by implementing checkpoints to restart training. Varuna [18] is designed for training in commodity clusters with low-bandwidth networks, frequent pre-emptions, and user-friendly features. However, Gemini [256] expedites failure recovery through in-memory checkpoints. Second, Bamboo [233] utilizes redundant computations where one node performs computations for both itself and its neighbors. Bamboo avoids the overhead of recovering but introduces the overhead during training. Third, activation checkpointing [131, 307], which avoids storing the activation and recomputes it when needed, falls between the checkpointing and redundant computation approaches. The fourth approach involves recovering partial layers, as demonstrated by Oobleck [109]. In the event of a failure, the affected pipeline can be restored using partial layers from other replicas, incurring less overhead than employing the entire checkpoint.

Parallelism. Parallelism plays a crucial role in distributed training, especially for large FMs. Three types of parallelism are commonly employed for training large FMs. *Data parallelism* involves distributing the data across workers to scale up distributed training. DeepSpeed ZeRO [213] optimizes memory usage by splitting the model states. *Model parallelism* partitions the model in intra-layer paradigm (tensor parallelism [190]) or inter-layer paradigm (pipeline parallelism [134, 198]). Tensor parallelism improves the training speed while leading to more communication. Pipeline parallelism improves GPU utilization by filling the bubbles. Breadth-first pipeline parallelism [134] designs a looping placement and breadth-first schedule to achieve both high GPU utilization and low cost. PipeFisher [198] assigns extra work to the bubbles for further benefits. Mobius [64] is designed for fine-tuning with a novel pipeline parallelism scheme and heterogeneous memory. FTPipe [59] partitions the model into finer-grained blocks rather than layers for flexible execution and low resource demand. *Sequence parallelism* [131, 145] is designed for the trend of long sequence training where training one sentence exceeds the memory capacity of one worker. Sequence parallelism divides the long sequence into multiple chunks and puts them on different workers. In practice, these parallelisms are usually used in a hybrid way. Galvatron [185] can automatically determine the most efficient hybrid parallelism strategy.

Communication. The large scale and complex parallelism lead to significant communication overhead. We summarize the optimization of communication into two categories: reducing the communication time directly and hiding the communication. Some work explores parallelism-aware communication compression [226] and heterogeneity-aware traffic reduction [307]. Existing work usually overlaps the communication with computation, by unifying the abstraction of computation and communication [110], decomposing the original communication collective [251], or designing a novel pipelining schedule [317].

Storage. Large FMs require a significant amount of storage resources, such as GPU memory for model states, host memory for model analysis, and disk for dataset and checkpoint. Various approaches have been proposed to alleviate the storage constraints for efficiency. Offloading is a common way to reduce the stress of GPU memory. ZeRO-Offload [216] offloads data and computations to CPU to train large models on a single GPU. FlashNeuron [19], however, offloads selective data to the SSD for higher throughput. Additionally, Behemoth [124] replaces low-capacity, high-performance HBM with high-capacity, low-performance NAND flash to enable data-parallel training for large FMs.

Heterogeneous GPUs. Training on specialized high-performance GPU clusters is impossible for most people or enterprises. Moreover, heterogeneous GPUs commonly exist even in specialized GPU clusters. Therefore, some efforts try to train large FMs on heterogeneous GPUs. Hetpipe [202] accelerates training with low-performance GPUs and Wave Synchronous Parallel to synchronize parameters among heterogeneous GPUs. Whale [111] introduces a hardware-aware load-balancing algorithm to speed up training.

Mixture-of-Experts. MoE is an efficient approach to scaling up DNN models. The goals of optimizing MoE training systems are mainly efficiency and scalability. Existing work mainly optimizes the dynamism-related mechanisms, parallelism, and communication in MoE training. MegaBlocks [74] leverages sparse primitives to handle dynamic routing and load-imbalanced computation. Brainstorm [41] is a framework for dynamic DNNs by abstracting the dynamism and profile-based optimization. FlexMoE [193] focuses on the dynamic expert management and device placement problem. Additionally, Tutel [107] designs dynamic adaptive parallelism and pipelining strategies. SmartMoE [297] optimizes the parallelism strategy for efficient MoE training with a combination of offline and online mechanisms. Janus [163] changes communication from an expert-centric paradigm to a data-centric paradigm for faster communication in MoE training. MoE-Mamba [206] integrates MoE with Mamba [81] to enable selective SSMs, reaching the same performance as Mamba in 2.35× fewer training steps.

5.2 Hardware-Aware Optimizations

Some hardware-aware methods are also proposed to optimize FM. For instance, EdgeBERT [231] proposes an in-depth algorithm-hardware co-design for latency-aware energy optimization for multi-task NLP. Its core is an entropy-based early exit prediction for dynamic DVFS at a sentence granularity. FlightLLM [296] is an end-to-end LLM inference mapping flow on FPGAs. Its core is the computation and memory overhead of LLMs can be solved by utilizing FPGA-specific resources (e.g., DSP48 and heterogeneous memory hierarchy). SpAtten [248] proposes a sparse attention mechanism with cascade token and head pruning. It designs a novel top-k engine to rank token and head importance scores with high throughput, and along with other careful optimizations like progressive quantization. A3 [88] makes a key insight that the attention mechanism is semantically a content-based search where a large portion of computations ends up not being used. Recognizing that, it proposes an architecture with algorithmic approximation and hardware specialization.

5.3 Serving on Cloud

FM serving has two main phases: the prefill phase and the decoding phase. The prefill phase often processes a long sequence of input tokens in parallel, which is compute-intensive and can lead to potential bottlenecks if resources are not carefully allocated. In contrast, the decoding phase generates one token at a time, making it more bandwidth-bound [314]. Therefore, a series of optimizations for FM serving systems have been introduced to accelerate this process.

Inference Accelerating. To accelerate the computation in a single accelerator, kernel optimization is a common approach. FlashAttention [43] and FlashAttention-2 [42] design for FM training can be simply used to accelerate the prefill phase. However, due to the unique characteristics of the decoding phase, Flash-Decoding [45] proposes a specific NVIDIA CUDA kernel to accelerate the decoding phase. FlashDecoding++ [98] further improves the performance of Flash-Decoding by optimizing the softmax operation and flat GEMM operation in the decoding phase and provides additional AMD GPU support. DeepSpeed-Inference [15], ByteTransformer [299], and Google's PaLM serving system [209] also optimize GPU/TPU optimizations for small batch size scenarios, which is common in FM serving but rare in FM training. When scaling FM inference to numerous GPUs at a large scale, many works [15, 209] exploit combinations of various parallelism strategies, such as data parallelism, pipeline parallelism, tensor parallelism, and expert parallelism. These works efficiently serve FM inference on multiple modern accelerators, such as GPUs/TPUs.

Given the autoregressive nature of FMs, various requests may feature distinct lengths of input tokens and output tokens. To address this issue, request batching and scheduling constitute another set of methods to enhance the computational efficiency of request processing. Orca [291] proposes selective batching and iteration-level scheduling to batch requests of different lengths at the granularity of iterations to increase the maximum batch size. FlexGen [223] proposes a request scheduling algorithm to mitigate the impact of offloading on the performance of latency-insensitive FM serving in a single GPU. FastServe [263] proposes an iteration-level preemptive scheduling and proactive KV cache swapping to mitigate the impact of head-of-line blocking on the performance of distributed FM serving. SARATHI [13] and DeepSpeed-FastGen [1] split the computation of the prefill phase into small chunks and schedule these chunks with the decoding phase to mitigate the impact of the prefill phase on the performance of large FMs serving. Splitwise [203] splits the prefill phase and the decoding phase onto different machines according to their different computation and memory requirements. Sarathi-Serve [12] introduces a chunked-prefills scheduler which splits a prefill request into near equal sized chunks and creates stall-free schedules that adds new requests in a batch without pausing ongoing decodes. dLoRA [264] dynamically merges/unmerges adapters with the base model and migrating requests/adapters between worker replicas, significantly improving the serving throughput.

Memory Saving. An FM consumes a large amount of memory during the serving process. To reduce the memory consumption of FM serving, many works propose various memory management techniques. As for FMs' parameters and activations, DeepSpeed-Inference [15] and FlexGen [223] offload activations or model parameters to the DRAM or NVMe memories when the GPU memory is insufficient.

KV cache is another important memory component in FM serving. To reduce the memory consumption of KV cache, vLLM [132] adopts a block-level on-demand memory allocation mechanism, which only allocates memory to intermediate states when needed. vLLM also proposes a new operator, Paged Attention, to support attention operation when using this memory allocation mechanism. S-LoRA [222] extends this idea to Unified Paging to manage multiple LoRA adapters at the same time. SGLang [313] further exposes prompt programming primitives to users to enable more complex KV cache management among all requests with the help of RadixAttention.

Emerging Platforms. Typical FM serving systems are usually deployed on data centers equipped with plenty of homogeneous high-performance servers. Due to the scarcity and cost of these high-performance servers, there are also some FM serving systems specifically designed for other deployment platforms. SpotServe [184] tries to serve FMs on spot instances, which are low-cost but unreliable cloud instances. SpotServe dynamically adjusts its parallelism strategy to accommodate the impact of spot instance preemption. As for FM serving on heterogeneous GPUs, HexGen [115] uses an evolutionary algorithm to search for high-performance FM placement on heterogeneous GPUs.

5.4 Serving on Edge

Large FMs have been widely adopted in many real-world mobile applications, such as search engines [10], chatbots [282], and intelligent agents [149]. With ever-increasing data privacy concerns and the stringent response latency requirement, running large FM on mobile devices locally (i.e., on-device inference) has recently attracted attention from both academia and industry. While small language models [176, 289] have been developed for on-device deployment, the runtime efficiency (decoding speed, memory footprint, energy consumption, etc.) still remains a key challenge. Thereby, many on-device inference optimization techniques have been introduced.

Edge-Cloud Collaboration. A common strategy to tackle the scarce resources on mobile devices is to speed up the intensive inference with a powerful edge/cloud server collaboration. For instance, EdgeFM [281] queries and adapts the large FMs to the specific edge models with customized knowledge and architectures so that the dynamic edge model can ensure both low latency and close accuracy to the original large FMs.

On-Device MoE. On-device MoE models are proposed to only execute in routed sparse parameters during inference, which can decrease computation (detailed in Section 3.2). EdgeMoe [288] identifies the problem that experts have to be dynamically loaded into memory during inference. To tackle this issue, this approach proposes expert-wise bit-width adaptation to reduce the size of expert parameters with acceptable accuracy loss, saving parameters' loading time. PC-MoE [128] is based on a crucial observation that expert activations are subject to temporal locality. Based on this observation, PC-MoE proposes Parameter Committee, which intelligently maintains a subset of crucial experts in use to reduce resource consumption.

Memory Optimization. Since large FMs often rely on large parameter sizes and on-device memory resources are scarce (e.g., 8 GB), inferring large FMs on devices faces the challenge of "memory wall." To tackle this issue, LLMCad [272] utilizes speculative decoding [139], which can offload most workloads to a smaller memory-resident draft model. PowerInfer [227] relies on large FMs runtime sparsity (i.e., only hot neurons are consistently activated across inputs). To that end, PowerInfer pre-loads hot-activated neurons onto the GPU for fast access, whereas cold-activated neurons are computed on the CPU, thus significantly reducing GPU memory demands and CPU-GPU data transfers.

I/O Optimization. As parameter size increasing speed is larger than edge devices' memory increasing speed, dynamically loading parameters from disks to memory is avoidable. STI [86] identifies that loading parameters time is highly longer than computation time. To address this problem, STI proposes dynamically adapting weights bit-width during the loading procedure according to parameters importance, minimizing loading overhead under maximum inference accuracy. LLM in a flash [14] solves this problem by fine-grained management of flash storage to reduce the volume of data transferred from flash to memory as well as reading data in larger, more contiguous chunks.

Kernel Optimization. Computing resources are also crucial while limiting resources on the devices. A prior study [304] implements the first 32-bit integer-based edge kernel for ViTs with post-training integer-only quantization to speed up the inference process. This method also introduces a range-constrained quantization technique for activation and normalization operators in transformers to tradeoff data range and inference accuracy. Llm.npu [273] offloads most of the LLM inference computation to a hardware accelerator (NPU) to significantly improve the runtime efficiency.

6 Conclusion and Future Directions

This survey provided a holistic, systematic overview of recent literature toward resource-efficient large FMs. We first presented the preliminary background and cost analysis of the popular FMs, including language, vision, and multimodal. We then dived into the model architecture, algorithm, and system designs to enable a more resource-efficient large FM lifecycle. In the future, the research of this domain will continue to be (or even more) crucial since the scaling law guarantees a promising future of more powerful AI with larger and larger models. Such research is also highly interdisciplinary, involving various CS communities such as machine learning, NLP/CV/Speech, networking, cloud computing, and edge computing.

The research opportunities for resource-efficient large FMs are extremely large, as presented next.

Cloud-Edge Hybrid Deployment. To enable ubiquitous, privacy-preserving, and highly available general intelligence, many FMs will ultimately sink to near-user devices. Preliminary efforts have been already conducted to bring LLaMA-7B to smartphones and PCs. The killer applications include personal assistants/agents [149, 260] and multimodal information retrieval [140], among others. In the future, at what size and speed the FMs can run on devices will become a key competitive force in the business model of hardware vendors.

Exploiting the Model Sparsity. With the model being larger, the activated ratio of the model will become smaller for a given task. Recent literature [174] finds that even a densely trained non-MoE model exhibits runtime activation sparsity, which can be exploited to reduce the inference time and memory footprint. We believe that exploiting the model and activation sparsity will be a promising direction toward sustainable model size scaling. More efficient sparse architectures other than MoE could emerge.

Large FM as a Service. On both clouds and devices, large FMs are unifying the DNN ecosystem [292]. Ultimately, it becomes a universal service to be invoked just as today's Web and Database. On the one hand, it opens the opportunity for highly hardware-algorithm co-design and optimizations, and on the other hand, it poses new challenges in system and infrastructure design for scheduling, load balancing, and security and isolation.

Agent as a Holistic System to Optimize. In the future, FMs, especially LLMs, will be used as a key building block for establishing agents [149, 260]. Its efficiency shall not be considered as in a stand-alone LLM service; instead, the algorithm and system designs need to cater to the specific agent workflow. For example, an agent system might require multiple FMs to cooperate, where there exists inherent logic dependency. In this process, the design space of selecting the proper FMs for each task and scheduling them on a given set of hardware resources to maximize the agent performance is huge.

Practical Privacy-Preserving FM. As the volume of user data uploaded to the cloud for FM processing continues to increase, the severity of privacy concerns correspondingly escalates. Existing methods include federated learning,[2] homomorphic encryption, and disentanglement learning. While being theoretically sound, those methods still confront significant performance challenges, hindering their large-scale in-the-wild deployment. A promising direction involves the development of innovative privacy-preserving techniques specifically designed for large FMs, or the refinement of existing methods, to effectively balance privacy with performance.

Acknowledgments

This project was supported by NSFC (No. 62102045 and No. 62325201), and Super Computing Platform of Beijing University of Posts and Telecommunications.

References

[1] GitHub. 2023. DeepSpeed-FastGen: High-Throughput Text Generation for LLMs via MII and DeepSpeed-Inference. Retrieved December 3, 2024 from https://github.com/microsoft/DeepSpeed/tree/master/blogs/deepspeed-fastgen

[2] GitHub. 2023. Huggingface PEFT. RetrievedDecember3,2024fromhttps://github.com/huggingface/peft

[3] GitHub. 2023. HuggingFace Text Generation Inference. Retrieved December 3, 2024 from https://github.com/huggingface/text-generation-inference

[4] GitHub. 2023. LightLLM. Retrieved December 3, 2024 from https://github.com/ModelTC/lightllm

[5] GitHub. 2023. mlc-llm. Retrieved December 3, 2024 from https://github.com/mlc-ai/mlc-llm

[6] GitHub. 2023. mllm. Retrieved December 3, 2024 from https://github.com/UbiquitousLearning/mllm

[2]A brief literature survey of resource-efficient federated learning can be found in the appendix.

[7] GitHub. 2023. mnn-llm. Retrieved December 3, 2024 from https://github.com/wangzhaode/mnn-llm
[8] GitHub. 2023. NVIDIA TensorRT-LLM. Retrieved December 3, 2024 from https://github.com/NVIDIA/TensorRT-LLM
[9] GitHub. 2023. Ray LLM. Retrieved December 3, 2024 from https://github.com/ray-project/ray-llm
[10] Microsoft. 2024. Microsoft Recall. Retrieved December 3, 2024 from https://support.microsoft.com/en-us/windows/retrace-your-steps-with-recall-aa03f8a0-a78b-4b3e-b0a1-2eb8ac48701c
[11] Rishabh Agarwal, Nino Vieillard, Piotr Stanczyk, and Sabela Ramos. 2023. GKD: Generalized knowledge distillation for auto-regressive sequence models. *arXiv preprint arXiv:2306.13649* (2023).
[12] Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. 2024. Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve. In *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI'24)*. 117–134.
[13] Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, and Ramachandran Ramjee. 2023. SARATHI: Efficient LLM inference by piggybacking decodes with chunked prefills. *arXiv preprint arXiv:2308.16369* (2023).
[14] Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, Karen Khatamifard, Minsik Cho, Carlo C. Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. 2023. LLM in a flash: Efficient large language model inference with limited memory. *arXiv:2312.11514 [cs.CL]* (2023).
[15] 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 *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'22)*. IEEE, 1–15.
[16] Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hofmann. 2023. Dynamic context pruning for efficient and interpretable autogressive transformers. *arXiv:2305.15805 [cs.CL]* (2023).
[17] Akari Asai, Mohammadreza Salehi, Matthew E. Peters, and Hannaneh Hajishirzi. 2022. Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 6655–6672.
[18] Sanjith Athlur, Nitika Saran, Muthian Sivathanu, Ramachandran Ramjee, and Nipun Kwatra. 2022. Varuna: Scalable, low-cost training of massive deep learning models. In *Proceedings of the 17th European Conference on Computer Systems*. 472–487.
[19] Jonghyun Bae, Jongsung Lee, Yunho Jin, Sam Son, Shine Kim, Hakbeom Jang, Tae Jun Ham, and Jae W. Lee. 2021. FlashNeuron: SSD-enabled large-batch training of very deep neural networks. In *Proceedings of the 19th USENIX Conference on File and Storage Technologies*. 387–401.
[20] Sangmin Bae, Jongwoo Ko, Hwanjun Song, and Se-Young Yun. 2023. Fast and robust early-exiting framework for autoregressive language models with synchronized parallel decoding. *arXiv preprint arXiv:2310.05424* (2023).
[21] Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. 2020. BinaryBERT: Pushing the limit of BERT quantization. *arXiv preprint arXiv:2012.15701* (2020).
[22] Arian Bakhtiarnia, Qi Zhang, and Alexandros Iosifidis. 2021. Multi-exit vision transformer for dynamic inference. *arXiv:2106.15183* (2021).
[23] Peter Belcak and Roger Wattenhofer. 2023. Fast feedforward networks. *arXiv:2308.14711 [cs.LG]* (2023).
[24] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* (2020).
[25] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems* 33 (2020), 1877–1901.
[26] Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. *Advances in Neural Information Processing Systems* 35 (2022), 11079–11091.
[27] Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. 2023. Scaling transformer to 1M tokens and beyond with RMT. *arXiv preprint arXiv:2304.11062* (2023).
[28] Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. 2024. EfficientViT: Multi-scale linear attention for high-resolution dense prediction. *arXiv:2205.14756 [cs.CV]* (2024). https://arxiv.org/abs/2205.14756
[29] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2023. Medusa: Simple framework for accelerating LLM generation with multiple decoding heads. Retrieved December 3, 2024 from https://github.com/FasterDecoding/Medusa
[30] Qingqing Cao, Bhargavi Paranjape, and Hannaneh Hajishirzi. 2023. PuMer: Pruning and merging tokens for efficient vision language models. *arXiv:2305.17530 [cs.CV]* (2023).
[31] Ayan Chakrabarti and Benjamin Moseley. 2019. Backprop with approximate activations for memory-efficient network training. *Advances in Neural Information Processing Systems* 32 (2019), 1–10.

[32] Arnav Chavan, Zhuang Liu, Deepak Gupta, Eric Xing, and Zhiqiang Shen. 2023. One-for-All: Generalized LoRA for parameter-efficient fine-tuning. *arXiv preprint arXiv:2306.07967* (2023).
[33] Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. 2023. QuIP: 2-Bit quantization of large language models with guarantees. *arXiv preprint arXiv:2307.13304* (2023).
[34] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv* abs/2302.01318 (2023). https://api.semanticscholar.org/CorpusID:256503945
[35] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. 2021. CrossViT: Cross-attention multi-scale vision transformer for image classification. *arXiv:2103.14899 [cs.CV]* (2021).
[36] Guoxin Chen, Yiming Qian, Bowen Wang, and Liangzhi Li. 2023. MPrompt: Exploring multi-level prompt tuning for machine reading comprehension. *arXiv preprint arXiv:2310.18167* (2023).
[37] Xuxi Chen, Tianlong Chen, Weizhu Chen, Ahmed Hassan Awadallah, Zhangyang Wang, and Yu Cheng. 2021. DSEE: Dually sparsity-embedded efficient tuning of pre-trained language models. *arXiv preprint arXiv:2111.00160* (2021).
[38] Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023. LongLoRA. *arXiv preprint arXiv:2309.12307* (2023).
[39] Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. 2021. Meta-learning via language model in-context tuning. *arXiv preprint arXiv:2110.07814* (2021).
[40] Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *arXiv:2305.14788 [cs.CL]* (2023).
[41] Weihao Cui, Zhenhua Han, Lingji Ouyang, Yichuan Wang, Ningxin Zheng, Lingxiao Ma, Yuqing Yang, Fan Yang, Jilong Xue, Lili Qiu, et al. 2023. Optimizing dynamic neural networks with Brainstorm. In *Proceedings of the 17th USENIX Symposium on Operating Systems Design and Implementation.* 797–815.
[42] Tri Dao. 2023. FlashAttention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691* (2023).
[43] Tri Dao, Dan Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Re. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *Advances in Neural Information Processing Systems* 35 (2022), 16344–16359.
[44] Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Re. 2022. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052* (2022).
[45] Tri Dao, Daniel Haziza, Francisco Massa, and Grigory Sizov. 2023. Flash-Decoding for Long-Context Inference. Retrieved December 3, 2024 from https://crfm.stanford.edu/2023/10/12/flashdecoding.html
[46] Sarkar Snigdha Sarathi Das, Ranran Haoran Zhang, Peng Shi, Wenpeng Yin, and Rui Zhang. 2023. Unified low-resource sequence labeling by sample-aware dynamic sparse finetuning. *arXiv preprint arXiv:2311.03748* (2023).
[47] Jyotikrishna Dass, Shang Wu, Huihong Shi, Chaojian Li, Zhifan Ye, Zhongfeng Wang, and Yingyan Lin. 2023. ViTALiTy: Unifying low-rank and sparse approximation for vision transformer acceleration with a linear Taylor attention. In *Proceedings of the 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA'23).* IEEE, 415–428.
[48] Alex de Vries. 2023. The growing energy footprint of artificial intelligence. *Joule* 7, 10 (2023), 2191–2194.
[49] Luciano Del Corro, Allie Del Giorno, Sahaj Agarwal, Bin Yu, Ahmed Awadallah, and Subhabrata Mukherjee. 2023. SkipDecode: Autoregressive skip decoding with batching and caching for efficient LLM inference. *arXiv preprint arXiv:2307.02628* (2023).
[50] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. LLM.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv:2208.07339 [cs.LG]* (2022).
[51] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized LLMs. *arXiv preprint arXiv:2305.14314* (2023).
[52] 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 preprint arXiv:2306.03078* (2023).
[53] Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. 2023. LongNet: Scaling transformers to 1,000,000,000 tokens. *arXiv:2307.02486 [cs.CL]* (2023).
[54] Tianyu Ding, Tianyi Chen, Haidong Zhu, Jiachen Jiang, Yiqi Zhong, Jinxin Zhou, Guangzhi Wang, Zhihui Zhu, Ilya Zharkov, and Luming Liang. 2023. The efficiency spectrum of large language models: An algorithmic survey. *arXiv preprint arXiv:2312.00678* (2023).
[55] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234* (2022).
[56] 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).

[57] 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 *Proceedings of the International Conference on Machine Learning*. 5547–5569.

[58] Julian Eisenschlos, Maharshi Gor, Thomas Mueller, and William Cohen. 2021. MATE: Multi-view attention for table transformer efficiency. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 7606–7619.

[59] Saar Eliad, Ido Hakimi, Alon De Jagger, Mark Silberstein, and Assaf Schuster. 2021. Fine-tuning giant neural networks on commodity hardware with automatic pipeline model parallelism. In *Proceedings of the USENIX Annual Technical Conference*. 381–396.

[60] Beyza Ermis, Giovanni Zappella, Martin Wistuba, Aditya Rawal, and Cedric Archambeau. 2022. Memory efficient continual learning with transformers. *Advances in Neural Information Processing Systems* 35 (2022), 10629–10642.

[61] FairScale Authors. 2021. FairScale: A general purpose modular PyTorch library for high performance and large scale training. *GitHub*. Retrieved December 3, 2024 from https://github.com/facebookresearch/fairscale

[62] Alex Fang, Albin Madappally Jose, Amit Jain, Ludwig Schmidt, Alexander Toshev, and Vaishaal Shankar. 2023. Data filtering networks. *arXiv:2309.17425 [cs.AI]* (2023). https://arxiv.org/abs/2309.17425

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

[64] Yangyang Feng, Minhui Xie, Zijie Tian, Shuo Wang, Youyou Lu, and Jiwu Shu. 2023. Mobius: Fine tuning large-scale models on commodity GPU servers. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, Vol. 2. 489–501.

[65] Zhida Feng, Zhenyu Zhang, Xintong Yu, Yewei Fang, Lanxin Li, Xuyi Chen, Yuxiang Lu, Jiaxiang Liu, Weichong Yin, Shikun Feng, et al. 2023. ERNIE-ViLG 2.0: Improving text-to-image diffusion model with knowledge-enhanced mixture-of-denoising-experts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10135–10145.

[66] Quentin Fournier, Gaétan Marceau Caron, and Daniel Aloise. 2023. A practical survey on faster and lighter transformers. *ACM Computing Surveys* 55, 14s (2023), 1–40.

[67] Elias Frantar and Dan Alistarh. 2022. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems* 35 (2022), 4475–4488.

[68] Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the International Conference on Machine Learning*. 10323–10337.

[69] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323* (2022).

[70] Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2023. Breaking the Sequential Dependency of LLM Inference Using Lookahead Decoding. Retrieved December 2, 2024 from https://lmsys.org/blog/2023-11-21-lookahead-decoding/

[71] Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. *arXiv preprint arXiv:2301.12726* (2023).

[72] Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. 2023. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 12799–12807.

[73] Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, et al. 2023. DataComp: In search of the next generation of multimodal datasets. *arXiv:2304.14108 [cs.CV]* (2023). https://arxiv.org/abs/2304.14108

[74] Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. 2023. MegaBlocks: Efficient sparse training with mixture-of-experts. In *Proceedings of Machine Learning and Systems 5 (MLSys'23)*.

[75] Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context autoencoder for context compression in a large language model. *arXiv:2307.06945 [cs.CL]* (2023).

[76] Georgi Gerganov. 2023. llama.cpp: A C++ Implementation of the Large Language Models. Retrieved December 3, 2024 from https://github.com/ggerganov/llama.cpp

[77] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. 2023. ImageBind: One embedding space to bind them all. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15180–15190.

[78] 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 International Conference on Machine Learning*. 2337–2346.

[79] Saurabh Goyal, Anamitra R. Choudhury, Saurabh M. Raje, Venkatesan T. Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. PoWER-BERT: Accelerating BERT Inference via progressive word-vector elimination. *arXiv:2001.08950 [cs.LG]* (2020).

[80] Ben Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Herve Jegou, and Matthijs Douze. 2021. LeViT: A vision transformer in ConvNet's clothing for faster inference. *arXiv:2104.01136 [cs.CV]* (2021).

[81] Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752* (2023).
[82] Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently modeling long sequences with structured state spaces. *arXiv:2111.00396* (2022).
[83] Xiaotao Gu, Liyuan Liu, Hongkun Yu, Jing Li, Chen Chen, and Jiawei Han. 2021. On the transformer growth for progressive BERT training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 5174–5180.
[84] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543* (2023).
[85] Cong Guo, Jiaming Tang, Weiming Hu, Jingwen Leng, Chen Zhang, Fan Yang, Yunxin Liu, Minyi Guo, and Yuhao Zhu. 2023. OliVe: Accelerating large language models via hardware-friendly outlier-victim pair quantization. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA'23)*. ACM, New York, NY, USA. https://doi.org/10.1145/3579371.3589038
[86] Liwei Guo, Wonkyo Choe, and Felix Xiaozhu Lin. 2023. STI: Turbocharge NLP inference at the edge via elastic pipelining. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, Vol. 2. 791–803.
[87] Yangyang Guo, Guangzhi Wang, and Mohan Kankanhalli. 2023. PELA: Learning parameter-efficient models with low-rank approximation. *arXiv preprint arXiv:2310.10700* (2023).
[88] 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. A^3: Accelerating attention mechanisms in neural networks with approximation. *arXiv:2002.10941 [cs.DC]* (2020). https://arxiv.org/abs/2002.10941
[89] Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024. LM-Infinite: Simple on-the-fly length generalization for large language models. In *Proceedings of the 12th International Conference on Learning Representations (ICLR'24)*.
[90] Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv:1510.00149 [cs.CV]* (2016). https://arxiv.org/abs/1510.00149
[91] Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural networks. *arXiv:1506.02626 [cs.NE]* (2015). https://arxiv.org/abs/1506.02626
[92] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. LoRA+: Efficient low rank adaptation of large models. *arXiv:2402.12354* (2024).
[93] Bobby He and Thomas Hofmann. 2023. Simplifying transformer blocks. *arXiv preprint arXiv:2311.01906* (2023).
[94] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollar, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16000–16009.
[95] Yefei He, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. 2023. EfficientDM: Efficient quantization-aware fine-tuning of low-bit diffusion models. *arXiv preprint arXiv:2310.03270* (2023).
[96] Yingqing He, Shaoshu Yang, Haoxin Chen, Xiaodong Cun, Menghan Xia, Yong Zhang, Xintao Wang, Ran He, Qifeng Chen, and Ying Shan. 2023. ScaleCrafter: Tuning-free higher-resolution visual generation with diffusion models. *arXiv preprint arXiv:2310.07702* (2023).
[97] Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071* (2022).
[98] Ke Hong, Guohao Dai, Jiaming Xu, Qiuli Mao, Xiuhong Li, Jun Liu, Kangdi Chen, Yuhan Dong, and Yu Wang. 2023. FlashDecoding++: Faster large language model inference on GPUs. *arXiv preprint arXiv:2311.01282* (2023).
[99] 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).
[100] Zi-Yuan Hu, Yanyang Li, Michael R. Lyu, and Liwei Wang. 2023. VL-PET: Vision-and-language parameter-efficient tuning via granularity control. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3010–3020.
[101] Jane Huang and Kevin Williams. 2023. GPT-4 for creative writing: A case study of content generation in digital media. *arXiv preprint arXiv:2305.11234* (2023).
[102] Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. *arXiv preprint arXiv:2104.02112* (2021).
[103] Tao Huang, Lang Huang, Shan You, Fei Wang, Chen Qian, and Chang Xu. 2022. LightViT: Towards light-weight convolution-free vision transformers. *arXiv:2207.05557 [cs.CV]* (2022). https://arxiv.org/abs/2207.05557
[104] Xijie Huang, Li Lyna Zhang, Kwang-Ting Cheng, and Mao Yang. 2023. Boosting LLM reasoning: Push the limits of few-shot learning with reinforced in-context pruning. *arXiv:2312.08901 [cs.CL]* (2023).

[105] 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 preprint arXiv:2212.10670* (2022).

[106] Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Seffi Naor, and Daniel Soudry. 2021. Accelerated sparse neural training: A provable and efficient method to find N:M transposable masks. *arXiv:2102.08124 [cs.AI]* (2021).

[107] 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. *Microsoft.* Retrieved December 3, 2024 from https://mlsys.org/media/mlsys-2023/Slides/2477.pdf

[108] 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.

[109] Insu Jang, Zhenning Yang, Zhen Zhang, Xin Jin, and Mosharaf Chowdhury. 2023. Oobleck: Resilient distributed training of large models using pipeline templates. In *Proceedings of the 29th Symposium on Operating Systems Principles.* 382–395.

[110] Abhinav Jangda, Jun Huang, Guodong Liu, Amir Hossein Nodehi Sabet, Saeed Maleki, Youshan Miao, Madanlal Musuvathi, Todd Mytkowicz, and Olli Saarikivi. 2022. Breaking the computation and communication abstraction barrier in distributed machine learning workloads. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems.* 402–416.

[111] Xianyan Jia, Le Jiang, Ang Wang, Wencong Xiao, Ziji Shi, Jie Zhang, Xinyuan Li, Langshi Chen, Yong Li, Zhen Zheng, et al. 2022. Whale: Efficient giant model training over heterogeneous GPUs. In *Proceedings of the USENIX Annual Technical Conference.* 673–688.

[112] Chaoya Jiang, Haiyang Xu, Chenliang Li, Ming Yan, Wei Ye, Shikun Zhang, Bin Bi, and Songfang Huang. 2022. TRIPS: Efficient vision-and-language pre-training with text-relevant image patch selection. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing.* 4084–4096.

[113] Chaoya Jiang, Haiyang Xu, Wei Ye, Qinghao Ye, Chenliang Li, Ming Yan, Bin Bi, Shikun Zhang, Ji Zhang, and Fei Huang. 2023. COPA: Efficient vision-language pre-training through collaborative object-and patch-text alignment. In *Proceedings of the 31st ACM International Conference on Multimedia.* 4480–4491.

[114] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LLMLingua: Compressing prompts for accelerated inference of large language models. *arXiv:2310.05736 [cs.CL]* (2023).

[115] Youhe Jiang, Ran Yan, Xiaozhe Yao, Yang Zhou, Beidi Chen, and Binhang Yuan. 2023. HexGen: Generative inference of foundation model over heterogeneous decentralized environment. *arXiv preprint arXiv:2311.11514* (2023).

[116] Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. 2023. PolySketchFormer: Fast transformers via sketches for polynomial kernels. *arXiv preprint arXiv:2310.01655* (2023).

[117] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

[118] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the International Conference on Machine Learning.* 5156–5165.

[119] Ayush Kaushal, Tejas Vaidhya, and Irina Rish. 2023. LoRD: Low rank decomposition of monolingual code LLMs for one-shot compression. *arXiv preprint arXiv:2309.14021* (2023).

[120] Gyuwan Kim and Kyunghyun Cho. 2021. Length-adaptive transformer: Train once with length drop, use anytime with search. *arXiv:2010.07003 [cs.CL]* (2021).

[121] 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 preprint arXiv:2305.14152* (2023).

[122] Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. 2023. SqueezeLLM: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629* (2023).

[123] Sehoon Kim, Coleman Hooper, Thanakul Wattanawong, Minwoo Kang, Ruohan Yan, Hasan Genc, Grace Dinh, Qijing Huang, Kurt Keutzer, Michael W. Mahoney, et al. 2023. Full stack optimization of transformer inference: A survey. *arXiv preprint arXiv:2302.14017* (2023).

[124] Shine Kim, Yunho Jin, Gina Sohn, Jonghyun Bae, Tae Jun Ham, and Jae W. Lee. 2021. Behemoth: A flash-centric training accelerator for extreme-scale DNNs. In *Proceedings of the 19th USENIX Conference on File and Storage Technologies.* 371–385.

[125] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo et al. 2023. Segment anything. *arXiv:2304.02643* (2023).

[126] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *Proceedings of the International Conference on Learning Representations.*

[127] Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. 2023. Sparse upcycling: Training mixture-of-experts from dense checkpoints. *arXiv:2212.05055 [cs.LG]* (2023).
[128] Rui Kong, Yuanchun Li, Qingtian Feng, Weijun Wang, Linghe Kong, and Yunxin Liu. 2023. Serving MoE models on resource-constrained edge devices via dynamic expert swapping. *arXiv preprint arXiv:2308.15030* (2023).
[129] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Mengshu Sun, Wei Niu, Xuan Shen, Geng Yuang, Bin Ren, Minghai Qin, et al. 2022. SPViT: Enabling faster vision transformers via soft token pruning. *arXiv:2112.13890 [cs.CV]* (2022).
[130] Juil Koo, Seungwoo Yoo, Minh Hieu Nguyen, and Minhyuk Sung. 2023. Salad: Part-level latent diffusion for 3D shape generation and manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14441–14451.
[131] Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. 2023. Reducing activation recomputation in large transformer models. In *Proceedings of the Machine Learning and Systems Conference (MLSys'23).*
[132] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*. 611–626.
[133] François Lagunas, Ella Charlaix, Victor Sanh, and Alexander M. Rush. 2021. Block pruning for faster transformers. *arXiv preprint arXiv:2109.04838* (2021).
[134] Joel Lamy-Poirier. 2023. Breadth-first pipeline parallelism. In *Proceedings of the Machine Learning and Systems Conference (MLSys'23).*
[135] 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 preprint arXiv:2306.02272* (2023).
[136] Jung Hyun Lee, Jeonghoon Kim, Se Jung Kwon, and Dongsoo Lee. 2023. FlexRound: Learnable rounding based on element-wise division for post-training quantization. *arXiv preprint arXiv:2306.00317* (2023).
[137] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 8424–8445.
[138] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691* (2021).
[139] Yaniv Leviathan, Matan Kalman, and Y. Matias. 2022. Fast inference from transformers via speculative decoding. In *Proceedings of the International Conference on Machine Learning*. https://api.semanticscholar.org/CorpusID:254096365
[140] Chunyuan Li, Zhe Gan, Zhengyuan Yang, Jianwei Yang, Linjie Li, Lijuan Wang, and Jianfeng Gao. 2023. Multimodal foundation models: From specialists to general-purpose assistants. *arXiv preprint arXiv:2309.10020* (2023).
[141] Changlin Li, Bohan Zhuang, Guangrun Wang, Xiaodan Liang, Xiaojun Chang, and Yi Yang. 2022. Automated progressive learning for efficient training of vision transformers. *arXiv:2203.14509 [cs.CV]* (2022). https://arxiv.org/abs/2203.14509
[142] Yujia Li, David Choi, Junyong Chung, Nate Kushman, Julian Schrittwieser, Remi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Augustin Dal Lago, et al. 2022. Competition-level code generation with AlphaCode. *Science* 378, 6624 (2022), 1107–1114.
[143] Kai Li, Runxuan Yang, and Xiaolin Hu. 2022. An efficient encoder-decoder architecture with top-down attention for speech separation. *arXiv preprint arXiv:2209.15200* (2022).
[144] Shenggui Li, Hongxin Liu, Zhengda Bian, Jiarui Fang, Haichen Huang, Yuliang Liu, Boxiang Wang, and Yang You. 2023. Colossal-AI: A unified deep learning system for large-scale parallel training. In *Proceedings of the 52nd International Conference on Parallel Processing*. ACM, New York, NY, USA, 766–775.
[145] Shenggui Li, Fuzhao Xue, Chaitanya Baranwal, Yongbin Li, and Yang You. 2023. Sequence parallelism: Long sequence training from system perspective. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. 2391–2404.
[146] Sheng Li, Geng Yuan, Yue Dai, Youtao Zhang, Yanzhi Wang, and Xulong Tang. 2022. SmartFRZ: An efficient training framework using attention-based layer freezing. In *Proceedings of the 11th International Conference on Learning Representations.*
[147] Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. 2023. Q-diffusion: Quantizing diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 17535–17545.
[148] Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. 2023. SnapFusion: Text-to-image diffusion model on mobile devices within two seconds. *arXiv preprint arXiv:2306.00980* (2023).

[149] Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, et al. 2024. Personal LLM agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459* (2024).

[150] Yanjing Li, Sheng Xu, Baochang Zhang, Xianbin Cao, Peng Gao, and Guodong Guo. 2022. Q-ViT: Accurate and fully quantized low-bit vision transformer. *Advances in Neural Information Processing Systems* 35 (2022), 34451–34463.

[151] 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).

[152] 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. *arXiv preprint arXiv:2306.11222* (2023).

[153] Yanyu Li, Geng Yuan, Yang Wen, Ju Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. 2022. EfficientFormer: Vision transformers at MobileNet speed. *arXiv:2206.01191 [cs.CV]* (2022).

[154] Zhikai Li and Qingyi Gu. 2023. I-ViT: Integer-only quantization for efficient vision transformer inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 17065–17075.

[155] 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 *Proceedings of the International Conference on Machine Learning*. 20852–20867.

[156] Baohao Liao, Yan Meng, and Christof Monz. 2023. Parameter-efficient fine-tuning without introducing new latency. *arXiv preprint arXiv:2305.16742* (2023).

[157] Baohao Liao, Shaomu Tan, and Christof Monz. 2023. Make your pre-trained model reversible: From parameter to memory efficient fine-tuning. *arXiv preprint arXiv:2306.00477* (2023).

[158] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2023. AWQ: Activation-aware weight quantization for LLM compression and acceleration. *arXiv preprint arXiv:2306.00978* (2023).

[159] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2022. A survey of transformers. *AI Open* 3 (2022), 111–132.

[160] Guisheng Liu, Yi Li, Zhengcong Fei, Haiyan Fu, Xiangyang Luo, and Yanqing Guo. 2023. Prefix-diffusion: A lightweight diffusion model for diverse image captioning. *arXiv preprint arXiv:2309.04965* (2023).

[161] 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:2310.08041 [cs.CL]* (2023).

[162] Jihao Liu, Xin Huang, Jinliang Zheng, Yu Liu, and Hongsheng Li. 2023. MixMAE: Mixed and masked autoencoder for efficient pretraining of hierarchical vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6252–6261.

[163] Juncai Liu, Jessie Hui Wang, and Yimin Jiang. 2023. Janus: A unified distributed training framework for sparse mixture-of-experts models. In *Proceedings of the 2023 ACM SIGCOMM Conference*. 486–498.

[164] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. 2022. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778* (2022).

[165] Shih-Yang Liu, Zechun Liu, and Kwang-Ting Cheng. 2023. Oscillation-free quantization for low-bit vision transformers. *arXiv preprint arXiv:2302.02210* (2023).

[166] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. DoRA: Weight-decomposed low-rank adaptation. *arXiv:2402.09353 [cs]* (2024). http://arxiv.org/abs/2402.09353

[167] Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. 2023. EfficientViT: Memory efficient vision transformer with cascaded group attention. *arXiv:2305.07027 [cs.CV]* (2023). https://arxiv.org/abs/2305.07027

[168] 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 *Proceedings of the International Conference on Machine Learning*. 14139–14152.

[169] Yi Liu, Yun Ma, Xusheng Xiao, Tao Xie, and Xuanzhe Liu. 2023. LegoDroid: flexible Android app decomposition and instant installation. *Sci. China Inf. Sci.* 66, 4 (2023). DOI: https://doi.org/10.1007/S11432-021-3528-7

[170] Yue Liu, Christos Matsoukas, Fredrik Strand, Hossein Azizpour, and Kevin Smith. 2023. PatchDropout: Economizing vision transformers using PatchDropout. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 3953–3962.

[171] Yuqi Liu, Luhui Xu, Pengfei Xiong, and Qin Jin. 2023. Token mixing: Parameter-efficient transfer learning from image-language to video-language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 1781–1789.

[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:2305.17118 [cs.LG]* (2023).

[173] Zechun Liu, Barlas Oguz, 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 preprint arXiv:2305.17888* (2023).

[174] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christoper Re, et al. 2023. DejaVu: Contextual sparsity for efficient LLMs at inference time. In *Proceedings of the International Conference on Machine Learning*. 22137–22176.

[175] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. 2022. DPM-Solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems* 35 (2022), 5775–5787.

[176] Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicolas D. Lane, and Mengwei Xu. 2024. Small language models: Survey, measurements, and insights. *arXiv preprint arXiv:2409.15790* (2024).

[177] Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-Pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627* (2023).

[178] Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. 2022. Mega: Moving average equipped gated attention. *arXiv preprint arXiv:2209.10655* (2022).

[179] Sachin Mehta and Mohammad Rastegari. 2022. MobileViT: Light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv:2110.02178 [cs.CV]* (2022).

[180] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. PiSSA: Principal singular values and singular vectors adaptation of large language models. *arXiv:2404.02948 [cs]* (2024). http://arxiv.org/abs/2404.02948

[181] Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. 2021. AdaViT: Adaptive vision transformers for efficient image recognition. *arXiv:2111.15668 [cs.CV]* (2021).

[182] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. 2023. Towards efficient generative large language model serving: A survey from algorithms to systems. *arXiv preprint arXiv:2312.15234* (2023).

[183] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al 2023. SpecInfer: Accelerating generative large language model serving with speculative inference and token tree verification. *arXiv:2305.09781 [cs.CL]* (2023).

[184] Xupeng Miao, Chunan Shi, Jiangfei Duan, Xiaoli Xi, Dahua Lin, Bin Cui, and Zhihao Jia. 2023. SpotServe: Serving generative large language models on preemptible instances. *arXiv preprint arXiv:2311.15566* (2023).

[185] Xupeng Miao, Yujie Wang, Youhe Jiang, Chunan Shi, Xiaonan Nie, Hailin Zhang, and Bin Cui. 2023. Galvatron: Efficient transformer training over multiple GPUs using automatic parallelism. *Proceedings of the VLDB Endowment* 16, 3 (2023), 470–479.

[186] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2021. MetaICL: Learning to learn in context. *arXiv preprint arXiv: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] Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. 2022. Multimodal contrastive learning with LIMoE: The language-image mixture of experts. *Advances in Neural Information Processing Systems* 35 (2022), 9564–9576.

[189] Pranav Ajit Nair, Sukomal Pal, and Pradeepika Verm. 2023. Domain aligned prefix averaging for domain generalization in abstractive summarization. *arXiv preprint arXiv:2305.16820* (2023).

[190] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. 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*. 1–15.

[191] Piotr Nawrot, Jan Chorowski, Adrian Łańcucki, and Edoardo M. Ponti. 2022. Efficient transformers with dynamic token pooling. *arXiv preprint arXiv:2211.09761* (2022).

[192] Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *Proceedings of the International Conference on Machine Learning*. 8162–8171.

[193] Xiaonan Nie, Xupeng Miao, Zilong Wang, Zichao Yang, Jilong Xue, Lingxiao Ma, Gang Cao, and Bin Cui. 2023. FlexMoE: Scaling large-scale sparse pre-trained model training via dynamic device placement. *Proceedings of the ACM on Management of Data* 1, 1 (2023), Article 110, 19 pages.

[194] Wei Niu, Jiexiong Guan, Yanzhi Wang, Gagan Agrawal, and Bin Ren. 2021. DNNFusion: Accelerating deep neural networks execution with advanced operator fusion. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*. 883–898.

[195] Georgii Sergeevich Novikov, Daniel Bershatsky, Julia Gusak, Alex Shonenkov, Denis Dimitrov, and Ivan Oseledets. 2023. Few-bit backward: Quantized gradients of activation functions for memory footprint reduction. In *Proceedings of the International Conference on Machine Learning*. 26363–26381.

[196] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Forencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, et al. 2023. GPT-4 technical report. *arXiv:2303.08774 [cs.CL]* (2023).

[197] Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. 2023. Resurrecting recurrent neural networks for long sequences. *arXiv preprint arXiv:2303.06349* (2023).

[198] Kazuki Osawa, Shigang Li, and Torsten Hoefler. 2023. PipeFisher: Efficient training of large language models using pipelining and Fisher information matrices. In *Proceedings of the Machine Learning and Systems Conference.*

[199] Junting Pan, Adrian Bulat, Fuwen Tan, Xiatian Zhu, Lukasz Dudziak, Hongsheng Li, Georgios Tzimiropoulos, and Brais Martinez. 2022. EdgeViTs: Competing light-weight CNNs on mobile devices with vision transformers. *arXiv:2205.03436 [cs.CV]* (2022). https://arxiv.org/abs/2205.03436

[200] Junting Pan, Ziyi Lin, Xiatian Zhu, Jing Shao, and Hongsheng Li. 2022. ST-Adapter: Parameter-efficient image-to-video transfer learning. *Advances in Neural Information Processing Systems* 35 (2022), 26462–26477.

[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] Jay H. Park, Gyeongchan Yun, Chang M. Yi, Nguyen T. Nguyen, Seungmin Lee, Jaesik Choi, Sam H. Noh, and Young-Ri Choi. 2020. HetPipe: Enabling large DNN training on (whimpy) heterogeneous GPU clusters through integration of pipelined model parallelism and data parallelism. In *Proceedings of the USENIX Annual Technical Conference.* 307–321.

[203] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Inigo Goiri, Saeed Maleki, and Ricardo Bianchini. 2023. Splitwise: Efficient generative LLM inference using phase splitting. *arXiv preprint arXiv:2311.18677* (2023).

[204] William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. *arXiv:2212.09748 [cs.CV]* (2023). https://arxiv.org/abs/2212.09748

[205] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. 2023. RWKV: Reinventing RNNs for the transformer era. *arXiv preprint arXiv:2305.13048* (2023).

[206] Maciej Pióro, Kamil Ciebiera, Krystian Król, Jan Ludziejewski, Michal Krutul, Jakub Krajewski, Szymon Antoniak, Piotr Milos, Marek Cygan, and Sebastian Jaszczur. 2024. MoE-Mamba: Efficient selective state space models with mixture of experts. *arXiv preprint arXiv:2401.04081* (2024).

[207] Koutilya Pnvr, Bharat Singh, Pallabi Ghosh, Behjat Siddiquie, and David Jacobs. 2023. LD-ZNet: A latent diffusion approach for text-based image segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 4157–4168.

[208] Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Re. 2023. Hyena hierarchy: Towards larger convolutional language models. *arXiv preprint arXiv:2302.10866* (2023).

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

[210] 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. *arXiv:2310.02409 [cs.CL]* (2023).

[211] Yujia Qin, Yankai Lin, Jing Yi, Jiajie Zhang, Xu Han, Zhengyan Zhang, Yusheng Su, Zhiyuan Liu, Peng Li, Maosong Sun, et al. 2022. Knowledge inheritance for pre-trained language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* 3921–3937.

[212] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning.* 8748–8763.

[213] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'20).* IEEE, 1–16.

[214] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. 2021. DynamicViT: Efficient vision transformers with dynamic token sparsification. *arXiv:2106.02034 [cs.CV]* (2021).

[215] 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.

[216] 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 *Proceedings of the USENIX Annual Technical Conference.* 551–564.

[217] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, Andre Susano Pinto, Daniel Keysers, and Neil Houlsby. 2021. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems* 34 (2021), 8583–8595.

[218] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10684–10695.

[219] Rajarshi Saha, Varun Srivastava, and Mert Pilanci. 2023. Matrix compression via randomized low rank and low precision factorization. *arXiv preprint arXiv:2310.11028* (2023).

[220] Michael Santacroce, Zixin Wen, Yelong Shen, and Yuanzhi Li. 2023. What matters in the structured pruning of generative language models? *arXiv preprint arXiv:2302.03773* (2023).

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

[222] Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, et al. 2023. S-LoRA: Serving thousands of concurrent LoRA adapters. *arXiv preprint arXiv:2311.03285* (2023).

[223] 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. FlexGen: High-throughput generative inference of large language models with a single GPU. In *Proceedings of the International Conference on Machine Learning*. 31094–31116.

[224] Dachuan Shi, Chaofan Tao, Ying Jin, Zhendong Yang, Chun Yuan, and Jiaqi Wang. 2023. UPop: Unified and progressive pruning for compressing vision-language transformers. *arXiv preprint arXiv:2301.13741* (2023).

[225] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2020. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502* (2020).

[226] Jaeyong Song, Jinkyu Yim, Jaewon Jung, Hongsun Jang, Hyung-Jin Kim, Youngsok Kim, and Jinho Lee. 2023. Optimus-CC: Efficient large NLP model training with 3D parallelism aware communication compression. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, Vol. 2. 560–573.

[227] Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. 2023. PowerInfer: Fast large language model serving with a consumer-grade GPU. *arXiv:2312.12456 [cs.LG]* (2023).

[228] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695* (2023).

[229] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621* (2023).

[230] Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. 2023. SpecTR: Fast speculative decoding via optimal transport. In *Proceedings of the Workshop on Efficient Systems for Foundation Models @ ICML2023*. https://openreview.net/forum?id=d0mGsaheuT

[231] Thierry Tambe, Coleman Hooper, Lillian Pentecost, Tianyu Jia, En-Yu Yang, Marco Donato, Victor Sanh, Paul N. Whatmough, Alexander M. Rush, David Brooks, et al. 2021. EdgeBERT: Sentence-level energy optimizations for latency-aware multi-task NLP inference. *arXiv:2011.14203 [cs.AR]* (2021). https://arxiv.org/abs/2011.14203

[232] 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).

[233] John Thorpe, Pengzhan Zhao, Jonathan Eyolfson, Yifan Qiao, Zhihao Jia, Minjia Zhang, Ravi Netravali, and Quoqing Harry Xu. 2023. Bamboo: Making preemptible instances resilient for affordable training of large DNNs. In *Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation*. 497–513.

[234] Lin Tian, Xiuzhen Zhang, and Jey Han Lau. 2023. MetaTroll: Few-shot detection of state-sponsored trolls with transformer adapters. In *Proceedings of the 2023 ACM Web Conference*. 1743–1753.

[235] 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).

[236] Katrin Tomanek, Vicky Zayats, Dirk Padfield, Kara Vaillancourt, and Fadi Biadsy. 2021. Residual adapters for parameter-efficient ASR adaptation to atypical and accented speech. *arXiv preprint arXiv:2109.06952* (2021).

[237] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. 2012. Training data-efficient image transformers and distillation through attention arXiv:2012.12877 (2012).

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

[239] Hieu Tran, Zhichao Yang, Zonghai Yao, and Hong Yu. 2023. BioInstruct: Instruction tuning of large language models for biomedical natural language processing. *arXiv preprint arXiv:2310.19975* (2023).

[240] Alexander Tsvetkov and Alon Kipnis. 2023. EntropyRank: Unsupervised keyphrase extraction via side-information optimization for language model-based text compression. In *Proceedings of the Workshop on Neural Compression: From Information Theory to Applications (ICML'23)*.

[241] Chandra Shekhara Kaushik Valmeekam, Krishna Narayanan, Dileep Kalathil, Jean-Francois Chamberland, and Srinivas Shakkottai. 2023. LLMZip: Lossless text compression using large language models. *arXiv:2306.04050 [cs.IT]* (2023).

[242] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. 2023. FastViT: A fast hybrid vision transformer using structural reparameterization. *arXiv:2303.14189 [cs.CV]* (2023). https://arxiv.org/abs/2303.14189

[243] Pavan Kumar Anasosalu Vasu, Hadi Pouransari, Fartash Faghri, Raviteja Vemulapalli, and Oncel Tuzel. 2024. MobileCLIP: Fast image-text models through multi-modal reinforced training. *arXiv:2311.17049 [cs.CV]* (2024). https://arxiv.org/abs/2311.17049

[244] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30 (2017), 1–11.

[245] Yixin Wan, Kuan-Hao Huang, and Kai-Wei Chang. 2023. PIP: Parse-instructed prefix for syntactically controlled paraphrase generation. *arXiv preprint arXiv:2305.16701* (2023).

[246] Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, et al. 2023. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863* 1 (2023).

[247] 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. *arXiv preprint arXiv:2310.11453* (2023).

[248] Hanrui Wang, Zhekai Zhang, and Song Han. 2021. SpAtten: Efficient sparse attention architecture with cascade token and head pruning. In *Proceedings of the 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA'21)*. IEEE. https://doi.org/10.1109/hpca51647.2021.00018

[249] Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. 2022. Learning to grow pretrained models for efficient transformer training. In *Proceedings of the 11th International Conference on Learning Representations*.

[250] 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).

[251] Shibo Wang, Jinliang Wei, Amit Sabne, Andy Davis, Berkin Ilbeyi, Blake Hechtman, Dehao Chen, Karthik Srinivasa Murthy, Marcello Maggioni, Qiao Zhang, et al. 2023. Overlap communication with dependent computation via decomposition in large deep learning models. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, Vol. 1. 93–106.

[252] Xuan Wang, Guanhong Wang, Wenhao Chai, Jiayu Zhou, and Gaoang Wang. 2023. User-aware prefix-tuning is a good learner for personalized image captioning. *arXiv preprint arXiv:2312.04793* (2023).

[253] Xudong Wang, Li Lyna Zhang, Yang Wang, and Mao Yang. 2022. Towards efficient vision transformer inference: A first study of transformers on mobile devices. In *Proceedings of the 23rd Annual International Workshop on Mobile Computing Systems and Applications*. 1–7.

[254] Xinlong Wang, Xiaosong Zhang, Yue Cao, Wen Wang, Chunhua Shen, and Tiejun Huang. 2023. SegGPT: Segmenting everything in context. *arXiv:2304.03284* (2023).

[255] Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Guo, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. AdaMix: Mixture-of-adaptations for parameter-efficient model tuning. *arXiv preprint arXiv:2210.17451* (2022).

[256] Zhuang Wang, Zhen Jia, Shuai Zheng, Zhen Zhang, Xinwei Fu, T. S. Eugene Ng, and Yida Wang. 2023. GEMINI: Fast failure recovery in distributed training with in-memory checkpoints. In *Proceedings of the 29th Symposium on Operating Systems Principles*. 364–381.

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

[258] 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:2304.09145 [cs.CL]* (2023).

[259] Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. 2022. Outlier Suppression: Pushing the limit of low-bit transformer language models. *Advances in Neural Information Processing Systems* 35 (2022), 17402–17414.

[260] Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2023. Empowering LLM to use smartphone for intelligent task automation. *arXiv preprint arXiv:2308.15272* (2023).

[261] Qizhen Weng, Wencong Xiao, Yinghao Yu, and others. 2022. MLaaS in the wild: Workload analysis and scheduling in large-scale heterogeneous GPU clusters. In *Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation*. 945–960.

[262] David Wingate, Mohammad Shoeybi, and Taylor Sorensen. 2022. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models. *arXiv:2210.03162 [cs.CL]* (2022).
[263] Bingyang Wu, Yinmin Zhong, Zili Zhang, Shengyu Liu, Fangyue Liu, Yuanhang Sun, Gang Huang, Xuanzhe Liu, and Xin Jin. 2023. Fast distributed inference serving for large language models. *arXiv preprint arXiv:2305.05920* (2023).
[264] Bingyang Wu, Ruidong Zhu, Zili Zhang, Peng Sun, Xuanzhe Liu, and Xin Jin. 2024. dLoRA: Dynamically orchestrating requests and adapters for LoRA LLM serving. In *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI'24)*. 911–927.
[265] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga Behram, James Huang, Charles Bai, et al. 2022. Sustainable AI: Environmental implications, challenges and opportunities. In *Proceedings of the Machine Learning and Systems Conference*. 795–813.
[266] Xiaoxia Wu, Cheng Li, Reza Yazdani Aminabadi, Zhewei Yao, and Yuxiong He. 2023. Understanding INT4 quantization for language models: Latency speedup, composability, and failure cases. In *Proceedings of the International Conference on Machine Learning*. 37524–37539.
[267] Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694* (2023).
[268] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. 2022. Vision transformer with deformable attention. *arXiv:2201.00520 [cs.CV]* (2022).
[269] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and efficient post-training quantization for large language models. *arXiv:2211.10438 [cs.CL]* (2023).
[270] 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).
[271] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. *arXiv preprint arXiv:2004.12993* (2020).
[272] Daliang Xu, Wangsong Yin, Xin Jin, Ying Zhang, Shiyun Wei, Mengwei Xu, and Xuanzhe Liu. 2023. LLMCad: Fast and scalable on-device large language model inference. *arXiv:2309.04255 [cs.NI]* (2023).
[273] Daliang Xu, Hao Zhang, Liming Yang, Ruiqi Liu, Gang Huang, Mengwei Xu, and Xuanzhe Liu. 2024. Empowering 1000 tokens/second on-device LLM prefilling with mllm-NPU. *arXiv preprint arXiv:2407.05858* (2024).
[274] Guanyu Xu, Jiawei Hao, Li Shen, Han Hu, Yong Luo, Hui Lin, and Jialie Shen. 2023. LGViT: Dynamic early exiting for accelerating vision transformer. In *Proceedings of the 31st ACM International Conference on Multimedia*. 9103–9114.
[275] Mengwei Xu, Jiawei Liu, Yuanqiang Liu, Felix Xiaozhu Lin, Yunxin Liu, and Xuanzhe Liu. 2019. A first look at deep learning apps on smartphones. In *Proceedings of the World Wide Web Conference*. 2125–2136.
[276] 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 preprint arXiv:2307.00526* (2023).
[277] Mengwei Xu, Wangsong Yin, Dongqi Cai, Rongjie Yi, Daliang Xu, Qipeng Wang, Bingyang Wu, Yihao Zhao, Chen Yang, Shihe Wang, et al. 2024. A survey of resource-efficient LLM and multimodal foundation models. *arXiv preprint arXiv:2401.08092* (2024).
[278] Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. 2023. QA-LoRA: Quantization-aware low-rank adaptation of large language models. *arXiv preprint arXiv:2309.14717* (2023).
[279] Liu Xuanzhe, Yihao Zhao, Shufan Liu, Xiang Li, Xin Zhu Yibo Liu, and Xin Jin. 2024. MuxFlow: Efficient GPU sharing in production-level clusters with more than 10,000 GPUs. *Science China Information Sciences* (2024). DOI: https://doi.org/10.1007/s11432-024-4227-2
[280] Yu Yan, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-sequence pre-training. *arXiv:abs/2001.04083* (2020). https://api.semanticscholar.org/CorpusID:210164665
[281] Bufang Yang, Lixing He, Neiwen Ling, Zhenyu Yan, Guoliang Xing, Xian Shuai, Xiaozhe Ren, and Xin Jiang. 2023. EdgeFM: Leveraging foundation model for open-set learning on the edge. *arXiv preprint arXiv:2311.10986* (2023).
[282] 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:2304.13712 [cs.CL]* (2023).
[283] Yuedong Yang, Hung-Yueh Chiang, Guihong Li, Diana Marculescu, and Radu Marculescu. 2023. Efficient low-rank backpropagation for vision transformer adaptation. *arXiv preprint arXiv:2309.15275* (2023).
[284] Yuting Yang, Wenqiang Lei, Pei Huang, Juan Cao, Jintao Li, and Tat-Seng Chua. 2023. A dual prompt learning framework for few-shot dialogue state tracking. In *Proceedings of the 2023 ACM Web Conference*. 1468–1477.
[285] Shih yang Liu, Zechun Liu, Xijie Huang, Pingcheng Dong, and Kwang-Ting Cheng. 2023. LLM-FP4: 4-bit floating-point quantized transformers. *arXiv:2310.16836 [cs.CL]* (2023).
[286] 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. *Advances in Neural Information Processing Systems* 35 (2022), 27168–27183.

[287] Deming Ye, Yankai Lin, Yufei Huang, and Maosong Sun. 2021. TR-BERT: Dynamic token reduction for accelerating BERT inference. *arXiv:2105.11618 [cs.CL]* (2021).

[288] 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).

[289] Rongjie Yi, Xiang Li, Weikai Xie, Zhenyan Lu, Chenghua Wang, Ao Zhou, Shangguang Wang, Xiwen Zhang, and Mengwei Xu. 2024. PhoneLM: An efficient and capable small language model family through principled pre-training. *arXiv preprint arXiv:2411.05046* (2024).

[290] Hongxu Yin, Arash Vahdat, Jose Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. 2022. AdaViT: Adaptive tokens for efficient vision transformer. *arXiv:2112.07658 [cs.CV]* (2022).

[291] 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 *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI'22)*. 521–538.

[292] Jinliang Yuan, Chen Yang, Dongqi Cai, Shihe Wang, Xin Yuan, Zeling Zhang, Xiang Li, Dingge Zhang, Hanzi Mei, Xianqing Jia, et al. 2024. Mobile foundation model as firmware. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking (MobiCom'24)*. ACM, New York, NY, USA. https://doi.org/10.1145/3636534.3649361

[293] Zhihang Yuan, Lin Niu, Jiawei 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:2304.01089 [cs.CL]* (2023).

[294] Manzil Zaheer, Guru Guruganesh, 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.

[295] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. 2022. Restormer: Efficient transformer for high-resolution image restoration. *arXiv:2111.09881 [cs.CV]* (2022).

[296] Shulin Zeng, Jun Liu, Guohao Dai, Xinhao Yang, Tianyu Fu, Hongyi Wang, Wenheng Ma, Hanbo Sun, Shiyao Li, Xixiao Huang, et al. 2024. FlightLLM: Efficient large language model inference with a complete mapping flow on FPGAs. *arXiv:2401.03868 [cs.AR]* (2024). https://arxiv.org/abs/2401.03868

[297] 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 *Proceedings of the USENIX Annual Technical Conference*. 961–975.

[298] Shuangfei Zhai, Walter Talbott, Nitish Srivastava, Chen Huang, Hanlin Goh, Ruixiang Zhang, and Josh Susskind. 2021. An attention free transformer. *arXiv preprint arXiv:2105.14103* (2021).

[299] Yujia Zhai, Chengquan Jiang, Leyuan Wang, Xiaoying Jia, Shang Zhang, Zizhong Chen, Xin Liu, and Yibo Xhu. 2023. ByteTransformer: A high-performance transformer boosted for variable-length inputs. In *Proceedings of the 2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS'23)*. IEEE, 344–355.

[300] Jinchao Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2023. Draft & Verify: Lossless large language model acceleration via self-speculative decoding. *arXiv:abs/2309.08168* (2023). https://api.semanticscholar.org/CorpusID:262013673

[301] Mingyang Zhang, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. 2023. Pruning meets low-rank parameter-efficient fine-tuning. *arXiv preprint arXiv:2305.18403* (2023).

[302] Qiyang Zhang, Xiangying Che, Yijie Chen, Xiao Ma, Mengwei Xu, and Schahram Dustdar. 2024. A comprehensive deep learning library benchmark and optimal library selection. *IEEE Transactions on Mobile Computing* 23, 5 (2024), 5069–5082.

[303] Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2022. PLATON: Pruning large transformer models with upper confidence bound of weight importance. In *Proceedings of the International Conference on Machine Learning*. PMLR, 26809–26823.

[304] Zining Zhang, Bingsheng He, and Zhenjie Zhang. 2023. Practical edge kernels for integer-only vision transformers under post-training quantization. In *Proceedings of the Machine Learning and Systems Conference.*

[305] Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. MoEfication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*. Association for Computational Linguistics. 877–890.

[306] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, LIanmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, Clark Barrett, et al. 2023. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. *arXiv:2306.14048 [cs.LG]* (2023).

[307] Zhen Zhang, Shuai Zheng, Yida Wang, Justin Chiu, George Karypis, Trishul Chilimbi, Mu Li, and Xin Jin. 2023. MiCS: Near-linear scaling for training gigantic model on public cloud. *Proceedings of the VLDB Endowment* 16, 1 (2023), 37–50.

[308] Lulu Zhao, Fujia Zheng, Weihao Zeng, Keqing He, Weiran Xu, Huixing Jiang, Wei Wu, and Yanan Wu. 2022. Domain-oriented prefix-tuning: Towards efficient and generalizable fine-tuning for zero-shot dialogue summarization. *arXiv preprint arXiv:2204.04362* (2022).

[309] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. 2023. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277* (2023).

[310] Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. 2023. Atom: Low-bit quantization for efficient and accurate LLM serving. *arXiv:2310.19102 [cs.LG]* (2023).

[311] Yang Zhao, Yanwu Xu, Zhisheng Xiao, Haolin Ja, and Tingbo Hou. 2024. MobileDiffusion: Instant text-to-image generation on mobile devices. *arXiv:2311.16567 [cs.CV]* (2024). https://arxiv.org/abs/2311.16567

[312] Lianmin Zheng, Zhuohan Li, Hao Zhang, Yonghao Zhuang, Zhifeng Chen, Yanping Huang, Yida Wang, Yuanzhong Xu, Danyang Zho, Eric P. Xing, et al. 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*. 559–578.

[313] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Zozyrakis, Ion Stoica, Joseph E. Gonzalez, et al. 2023. Efficiently programming large language models using SGLang. *arXiv preprint arXiv:2312.07104* (2023).

[314] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. 2024. DistServe: Disaggregating prefill and decoding for goodput-optimized large langage model serving. In *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI'24)*. 193–210.

[315] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. BERT loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems* 33 (2020), 18330–18341.

[316] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2023. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633* (2023).

[317] 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 the Machine Learning and Systems Conference*.

[318] Bojia Zi, Xianbiao Qi, Lingzhi Wang, Jianan Wang, Kam-Fai Wong, and Lei Zhang. 2023. Delta-LoRA: Fine-tuning high-rank parameters with the delta of low-rank matrices. *arXiv:2309.02411 [cs]* (2023). http://arxiv.org/abs/2309.02411

Received 26 February 2024; revised 24 September 2024; accepted 18 November 2024