A Survey on Mixture of Experts in Large Language Models

Weilin Cai, *Graduate Student Member, IEEE*, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang, *Member, IEEE*

(Survey Paper)

***Abstract*—Large language models (LLMs) have garnered unprecedented advancements across diverse fields, ranging from natural language processing to computer vision and beyond. The prowess of LLMs is underpinned by their substantial model size, extensive and diverse datasets, and the vast computational power harnessed during training, all of which contribute to the emergent abilities of LLMs (e.g., in-context learning) that are not present in small models. Within this context, the mixture of experts (MoE) has emerged as an effective method for substantially scaling up model capacity with minimal computation overhead, gaining significant attention from academia and industry. Despite its growing prevalence, there lacks a systematic and comprehensive review of the literature on MoE. This survey seeks to bridge that gap, serving as an essential resource for researchers delving into the intricacies of MoE. We first briefly introduce the structure of the MoE layer, followed by proposing a new taxonomy of MoE. Next, we overview the core designs for various MoE models including both algorithmic and systemic aspects, alongside collections of available open-source implementations, hyperparameter configurations and empirical evaluations. Furthermore, we delineate the multifaceted applications of MoE in practice, and outline some potential directions for future research.**

***Index Terms*—Large language models, mixture of experts, gating functions.**

Received 6 July 2024; revised 15 January 2025; accepted 9 March 2025. Date of publication 25 March 2025; date of current version 28 May 2025. The work of Weilin Cai and Jiayi Huang was supported in part by the National Natural Science Foundation of China under Grant 62402411, in part by Guangdong Provincial Project under Grant 2023QN10X252, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515110353, in part by Guangzhou Municipal Science and Technology Project under Grant 2024A04J4528, and in part by Guangzhou-HKUST(GZ) Joint Funding Program under Grant 2024A03J0624. The work of Jing Tang work supported in part by the National Key R&D Program of China under Grant 2023YFF0725100 and Grant 2024YFA1012701, in part by the National Natural Science Foundation of China (NSFC) under Grant 62402410 and Grant U22B2060, in part by Guangdong Provincial Project under Grant 2023QN10X025, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515110131, in part by Guangzhou Municipal Science and Technology Bureau under Grant 2023A03J0667 and Grant 2024A04J4454, in part by Guangzhou Municipal Education Bureau under Grant 2024312263, in part by Guangzhou Municipality Big Data Intelligence Key Lab under Grant 2023A03J0012, in part by Guangzhou Industrial Information and Intelligent Key Laboratory Project under Grant 2024A03J0628, and in part by the Guangzhou Municipal Key Laboratory of Financial Technology Cutting-Edge Research under Grant 2024A03J0630. Recommended for acceptance by L. Nie. *(Weilin Cai, Juyong Jiang, and Fan Wang contributed equally to this work.) (Corresponding authors: Jing Tang; Sunghun Kim; Jiayi Huang.)*

The authors are with The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511458, China (e-mail: wcai738@connect.hkust-gz.edu.cn; jjiang472@connect.hkust-gz.edu.cn; fwang380@connect.hkust-gz.edu.cn; jingtang@hkust-gz.edu.cn; hunkim@hkust-gz.edu.cn; hjy@hkust-gz.edu.cn).

To facilitate ongoing updates and the sharing of cutting-edge advances in MoE research, we have established a resource repository at https://github.com/withinmiaov/A-Survey-on-Mixture-of-Experts-in-LLMs.

This article has supplementary downloadable material available at https://doi.org/10.1109/TKDE.2025.3554028, provided by the authors.

Digital Object Identifier 10.1109/TKDE.2025.3554028

I. Introduction

In THE current landscape of artificial general intelligence (AGI), the transformative impact of transformer-based large language models (LLMs) has permeated diverse fields such as natural language processing [1], [2], [3], [4], [5], computer vision [6], [7], and multimodality [8], [9], [10]. Building upon the foundational transformer architecture, LLMs demonstrate extraordinary capabilities, which are attributed to their sheer size, the breadth of data they are trained on, and the significant computational resources invested in their development [11], [12], [13]. Recognizing a scaling law [11], [14] that underpins their evolution, it is imperative to identify and implement efficient methodologies for the sustainable scaling of LLMs.

The concept of mixture of experts (MoE), initially introduced in [15], [16], has undergone extensive exploration and advancement as evidenced by subsequent studies [17], [18], [19], [20], [21], [22], [23]. The emergence of sparsely-gated MoE [24], particularly within the integration of transformer-based large language models [25], has brought new vitality to this three-decade-old technology. The MoE framework is based on a simple yet powerful idea: different parts of a model, known as experts, specialize in different tasks or aspects of the data. With this paradigm, only pertinent experts are engaged for a given input, keeping the computational cost in check while still benefiting from a large pool of specialized knowledge. This scalable and flexible innovation has offered an effective approach for adhering to the scaling law, allowing for increased model capacity without a corresponding surge in computational demands. As depicted in Fig. 1, MoE has maintained a robust trajectory of growth, particularly notable in 2024 with the advent of Mixtral-8x7B [26] and a variety of subsequent industrial-scale LLMs such as Grok-1 [27], DBRX [28], Arctic [29], DeepSeek-V2 [30], etc.

Despite the growing popularity and application of mixture of experts (MoE) models across various domains, comprehensive reviews that thoroughly examine and categorize advancements,

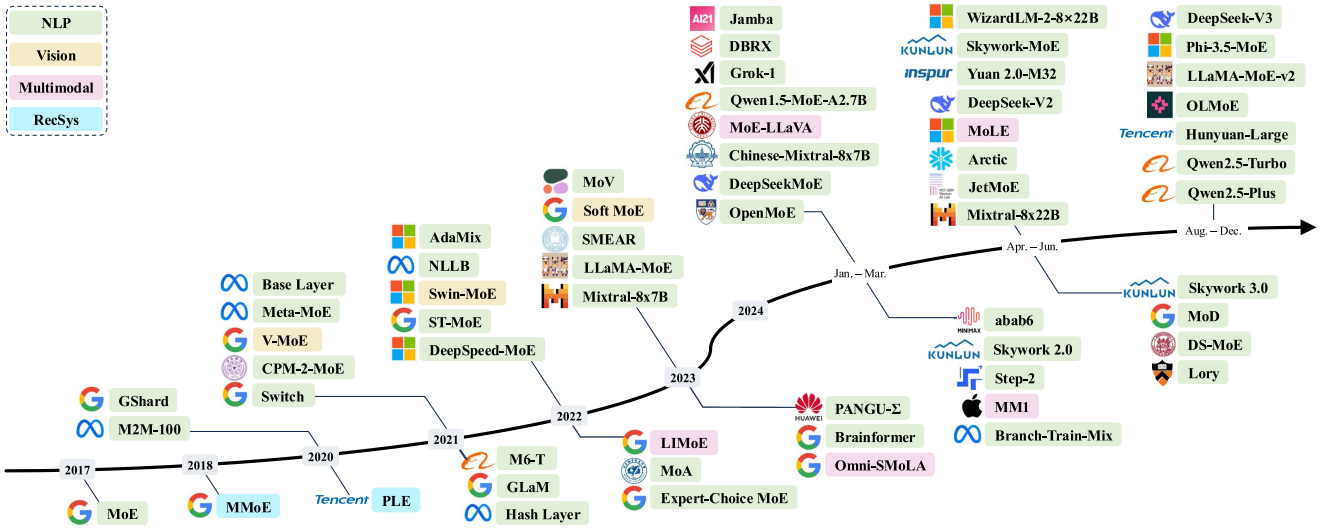


Fig. 1. A chronological overview of several representative mixture-of-experts (MoE) models in recent years. The timeline is primarily structured according to the release dates of the models. MoE models located above the arrow are open-source, while those below the arrow are proprietary and closed-source. MoE models from various domains are marked with distinct colors: Natural Language Processing (NLP) in green, Computer Vision in yellow, Multimodal in pink, and Recommender Systems (RecSys) in cyan.

particularly in the context of MoE in LLMs, remain scarce. Specifically, we identified two surveys preceding our work: the first, published in August 2012 [31], provides a comprehensive review of early studies on dense MoE, which significantly differ from the current mainstream focus on sparse MoE; the second, released in September 2022 [32], predates the major developments following the "ChatGPT moment" and, as a result, does not cover the substantial advancements and increased interest from academia and industry that have since emerged. This gap in the literature not only hinders the progress of MoE research but also limits the broader dissemination of knowledge on this topic. Our survey aims to address this deficit by providing a clear and comprehensive overview of MoE in LLMs, introducing a novel taxonomy that organizes recent progress into three categories: algorithm, system, and application.

Under this taxonomy, we first delve into MoE algorithmic advancements, particularly the prevalent substitution of feed-forward network (FFN) layers with MoE layers in transformer-based LLMs [25], [26], [30], [33], [34], [35], [36]. As each MoE layer integrates multiple FFNs—each designated as an expert—and employs a gating function to activate a selected subset of these experts, we explore the design choices of gating function and expert network, alongside collections of available open-source implementations, hyperparameter configurations, and empirical evaluations. Furthermore, to underscore the flexibility and versatility of MoE, we extend our analysis beyond the standard integration of MoE into model backbone, and discuss an array of novel MoE-related designs, such as soft MoE with token or expert merging [37], [38], [39], [40], [41], mixture of parameter-efficient experts (MoPEs) [40], [42], [43], [44], [45], [46], training and inference schemes with model transition between dense and sparse [47], [48], [49], [50], [51], [52], and various derivatives [53], [54], [55], [56], [57], [58].

With the gradual convergence of model architecture design in industrial products, system design has emerged as a pivotal factor in enhancing the quality of LLM services. Given the close association of MoE models with machine learning system design, we provide a comprehensive overview of MoE system design, including computation, communication, and storage enhancements tailored to address the unique challenges posed by the sparse and dynamic nature of its computational workload. Additionally, we overview the applications of MoE across various domains, including natural language processing, computer vision, recommender system, and multimodal contexts.

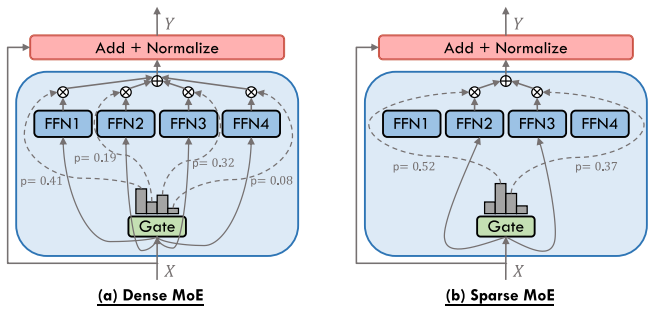


Fig. 2. An illustration of an MoE layer in Transformer-based models. For each input X, the linear-softmax gating will select all experts namely (a) Dense MoE or top-k experts namely (b) Sparse MoE to perform conditional computation. The expert layer returns the output of the selected expert multiplied by the gate score.

The remainder of this survey is organized as follows. Section II provides a foundational understanding of MoE, contrasting sparse and dense activation of experts. Section III introduces our proposed taxonomy for categorizing MoE advancements. Sections IV, V, and VI delve into the algorithmic designs, computing system support, and various applications of MoE models, respectively, following the structure outlined in our taxonomy in Fig. 3. Finally, in Section VII, we highlight the critical challenges and opportunities for bridging the research-practicality gap, culminating in Section VIII with our conclusions.

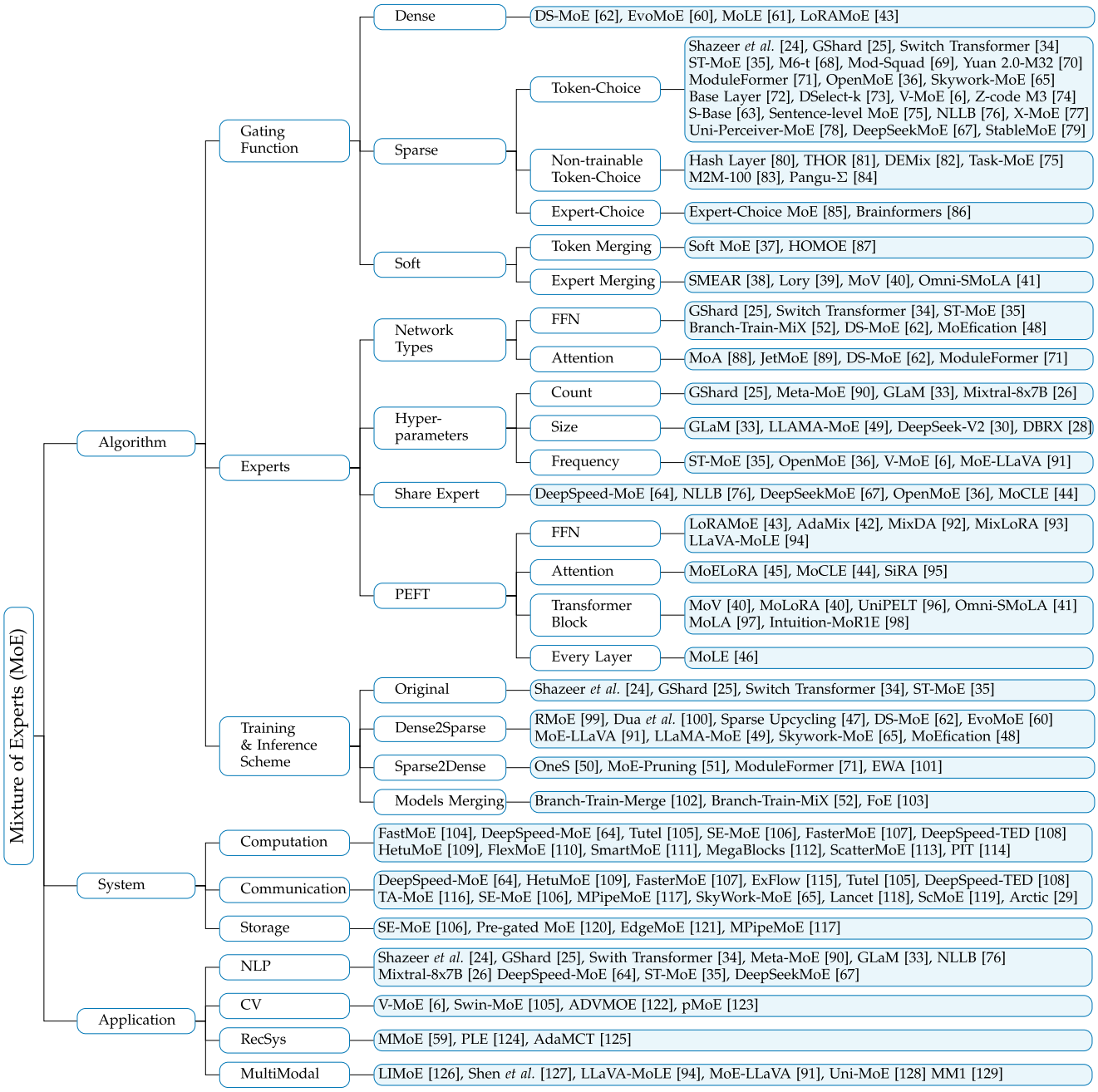


Fig. 3. Taxonomy of Mixture of Experts (MoE).

II. Background on Mixture of Experts

In transformer-based large language models (LLMs), each MoE layer typically consists of a set of N "expert networks" $\{f_1, \ldots, f_N\}$, alongside a "gating network" $\mathcal{G}$. The gating network, which often takes the form of a linear network with a softmax activation function, is to direct the input to the appropriate expert networks [24], [34]. The MoE layer is strategically placed to select the feed-forward network (FFN) within each Transformer block, typically following the self-attention (SA) sub-layer. This positioning is crucial because the FFN become increasingly computationally demanding as the model scales up. For instance, in the PaLM [3] model with the parameter number of 540B, the 90% of parameters are within its FFN layers.

Formally, each expert network f_i, usually a linear-ReLU-linear network, is parameterized by $\mathbf{W}_i$, accepting the same input $\mathbf{x}$ and generating an output $f_i(\mathbf{x}; \mathbf{W}_i)$. The gating network $\mathcal{G}$, parameterized by $\mathbf{\Theta}$ and typically consisting of a linear-softmax network, yields the output $\mathcal{G}(\mathbf{x}; \mathbf{\Theta})$. Based on the design of gating function, MoE layers can be broadly classified into two categories: dense MoE and sparse MoE, which are described in detail in the following subsections.

A. Dense MoE

The dense mixture-of-experts layer activates all expert networks $\{f_1, \ldots, f_N\}$ during each iteration. This strategy has

been extensively employed in a range of early proposals [15], [16], [17], [18], [19], [20], [21], [22], [23], [59]. Most recently, the dense MoE concept has been revisited by studies such as EvoMoE [60], MoLE [61], LoRAMoE [43], and DS-MoE [62]. The structure of the dense MoE layer is depicted in Fig. 2(a). Consequently, the output of the dense MoE layer can be formulated as

$$\mathcal{F}_{\text{dense}}^{\text{MoE}}\left(\mathbf{x};\boldsymbol{\Theta},\{\mathbf{W}_i\}_{i=1}^{N}\right)=\sum_{i=1}^{N}\mathcal{G}(\mathbf{x};\boldsymbol{\Theta})_i f_i\left(\mathbf{x};\mathbf{W}_i\right), \tag{1}$$

$$\mathcal{G}(\mathbf{x};\boldsymbol{\Theta})_i=\text{softmax}\left(g\left(\mathbf{x};\boldsymbol{\Theta}\right)\right)_i=\frac{\exp\left(g(\mathbf{x};\boldsymbol{\Theta})_i\right)}{\sum_{j=1}^{N}\exp\left(g(\mathbf{x};\boldsymbol{\Theta})_j\right)}, \tag{2}$$

where $g(\mathbf{x};\boldsymbol{\Theta})$ represents the gating value prior to the softmax operation.

B. Sparse MoE

While dense MoE typically yields higher prediction accuracy [6], it also incurs a significant increase in computational overhead. To address this, Shazeer et al. [24] introduced the sparsely-gated MoE layer, which is designed to activate only a selected subset of experts during each forward pass. This strategy achieves sparsity by computing a weighted sum of the outputs from only the top-k experts, rather than aggregating the outputs from all the experts. The structure of the sparse MoE layer is illustrated in Fig. 2(b). Building on the framework established by [24], (2) can be modified to reflect the sparsely-gated mechanism as follows:

$$\mathcal{G}(\mathbf{x};\boldsymbol{\Theta})_i=\text{softmax}\left(\text{TopK}\left(g(\mathbf{x};\boldsymbol{\Theta})+\mathcal{R}_{\text{noise}},k\right)\right)_i, \tag{3}$$

$$\text{TopK}(g(\mathbf{x};\boldsymbol{\Theta}),k)_i=\begin{cases}g(\mathbf{x};\boldsymbol{\Theta})_i, & \text{condition},\\ -\infty, & \text{otherwise.}\end{cases}, \tag{4}$$

$$\text{condition}: \text{if } g(\mathbf{x};\boldsymbol{\Theta})_i \text{ is in the top} - k \text{ elements of } g(\mathbf{x};\boldsymbol{\Theta}). \tag{5}$$

To explain, $\text{TopK}(\cdot,k)$ function retains only the top-k entries of a vector at their original values, while setting all other entries to $-\infty$. Following the softmax operation, those entries assigned $-\infty$ become approximately zero. The hyper-parameter k is selected based on the specific application, with common choices being $k=1$ [34], [63] or $k=2$ [25], [26], [33], [35], [64], [65]. The addition of a noise term $\mathcal{R}_{\text{noise}}$ is a prevalent strategy for training a sparsely-gated MoE layer, fostering exploration among experts and enhancing the stability of MoE training [24], [34].

Although the sparse gate $\mathcal{G}(\mathbf{x};\boldsymbol{\Theta})$ substantially expands the model's parameter space without a corresponding increase in computational cost, it can lead to a load balancing issue. Such an issue refers to the uneven distribution of workload across experts, with some being frequently utilized and others seldom engaged. To address this, each MoE layer incorporates an auxiliary loss function that promotes an even distribution of tokens across experts within each batch, as described in many studies [25], [26], [33], [34], [65], [66], [67]. To formulate this concept, consider a batch of queries $\mathcal{B}=\{\mathbf{x}_i,\mathbf{x}_2,\ldots,\mathbf{x}_T\}$, comprising T tokens, and N experts indexed from $i=1$ to N. Following [25], [34], the auxiliary load balancing loss for the batch is defined as

$$\mathcal{L}_{\text{load-balancing}}=N\sum_{i=1}^{N}\mathcal{D}_i\mathcal{P}_i, \tag{6}$$

$$\mathcal{D}_i=\frac{1}{T}\sum_{x\in\mathcal{B}}\mathbb{1}\{\text{argmax}\,\mathcal{G}(\mathbf{x};\boldsymbol{\Theta})=i\}, \tag{7}$$

$$\mathcal{P}_i=\frac{1}{T}\sum_{x\in\mathcal{B}}\mathcal{G}(\mathbf{x};\boldsymbol{\Theta})_i, \tag{8}$$

where $\mathcal{D}_i$ represents the proportion of tokens distributed to expert i, while $\mathcal{P}_i$ denotes the proportion of the gating probability assigned to expert i. To ensure an even distribution of the batch of tokens across the N experts, the load-balancing loss function $\mathcal{L}_{\text{load-balancing}}$ should be minimized. The optimal condition, i.e., $\min(\mathcal{L}_{\text{load-balancing}})=N\sum_{i=1}^{N}\mathcal{D}_i\mathcal{P}_i=N\sum_{i=1}^{N}\frac{1}{N}\frac{1}{N}=1$, is achieved when each expert receives an equal number of dispatched tokens $\mathcal{D}_i=\frac{1}{N}$, and an equal proportion of the gating probability $\mathcal{P}_i=\frac{1}{N}$. The balance is thus maintained across all the experts, ensuring that the workload is uniformly distributed at all times. Throughout the subsequent sections, unless explicitly stated otherwise, the term "MoE" refers to "sparse MoE".

III. Taxonomy of Mixture of Experts

To effectively scale model parameters without a corresponding increase in computational demand, MoE has emerged as a viable solution. MoE leverages a collection of specialized models and a gating mechanism to dynamically select the appropriate "expert networks" for processing a given input. This enables the model to allocate computational resources on an as-needed basis, a concept known as conditional computation. The incorporation of MoE architectures into LLMs is now a prevalent practice, allowing these models to achieve significant parameter scale-ups and consequent enhancements in capabilities [25], [26], [32], [34], [65].

For example, the Mixtral 8x7B [26], introduced by Mixtral AI, shares its foundational architecture with the earlier Mistral 7B [130], but with a notable difference: each layer comprises eight feed-forward networks (FFN) (i.e., experts). Despite utilizing only 13 billion active parameters, the Mixtral-8x7B demonstrates superior or equivalent performance to the Llama-2-70B [131] and GPT-3.5 [132] across various benchmarks. Similarly, the DeepSeek LLM [133], developed by DeepSeek, has been extended with an MoE variant known as DeepSeek-MoE [67]. The DeepSeekMoE 16B, while requiring approximately 40% less computation, attains performance on par with the Llama 2 7B [131].

To help researchers navigate the rapidly evolving landscape of MoE-equipped LLMs, we have developed a taxonomy that categorizes these models from three perspectives: algorithm, system, and application. Fig. 3 showcases our taxonomy alongside several representative studies. In the following sections, we provide a comprehensive and in-depth analysis of each category within our taxonomy.

TABLE I
OVERVIEW OF DIVERSE AUXILIARY LOSS FUNCTIONS AND THEIR TYPICAL COEFFICIENT CONFIGURATIONS

Reference	Auxiliary Loss	Coefficient
Shazeer *et al.* [24], V-MoE [6]	$\mathcal{L}_{importance} + \mathcal{L}_{load}$	$w_{importance} = 0.1, w_{load} = 0.1$
GShard [25], **Switch-T** [34], [33], [26], [28], DeepSeekMoE [67], [66], Skywork-MoE [65]	$\mathcal{L}_{aux}$	$w_{aux} = 0.01$
ST-MoE [35], [36], [88], [89]	$\mathcal{L}_{aux} + \mathcal{L}_z$	$w_{aux} = 0.01, w_z = 0.001$
Mod-Squad [69], DS-MoE [62], Moduleformer [71]	$\mathcal{L}_{MI}$	$w_{MI} = 0.001$

The originators introducing each auxiliary loss is highlighted as bolded reference, followed by references that adopts the same approach. Studies that have modified the original formulation are indicated with underlined reference.

IV. ALGORITHM DESIGN OF MIXTURE OF EXPERTS

A. Gating Function

The gating function (also known as the routing function or router), which stands as a fundamental component of all the MoE architectures, orchestrates the engagement of expert computations and the combination of their respective outputs. We categorize this mechanism into three distinct types Based on the processing methodologies of each input, we categorize the gating mechanism into three distinct types: sparse, which activates a subset of experts; dense, which activates all experts; and soft, which encompasses efficient fully-differentiable approaches.

1) Sparse: The sparse gating functions activate a selected subset of experts for processing each individual input token, which can be considered as a form of conditional computation [134], [135], [136]. The gating functions have been studied extensively, which may be trained by various forms of reinforcement learning and back-propagation, making binary or sparse and continuous, stochastic or deterministic gating decisions [20], [63], [137], [138], [139]. Shazeer et al. [24] pioneered a differentiable heuristic with auxiliary load balancing losses, in which the outputs from expert computations are weighted by their selection probabilities. This introduces a differentiable aspect to the gating process, thereby facilitating the derivation of gradients that can guide the gating function's optimization. This paradigm has subsequently become predominant in the realm of MoE research, recognized as token-choice gating.

Token-Choice Gating: Shazeer et al. [24] posited the necessity of gating inputs to the top-k experts, with $k > 1$, to enhance the efficacy of MoE. The rationale behind this approach is that by simultaneously consulting multiple experts for a given input, the network can effectively weigh and integrate their respective contributions, thereby improving performance. Subsequent research has largely affirmed that increasing the value of k enhances performance, which has led to the widespread adoption of this top-k strategy with $k > 1$. Notwithstanding, the Switch Transformer model [34] has shown that a top-1 gating strategy (as illustrated in Fig. 4(a)) can also yield competitive results, a finding that has been substantiated and adopted by later studies [63].

Auxiliary Loss for Token-Choice Gating: Token-choice gating algorithms frequently incorporate an auxiliary loss during training to promote equitable token distribution across experts. Table I shows prevalent auxiliary loss functions leveraged in the

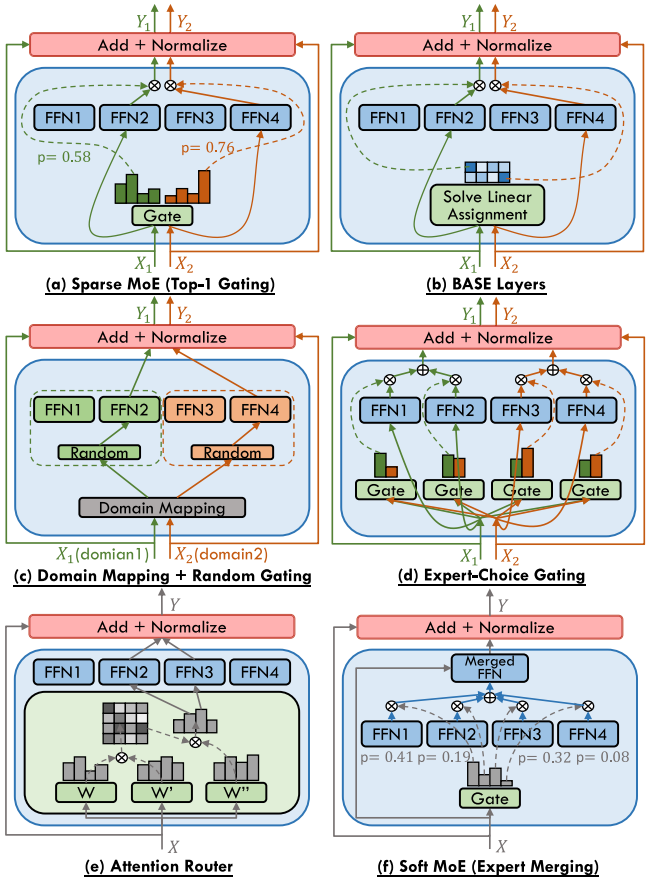


Fig. 4. The illustration of various gating functions employed in MoE models, including (a) sparse MoE with top-1 gating [34], (b) BASE layers [72], (c) the combination of grouped domain mapping and random gating [84], (d) expert-choice gating [85], (e) attention router [70], and (f) soft MoE with expert merging [38].

field. Shazeer et al. [24] quantify the importance of an expert in relation to a training batch via the batchwise sum of the gate values for that expert, defined as

$$\text{Importance}(\mathcal{B}) = \sum_{x \in \mathcal{B}} \mathcal{G}(\mathbf{x}; \mathbf{\Theta}). \tag{9}$$

Furthermore, they introduce an additional loss $\mathcal{L}_{\text{importance}}$, which is defined as the square of the coefficient of variation of the set of importance values, and can be formulated as

$$\mathcal{L}_{\text{importance}} = \text{CV}\left(\text{Importance}(\mathcal{B})\right)^2. \tag{10}$$

This loss is multiplied by a manually adjusted scaling factor $w_{\text{importance}}$, and then integrated into the overall loss function for the model, encouraging all experts to have equal importance. Although $\mathcal{L}_{\text{importance}}$ promotes balance in importance, it does not guarantee an even distribution of training examples among experts, which can lead to execution inefficiencies in distributed computing environments. To address this, they introduce a second loss $\mathcal{L}_{\text{load}}$, which employs a smooth estimator of the number of examples assigned to each expert for a batch of inputs, thereby facilitating gradient backpropagation. Simplifying the above design, GShard [25] defines a new differentiable auxiliary loss $\mathcal{L}_{\text{aux}}$ using a differentiable approximation (the

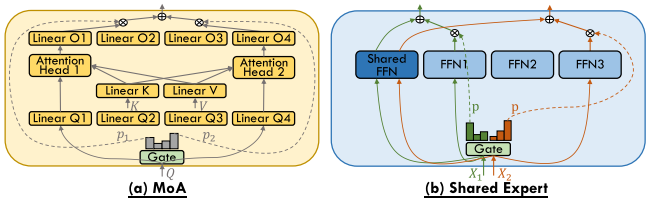


Fig. 5. The illustration of Mixture of Attention Heads [88] (a) and Shared Expert [64] (b) architectures.

dot-product of mean gates and mean gating decisions per expert). This is equivalent to $\mathcal{L}_{\text{load-balancing}}$ in (6)–(8), as detailed in Section II-B. Switch Transformers [34] and many other subsequent studies [26], [28], [33], [66] have embraced this $\mathcal{L}_{\text{aux}}$ design, and enhancements [30], [65], [67] have been made to cater to diverse requirements. Nevertheless, ST-MoE [35] identified limitations with $\mathcal{L}_{\text{aux}}$, particularly at larger scales, leading to unreliable training outcomes. To enhance training stability without compromising quality, the z-loss $\mathcal{L}_z$ is integrated, defined as

$$\mathcal{L}_z = \frac{1}{T}\sum_{x\in\mathcal{B}}\left(\log\sum_{i=1}^{N} e^{x_i}\right)^2, \tag{11}$$

which functions by penalizing large logits entering the gating network. Since this loss encourages absolute magnitude of values to be smaller, roundoff errors are reduced, which can be quite impactful for exponential functions such as gating. Additionally, Mod-Squad [69] posits the difficulty of training multi-task models under such an expert-balancing loss, which may inadvertently force experts to set parameters on conflicting tasks or hinder the potential synergies from parameter sharing across complementary tasks. Instead, it proposes to maximize the mutual information (MI) between experts and tasks to build task-expert alignment, which is accomplished through $\mathcal{L}_{\text{MI}}$. Differently, ModuleFormer [71] proposes to maximize the Mutual Information between experts and tokens. Furthermore, DS-MoE [62] extends the application of $\mathcal{L}_{\text{MI}}$, calibrating different weightings w_{MI}, in Mixture-of-Attention (MoA, as illustrated in Fig. 5(a)) and FFN MoE modules of different size models. Although existing research has introduced many different gating functions and auxiliary losses, the methods proposed by GShard [25] remain the predominant choice in industry [26], [28], [33], [66], [67].

Expert Capacity for Token-Choice Gating: In conjunction with load balancing via auxiliary loss, GShard [25] incorporates an expert capacity limit, defining a threshold for the number of tokens an expert can process. This can lead to token overflow, where excess tokens are not processed by the designated expert. GShard also proposes a random routing mechanism that selects a secondary expert with a probability proportional to its weight, under the intuition that the contribution of a secondary expert can be negligible, given that the output is a weighted average and the secondary weight is typically small. For the task of image classification with Vision Transformer (ViT) models, Riquelme et al. [6] enhance the top-k gating strategy with Batch Prioritized Routing (BPR), which assigns priority based on higher gating scores rather than the sequence order of tokens. Zoph et al. [35] have demonstrated the efficacy of BPR in the context of MoE language models. Kim et al. [74] suggest randomizing token prioritization within sequences to mitigate routing bias towards early-positioned tokens. OpenMoE [36] provides a comprehensive analysis of gating mechanisms, highlighting the "Drop-towards-the-End" phenomenon whereby tokens later in a sequence are at greater risk of being dropped due to experts reaching their maximum capacity limits, an issue that is exacerbated in instruction-tuning datasets. Moreover, OpenMoE [36] identifies a tendency within MoE systems to route tokens based on token-level semantic similarities, leading to "Context-independent Specialization".

Other Advancements on Token-Choice Gating: Despite the implementation of gating heuristics and auxiliary expert-balancing loss functions aimed at achieving a balanced workload distribution among experts, the issue of load imbalance persists as a prevalent challenge within MoE architectures. To solve it, the Balanced Assignment of Sparse Experts (BASE) layer, as conceptualized by Lewis et al. [72] and illustrated in Fig. 4(b), re-envisions the token-to-expert allocation process by casting it as a linear assignment problem, aiming to maximize the token-expert affinities under the constraints that each expert is assigned an equal quantity of tokens. Subsequently, Clark et al. [63] introduce a variant of the BASE layer, termed S-BASE, using an optimal transport formulation. Additionally, they devise a reinforcement learning based gating algorithm employing top-1 routing, with the reward function defined as the negative cross-entropy of the predicted token. The discrete optimization of gating function can lead to convergence and statistical performance issues when training with gradient-based methods. To address these issues, Hazimeh et al. [73] introduce DSelect-k, which is a smooth version of the top-k gating algorithm that improves over standard top-k gating. Kudugunta et al. [75] diverge from the prevalent token-level gating strategies by introducing a sentence-level gating mechanism. This approach involves generating a sentence representation by averaging the tokens within a sequence and subsequently routing it to an expert. Chi et al. [77] observe that prevailing gating mechanisms tend to push hidden representations clustering around expert centroids, implying a trend toward representation collapse, which in turn harms model performance. To counteract this issue, they project hidden vectors into a lower-dimensional space before gating and implement L2 normalization for both token representations and expert embeddings, thus calculating gating scores within a low-dimensional hypersphere. Yuan 2.0-M32 [70] proposes a new router network, Attention Router (as illustrated in Fig. 4(e)), which implements a more efficient selection of experts and yields an enhancement in model accuracy over classical linear router network. Zeng et al. [140] posit that the complexity of token feature abstraction may necessitate a variable number of experts to process. In response, they propose AdaMoE, a novel approach that enables token-adaptive gating, allowing for a dynamic number of selected experts per token.

Non-trainable Token-Choice Gating: The most significant benefit of non-trainable token-choice gating is that no additional gating network parameters are required and the full load balancing can be achieved through specific gating mechanisms. The Hash Layer [80] utilizes a random fixed gating approach by hashing the input token, achieving competitive results without the necessity of training the gating network. The load balancing is facilitated by the selection of hash functions prior to training, which can equitably distribute token batches. Zuo et al. [81] introduces THOR, an algorithm that randomly allocates two experts to each input during training and inference with a consistency regularized loss promoting consistent predictions. Gururangan et al. [82] propose the DEMix model, which explicitly assigns distinct experts to discrete pretraining domains, with domain matching being employed to select experts corresponding to the training inputs. Furthermore, DEMix adopts a parameter-free probabilistic method that dynamically estimates the domain-weighted mixture at inference. Kudugunta et al. [75] explore task-level gating incorporating prior knowledge tags, and similarly, M2M-100 model [83] utilizes explicit language-specific sublayers with deterministically routing input tokens based on their language. Building upon the aforementioned non-trainable gating strategies—random gating and domain mapping—PanGu-$\sum$ [84] presents the Random Routed Experts (RRE) mechanism. As illustrated in Fig. 4(c), this approach initially routes tokens to a domain-specific expert group, followed by a random selection within that group.

In contrast to explicit language-specific expert selection, NLLB [76] leverages trainable gating to manage multilingual machine translation tasks, outperforming the M2M-100 approach [83]. Addressing task interference in generalist models, Zhu et al. [78] introduce the Conditional MoE, which augments MoE with trainable gating by integrating conditional information at various levels, such as token-level, context-level, modality-level, task-level, and predefined token attributes. Ye et al. [141] further investigate the incorporation of trainable gating at task-level MoE. Additionally, STABLEMOE [79] employs a two-stage training process, with a freezed gate in the second stage.

Expert-Choice Gating: Zhou et al. [85] propose an inversion of the conventional token-choice gating paradigm, wherein each expert selects the top-k tokens they will process, as illustrated in Fig. 4(d). This approach circumvents the necessity for auxiliary load balancing losses during training, ensuring a uniform distribution of tokens across experts. Expert-choice gating demonstrates stronger empirical performance than token-choice top-2 gating in Gshard and GLaM models, as validated by Zhou et al. [85] and their subsequent Brainformers study [86]. However, OLMoE [142] finds that expert-choice gating exhibits lower accuracy compared to token-choice dropless MoE [112].

2) Dense: In Section II-A, we discuss the enduring relevance of dense MoE, which activates all the experts for each input process. This dense paradigm continues to inform current innovations in MoE training and inference methodologies, as elaborated in Section IV-D1. While sparse activation of experts, as a trade-off, may yield computational efficiency gains at the expense of some performance loss when compared to a densely activated MoE with an equivalent number of total parameters [62], [67], [71], it represents a strategic adjustment to balance computational demands with model capability. Notably, dense activation performs well in the context of MoPE, which will be discussed in Section IV-C.

3) Soft: Deciding the allocation of appropriate experts to each input token pose the fundamental discrete optimization challenge for sparse MoE. This often necessitates heuristic auxiliary losses to ensure balanced expert engagement and to minimize unassigned tokens. These issues become more pronounced in scenarios involving out-of-distribution data, such as small inference batches, novel inputs, or during transfer learning. Similar to dense MoE, the soft MoE approach maintains full differentiability by leveraging all the experts for processing each input, thus avoiding issues inherent to discrete expert selection. We distinguish soft MoE from dense MoE to highlight the characteristic that mitigates computational demands through the gating-weighted merging of input tokens or experts.

Token Merging: Puigcerver et al. [37] proposed the Soft MoE, which eschews the conventional sparse and discrete gating mechanism in favor of a soft assignment strategy that merges tokens. This method computes several weighted averages of all tokens, with weights depending on both tokens and experts, and processes each aggregate with its respective expert. Their experimental results in image classification demonstrate that soft MoE enhances the stability of gating function training and inherently maintains balance.

Expert Merging: In contrast to the merging of input tokens, Muqeeth et al. [38] introduced the Soft Merging of Experts with Adaptive Routing (SMEAR) framework, which circumvents discrete gating by merging all the experts' parameters through a weighted average, as illustrated in Fig. 4(f). By processing the input tokens through a single merged expert, SMEAR does not incur a significant increase in computational costs and enables standard gradient-based training without gradient estimation for non-differentiable gating decisions. Subsequent contributions by Zhong et al. [39] argue that SMEAR's demonstrated advantages are confined to downstream fine-tuning on classification tasks. They present Lory, an innovative approach for scaling such expert merging architectures to auto-regressive language model pretraining. Lory [39] introduces a causal segment routing strategy, conducting expert merging at the segment level while maintaining the auto-regressive nature of language models. Furthermore, it employs similarity-based data batching to direct expert specialization in particular domains or topics. Furthermore, the soft merging of experts shows strong performance in certain MoPE studies [40], [41].

B. Experts

In this section, we delineate the architecture of expert networks within MoE, following our discussion on the gating function that orchestrates the activation of these experts.

1) Network Types: Since the initial integration of MoE into transformer architectures [25], [34], [35], MoE has served as a substitute for feed-forward network (FFN) modules within these models. Typically, each expert within a MoE layer replicates

TABLE II
COMPARATIVE CONFIGURATIONS OF MOE WITH FFN EXPERTS IN SELECTED MODELS

Reference	Models	Expert Count (Activ./Total)	d_{model}	d_{ffn}	d_{expert}	#L	#H	d_{head}	Placement Frequency	Activation Function	Share Expert Count
GShard [25] (2020)	37B	2/128	1024	8192	d_{ffn}	36	16	128	1/2	ReLU	0
	150B	2/512	1024	8192	d_{ffn}	36	16	128	1/2	ReLU	0
	600B	2/2048	1024	8192	d_{ffn}	36	16	128	1/2	ReLU	0
Switch [34] (2021)	26B	1/128	1024	2816	d_{ffn}	24	16	64	1/2	GEGLU	0
	395B	1/64	4096	10240	d_{ffn}	24	64	64	1/2	GEGLU	0
	1571B	1/2048	2080	6144	d_{ffn}	15	32	64	1	ReLU	0
GLaM [33] (2021)	1.7B/27B	2/64	2048	8192	d_{ffn}	24	16	128	1/2	GEGLU	0
	8B/143B	2/64	4096	16384	d_{ffn}	32	32	128	1/2	GEGLU	0
	64B/1.2T	2/64	8192	32768	d_{ffn}	64	128	128	1/2	GEGLU	0
DeepSpeed-MoE [64] (2022)	1.3B/52B	2/128	2048	$4d_{model}$	d_{ffn}	24	16	128	1/2	GeLU	0
	PR-1.3B/31B	2/64-2/128	2048	$4d_{model}$	d_{ffn}	24	16	128	1/2, 10L-64E, 2L-128E	GeLU	1
ST-MoE [35] (2022)	0.8B/4.1B	2/32	1024	2816	d_{ffn}	27	16	64	1/4, add extra FFN	GEGLU	0
	32B/269B	2/64	5120	20480	d_{ffn}	27	64	128	1/4, add extra FFN	GEGLU	0
Mixtral [26] (2023)	13B/47B	2/8	4096	14336	d_{ffn}	32	32	128	1	SwiGLU	0
	39B/141B	2/8	6144	16384	d_{ffn}	56	48	128	1	SwiGLU	0
LLAMA-MoE [49] (2023)	3.5B/6.7B	4/16	4096	11008	688	32	32	128	1	SwiGLU	0
	3.5B/6.7B	2/8	4096	11008	1376	32	32	128	1	SwiGLU	0
DeepSeekMoE [67] (2024)	2.8B/16.4B	8/66	2048	10944	1408	28	16	128	1, except 1st layer	SwiGLU	2
	22B/145B	16/132	4096	-	$\frac{1}{8}d_{ffn}$	62	32	128	1, except 1st layer	SwiGLU	4
OpenMoE [36] (2024)	2.6B/8.7B	2/32	2048	8192	d_{ffn}	24	24	128	1/6	SwiGLU	1
	6.8B/34B	2/32	3072	12288	d_{ffn}	32	24	128	1/4	SwiGLU	1
Qwen1.5-MoE [143] (2024)	2.7B/14.3B	8/64	2048	5632	1408	24	16	128	1	SwiGLU	4
Jamba [66] (2024)	12B/52B	2/16	4096	14336	d_{ffn}	32	32	128	1/2, 1:7 Attention:Mamba	SwiGLU	0
Skywork-MoE [65] (2024)	22B/146B	2/16	4608	12288	d_{ffn}	52	36	128	1	SwiGLU	0
Yuan 2.0-M32 [70] (2024)	3.7B/40B	2/32	2048	8192	d_{ffn}	24	16	256	1	SwiGLU	0

Model differentiation in each reference is achieved by using the model size, indicated either by total or activated/total parameter count. Both activated and total expert counts encompass the count of shared experts when utilized. d_{model} is the hidden size, d_{ffn} is the intermediate size of FFNs, dexpert is the intermediate size of FFN experts, #L is the number of layers, #H and d_{head} are the number of attention heads and attention head dimensions.

the architecture of the FFN it replaces. Therefore, the structure of the MoE model, encompassing attention mechanisms and activation functions, has undergone significant changes with the advancement of the dense transformer model. This paradigm, wherein FFNs are utilized as experts, remains predominant, and subsequent refinements will be expounded upon in Sections IV-B2 to IV-B3.

Feed-Forward Network: As discussed in existing work [52], the predilection for leveraging MoE in the context of FFNs is rooted in the hypothesis that self-attention layers exhibit lower sparsity and less domain specificity than FFN layers. Pan et al. [62] provide empirical support for this, revealing marked sparsity in FFN layers compared to self-attention layers, through their analysis of downstream Wikitext tasks using their pretrained DS-MoE models. Their results indicate a mere 20% active expert engagement in FFN layers, in contrast to the 80% observed within self-attention layers. In earlier investigation of FFN computational patterns, Zhang et al. [48] observe that most inputs only activate a small proportion of neurons of FFNs, thus corroborating the inherent sparsity of FFNs.

Attention: While the focus of MoE research has predominantly been on FFN layers within the Transformer architecture, Zhang et al. [88] introduce the Mixture of Attention Heads (MoA), an innovative architecture that combines multi-head attention layers with MoE to further enhance performance and restrain computational cost. As delineated in Fig. 5(a), MoA employs two sets of experts, one for query projection and one for output projection. Both sets select the experts with the same indices through a common gating network. To reduce computational complexity, MoA shares the key and value projection weights across attention experts. Subsequent work such as DS-MoE [62], JetMoE [89], and ModuleFormer [71] follows MoA and further refines the combination of MoE and attention.

2) Hyperparameters: The scale of sparse MoE models is governed by several critical hyperparameters that extend beyond those of dense transformer models. These include (1) the count of experts per MoE layer, (2) the size of each expert, and (3) the placement frequency of MoE layers throughout the model. The selection of these hyperparameters is crucial, as it profoundly impacts model accuracy and efficiency across various tasks. Optimal hyperparameter choices are thus contingent upon the specific application requirements and the limits of the computational infrastructure. Our subsequent analysis, informed by the exemplified models listed in Table II, explores these hyperparameter decisions in depth.

Expert Count: Initial investigations employing thousands of experts per layer yielded impressive gains in pretraining and translation quality [24], [25], [34]. Nonetheless, the quality of sparse MoE models is disproportionately reduced under domain shift [90] or when fine-tuning on diverse task distributions [34]. GLaM [33] adopts a configuration of 64 experts, guided by their findings that a 64-expert setup with top-2 gating strikes an optimal balance between execution efficiency and performance across zero-shot, one-shot, and few-shot scenarios. Contrary to the earlier trend that reduced the number of experts from thousands to 64, the expert count increases from 8 to 64

TABLE III
A COLLECTION OF RECENT OPEN-SOURCE MODELS DETAILING ACTIVATED AND TOTAL PARAMETER COUNTS, ALONGSIDE PERFORMANCE BENCHMARKS SUCH AS MMLU [144] (5-SHOT), GSM8K [145] (3/5/8-SHOT), MATH [146] (4/8-SHOT), AND HUMANEVAL [147] (0-SHOT), UNLESS SPECIFIED OTHERWISE

Name	Time	Affiliation	Params.		Benchmarks				Link
			Activ.	Total	MMLU	GSM8K	MATH	HumanEval	
Mixtral-8x7B-v0.1	2023.12	Mistral	13B	47B	70.6	74.4 (8s)	28.4 (4s)	40.2	https://huggingface.co/mistralai/Mixtral-8x7B-v0.1
DeepSeekMoE-16B-Base	2024.1	DeepSeek	3B	16B	45.0	18.8 (8s)	4.3 (4s)	26.8	https://huggingface.co/deepseek-ai/deepseek-moe-16b-base
Grok-1	2024.3	xAI	86B	314B	73.0	62.9 (5s)	23.9 (4s)	63.2	https://github.com/xai-org/grok-1
Qwen1.5-MoE-A2.7B	2024.3	Alibaba	3B	14B	62.5	61.5 (8s)	-	34.2	https://huggingface.co/Qwen/Qwen1.5-MoE-A2.7B
DBRX Instruct	2024.3	Databricks	36B	132B	73.7	72.8 (5s)	-	70.1	https://huggingface.co/databricks/dbrx-instruct
Jamba-v0.1	2024.3	AI21 Labs	12B	52B	67.4	59.9 (3s)	-	29.3	https://huggingface.co/ai21labs/Jamba-v0.1
Mistral-8x22B-v0.1	2024.4	Mistral	39B	141B	77.8	88.4 (8s)	41.8 (4s)	45.1	https://huggingface.co/mistralai/Mixtral-8x22B-v0.1
Arctic Instruct	2024.4	Snowflake	17B	480B	67.3	74.2 (5s)	-	-	https://huggingface.co/Snowflake/snowflake-arctic-instruct
DeepSeek-V2	2024.5	DeepSeek	21B	236B	78.5	79.2 (8s)	43.6 (4s)	48.8	https://huggingface.co/deepseek-ai/DeepSeek-V2
DeepSeek-V2-Chat (RL)	2024.5	DeepSeek	21B	236B	77.8	92.2 (8s)	53.9 (4s)	81.1	https://huggingface.co/deepseek-ai/DeepSeek-V2-Chat
Yuan 2.0-M32	2024.5	IEIT	4B	40B	72.2	92.7 (8s)	55.9 (8s)	74.4	https://huggingface.co/IEITYuan/Yuan2-M32
Skywork-MoE-Base	2024.6	Kunlun	22B	146B	77.4	76.1 (5s)	31.9 (4s)	43.9	https://huggingface.co/Skywork/Skywork-MoE-Base

following the appearance of Mixtral-8x7B [26]. This is because several studies [28], [67], [142], [143] observe that dividing each coarse-grained expert into multiple finer-grained experts (as detailed in the next paragraph, "Expert Size") results in higher model quality. In conclusion, the most recent LLaMa-like MoE models typically select an expert count ranging from 8 to 64 [36], [65], [70]. Additionally, DeepSpeed-MoE [64] uses a Pyramid-MoE approach, positioning MoE layers with a higher expert count towards the network's end.

Expert Size: To scale the model effectively, GLaM [33] prioritizes the expansion of the intermediate hidden dimension per expert while standardizing the expert count at 64, a strategy that often requires the implementation of tensor parallelism across multiple accelerators to maintain computational efficiency [33], [34], [64]. From this period forward, MoE models [26], [28], [35], [65] typically featured larger expert dimensions. Differently, DeepSeekMoE [30], [67] introduces the concept of fine-grained expert segmentation by subdividing the intermediate hidden dimension of FFN expert, while preserving the overall parameter count. Specifically, DeepSeekMoE-145B employs a reduced intermediate hidden dimension at one-eighth that of its dense FFN counterpart, increasing both the number of experts (from 16 to 128) and the number of active experts (from top-2 to top-16) by a factor of eight. Fine-grained experts are thought to not only refine the decomposition of knowledge among experts, enabling more precise learning, but also to enhance the flexibility of expert activation combinations, which allows for more specialized and targeted knowledge capture. Qwen1.5-MoE [143], LLAMA-MoE [49], DBRX [28], and OLMoE [142] employ a similar fine-grained expert segmentation strategy and demonstrate its effectiveness.

Frequency of MoE Layers: Sparse MoE models typically evolve from dense architectures by interspersing MoE layers in place of the dense FFN layers at regular intervals. Although a higher frequency of MoE layers can enlarge the model size, it also introduces greater system overhead. In practice, most MoE models features alternate FFN replacement (1/2) with MoE layers [25], [33], [64], [90]. Nevertheless, variations exist, with some models incorporating MoE layers every fourth layer (1/4) [35], [36] or in every layer (1/1) [34], [67]. Following the introduction of LLaMa-like MoE model, Mixtral 8x7B [26], the trend seems to shift towards placing MoE in every layer of the model.

To illustrate the performance differences among various MoE models with different hyperparameter choices, we enumerate some recent open-source models in Table III, summarizing their parameter counts and benchmark performances. The performance of LLMs is influenced by numerous factors, including data, pre-training and post-training schemes, making it hard to directly correlate benchmark performance with specific configurations. However, a clear trend is that models with more activated and total parameters generally exhibit greater power, in line with the scaling law. Notably, DeepSeek-V2 [30] achieves relatively higher comprehensive score across four benchmarks despite having a moderate number of parameters, highlighting the effectiveness of its design features, including fine-grained expert segmentation and shared expert. Additionally, Qwen1.5-MoE-A2.7B [143] and Yuan 2.0-M32 [70] exhibit strong performance with relatively fewer activated parameters, suggesting they may offer cost-effective options for future expansions and research explorations.

In addition, research into the optimal configuration of MoE layers has been extensive. DeepSeekMoE-16B/-145B [67] replaces all FFNs with MoE layers, excluding the first, due to the observed slower convergence of load balance status in the first layer. MoE-LLaVA [91], a recently popular open Large Vision-Language Model (LVLM), demonstrates that alternating MoE placement yields superior model quality and execution efficiency on multimodal tasks, compared to every-layer MoE placement or "First-Half" and "Second-Half" configurations. Jamba [66] enhances efficiency by incorporating the architecture of Mamba [148], utilizing a 1:7 ratio of attention-to-Mamba layers.

3) Shared Expert: DeepSpeed-MoE [64] innovatively introduces the Residual-MoE architecture, wherein each token is processed by a fixed expert and another selected through gating, achieving two experts engagement per layer like top-2 gating. This method achieves a similar level of accuracy as top-2 gating while reducing the communication cost to that of top-1 gating. A conceptually similar approach, Conditional MoE Routing (CMR), is employed in NLLB [76], which also combines the outputs of dense FFN and MoE layers. This paradigm of integrating fixed FFN with sparse MoE, often referred to as shared expert and illustrated in Fig. 5(b), has gained traction in recent language models such as DeepSeekMoE [67], OpenMoE [36], Qwen1.5-MoE [143], and MoCLE [44], indicating its ascension

to a mainstream configuration. Instead of using a single shared expert, DeepSeekMoE [67] and Qwen1.5-MoE [143] employ multiple shared experts, due to their fine-grained expert segmentation design.

C. Mixture of Parameter-Efficient Experts

LLMs pretrained on massive generic datasets have shown impressive abilities, enabling their deployment across diverse tasks [97]. However, customizing generic LLMs for specific purposes requires fine-tuning. Traditional full fine-tuning (FT), which updates all model parameters, becomes resource-intensive as models grow larger [149]. This challenge has led to the development of parameter-efficient fine-tuning (PEFT) methods, which update only a small set of parameters, achieving remarkable performances in many NLP tasks with less computational cost [150], [151].

Despite the success, PEFT approaches often struggle with generalizing across multiple tasks due to limited trainable parameters and the potential for catastrophic forgetting [93]. To address it, a line of mixture of parameter-efficient experts (MoPE) research has emerged, focusing on integrating the MoE framework with PEFT [93], [152]. MoPE incorporates the MoE's gating mechanism and multi-expert architecture, but with each expert constructed using PEFT techniques [153]. This combination enhances PEFT's multi-task performances while maintaining efficiency by using fewer parameters than traditional MoE models [40], [95].

MoPE harnesses the best of both fields: the task versatility of MoE and the resource efficiency of PEFT [93], positioning it as a promising area of study that pushes the boundaries of both fields. In the following subsection, we will give a taxonomy of MoPE, as depicted in Fig. 6, based on their placement within the Transformer architecture. We will subsequently review recent MoPE research, summarizing the methodologies and contributions.

1) Feed-Forward Network: Following the conventional MoE structure, a series of investigations introduce the MoPE framework to the FFN layer of every Transformer block. As illustrated in Fig. 6(b), the forward process can be expressed as:

$$\mathrm{FFN}^{MoE}\left(\mathbf{x}'\right) = \mathrm{FFN}\left(\mathbf{x}'\right) + \sum_{i=1}^{n} \mathbf{x}' \Delta \mathbf{W}_i^{ffn} \cdot G^{ffn}\left(\mathbf{x}'\right)_i, \tag{12}$$

$$\mathbf{x}' = \mathrm{LayerNorm}(\mathrm{SA}(\mathbf{x}) + \mathbf{x}), \tag{13}$$

where $\Delta \mathbf{W}^{ffn}$ and $G^{ffn}(\mathbf{x})$ is the parameter-efficient expert and gating function applied to the FFN, respectively.

LoRAMoE [43] efficiently combines MoE with LoRA [154], which introduces two low-rank matrices to receive incremental updates associated with the task-specific fine-tuning. LoRAMoE plugs multiple plug-in LoRA experts and a gating mechanism into the FFN layer. To handle the data distribution diversity, LoRAMoE separates the experts into two groups: one learns various downstream tasks, and the other aligns pretrained world knowledge with human instructions. In addition, a novel localized balancing constraint loss is introduced to maintain each expert's significance within its group while optimizing group focus on distinct tasks. These designs enable LoRAMoE to resolve the knowledge forgetting issue and enhance model performance on downstream tasks. LLaVA-MoLE [94] extends MoPE to multimodal tasks by adopting multiple LoRA experts to handle inputs from different domains. This approach enforces top-1 routing to maintain computational costs close to a standard FFN with LoRA. LLaVA-MoLE is effective in addressing data conflicts and enhancing the model's versatility. MixLoRA [93] fuses different LoRA modules with a shared FFN layer to construct experts, closely aligning with the conventional MoE. Furthermore, MixLoRA implements a high-throughput framework that significantly reduces token computation latency and memory usage during training and inference, optimizing performance and efficiency. Many MoPE studies also explore integrating MoE with different PEFT methods, such as Adapter [155], AdaMix [42] and MixDA [92].

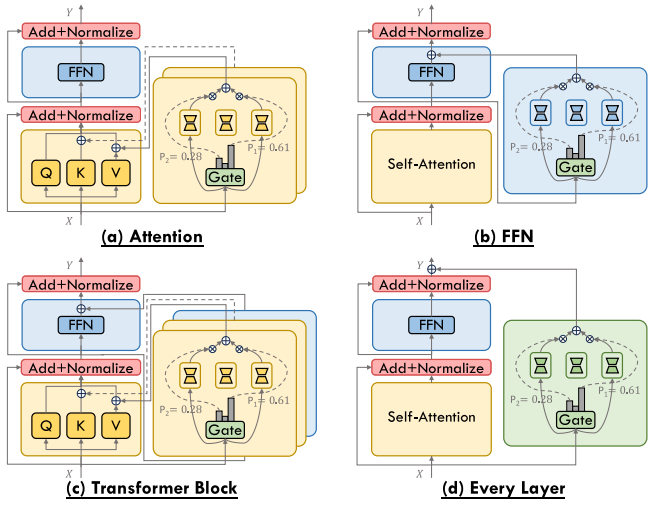


Fig. 6. The illustration of the taxonomy of MoPEs based placement within the Transformer model architecture. (a) exemplifies the integration of MoPE with the Key and Value modules of the attention mechanism, with applicability extending to Query and Output projection modules. (b) represents the application of MoPE to the FFN. (c) refers to the MoPE integration at the level of the Transformer block, wherein two distinct groups of experts are allocated to both attention and FFN, each regulated by its own gating mechanism. (d) illustrates a layer-wise integration of MoPE, in which each Transformer layer is regarded as a unified entity with a gating orchestrating the interplay among experts.

2) Attention: A branch of research has been exploring the application of the MoPE framework with the attention mechanism. These studies involve augmenting the attention mechanism with a gating network and a set of experts. The MoPE framework can be applied to the Query, Key, Value, and Output projection, individually or in various combinations, within the attention mechanism. For example, as shown in Fig. 6(a), the integration of MoPE with the Key and Value module of the attention mechanism can be formalized as follows:

$$\begin{aligned} &\mathrm{SA}^{MoE}(\mathbf{x}) \\ &= Softmax\left(\frac{\mathbf{Q}\left(\mathbf{K}^{\mathbf{T}} + \sum_{i=1}^{n} \mathbf{x}\Delta\mathbf{W}_i^k \cdot G^k(\mathbf{x})_i\right)}{\sqrt{d_{head}}}\right) \\ &\left(\mathbf{V} + \sum_{i=1}^{n} \mathbf{x}\Delta\mathbf{W}_i^v \cdot G^v(\mathbf{x})_i\right), \end{aligned} \tag{14}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ represents the Query, Key and Value modules, respectively. $\Delta\mathbf{W}^k$ and $G^k(\mathbf{x})$ denote the parameter-efficient expert and its corresponding gating function for the Key module. Similarly, $\Delta\mathbf{W}^v$ and $G^v(\mathbf{x})$ indicate the expert and the gating function for the Value module. Here, n is the number of experts, and d_{head} is the dimensions in the Multi-head Attention mechanism.

Recent research has been exploring the integration of MoE with the attention mechanism. One notable example is MoELoRA [45], which introduces multiple LoRA experts to the Query and Value matrices of the attention mechanism. It employs a contrastive learning loss to alleviate routing randomness and ensure distinct feature learning across experts. Another innovative approach, MoCLE [44], addresses task conflict by routing similar data to the corresponding task LoRA expert and a universal LoRA expert (similar to shared expert). SiRA [95] further extends the MoE-LoRA framework by incorporating a capacity constraint, auxiliary loss, and expert dropout mechanism.

3) Transformer Block: The integration of MoPE with the Transformer architecture has garnered interest in the pursuit of enhancing parameter efficiency and model performance. This approach assigns distinct groups of experts to the attention mechanism and FFN within each Transformer block, each governed by its dedicated gating mechanism. As exhibited in Fig. 6(c), the forward process under the MoPE framework integrated with the Transformer block can be denoted as:

$$y = \text{LayerNorm}\left(\mathbf{x}' + \text{FFN}^{MoE}\left(\mathbf{x}'\right)\right), \tag{15}$$

$$\mathbf{x}' = \text{LayerNorm}\left(\text{SA}^{MoE}(\mathbf{x}) + \mathbf{x}\right). \tag{16}$$

MoV [40] introduces tunable $(IA)^3$ [156] vectors as experts, adjusting the Key and Value modules of the attention mechanism and the activation within the FFN. MoV only tunes the lightweight experts and gatings, significantly reducing the computational burden and memory footprint. Similarly, MoLORA [40] employs multiple LoRA experts within the Transformer block, achieving performance parity with FT while maintaining high parameter efficiency. UniPELT [96] proposed a hybrid framework that combines Adapter [155], Prefix-tuning [157], and LoRA [154] as experts. UniPELT leverages different gating mechanisms to dynamically activate the experts, efficiently finding the approaches that best suit the given task. Omni-SMoLA [41] extends the MoPE with three sets of LoRA experts, each tailored to handle text tokens, visual tokens, and multimodal tokens, respectively. The specialization feature of Omni-SMoLA can enhance model performance on various vision-and-language tasks. Aiming to explore the expert allocation issue, MoLA [97] introduces a LoRA-MoE framework with a Layer-wise Expert Allocation. The allocation strategy enables flexible employment of varying numbers of experts across different layers, improving the performance of the LoRA-MoE framework. Intuition-MoR1E [98] fuses the semantic clustering of data into the MoE-based LLM for boosting the gating efficacy, and proposes a novel rank-1 expert architecture that enables the flexible combination of down and up projections of LoRA to construct experts.

4) Every Layer: Recent advancements consider the Transformer layer as an integrated whole, applying the MoPE framework to the entire Transformer layer. As illustrated in Fig. 6(d), the forward process under the MoPE framework integrated with every layer is as follows:

$$y = \text{LayerNorm}\left(\mathbf{x}' + \text{FFN}\left(\mathbf{x}'\right)\right) + \sum_{i=1}^{n} \mathbf{x}\Delta\mathbf{W}_i^{layer} \cdot G^{layer}(\mathbf{x})_i, \tag{17}$$

$$\mathbf{x}' = \text{LayerNorm}\left(\text{SA}(\mathbf{x}) + \mathbf{x}\right), \tag{18}$$

where $\Delta\mathbf{W}^{layer}$ and $G^{layer}(\mathbf{x})$ is the parameter-efficient expert and gating function applied to the entire layer, respectively. For instance, MoLE [46] provides innovative insights into MoPE from a layer-wise level. MoLE integrates a set of trained LoRAs and a gating function into each layer. It treats each layer of trained LoRAs as an individual expert and only trains the gating to learn the optimal composition weights for a specified domain. This enables dynamic adjusting of layer-specific weights, extending the model versatility.

D. Training & Inference Scheme

The architectural advancements of MoE models have been complemented by extensive research into training and inference schemes, with the objective of synergizing the strengths of dense and sparse models while mitigating their respective weaknesses.. Original training & inference schemes, as established in seminal works [6], [24], [25], [34], [35], involve constructing an MoE model and training it from scratch, with inference directly following the model configurations of training. We divide the emerging schemes into three distinct categories: Dense-to-Sparse, which entails initiating with dense model training and progressively transitioning to a sparse MoE configuration; Sparse-to-Dense, which involves degrading a sparse MoE model to a dense form that is more efficient for inference; and Expert Models Merging, a process of integrating multiple pretrained dense expert models into a unified model.

1) Dense-to-Sparse: Komatsuzaki et al. [47] highlight the efficiency of sparse models in terms of quality and computational cost, yet acknowledge their significant data requirements and the expense of training from scratch at scale. To address this, they introduce a scheme termed "sparse upcycling," which leverages pre-existing training investments by initializing a sparsely activated MoE model from a pretrained dense checkpoint. The new MoE layers are populated with identical copies of the original dense model's FFN layers, and the gating mechanism weights are initialized randomly. Building upon the sparse upcycling technique [47], the Skywork-MoE model [65] leverages the foundational architecture of its pre-developed Skywork-13B model [158], adopting its dense checkpoints as a foundation for initial states. Similar methods are explored by RMoE [99] and Dua et al. [100].

Nie et al. [60] present EvoMoE, an efficient end-to-end MoE training framework. EvoMoE decouples the joint learning of experts and the sparse gate, emphasizing the acquisition of

foundational knowledge through a single expert at the inception of training. Subsequently, it spawns multiple diverse experts and advances the diversification of experts by training with the novel Dense-to-Sparse gate (DTS-Gate). A similar strategy is employed in the development of the MoE-LLaVA [91] large vision-language model, which commences with a dense model, subsequently multiplies the feed-forward network (FFN) to create expert initializations, and proceeds to train exclusively the MoE layers, while keeping the remaining model components static.

Pan et al. [62] posit that the parameter inefficiency observed in MoE models stems from conventional sparse training methodologies, where only a selected group of experts is engaged and refined for each input token. To counteract this, they introduce a hybrid framework for MoE models, denoted as DS-MoE, which integrates dense training (activating all the experts) with sparse inference (sparse expert activation) to achieve higher computation and parameter efficiency. MoEfication [48] investigates various strategies for expert construction modularizing the FFN into a sequence of smaller FFNs, including random splitting, parameter clustering, and building co-activation graphs. MoEBERT [159] implements an importance-based method for adapting FFNs into experts within BERT models. LLaMA-MoE [49] conducts an extensive examination of different expert construction methods, ultimately proposing a straightforward random division approach that partitions the parameters of FFNs into non-overlapping experts.

2) Sparse-to-Dense: Switch Transformer [34] studies the distillation of large sparse models into smaller dense counterparts to achieve parameter efficiency for deployment. The study reports that initializing the corresponding weights of dense model from non-expert layers of MoE model modestly enhances performance, facilitated by the consistent dimension of non-expert layers. Similarly, Xue et al. [50] address the challenges of overfitting, deployment difficulty, and hardware constraints associated with sparse MoE models. Drawing inspiration from human learning paradigms, they propose a new concept referred to as 'knowledge integration' aimed at creating a dense student model (OneS) that encapsulates the expertise of a sparse MoE model. Further investigations into MoE model distillation are also conducted by other researchers [64], [76].

Chen et al. [51] highlight the challenges associated with deploying MoE models on resource-constrained platforms, such as cloud or mobile environments. Observing that only a fraction of experts contribute significantly to MoE fine-tuning and inference, they propose a method for the progressive elimination of non-essential experts. Similarly, ModuleFormer [71] applies a comparable pruning technique, removing task-unrelated experts for a lightweight deployment. Huang et al. [101] assign tokens to experts using a random uniform partition and incorporate experts weights averaging on these MoEs at the end of each training iteration, then revert the model to dense for inference.

3) Expert Models Merging: Li et al. [102] introduce the Branch-Train-Merge (BTM) algorithm, a method for the communication-efficient training of language models (LMs). BTM independently trains a set of expert LMs (ELMs), each tailored to a specific domain within the training corpus, such as scientific or legal text. These ELMs, which operate without shared parameters, can be ensembled or parameter-averaged at inference to coalesce into a singular LM. Expanding on this concept, Sukhbaatar et al. [52] present Branch-Train-MiX (BTX), designed to combine the strengths of BTM and MoE while addressing their respective limitations. BTX maintains separate training for multiple expert LLMs, akin to BTM, but subsequently integrates these experts within a unified MoE model. Specifically, it consolidates the FFNs from all ELMs into a singular MoE module at each layer, with a gating network determining the appropriate FFN expert for each token. Other components, such as the self-attention layers from ELMs, are merged by averaging their weights. Wang et al. [103] explore the Fusion of Experts (FoE) challenge, which aims to integrate outputs from expert models that provide diverse but complementary insights into the data distribution, formulating it as supervised learning task.

V. System Design of Mixture of Experts

While MoE has been increasingly leveraged to enhance the capabilities of LLMs, its adoption introduces new challenges to existing training and inference systems, due to the inherently sparse and dynamic nature of its computational workload. GShard [25] introduces expert parallelism that implements parallel gating and expert computation by dispatching partitioned local tokens with load balancing limit of expert capacity. Since then, expert parallelism has emerged as a fundamental strategy to facilitate efficient scaling of MoE models. This approach can be viewed as an augmentation of data parallelism [160], [161], [162], where each expert in an MoE layer is assigned to a distinct device, while all non-expert layers are duplicated across devices. As depicted in Fig. 7(a), the process flow of expert parallelism consists of the following sequential operations: gate routing, input encode, All-to-All dispatch, expert computation, All-to-All combine, and output decode. Input encode is employed to aggregate the input tokens of a same expert into a contiguous memory space, as determined by the token-expert mapping from gate routing. Subsequent All-to-All dispatch is employed to send the input tokens to their corresponding experts across the distributed devices. Following the localized computation by the experts, the All-to-All combine and output decode reinstate the original data layout according to the gating indices.

Furthermore, the synergy of expert parallelism [34], [105], [108], [163], [164] with other existing parallel strategies (tensor [165], [166], [167], pipeline [168], [169], [170], sequence parallelism [171], [172], [173]) has been investigated to enhance the scalability and efficiency of MoE models in large-scale distributed environments. As shown in Fig. 7, we illustrate several examples of hybrid parallelism, encompassing (b) data + expert + tensor parallelism [34], [64], [105], [108], [111], (c) data + expert + pipeline parallelism [105], [107], [111], and (d) expert + tensor parallelism [65]. It is imperative to recognize that the choice of distributed parallelism strategies influences a complex interplay between computation efficiency, communication overhead, memory occupation, potentially affected by various hardware configurations. Notably, since both

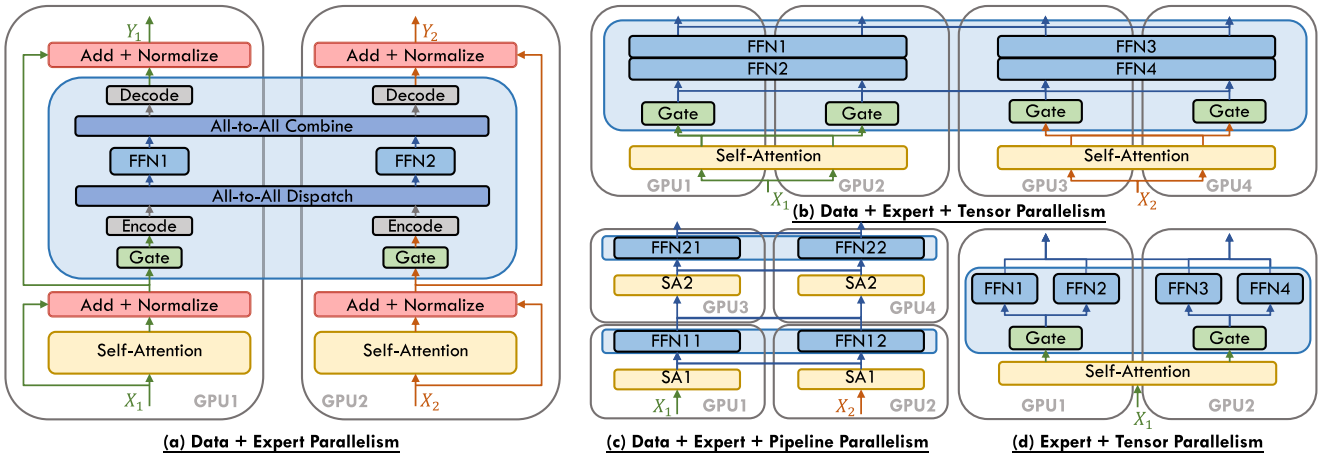


Fig. 7. Schematic depiction of diverse parallel strategies for MoE.

TABLE IV
COMPARATIVE OVERVIEW OF THE OPEN-SOURCE MOE SYSTEM FRAMEWORKS, ARRANGED CHRONOLOGICALLY BY REFERENCE PUBLICATION DATE FROM NEWEST TO OLDEST

Reference	Affiliation	Optimizations			Link	Star
		Computation	Communication	Storage		
OpenMoE [36]	Colossal-AI	✓	✓		https://github.com/hpcaitech/ColossalAI	38K
ScatterMoE [113]	Mila Quebec	✓			https://github.com/shawntan/scattermoe	140
Megablocks [112]	Stanford University	✓			https://github.com/stanford-futuredata/megablocks	1.1K
Tutel [105]	Microsoft	✓	✓		https://github.com/microsoft/tutel	672
SE-MoE [106]	Baidu	✓	✓	✓	https://github.com/PaddlePaddle/Paddle	21K
HetuMoE [109]	Peking University	✓	✓		https://github.com/PKU-DAIR/Hetu	236
Deepspeed-MoE [64]	Microsoft	✓	✓		https://github.com/microsoft/DeepSpeed	33K
FastMoE [104]	Tsinghua University	✓	✓		https://github.com/laekov/fastmoe	1.4K
Fairseq [90], [174]	Meta				https://github.com/facebookresearch/fairseq/tree/moe	29K
Mesh-TensorFlow [175]	Google				https://github.com/tensorflow/mesh	1.6K

tensor parallelism and expert parallelism effectively distribute model parameters and reduce memory costs, expert parallelism is generally preferred in practical MoE models. For instance, DeepSeek-V3 [30] employs a parallel strategy similar to that shown in Fig. 7(c) for distributed training. This preference is due to the fact that tensor parallelism typically incurs a relatively higher communication overhead. Consequently, the deployment strategies for applications require nuanced trade-offs and bespoke designs tailored to use-case scenarios.

In the subsequent discussion, we delineate the challenges introduced by MoE models from computation, communication, and storage aspects, concurrently reviewing existing research addressing these issues. Table IV shows an overview of the open-source MoE frameworks.

A. Computation

Despite MoE is designed to scale model parameters efficiently without increasing computational demand, it encounters challenges pertaining to computational efficiency. One concern is the imbalance of computational load across distributed devices employing expert parallelism, which incurs significant synchronization overhead as the system awaits the processing completion of the most heavily loaded expert. Such issues are typically addressed through algorithmic strategies, such as optimized gating mechanisms and expert capacity adjustments, as discussed in Section IV-A. Besides, solutions like SE-MoE [106], Tutel [105], FlexMoE [110] and SmartMoE [111] have introduced dynamic expert placement strategies to distribute the workload as equally as possible among devices. Additionally, FasterMoE [107] has implemented a novel dynamic shadowed expert strategy, replicating experts on multiple devices to mitigate severe load imbalance. These model placement related strategies impact both computation and communication efficiency.

Another concern is that MoE introduces additional computational overhead through operations including gate routing, input encode, and output decode. Unlike expert computations, which mirror operations in dense models and benefit from extensive optimization on prevalent hardware such as GPUs, these MoE operations are characterized by redundant computation and memory movement, resulting in low efficiency on computing devices. Therefore, recent studies like DeepSpeed-MoE [64], FastMoE [104], HetuMoE [109] and Tutel [105] have focused on the development of tailored GPU kernels to enhance the operation efficiency.

In contexts where multiple experts are deployed on a single GPU device, MegaBlocks [112] reformulates MoE computation in terms of block-sparse operations, developing specialized block-sparse GPU kernels that efficiently handle the dynamic workloads without dropping tokens. Zheng et al. [114] propose PIT, a deep-learning compiler tailored for dynamic sparsity of MoE, which can find feasible PIT rules for all the operators

within a model and generate optimized GPU kernels for them. Despite these advancements, Tan et al. [113] highlight remaining optimization potential within current MoE frameworks such as MegaBlocks and PIT, which commence with an initial scatter-to-group data copy that increases memory footprint and requires a translation of the MoE problem into the sparse matrix format. Although this translation contributes minimally to computation overhead, it imposes limitations on the transparency and adaptability of extending MegaBlocks to modules beyond the FFN. To address these issues, Tan et al. [113] propose ScatterMoE, a MoE implementation designed to effectively minimize the memory footprint. ScatterMoE leverages ParallelLinear, a linear module capable of executing grouped matrix operations on scattered groups.

B. Communication

In expert parallelism, the quadruple invocation of All-to-All communication during both the forward and backward propagation phases within each MoE layer causes a significant overhead, even emerging as the primary constraint on efficiency. The All-to-All communication paradigm encompasses both intra-node (via PCIe, pre-4th-generation NVLink) and inter-node (Ethernet, Infiniband, 4th-generation NVLink) communication channels. The efficiency of such communication is contingent upon a multitude of factors, including the heterogeneity of channel bandwidths, network topology, and the collective communication algorithms. Moreover, load imbalances intrinsic to MoE may exacerbate these inefficiencies by inducing synchronization delays.

To optimize the use of high intra-node bandwidth and low inter-node bandwidth, DeepSpeed-MoE [64] and Hetu-MoE [109] have introduced a hierarchical All-to-All communication strategy that enhances intra-node process and reduces inter-node data exchanges. Besides, FasterMoE [107], TA-MoE [116] and SE-MoE [106] have introduced topology-aware routing strategies aimed at mitigating cross-node expert selection, thereby reducing inter-node communication burdens. Additionally, ExFlow [115] exploits expert affinity, anticipating expert allocation across layers to maximize the retention of token processing within local GPU confines. The strategic allocation of experts to minimize network traffic and leverage high-bandwidth connections is a prevalent approach in distributed MoE system [64], [65], [108]. Moreover, this is often integrated with the placement design of non-expert modules to optimize overall system performance.

Given the concurrent feature of communication and computation, pipelining [168], [169], [170] is commonly employed to overlap their execution, thereby reducing the total time cost. This technique, which is integrated in systems such as Tutel [105], FasterMoE [107], and MPipeMoE [117], orchestrates overlapping between All-to-All communication and expert computation. Notably, Lancet [118] partitions non-MoE computations and integrates them into the pipeline during forward pass, and strategically schedules gradient weight computations to augment overlap in the backward pass. With the same objective of extending the overlap duration, ScMoE [119] restructures the MoE architecture to simultaneously process representations from preceding layers while engaging with current-layer representations. This decoupling of communication dependencies facilitates substantial, and in certain cases, complete overlapping between communication and computation. Snowflake Arctic [29] employs a similar design, utilizing a Dense-MoE hybrid transformer architecture to effectively overlap communication with computation.

C. Storage

The ever-increasing parameters in MoE models exacerbate the constraints posed by memory capacity in compute devices, a challenge already pronounced in dense models. While expert parallelism offers a mitigation strategy through the distribution of experts across multiple devices, individual devices may still struggle to accommodate numerous experts, particularly in inference contexts where device capacity——such as that of edge devices (PCs, smartphones, IoTs)——is inherently more restricted.

Considering the hierarchical storage pyramid, solutions like SE-MoE [106], Pre-gated MoE [120], and EdgeMoE [121] selectively retain only essential non-expert parameters and the active expert parameters within the GPU's High-Bandwidth Memory (HBM), offloading inactive expert parameters to CPU memory or SSDs. These patterns incur additional overhead from data transfer across the storage hierarchy, thus they integrate expert selection forecasting and expert parameter prefetching techniques to overlap parameter access with computation.

VI. Applications of Mixture of Experts

In the current landscape dominated by Transformer-based LLMs, the MoE paradigm offers a compelling method to significantly expand model capacity while avoiding a corresponding surge in computational demands during training and inference phases. These models have been instrumental in enhancing the performance of LLMs across a spectrum of downstream tasks, with some applications achieving results that eclipse human performance [26], [32], [67]. Rumors suggest that the proprietary GPT-4 may employ an MoE architecture with an array of 8×220B experts, trained on diverse datasets and tasks, and utilizing a 16-iteration inference process.[1] Given these, MoE has garnered widespread adoption across fields such as natural language processing, computer vision, recommender systems, multimodal applications, finance, traffic prediction, healthcare, and others. The essence of these applications lies in leveraging conditional computation to significantly boost the number of model parameters, thereby augmenting model capacities with a fixed computational cost, or implementing dynamic expert selection through gating mechanisms for efficient multi-task or multi-modal data learning. In the following, we examine specific applications and case studies of MoE models, aiming to provide a comprehensive understanding of how MoE can be effectively utilized and to demonstrate how MoE models address specific tasks across various real-world scenarios.

[1] https://x.com/soumithchintala/status/1671267150101721090

Natural Language Processing: The integration of MoE architectures with LLMs has unlocked extraordinary capabilities in a range of natural language understanding (NLU) and generation (NLG) tasks, including machine translation [24], [76], open-domain question answering [33], [90], code generation [26], [65], [67], [176], and mathematical problem-solving [26], [30], [67], [176]. The methods of integrating MoE into LLMs have been thoroughly discussed and analyzed in Section IV (algorithm design) and Section V (system design), and will not be reiterated in depth here.

Computer Vision: The great success of sparsely-gated MoE networks in NLP has inspired their application in computer vision. For example, Riquelme et al. [6] introduced Vision MoE (V-MoE), which incorporates a sparsely activated mixture of MLPs into selected ViT [177] blocks. In image recognition tasks, V-MoE rivals the performance of state-of-the-art networks while requiring substantially less computational power during inference. This demonstrates the potential of MoE to discern distinct image semantics through specialized experts. Hwang et al. [105] develop Tutel, a scalable system design and implementation for MoE with dynamic parallelism and pipelining, which they demonstrate with SwinV2-MoE, built upon Swin Transformer V2 [7]. Moreover, Zhang et al. [122] explore adversarial robustness in CNN-based MoE models, proposing a novel router-expert alternating adversarial training framework called ADVMOE. In most recent work, Chowdhury et al. [123] introduce the concept of patch-level routing in MoE (pMoE) that segments each input image into n patches (or tokens) and allocates l patches ($l \ll n$) to each expert for processing through prioritized routing to enhance efficiency.

Recommender System: Recommender systems are quintessential in various large-scale applications where they are required to balance and optimize multiple objectives simultaneously [178]. A prime example is in the domain of movie recommendations, where the aim is not only to suggest movie that align with users' immediate preferences but also to ensure subsequent user satisfaction for the selected movies [59]. The effectiveness of multi-task models hinges on the intricate interplay between task-specific goals and the relationships between tasks. MoE models with gating mechanisms have emerged as a popular paradigm for tackling the complexities of multi-task learning in recommender systems. Ma et al. [59] introduce the multi-gate mixture-of-experts (MMOE) approach, which capitalizes on the concept of shared expert submodels across all tasks, guided by a gating network tailored to each individual task. Addressing the "seesaw phenomenon" where the improvement of one task's performance can detrimentally affect another is another challenge in multi-task learning. To counteract this, Tang et al. [124] propose the Progressive Layered Extraction (PLE) model for personalized recommendations. PLE distinctly segregates shared and task-specific components and employs a progressive routing mechanism to incrementally extract and refine the semantic knowledge, thereby enhancing the efficacy of joint representation learning and the routing of information across tasks. Recently, in the pursuit of capturing both the long-term and short-term user preferences that are particularly salient in sequential recommendation scenarios, a method named AdaMCT [125] has been proposed, which utilizes layer-aware adaptive mixture units to dynamically blend CNN and Transformer experts.

Multimodal Applications: Multimodal models are designed to process and integrate various data types within a single neural network framework [179]. These models often simultaneously encompass two primary data modalities: images and text [180], [181], [182]. The MoE architecture has gained considerable traction as the foundation of multimodal models due to its capacity for expert layers to learn distinct modality partitioning [126]. One notable implementation of this approach is the LIMoE model [126], a sparse mixture of expert models tailored for multimodal learning. LIMoE is trained on both images and text data, employing contrastive loss and an entropy-based regularization technique to address load balancing challenges inherent in MoE systems. Subsequently, Shen et al. [127] and Lin et al. [91] have further investigated the potential of MoE for scaling vision-language models, offering valuable insights that contribute to the development of more efficient and effective multimodal learning systems. Furthermore, to mitigate data conflicts, LLaVA-MoLE [94] deploys a set of LoRA experts, specifically for the MLP layer, combined with a top-1 gating mechanism to refine instruction tuning in Multimodal Large Language Models (MLLMs). While the MLLMs employing MoE architectures have demonstrated impressive performances, they generally involve a limited number of experts and modalities [128]. To address this limitation, Li et al. [128] introduce the pioneering Uni-MoE, a unified MLLM with LoRA MoE architecture capable of managing an extensive range of modalities.

Due to page limitations, more applications related to finance, traffic prediction, healthcare, and others are shown in Appendix.

VII. Challenges & Opportunities

MoE models present a compelling approach for significantly increasing model capacity at a constant computational cost. Despite their promise, several intrinsic challenges remain, necessitating further collaborative design and engineering across algorithm, system, and application aspects. In this section, we identify critical challenges and promising directions for future investigation as follows:

Training Stability and Load Balancing: MoE models that utilize sparse gating have become a popular means to expand model capacity without proportionally increasing computational demands. However, the discrete nature of assigning a fixed number of experts to tokens leads to significant challenges in maintaining balanced workloads of experts and training stability across varying inputs [24], [25], [34], [54], [85]. Load imbalances, where certain experts become over-utilized while others are underutilized can hinder expert specialization and further degrade model performance. Although current efforts [25], [26], [33], [34], [65], [66], [67] have attempted to address this challenge by incorporating auxiliary loss functions to encourage even token distribution across experts, these solutions can still lead to training instability [35] and often neglect the relative importance of different tokens [85]. Therefore, future studies should focus on more effective regularization techniques [35] or

innovative gating algorithms [38], [39], [54], [85] that encourage equitable load distribution among experts and enhance model training stability.

Scalability and Communication Overhead: As the escalating sizes of LLMs with MoE necessitate more expansive distributed systems, the imperative for efficient communication during model training becomes increasingly critical, as elaborated in Section V-B. The trade-off between model complexity, indicated by the number of parameters, and the communication overhead represents a significant bottleneck in distributed training processes [25]. To address these challenges, it is essential to develop and implement effective strategies that enhance the efficiency of information transfer from system aspect or streamline information exchange without compromising model performance from algorithm aspect. Innovations such as DeepSpeed [64], FasterMoE [107], and ScMoE [119] are at the forefront of minimizing communication overhead. For example, the shared expert approach [29], [64], [67], [143], advancing MoE with parameter-sharing frameworks, holds promise for reducing the volume of data transmitted between distributed systems while concurrently enhancing model performance in natural language processing tasks. Such innovations are pivotal in facilitating more scalable and efficient distributed training architectures for MoE models.

Expert Specialization and Collaboration: Expert specialization refers to the concept where each expert develops non-overlapping and focused knowledge. Encouraging experts to concentrate their skills on distinct sub-tasks or domains has been shown to enhance the performance and generalization of the MoE model. The prevailing strategy involves designating a select number of experts as shared ones, with the goal of capturing commonalities in knowledge and reducing redundancy among those experts that are routed dynamically [36], [64], [67], [143]. However, fostering effective collaboration among these specialized experts is an ongoing challenge. Relying solely on a sparsely computed weighted sum of outputs from the top-k experts can overlook the intricate internal relationships that exist across the entire experts. Consequently, exploring new mechanisms for enhancing both the specialization and collaboration among experts is crucial for improving MoE models.

Interpretability and Transparency: The inherent complexity of MoE models, coupled with their dynamic gating of inputs to specialized experts, poses significant challenges to interpretability. This becomes particularly problematic in contexts where comprehending the rationale behind the model's decisions is essential. Enhancing the interpretability of MoE models is therefore critical, not only to facilitate a clearer understanding of their decision-making processes but also to address underlying challenges such as load balancing [25], [33], [34] and the mitigation of knowledge redundancy [67], [143]. In light of these considerations, there is a pressing need for future studies focused on the development of methods and tools that can effectively visualize and explain the behavior of individual experts within MoE models, as well as the nature of their interactions. Such advancements would significantly improve our grasp of MoE models and bolster their ongoing development, ensuring their gating decisions are transparent and trustworthy.

Optimal Expert Architecture: The design of MoE architectures, encompassing the selection of network types and the quantity of experts, significantly influences the efficacy of multi-task learning across various domains. A plethora of network architectures has been adopted as experts, including LSTM [24], CNN [122], [123], FFNs (MLPs) [25], [34], [35], [62], Attention [71], [88], and LoRA [43], [45], [93]. Among these, FFNs as experts remain the most prevalent. Despite their considerable achievements, the exploration of various hybrids of network types within experts (as the distinct features processing capabilities of different network architectures), the strategic allocation of a varying number of experts across different layers, as well as the development of innovative expert architectures, remains nascent areas of research. For example, the development of automated architecture search methods specifically designed for MoE models is imperative [86]. Such approaches could systematically identify optimal configurations, balancing the trade-offs between computational efficiency and model quality.

VIII. Conclusion

In this survey, we present a comprehensive review of the literature on MoE, serving as a valuable compendium for researchers exploring the landscape of MoE technologies. We introduce a new taxonomy for MoE and provide an in-depth analysis that encompasses three distinct vantage points: algorithm design, system design, and practical applications, complemented by a curated collection of open-source implementations, detailed hyperparameter configurations, and thorough empirical assessments. Moreover, we highlight the critical challenges faced in the field and outline the most promising avenues for future investigation. To support the continuous dissemination of knowledge and advancements, we have established a dedicated resource repository to facilitate ongoing updates and the sharing of cutting-edge developments in MoE research. We hope this survey can contribute to an essential reference for researchers seeking to rapidly acquaint themselves with MoE models, and that it will actively contribute to the vibrant progression.

References

[1] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.
[2] T. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.
[3] A. Chowdhery et al., "PaLM: Scaling language modeling with pathways," *J. Mach. Learn. Res.*, vol. 24, no. 240, pp. 1–113, 2023.
[4] J. Achiam et al., "GPT-4 technical report," 2023, *arXiv:2303.08774*.
[5] J. Jiang, F. Wang, J. Shen, S. Kim, and S. Kim, "A survey on large language models for code generation," 2024, *arXiv:2406.00515*.
[6] C. Riquelme et al., "Scaling vision with sparse mixture of experts," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 8583–8595.
[7] Z. Liu et al., "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 10012–10022.
[8] J. Lu, D. Batra, D. Parikh, and S. Lee, "ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 13–23.
[9] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *Int. J. Comput. Vis.*, vol. 130, no. 9, pp. 2337–2348, 2022.

[10] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, "MiniGPT-4: Enhancing vision-language understanding with advanced large language models," 2023, *arXiv:2304.10592*.

[11] J. Kaplan et al., "Scaling laws for neural language models," 2020, *arXiv:2001.08361*.

[12] J. Wei et al., "Emergent abilities of large language models," 2022, *arXiv:2206.07682*.

[13] K. M. Yoo et al., "HyperCLOVA X technical report," 2024, *arXiv:2404.01954*.

[14] J. Hoffmann et al., "Training compute-optimal large language models," 2022, *arXiv:2203.15556*.

[15] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, 1991.

[16] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Comput.*, vol. 6, no. 2, pp. 181–214, 1994.

[17] R. Collobert, S. Bengio, and Y. Bengio, "A parallel mixture of SVMs for very large scale problems," in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, pp. 633–640.

[18] C. Rasmussen and Z. Ghahramani, "Infinite mixtures of Gaussian process experts," in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, pp. 881–888.

[19] B. Shahbaba and R. Neal, "Nonlinear models using Dirichlet process mixtures," *J. Mach. Learn. Res.*, vol. 10, no. 8, pp. 1829–1850, 2009.

[20] D. Eigen, M. Ranzato, and I. Sutskever, "Learning factored representations in a deep mixture of experts," 2013, *arXiv:1312.4314*.

[21] L. Theis and M. Bethge, "Generative image modeling using spatial LSTMs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1927–1935.

[22] M. Deisenroth and J. W. Ng, "Distributed Gaussian processes," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 1481–1490.

[23] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3366–3375.

[24] N. Shazeer et al., "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," 2017, *arXiv:1701.06538*.

[25] D. Lepikhin et al., "GShard: Scaling giant models with conditional computation and automatic sharding," 2020, *arXiv:2006.16668*.

[26] A. Q. Jiang et al., "Mixtral of experts," 2024, *arXiv:2401.04088*.

[27] xAI, "Grok-1," Mar. 2024. [Online]. Available: https://github.com/xai-org/grok-1

[28] Databricks, "Introducing DBRX: A new State-of-the-Art open LLM," Mar. 2024. [Online]. Available: https://www.databricks.com/blog/introducing-dbrx-new-state-art-open-llm

[29] S. A. R. Team, "Snowflake arctic: The best LLM for enterprise AI — Efficiently intelligent, truly open," Apr. 2024. [Online]. Available: https://www.snowflake.com/blog/arctic-open-efficient-foundation-language-models-snowflake/

[30] A. Liu et al., "Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model," 2024, *arXiv:2405.04434*.

[31] S. E. Yuksel, J. N. Wilson, and P. D. Gader, "Twenty years of mixture of experts," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 23, no. 8, pp. 1177–1193, Aug. 2012.

[32] W. Fedus, J. Dean, and B. Zoph, "A review of sparse expert models in deep learning," 2022, *arXiv:2209.01667*.

[33] N. Du et al., "GLaM: Efficient scaling of language models with mixture-of-experts," in *Proc. 39th Int. Conf. Mach. Learn.*, 2022, pp. 5547–5569.

[34] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *J. Mach. Learn. Res.*, vol. 23, no. 120, pp. 1–39, 2022.

[35] B. Zoph et al., "ST-MoE: Designing stable and transferable sparse expert models," 2022, *arXiv:2202.08906*.

[36] F. Xue et al., "OpenMoE: An early effort on open mixture-of-experts language models," 2024, *arXiv:2402.01739*.

[37] J. Puigcerver, C. R. Ruiz, B. Mustafa, and N. Houlsby, "From sparse to soft mixtures of experts," in *Proc. 12th Int. Conf. Learn. Representations*, 2023.

[38] M. Muqeeth, H. Liu, and C. Raffel, "Soft merging of experts with adaptive routing," 2023, *arXiv:2306.03745*.

[39] Z. Zhong, M. Xia, D. Chen, and M. Lewis, "Lory: Fully differentiable mixture-of-experts for autoregressive language model pre-training," 2024, *arXiv:2405.03133*.

[40] T. Zadouri, A. Üstün, A. Ahmadian, B. Ermiş, A. Locatelli, and S. Hooker, "Pushing mixture of experts to the limit: Extremely parameter efficient MoE for instruction tuning," 2023, *arXiv:2309.05444*.

[41] J. Wu, X. Hu, Y. Wang, B. Pang, and R. Soricut, "Omni-SMoLA: Boosting generalist multimodal models with soft mixture of low-rank experts," 2023, *arXiv:2312.00968*.

[42] Y. Wang et al., "AdaMix: Mixture-of-adaptations for parameter-efficient model tuning," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Abu Dhabi, UAE, 2022, pp. 5744–5760. [Online]. Available: https://aclanthology.org/2022.emnlp-main.388

[43] S. Dou et al., "LoRAMoE: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment," 2023, *arXiv:2312.09979*.

[44] Y. Gou et al., "Mixture of cluster-conditional LoRA experts for vision-language instruction tuning," 2023, *arXiv:2312.12379*.

[45] T. Luo et al., "MoELoRA: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models," 2024, *arXiv:2402.12851*.

[46] X. Wu, S. Huang, and F. Wei, "Mixture of LoRA experts," in *Proc. 12th Int. Conf. Learn. Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=uWvKBCYh4S

[47] A. Komatsuzaki et al., "Sparse upcycling: Training mixture-of-experts from dense checkpoints," in *Proc. 11th Int. Conf. Learn. Representations*, 2022.

[48] Z. Zhang, Y. Lin, Z. Liu, P. Li, M. Sun, and J. Zhou, "MoEfication: Transformer feed-forward layers are mixtures of experts," in *Proc. Findings Assoc. Comput. Linguistics*, 2022, pp. 877–890.

[49] L.-M. Team, "LLaMA-MoE: Building mixture-of-experts from LLaMA with continual pre-training," Dec. 2023. [Online]. Available: https://github.com/pjlab-sys4nlp/llama-moe

[50] F. Xue, X. He, X. Ren, Y. Lou, and Y. You, "One student knows all experts know: From sparse to dense," 2022, *arXiv:2201.10890*.

[51] T. Chen et al., "Task-specific expert pruning for sparse mixture-of-experts," 2022, *arXiv:2206.00277*.

[52] S. Sukhbaatar et al., "Branch-train-MiX: Mixing expert LLMs into a mixture-of-experts LLM," 2024, *arXiv:2403.07816*.

[53] W. Chen et al., "Lifelong language pretraining with distribution-specialized experts," in *Proc. 40th Int. Conf. Mach. Learn.*, 2023, pp. 5383–5395.

[54] S. Antoniak et al., "Mixture of tokens: Efficient LLMs through cross-example aggregation," 2023, *arXiv:2310.15961*.

[55] D. Raposo, S. Ritter, B. Richards, T. Lillicrap, P. C. Humphreys, and A. Santoro, "Mixture-of-depths: Dynamically allocating compute in transformer-based language models," 2024, *arXiv:2404.02258*.

[56] F. Xue, Z. Shi, F. Wei, Y. Lou, Y. Liu, and Y. You, "Go wider instead of deeper," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 8779–8787.

[57] S. Tan, Y. Shen, Z. Chen, A. Courville, and C. Gan, "Sparse universal transformer," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2023, pp. 169–179.

[58] J.-Y. Choi, J. Kim, J.-H. Park, W.-L. Mok, and S. Lee, "SMoP: Towards efficient and effective prompt tuning with sparse mixture-of-prompts," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2023, pp. 14306–14316.

[59] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi, "Modeling task relationships in multi-task learning with multi-gate mixture-of-experts," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1930–1939.

[60] X. Nie et al., "EvoMoE: An evolutional mixture-of-experts training framework via dense-to-sparse gate," 2021, *arXiv:2112.14397*.

[61] X. Wu, S. Huang, and F. Wei, "MoLE: Mixture of LoRA experts," in *Proc. 12th Int. Conf. Learn. Representations*, 2023.

[62] B. Pan et al., "Dense training, sparse inference: Rethinking training of mixture-of-experts language models," 2024, *arXiv:2404.05567*.

[63] A. Clark et al., "Unified scaling laws for routed language models," in *Proc. 39th Int. Conf. Mach. Learn.*, 2022, pp. 4057–4086.

[64] S. Rajbhandari et al., "DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale," in *Proc. 39th Int. Conf. Mach. Learn.*, 2022, pp. 18332–18346.

[65] T. Wei et al., "Skywork-MoE: A deep dive into training techniques for mixture-of-experts language models," 2024, *arXiv:2406.06563*.

[66] O. Lieber et al., "Jamba: A hybrid transformer-Mamba language model," 2024, *arXiv:2403.19887*.

[67] D. Dai et al., "DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models," 2024, *arXiv:2401.06066*.

[68] A. Yang et al., "M6-T: Exploring sparse expert models and beyond," 2021, *arXiv:2105.15082*.

[69] Z. Chen et al., "Mod-squad: Designing mixtures of experts as modular multi-task learners," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 11828–11837.

[70] S. Wu et al., "Yuan 2.0-M32: Mixture of experts with attention router," 2024, *arXiv:2405.17976*.

[71] Y. Shen, Z. Zhang, T. Cao, S. Tan, Z. Chen, and C. Gan, "ModuleFormer: Learning modular large language models from uncurated data," 2023, *arXiv:2306.04640*.
[72] M. Lewis, S. Bhosale, T. Dettmers, N. Goyal, and L. Zettlemoyer, "Base layers: Simplifying training of large, sparse models," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 6265–6274.
[73] H. Hazimeh et al., "DSelect-k: Differentiable selection in the mixture of experts with applications to multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 29335–29347.
[74] Y. J. Kim et al., "Scalable and efficient MoE training for multitask multilingual models," 2021, *arXiv:2109.10465*.
[75] S. Kudugunta et al., "Beyond distillation: Task-level mixture-of-experts for efficient inference," in *Proc. Findings Assoc. Comput. Linguistics*, 2021, pp. 3577–3599.
[76] M. R. Costa-jussà et al., "No language left behind: Scaling human-centered machine translation," 2022, *arXiv:2207.04672*.
[77] Z. Chi et al., "On the representation collapse of sparse mixture of experts," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 34600–34613.
[78] J. Zhu et al., "Uni-perceiver-MoE: Learning sparse generalist models with conditional MoEs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 2664–2678.
[79] D. Dai et al., "StableMoE: Stable routing strategy for mixture of experts," in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics*, 2022, pp. 7085–7095.
[80] S. Roller et al., "Hash layers for large sparse models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 17555–17566.
[81] S. Zuo et al., "Taming sparsely activated transformer with stochastic experts," in *Proc. Int. Conf. Learn. Representations*, 2021.
[82] S. Gururangan, M. Lewis, A. Holtzman, N. A. Smith, and L. Zettlemoyer, "DEMix layers: Disentangling domains for modular language modeling," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2022, pp. 5557–5576.
[83] A. Fan et al., "Beyond english-centric multilingual machine translation," *J. Mach. Learn. Res.*, vol. 22, no. 107, pp. 1–48, 2021.
[84] X. Ren et al., "PanGu-Σ: Towards trillion parameter language model with sparse heterogeneous computing," 2023, *arXiv:2303.10845*.
[85] Y. Zhou et al., "Mixture-of-experts with expert choice routing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 7103–7114.
[86] Y. Zhou et al., "Brainformers: Trading simplicity for efficiency," in *Proc. 40th Int. Conf. Mach. Learn.*, 2023, pp. 42531–42542.
[87] D. H. Dat, P. Y. Mao, T. H. Nguyen, W. Buntine, and M. Bennamoun, "HOMOE: A memory-based and composition-aware framework for zero-shot learning with hopfield network and soft mixture of experts," 2023, *arXiv:2311.14747*.
[88] X. Zhang, Y. Shen, Z. Huang, J. Zhou, W. Rong, and Z. Xiong, "Mixture of attention heads: Selecting attention heads per token," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2022, pp. 4150–4162.
[89] Y. Shen, Z. Guo, T. Cai, and Z. Qin, "JetMoE: Reaching Llama2 performance with 0.1 M dollars," 2024, *arXiv:2404.07413*.
[90] M. Artetxe et al., "Efficient large scale language modeling with mixtures of experts," 2021, *arXiv:2112.10684*.
[91] B. Lin et al., "MoE-LLaVA: Mixture of experts for large vision-language models," 2024, *arXiv:2401.15947*.
[92] S. Diao, T. Xu, R. Xu, J. Wang, and T. Zhang, "Mixture-of-domain-Adapters: Decoupling and injecting domain knowledge to pre-trained language models' memories," in *Proc. 61st Annu. Meeting Assoc. Comput. Linguistics*, 2023, pp. 5113–5129.
[93] D. Li et al., "MixLoRA: Enhancing large language models fine-tuning with LoRA based mixture of experts," 2024, *arXiv:2404.15159*.
[94] S. Chen, Z. Jie, and L. Ma, "LLaVA-MoLE: Sparse mixture of LoRA experts for mitigating data conflicts in instruction finetuning MLLMs," 2024, *arXiv:2401.16160*.
[95] Y. Zhu et al., "SiRA: Sparse mixture of low rank adaptation," 2023, *arXiv:2311.09179*.
[96] Y. Mao et al., "UniPELT: A unified framework for parameter-efficient language model tuning," in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics*, 2022, pp. 6253–6264.
[97] C. Gao et al., "Higher layers need more LoRA experts," 2024, *arXiv:2402.08562*.
[98] Y. Liu et al., "Intuition-aware mixture-of-rank-1-experts for parameter efficient finetuning," 2024, *arXiv:2404.08985*.
[99] L. Wu, M. Liu, Y. Chen, D. Chen, X. Dai, and L. Yuan, "Residual mixture of experts," 2022, *arXiv:2204.09636*.
[100] D. Dua, S. Bhosale, V. Goswami, J. Cross, M. Lewis, and A. Fan, "Tricks for training sparse translation models," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2022, pp. 3340–3345.
[101] Y. Huang, P. Ye, X. Huang, S. Li, T. Chen, and W. Ouyang, "Experts weights averaging: A new general training scheme for vision transformers," 2023, *arXiv:2308.06093*.
[102] M. Li et al., "Branch-train-Merge: Embarrassingly parallel training of expert language models," in *Proc. 1st Workshop Interpolation Regularizers Beyond NeurIPS*, 2022.
[103] H. Wang, F. M. Polo, Y. Sun, S. Kundu, E. Xing, and M. Yurochkin, "Fusing models with complementary expertise," in *Proc. 12th Int. Conf. Learn. Representations*, 2023.
[104] J. He, J. Qiu, A. Zeng, Z. Yang, J. Zhai, and J. Tang, "FastMoE: A fast mixture-of-expert training system," 2021, *arXiv:2103.13262*.
[105] C. Hwang et al., "Tutel: Adaptive mixture-of-experts at scale," *Proc. Mach. Learn. Syst.*, vol. 5, pp. 269–287, 2023.
[106] L. Shen et al., "SE-MoE: A scalable and efficient mixture-of-experts distributed training and inference system," 2022, *arXiv:2205.10034*.
[107] J. He et al., "FasterMoE: Modeling and optimizing training of large-scale dynamic pre-trained models," in *Proc. 27th ACM SIGPLAN Symp. Princ. Pract. Parallel Program.*, 2022, pp. 120–134.
[108] S. Singh, O. Ruwase, A. A. Awan, S. Rajbhandari, Y. He, and A. Bhatele, "A hybrid tensor-expert-data parallelism approach to optimize mixture-of-experts training," in *Proc. 37th Int. Conf. Supercomputing*, 2023, pp. 203–214.
[109] X. Nie, P. Zhao, X. Miao, T. Zhao, and B. Cui, "HetuMoE: An efficient trillion-scale mixture-of-expert distributed training system," 2022, *arXiv:2203.14685*.
[110] X. Nie et al., "FleXMoE: Scaling large-scale sparse pre-trained model training via dynamic device placement," *Proc. ACM Manage. Data*, vol. 1, no. 1, pp. 1–19, 2023.
[111] M. Zhai, J. He, Z. Ma, Z. Zong, R. Zhang, and J. Zhai, "SmartMoE: Efficiently training Sparsely-Activated models through combining offline and online parallelization," in *Proc. USENIX Annu. Tech. Conf.*, 2023, pp. 961–975.
[112] T. Gale, D. Narayanan, C. Young, and M. Zaharia, "MegaBlocks: Efficient sparse training with mixture-of-experts," *Proc. Mach. Learn. Syst.*, vol. 5, pp. 288–304, 2023.
[113] S. Tan, Y. Shen, R. Panda, and A. Courville, "Scattered mixture-of-experts implementation," 2024, *arXiv:2403.08245*.
[114] N. Zheng et al., "PIT: Optimization of dynamic sparse deep learning models via permutation invariant transformation," in *Proc. 29th Symp. Operating Syst. Princ.*, 2023, pp. 331–347.
[115] J. Yao et al., "Exploiting inter-layer expert affinity for accelerating mixture-of-experts model inference," 2024, *arXiv:2401.08383*.
[116] C. Chen, M. Li, Z. Wu, D. Yu, and C. Yang, "TA-MoE: Topology-aware large scale mixture-of-expert training," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 22173–22186.
[117] Z. Zhang et al., "MPMoE: Memory efficient MoE for pre-trained models with adaptive pipeline parallelism," *IEEE Trans. Parallel Distrib. Syst.*, vol. 35, no. 6, pp. 998–1011, Jun. 2024.
[118] C. Jiang, Y. Tian, Z. Jia, S. Zheng, C. Wu, and Y. Wang, "Lancet: Accelerating mixture-of-experts training via whole graph computation-communication overlapping," 2024, *arXiv:2404.19429*.
[119] W. Cai, J. Jiang, L. Qin, J. Cui, S. Kim, and J. Huang, "Shortcut-connected expert parallelism for accelerating mixture-of-experts," 2024, *arXiv:2404.05019*.
[120] R. Hwang et al., "Pre-gated MoE: An algorithm-system co-design for fast and scalable mixture-of-expert inference," 2023, *arXiv:2308.12066*.
[121] R. Yi, L. Guo, S. Wei, A. Zhou, S. Wang, and M. Xu, "EdgeMoE: Fast on-device inference of MoE-based large language models," 2023, *arXiv:2308.14352*.
[122] Y. Zhang et al., "Robust mixture-of-expert training for convolutional neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 90–101.
[123] M. N. R. Chowdhury, S. Zhang, M. Wang, S. Liu, and P.-Y. Chen, "Patch-level routing in mixture-of-experts is provably sample-efficient for convolutional neural networks," in *Proc. 40th Int. Conf. Mach. Learn.*, 2023, pp. 6074–6114.
[124] H. Tang, J. Liu, M. Zhao, and X. Gong, "Progressive layered extraction (PLE): A novel multi-task learning (MTL) model for personalized recommendations," in *Proc. 14th ACM Conf. Recommender Syst.*, 2020, pp. 269–278.
[125] J. Jiang et al., "AdaMCT: Adaptive mixture of CNN-transformer for sequential recommendation," in *Proc. 32nd ACM Int. Conf. Inf. Knowl. Manage.*, 2023, pp. 976–986.
[126] B. Mustafa, C. Riquelme, J. Puigcerver, R. Jenatton, and N. Houlsby, "Multimodal contrastive learning with LiMoE: The language-image mixture of experts," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 9564–9576.

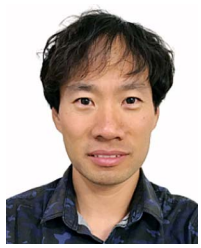
[127] S. Shen, Z. Yao, C. Li, T. Darrell, K. Keutzer, and Y. He, "Scaling vision-language models with sparse mixture of experts," 2023, *arXiv:2303.07226*.
[128] Y. Li et al., "Uni-MoE: Scaling unified multimodal LLMs with mixture of experts," 2024, *arXiv:2405.11273*.
[129] B. McKinzie et al., "MM1: Methods, analysis & insights from multimodal LLM pre-training," 2024, *arXiv:2403.09611*.
[130] A. Q. Jiang et al., "Mistral 7B," 2023, *arXiv:2310.06825*.
[131] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," 2023, *arXiv:2307.09288*.
[132] OpenAI, "ChatGPT: Optimizing language models for dialogue," 2022. [Online]. Available: https://openai.com/blog/chatgpt
[133] X. Bi et al., "DeepSeek LLM: Scaling open-source language models with longtermism," 2024, *arXiv:2401.02954*.
[134] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.
[135] A. Davis and I. Arel, "Low-rank approximations for conditional feedforward computation in deep neural networks," 2013, *arXiv:1312.4461*.
[136] A. Almahairi, N. Ballas, T. Cooijmans, Y. Zheng, H. Larochelle, and A. Courville, "Dynamic capacity networks," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 2549–2558.
[137] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup, "Conditional computation in neural networks for faster models," 2015, *arXiv:1511.06297*.
[138] C. Rosenbaum, T. Klinger, and M. Riemer, "Routing networks: Adaptive selection of non-linear functions for multi-task learning," 2017, *arXiv:1711.01239*.
[139] C. Rosenbaum, I. Cases, M. Riemer, and T. Klinger, "Routing networks and the challenges of modular and compositional computation," 2019, *arXiv:1904.12774*.
[140] Z. Zeng, Y. Miao, H. Gao, H. Zhang, and Z. Deng, "AdaMoE: Token-adaptive routing with null experts for mixture-of-experts language models," 2024, *arXiv:2406.13233*.
[141] Q. Ye, J. Zha, and X. Ren, "Eliciting and understanding cross-task skills with task-level mixture-of-experts," in *Proc. Findings Assoc. Comput. Linguistics*, 2022, pp. 2567–2592.
[142] N. Muennighoff et al., "OLMoE: Open mixture-of-experts language models," 2024, *arXiv:2409.02060*.
[143] Q. Team, "Qwen1.5-MoE: Matching 7B model performance with 1/3 activated parameters," Feb. 2024. [Online]. Available: https://qwenlm.github.io/blog/qwen-moe/
[144] D. Hendrycks et al., "Measuring massive multitask language understanding," 2020, *arXiv:2009.03300*.
[145] K. Cobbe et al., "Training verifiers to solve math word problems," 2021, *arXiv:2110.14168*.
[146] D. Hendrycks et al., "Measuring mathematical problem solving with the math dataset," 2021, *arXiv:2103.03874*.
[147] M. Chen et al., "Evaluating large language models trained on code," 2021, *arXiv:2107.03374*.
[148] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," 2023, *arXiv:2312.00752*.
[149] N. Ding et al., "Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models," 2022, *arXiv:2203.06904*.
[150] Z. Han et al., "Parameter-efficient fine-tuning for large models: A comprehensive survey," 2024, *arXiv:2403.14608*.
[151] V. Lialin, V. Deshpande, and A. Rumshisky, "Scaling down to scale up: A guide to parameter-efficient fine-tuning," 2023, *arXiv:2303.15647*.
[152] Q. Liu et al., "MOELoRA: An MoE-based parameter efficient fine-tuning method for multi-task medical applications," 2023, *arXiv:2310.18339*.
[153] O. Ostapenko, L. Caccia, Z. Su, N. Le Roux, L. Charlin, and A. Sordoni, "A case study of instruction tuning with mixture of parameter-efficient experts," in *Proc. NeurIPS 2023 Workshop Instruct. Tuning Instruct. Following*, 2023.
[154] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," in *Proc. Int. Conf. Learn. Representations*, 2021.
[155] N. Houlsby et al., "Parameter-efficient transfer learning for NLP," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 2790–2799.
[156] H. Liu et al., "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 1950–1965.
[157] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics, 11th Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 4582–4597.
[158] T. Wei et al., "Skywork: A more open bilingual foundation model," 2023, *arXiv:2310.19341*.
[159] S. Zuo, Q. Zhang, C. Liang, P. He, T. Zhao, and W. Chen, "MoEBERT: From BERT to mixture-of-experts via importance-guided adaptation," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2022, pp. 1610–1623.
[160] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "ZeRO: Memory optimizations toward training trillion parameter models," in *Proc. SC20, Int. Conf. High Perform. Comput. Netw., Storage Anal.*, 2020, pp. 1–16.
[161] J. Ren et al., "ZeRO-offload: Democratizing billion-scale model training," in *Proc. 2021 USENIX Annu. Tech. Conf.*, 2021, pp. 551–564.
[162] S. Rajbhandari, O. Ruwase, J. Rasley, S. Smith, and Y. He, "ZeRO-infinity: Breaking the GPU memory wall for extreme scale deep learning," in *Proc. Int. Conf. High Perform. Comput. Netw., Storage Anal.*, 2021, pp. 1–14.
[163] Z. Ma et al., "BaGuaLu: Targeting brain scale pretrained models with over 37 million cores," in *Proc. 27th ACM SIGPLAN Symp. Princ. Pract. Parallel Program.*, 2022, pp. 192–204.
[164] L. Zheng et al., "Alpa: Automating inter-and Intra-Operator parallelism for distributed deep learning," in *Proc. 16th USENIX Symp. Operating Syst. Des. Implementation*, 2022, pp. 559–578.
[165] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-LM: Training multi-billion parameter language models using model parallelism," 2019, *arXiv:1909.08053*.
[166] S. Smith et al., "Using DeepSpeed and megatron to train megatron-turing NLG 530B, a large-scale generative language model," 2022, *arXiv:2201.11990*.
[167] D. Narayanan et al., "Efficient large-scale language model training on GPU clusters using Megatron-LM," in *Proc. Int. Conf. High Perform. Comput. Netw., Storage Anal.*, 2021, pp. 1–15.
[168] Y. Huang et al., "GPipe: Efficient training of giant neural networks using pipeline parallelism," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 103–112.
[169] D. Narayanan et al., "PipeDream: Generalized pipeline parallelism for DNN training," in *Proc. 27th ACM Symp. Operating Syst. Princ.*, 2019, pp. 1–15.
[170] P. Qi, X. Wan, G. Huang, and M. Lin, "Zero bubble pipeline parallelism," in *Proc. 12th Int. Conf. Learn. Representations*, 2023.
[171] S. Li, F. Xue, C. Baranwal, Y. Li, and Y. You, "Sequence parallelism: Long sequence training from system perspective," 2021, *arXiv:2105.13120*.
[172] V. A. Korthikanti et al., "Reducing activation recomputation in large transformer models," *Proc. Mach. Learn. Syst.*, vol. 5, pp. 341–353, 2023.
[173] S. A. Jacobs et al., "DeepSpeed ulysses: System optimizations for enabling training of extreme long sequence transformer models," 2023, *arXiv:2309.14509*.
[174] M. Ott et al., "fairseq: A fast, extensible toolkit for sequence modeling," 2019, *arXiv:1904.01038*.
[175] N. Shazeer et al., "Mesh-TensorFlow: Deep learning for supercomputers," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 10435–10444.
[176] Q. Team, "Introducing Qwen1.5," Feb. 2024. [Online]. Available: https://qwenlm.github.io/blog/qwen1.5/
[177] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.
[178] Y. Zheng and D. X. Wang, "A survey of recommender systems with multi-objective optimization," *Neurocomputing*, vol. 474, pp. 141–153, 2022.
[179] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 689–696.
[180] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, "Multimodal machine learning: A survey and taxonomy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 2, pp. 423–443, Feb. 2019.
[181] S. Uppal et al., "Multimodal research in vision and language: A review of current and emerging trends," *Inf. Fusion*, vol. 77, pp. 149–171, 2022.
[182] L. Zhou, H. Palangi, L. Zhang, H. Hu, J. Corso, and J. Gao, "Unified vision-language pre-training for image captioning and VQA," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 13041–13049.

Weilin Cai (Graduate Student Member, IEEE) is currently working toward the PhD degree in microelectronics thrust with The Hong Kong University of Science and Technology, Guangzhou, China, under the supervision of Prof. Jiayi Huang. His research interests include machine learning systems, high performance computing and artificial intelligence, with a special focus on the system-algorithm co-design of mixture-of-experts.

Juyong Jiang is currently working toward the PhD degree in data science and analytics thrust with The Hong Kong University of Science and Technology, Guangzhou, China, under the supervision of Prof. Sunghun Kim. He has authored or coauthored several papers in IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, CIKM, EMNLP, and ICME. His research interests include large language models and code generation. He is a Reviewer for IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and IEEE TRANSACTIONS ON CYBERNETICS.

Fan Wang is currently working toward the MPhil degree in data science and analytics thrust with The Hong Kong University of Science and Technology, Guangzhou, China, under the supervision of Prof. Jing Tang. Her research interests include parameter-efficient fine-tuning and the mixture of experts.

Jing Tang received the PhD degree in computer science from Nanyang Technological University, Singapore, in 2018. He is currently an Assistant Professor with The Hong Kong University of Science and Technology, Guangzhou, China. His research interests include Big Data management and analytics, social network and graph analysis, machine learning, and blockchains.



Sunghun Kim received the PhD degree from the Computer Science Department, University of California, Santa Cruz, CA, USA. He was a Postdoctoral Researcher with Program Analysis Group, MIT. He is currently an Associate Professor with The Hong Kong University of Science and Technology, Guangzhou, China. His research interests include practical deep learning systems, including recommendation systems, neural search (NLP), and source code generation. He has made significant contributions to the field of software engineering, with a particular emphasis on deep learning-based code generation, software evolution, repository data mining, development social network mining, program analysis, and empirical studies. He was the recipient of the numerous accolades for his work, including four SIGSOFT Distinguished Paper Awards and the ten years Most Influential Paper Award from ICME 2008. He was also the Founding Head of Clova AI at Naver/Line, where he led a team of 250 people. He was an Associate Editor for IEEE TRANSACTIONS ON SOFTWARE ENGINEERING and was on the Editorial Board of the Empirical Software Engineering journal.

Jiayi Huang (Member, IEEE) received the BEng degree in information and communication engineering from Zhejiang University, Hangzhou, China, in 2014, and the PhD degree in computer engineering from Texas A&M University, College Station, TX, USA, in 2020. He is currently an Assistant Professor with The Hong Kong University of Science and Technology, Gaungzhou, China. His research interests include computer architecture, computer systems, and security. He is a Member of the ACM and the IEEE Computer Society.