DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models

Damai Dai*[1,2], Chengqi Deng[1], Chenggang Zhao*[1,3], R.X. Xu[1], Huazuo Gao[1], Deli Chen[1], Jiashi Li[1], Wangding Zeng[1], Xingkai Yu*[1,4], Y. Wu[1], Zhenda Xie[1], Y.K. Li[1], Panpan Huang[1], Fuli Luo[1], Chong Ruan[1], Zhifang Sui[2], Wenfeng Liang[1]

[1]DeepSeek-AI
[2]National Key Laboratory for Multimedia Information Processing, Peking University
[3]Institute for Interdisciplinary Information Sciences, Tsinghua University
[4]National Key Laboratory for Novel Software Technology, Nanjing University
{daidamai, szf}@pku.edu.cn, {wenfeng.liang}@deepseek.com
https://github.com/deepseek-ai/DeepSeek-MoE

Abstract

In the era of large language models, Mixture-of-Experts (MoE) is a promising architecture for managing computational costs when scaling up model parameters. However, conventional MoE architectures like GShard, which activate the top-K out of N experts, face challenges in ensuring expert specialization, i.e. each expert acquires non-overlapping and focused knowledge. In response, we propose the **DeepSeekMoE** architecture towards ultimate expert specialization. It involves two principal strategies: (1) finely segmenting the experts into mN ones and activating mK from them, allowing for a more flexible combination of activated experts; (2) isolating K_s experts as shared ones, aiming at capturing common knowledge and mitigating redundancy in routed experts. Starting from a modest scale with 2B parameters, we demonstrate that DeepSeekMoE 2B achieves comparable performance with GShard 2.9B, which has 1.5× expert parameters and computation. In addition, DeepSeekMoE 2B nearly approaches the performance of its dense counterpart with the same number of total parameters, which set the upper bound of MoE models. Subsequently, we scale up DeepSeekMoE to 16B parameters and show that it achieves comparable performance with LLaMA2 7B, with only about 40% of computations. Further, our preliminary efforts to scale up DeepSeekMoE to 145B parameters consistently validate its substantial advantages over the GShard architecture, and show its performance comparable with DeepSeek 67B, using only 28.5% (maybe even 18.2%) of computations.

1. Introduction

Recent research and practices have empirically demonstrated that, with sufficient training data available, scaling language models with increased parameters and computational budgets can yield remarkably stronger models (Brown et al., 2020; Hoffmann et al., 2022; OpenAI, 2023; Touvron et al., 2023a). It is imperative to acknowledge, however, that the endeavor to scale models to an extremely large scale is also associated with exceedingly high computational costs. Considering the substantial costs, the Mixture-of-Experts (MoE) architecture (Jacobs et al., 1991; Jordan and Jacobs, 1994; Shazeer et al., 2017) has emerged as a popular solution. It can

*Contribution during internship at DeepSeek-AI.

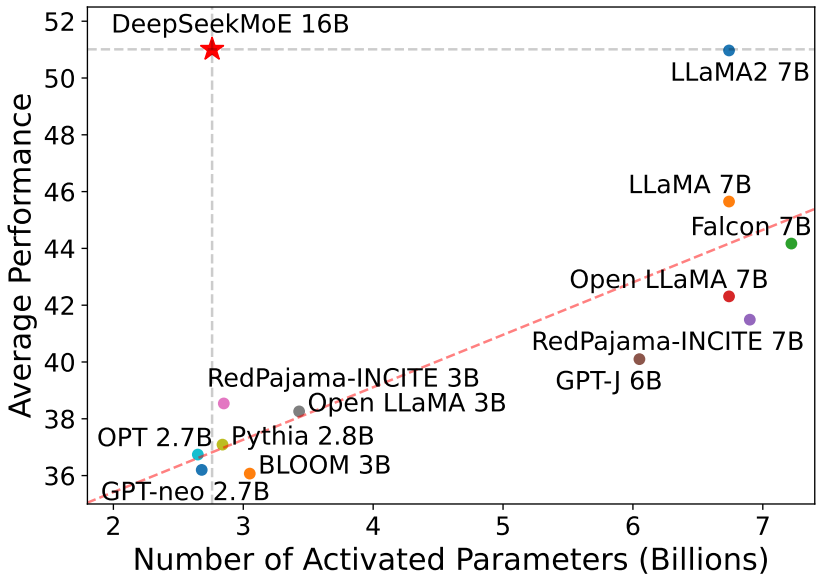

Figure 1 | Comparison between DeepSeekMoE 16B and open source models on the Open LLM Leaderboard. The red dashed line is linearly fitted from data points of all models except DeepSeekMoE 16B. DeepSeekMoE 16B consistently outperforms models with a similar number of activated parameters by a large margin, and achieves comparable performance with LLaMA2 7B, which has approximately 2.5 times the activated parameters.

enable parameter scaling, while concurrently keeping computational costs at a modest level. Recent applications of MoE architectures in Transformers (Vaswani et al., 2017) have yielded successful attempts at scaling language models to a substantial size (Du et al., 2022; Fedus et al., 2021; Lepikhin et al., 2021; Zoph, 2022), accompanied with remarkable performance. These achievements underscore the considerable potential and promise of MoE language models.

Despite the promising potential of MoE architectures, existing MoE architectures potentially suffer from issues of knowledge hybridity and knowledge redundancy, which limit the expert specialization, i.e., each expert acquires non-overlapping and focused knowledge. Conventional MoE architectures substitute the Feed-Forward Networks (FFNs) in a Transformer with MoE layers. Each MoE layer consists of multiple experts, with each structurally identical to a standard FFN, and each token is assigned to one (Fedus et al., 2021) or two (Lepikhin et al., 2021) experts. This architecture manifests two potential issues: (1) **Knowledge Hybridity**: existing MoE practices often employ a limited number of experts (e.g., 8 or 16), and thus tokens assigned to a specific expert will be likely to cover diverse knowledge. Consequently, the designated expert will intend to assemble vastly different types of knowledge in its parameters, which are hard to utilize simultaneously. (2) **Knowledge Redundancy**: tokens assigned to different experts may require common knowledge. As a result, multiple experts may converge in acquiring shared knowledge in their respective parameters, thereby leading to redundancy in expert parameters. These issues collectively hinder the expert specialization in existing MoE practices, preventing them from reaching the theoretical upper-bound performance of MoE models.

In response to the aforementioned issues, we introduce **DeepSeekMoE**, an innovative MoE architecture specifically designed towards ultimate expert specialization. Our architecture involves two principal strategies: (1) **Fine-Grained Expert Segmentation:** while maintaining the number of parameters constant, we segment the experts into a finer grain by splitting the

FFN intermediate hidden dimension. Correspondingly, keeping a constant computational cost, we also activate more fine-grained experts to enable a more flexible and adaptable combination of activated experts. Fine-grained expert segmentation allows diverse knowledge to be decomposed more finely and be learned more precisely into different experts, where each expert will retain a higher level of specialization. In addition, the increased flexibility in combining activated experts also contributes to a more accurate and targeted knowledge acquisition. (2) **Shared Expert Isolation:** we isolate certain experts to serve as shared experts that are always activated, aiming at capturing and consolidating common knowledge across varying contexts. Through compressing common knowledge into these shared experts, redundancy among other routed experts will be mitigated. This can enhance the parameter efficiency and ensure that each routed expert retains specialized by focusing on distinctive aspects. These architectural innovations in DeepSeekMoE offer opportunities to train a parameter-efficient MoE language model where each expert is highly specialized.

Starting from a modest scale with 2B parameters, we validate the advantages of the DeepSeekMoE architecture. We conduct evaluations on 12 zero-shot or few-shot benchmarks spanning diverse tasks. Empirical results indicate that DeepSeekMoE 2B surpasses GShard 2B (Lepikhin et al., 2021) by a substantial margin, and even matches GShard 2.9B, a larger MoE model with 1.5× expert parameters and computation. Remarkably, we find that DeepSeekMoE 2B nearly approaches the performance of its dense counterpart with an equivalent number of parameters, which sets the strict upper bound of MoE language models. In pursuit of deeper insights, we conduct elaborate ablation studies and analysis on the expert specialization for DeepSeekMoE. These studies validate the effectiveness of fine-grained expert segmentation and shared expert isolation, and provide empirical evidence supporting the assertion that DeepSeekMoE can achieve a high level of expert specialization.

Leveraging our architecture, we subsequently scale up the model parameters to 16B and train DeepSeekMoE 16B on a large-scale corpus with 2T tokens. Evaluation results reveal that with only about 40% of computations, DeepSeekMoE 16B achieves comparable performance with DeepSeek 7B (DeepSeek-AI, 2024), a dense model trained on the same 2T corpus. We also compare DeepSeekMoE with open source models and the evaluations demonstrate that DeepSeekMoE 16B consistently outperforms models with a similar number of activated parameters by a large margin, and achieves comparable performance with LLaMA2 7B (Touvron et al., 2023b), which has approximately 2.5 times the activated parameters. Figure 1 demonstrates the evaluation results on the Open LLM Leaderboard[1]. Additionally, we conduct supervised fine-tuning (SFT) for alignment, transforming the model into a chat model. Evaluation results show that DeepSeekMoE Chat 16B also achieves comparable performance with DeepSeek Chat 7B and LLaMA2 SFT 7B in the chat setting. Encouraged by these results, we further undertake a preliminary endeavor to scale up DeepSeekMoE to 145B. The experimental results still validate its substantial advantages over the GShard architecture consistently. In addition, it shows performance comparable with DeepSeek 67B, using only 28.5% (maybe even 18.2%) of computations.

Our contributions are summarized as follows:

- **Architectural Innovation.** We introduce DeepSeekMoE, an innovative MoE architecture aiming at achieving ultimate expert specialization, which employs two principal strategies of fine-grained expert segmentation and shared expert isolation.
- **Empirical Validation.** We conduct extensive experiments to empirically validate the effectiveness of the DeepSeekMoE architecture. Experimental results validate the high

[1]https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

level of expert specialization in DeepSeekMoE 2B, and indicate that DeepSeekMoE 2B can nearly approach the upper bound performance for MoE models
- **Scalability.** We scale up DeepSeekMoE to train a 16B model and show that with only about 40% of computations, DeepSeekMoE 16B achieves comparable performance with DeepSeek 7B and LLaMA2 7B. We also undertake a preliminary endeavor to scale up DeepSeekMoE to 145B, highlighting its consistent advantages over the GShard architecture and showing a comparable performance with DeepSeek 67B.
- **Alignment for MoE.** We successfully perform supervised fine-tuning on DeepSeekMoE 16B to create an aligned chat model, showcasing the adaptability and versatility of DeepSeekMoE 16B.
- **Public Release.** In the spirit of open research, we release the model checkpoint of DeepSeekMoE 16B to the public. Notably, this model can be deployed on a single GPU with 40GB of memory without the need for quantization.

2. Preliminaries: Mixture-of-Experts for Transformers

We first introduce a generic MoE architecture commonly used in Transformer language models. A standard Transformer language model is constructed by stacking L layers of standard Transformer blocks, where each block can be represented as follows:

$$\mathbf{u}_{1:T}^{l} = \text{Self-Att}\left(\mathbf{h}_{1:T}^{l-1}\right) + \mathbf{h}_{1:T}^{l-1}, \tag{1}$$

$$\mathbf{h}_{t}^{l} = \text{FFN}\left(\mathbf{u}_{t}^{l}\right) + \mathbf{u}_{t}^{l}, \tag{2}$$

where T denotes the sequence length, Self-Att$(\cdot)$ denotes the self-attention module, FFN$(\cdot)$ denotes the Feed-Forward Network (FFN), $\mathbf{u}_{1:T}^{l} \in \mathbb{R}^{T \times d}$ are the hidden states of all tokens after the l-th attention module, and $\mathbf{h}_{t}^{l} \in \mathbb{R}^{d}$ is the output hidden state of the t-th token after the l-th Transformer block. For brevity, we omit the layer normalization in the above formulations.

A typical practice to construct an MoE language model usually substitutes FFNs in a Transformer with MoE layers at specified intervals (Du et al., 2022; Fedus et al., 2021; Lepikhin et al., 2021; Zoph, 2022). An MoE layer is composed of multiple experts, where each expert is structurally identical to a standard FFN. Then, each token will be assigned to one (Fedus et al., 2021) or two (Lepikhin et al., 2021) experts. If the l-th FFN is substituted with an MoE layer, the computation for its output hidden state $\mathbf{h}_{t}^{l}$ is expressed as:

$$\mathbf{h}_{t}^{l} = \sum_{i=1}^{N} \left(g_{i,t}\, \text{FFN}_{i}\left(\mathbf{u}_{t}^{l}\right)\right) + \mathbf{u}_{t}^{l}, \tag{3}$$

$$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \text{Topk}(\{s_{j,t} | 1 \leqslant j \leqslant N\}, K), \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

$$s_{i,t} = \text{Softmax}_{i}\left(\mathbf{u}_{t}^{l^{T}} \mathbf{e}_{i}^{l}\right), \tag{5}$$

where N denotes the total number of experts, FFN$_{i}(\cdot)$ is the i-th expert FFN, $g_{i,t}$ denotes the gate value for the i-th expert, $s_{i,t}$ denotes the token-to-expert affinity, Topk$(\cdot, K)$ denotes the set comprising K highest affinity scores among those calculated for the t-th token and all N experts, and $\mathbf{e}_{i}^{l}$ is the centroid of the i-th expert in the l-th layer. Note that $g_{i,t}$ is sparse, indicating that only K out of N gate values are nonzero. This sparsity property ensures computational efficiency within an MoE layer, i.e., each token will be assigned to and computed in only K experts. Also, in the above formulations, we omit the layer normalization operation for brevity.

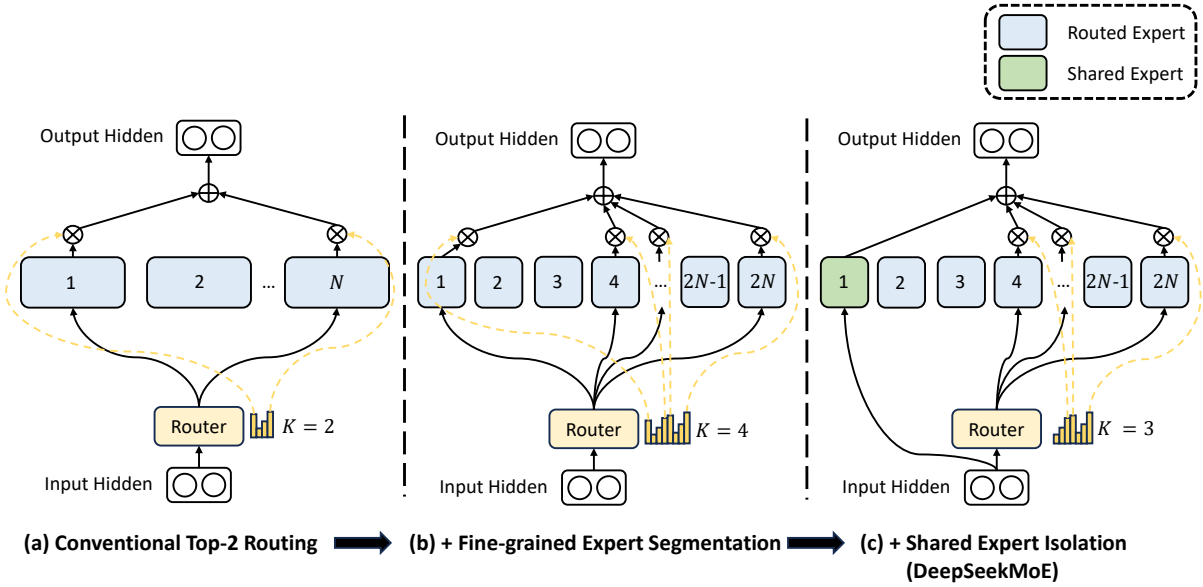

Figure 2 | Illustration of DeepSeekMoE. Subfigure (a) showcases an MoE layer with the conventional top-2 routing strategy. Subfigure (b) illustrates the fine-grained expert segmentation strategy. Subsequently, subfigure (c) demonstrates the integration of the shared expert isolation strategy, constituting the complete DeepSeekMoE architecture. It is noteworthy that across these three architectures, the number of expert parameters and computational costs remain constant.

3. DeepSeekMoE Architecture

On top of the generic MoE architecture outlined in Section 2, we introduce DeepSeekMoE, which is specifically designed to exploit the potential of expert specialization. As illustrated in Figure 2, our architecture incorporates two principal strategies: fine-grained expert segmentation and shared expert isolation. Both of these strategies are designed to elevate the level of expert specialization.

3.1. Fine-Grained Expert Segmentation

In scenarios where the number of experts is limited, tokens assigned to a particular expert will be more likely to cover diverse types of knowledge. As a consequence, the designated expert will intend to learn vastly different types of knowledge in its parameters, and they are hard to be simultaneously utilized. However, if each token can be routed to more experts, diverse knowledge will gain the potential to be decomposed and learned in different experts respectively. In this context, each expert can still retain a high level of expert specialization, contributing to a more focused knowledge distribution across experts.

In pursuit of the goal, while maintaining a consistent number of expert parameters and computational cost, we segment the experts with a finer grain. The finer expert segmentation enables a more flexible and adaptable combination of activated experts. To be specific, on top of a typical MoE architecture shown in Figure 2(a), we segment each expert FFN into m smaller experts by reducing the FFN intermediate hidden dimension to $\frac{1}{m}$ times its original size. Since each expert becomes smaller, in response, we also increase the number of activated experts to m times to keep the same computation cost, as illustrated in Figure 2(b). With the fine-grained

expert segmentation, the output of an MoE layer can be expressed as:

$$\mathbf{h}_t^l = \sum_{i=1}^{mN} \left(g_{i,t} \, \mathrm{FFN}_i \left(\mathbf{u}_t^l \right) \right) + \mathbf{u}_t^l, \tag{6}$$

$$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \mathrm{Topk}(\{s_{j,t} | 1 \leqslant j \leqslant mN\}, mK), \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

$$s_{i,t} = \mathrm{Softmax}_i \left(\mathbf{u}_t^{l^T} \mathbf{e}_i^l \right), \tag{8}$$

where the total number of expert parameters is equal to N times the number of parameters in a standard FFN, and mN denotes the total number of fine-grained experts. With the fine-grained expert segmentation strategy, the number of nonzero gates will also increases to mK.

From a combinatorial perspective, the fine-grained expert segmentation strategy substantially enhances the combinatorial flexibility of activated experts. As an illustrative example, we consider the case where $N = 16$. A typical top-2 routing strategy can yield $\binom{16}{2} = 120$ possible combinations. By contrast, if each expert is split into 4 smaller experts, the fine-grained routing strategy can yield $\binom{64}{8} = 4,426,165,368$ potential combinations. The surge in combinatorial flexibility enhances the potential for achieving more accurate and targeted knowledge acquisition.

3.2. Shared Expert Isolation

With a conventional routing strategy, tokens assigned to different experts may necessitate some common knowledge or information. As a result, multiple experts may converge in acquiring shared knowledge in their respective parameters, thereby resulting in redundancy in expert parameters. However, if there are shared experts dedicated to capturing and consolidating common knowledge across varying contexts, the parameter redundancy among other routed experts will be alleviated. This alleviation of redundancy will contribute to a more parameter-efficient model with more specialized experts.

Towards this objective, in addition to the fine-grained expert segmentation strategy, we further isolate K_s experts to serve as shared experts. Regardless of the router module, each token will be deterministically assigned to these shared experts. In order to maintain a constant computational cost, the number of activated experts among the other routed experts will be decreased by K_s, as depicted in Figure 2(c). With the shared expert isolation strategy integrated, an MoE layer in the complete DeepSeekMoE architecture is formulated as follows:

$$\mathbf{h}_t^l = \sum_{i=1}^{K_s} \mathrm{FFN}_i \left(\mathbf{u}_t^l \right) + \sum_{i=K_s+1}^{mN} \left(g_{i,t} \, \mathrm{FFN}_i \left(\mathbf{u}_t^l \right) \right) + \mathbf{u}_t^l, \tag{9}$$

$$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \mathrm{Topk}(\{s_{j,t} | K_s + 1 \leqslant j \leqslant mN\}, mK - K_s), \\ 0, & \text{otherwise}, \end{cases} \tag{10}$$

$$s_{i,t} = \mathrm{Softmax}_i \left(\mathbf{u}_t^{l^T} \mathbf{e}_i^l \right). \tag{11}$$

Finally, in DeepSeekMoE, the number of shared expert is K_s, the total number of routed experts is $mN - K_s$, and the number of nonzero gates is $mK - K_s$.

It is worth noting that the prototype of shared expert isolation can be credited to Rajbhandari et al. (2022). The key distinction lies in the fact that they derive this strategy from an engineering perspective, while we approach it from an algorithmic standpoint.

3.3. Load Balance Consideration

Automatically learned routing strategies may encounter the issue of load imbalance, which manifests two notable defects. Firstly, there is a risk of routing collapse (Shazeer et al., 2017), i.e., the model always selects only a few experts, preventing other experts from sufficient training. Secondly, if experts are distributed across multiple devices, load imbalance can exacerbate computation bottlenecks.

Expert-Level Balance Loss. In order to mitigate the risk of routing collapse, we also employ an expert-level balance loss. The computation of the balance loss is as follows:

$$\mathcal{L}_{\text{ExpBal}} = \alpha_1 \sum_{i=1}^{N'} f_i P_i, \tag{12}$$

$$f_i = \frac{N'}{K'T} \sum_{t=1}^{T} \mathbb{1}(\text{Token } t \text{ selects Expert } i), \tag{13}$$

$$P_i = \frac{1}{T} \sum_{t=1}^{T} s_{i,t}, \tag{14}$$

where α_1 is a hyper-parameter called expert-level balance factor, N' is equal to $(mN - K_s)$ and K' is equal to $(mK - K_s)$ for brevity. $\mathbb{1}(\cdot)$ denotes the indicator function.

Device-Level Balance Loss. In addition to the expert-level balance loss, we introduce a device-level balance loss. When aiming to alleviate computation bottlenecks, it becomes unnecessary to enforce strict balance constraints at the expert level, because excessive constraints on load balance will compromise model performance. Instead, our primary objective is to ensure balanced computation across the devices. If we partition all routed experts into D groups $\{\mathcal{E}_1, \mathcal{E}_2, ..., \mathcal{E}_D\}$, and deploy each group on a single device, the device-level balance loss is computed as follows:

$$\mathcal{L}_{\text{DevBal}} = \alpha_2 \sum_{i=1}^{D} f'_i P'_i, \tag{15}$$

$$f'_i = \frac{1}{|\mathcal{E}_i|} \sum_{j \in \mathcal{E}_i} f_j, \tag{16}$$

$$P'_i = \sum_{j \in \mathcal{E}_i} P_j, \tag{17}$$

where α_2 is a hyper-parameter called device-level balance factor. In practice, we set a small expert-level balance factor to mitigate the risk of routing collapse, and meanwhile set a larger device-level balance factor to promote balanced computation across the devices.

4. Validation Experiments

4.1. Experimental Setup

4.1.1. Training Data and Tokenization

Our training data is sampled from a large-scale multilingual corpus created by DeepSeek-AI. The corpus primarily focuses on English and Chinese but also encompasses other languages. It is de-

rived from diverse sources, including web text, mathematical material, coding scripts, published literature, and various other textual materials. For the purpose of validation experiments, we sample a subset containing 100B tokens from the corpus to train our models. For tokenization, we utilize the HuggingFace Tokenizer[2] tools to train byte pair encoding (BPE) (Sennrich et al., 2016) tokenizers on a smaller subset of the training corpus. In the validation experiments, we prepare a tokenizer with a vocabulary size of 8K, and the vocabulary size will be scaled up when training larger models.

4.1.2. Infrastructures

We conduct experiments based on HAI-LLM (High-Flyer, 2023), an efficient and light-weight training framework which integrates multiple parallelism strategies, including tensor parallelism (Korthikanti et al., 2023; Narayanan et al., 2021; Shoeybi et al., 2019), ZeRO data parallelism (Rajbhandari et al., 2020), PipeDream pipeline parallelism (Harlap et al., 2018), and more specifically, expert parallelism (Lepikhin et al., 2021) by combining data and tensor parallelism. In order to optimize performance, we develop GPU kernels with CUDA and Triton (Tillet et al., 2019) for gating algorithms and fusing computations across linear layers in different experts.

All experiments are carried out on clusters equipped with NVIDIA A100 or H800 GPUs. Each node in the A100 cluster contains 8 GPUs connected pairwise via the NVLink bridge. The H800 cluster also features 8 GPUs per node, interconnected using NVLink and NVSwitch within nodes. For both A100 and H800 clusters, InfiniBand interconnects are utilized to facilitate communication across nodes.

4.1.3. Hyper-Parameters

Model Settings. In the validation experiments, we set the number of Transformer layers to 9 and the hidden dimension to 1280. We employ the multi-head attention mechanism with a total of 10 attention heads, where each head has a dimension of 128. For initialization, all learnable parameters are randomly initialized with a standard deviation of 0.006. We substitute all FFNs with MoE layers, and ensure that the total number of expert parameters equals 16 times that of a standard FFN. Additionally, we keep the activated expert parameters, including shared expert parameters and activated routed expert parameters, as 2 times that of a standard FFN. Under this configuration, each MoE model has approximately 2B total parameters, with the number of activated parameters around 0.3B.

Training Settings. We employ the AdamW optimizer (Loshchilov and Hutter, 2019) with hyper-parameters set to $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight_decay = 0.1. The learning rate is scheduled using a warmup-and-step-decay strategy. Initially, the learning rate linearly increases from 0 to the maximum value during the first 2K steps. Subsequently, the learning rate is multiplied by 0.316 at 80% of the training steps, and again by 0.316 at 90% of the training steps. The maximum learning rate for validation experiments is set to 1.08×10^{-3}, and the gradient clipping norm is set to 1.0. The batch size is set to 2K, and with a maximum sequence length of 2K, each training batch contains 4M tokens. Correspondingly, the total number of training steps is set to 25,000 to achieve 100B training tokens. Due to the abundance of training data, we do not use dropout during training. Given the relatively small model size, all parameters, including expert parameters, are deployed on a single GPU device to avoid unbalanced computation. Correspondingly, we do not drop any tokens during training and do not employ the device-level

[2]https://github.com/huggingface/tokenizers

balance loss. In order to prevent routing collapse, we set an expert-level balance factor of 0.01.

For readability, we also present an overview table of hyper-parameters for DeepSeekMoE across different sizes in Appendix A.

4.1.4. Evaluation Benchmarks

We conduct evaluations on a wide range of benchmarks covering various types of tasks. We list the benchmarks as follows.

Language Modeling. For language modeling, we evaluate the models on the test set of Pile (Gao et al., 2020), and the evaluation metric is the cross-entropy loss.

Language Understanding and Reasoning. For language understanding and reasoning, we consider HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), ARC-challenge and ARC-easy (Clark et al., 2018). The evaluation metric for these tasks is accuracy.

Reading Comprehension. For reading comprehension, we use RACE-high and RACE-middle Lai et al. (2017), and the evaluation metric is accuracy.

Code Generation. For code generation, we evaluate the models on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). The evaluation metric is Pass@1, which represents the pass rate for only one generation attempt.

Closed-Book Question Answering. For closed-book question answering, we consider TriviaQA (Joshi et al., 2017) and NaturalQuestions (Kwiatkowski et al., 2019). The evaluation metric is the Exactly Matching (EM) rate.

4.2. Evaluations

Baselines. Including DeepSeekMoE, we compare five models for validation experiments. **Dense** denotes a standard dense Transformer language model with 0.2B total parameters. **Hash Layer** (Roller et al., 2021) is an MoE architecture based on top-1 hash routing, with 2.0B total parameters and 0.2B activated parameters, aligned with the dense baseline. **Switch Transformer** (Fedus et al., 2021) is another well-known MoE architecture based on top-1 learnable routing, with total parameters and activated parameters the same as Hash Layer. **GShard** (Lepikhin et al., 2021) employs a top-2 learnable routing strategy, with 2.0B total parameters and 0.3B activated parameters since one more expert is activated compared to top-1 routing methods. **DeepSeekMoE** has 1 shared expert and 63 routed experts, where each expert is 0.25 times the size of a standard FFN. Including DeepSeekMoE, all compared models share the same training corpus and training hyper-parameters. All compared MoE models have the same number of total parameters, and GShard has the same number of activated parameters as DeepSeekMoE.

Results. We present the evaluation results in Table 1. For all demonstrated models, we report the final evaluation results after training on 100B tokens. From the table, we make the following observations: (1) With sparse architectures and more total parameters, Hash Layer

Metric	# Shot	Dense	Hash Layer	Switch	GShard	DeepSeekMoE
# Total Params	N/A	0.2B	2.0B	2.0B	2.0B	2.0B
# Activated Params	N/A	0.2B	0.2B	0.2B	0.3B	0.3B
FLOPs per 2K Tokens	N/A	2.9T	2.9T	2.9T	4.3T	4.3T
# Training Tokens	N/A	100B	100B	100B	100B	100B
Pile (Loss)	N/A	2.060	1.932	1.881	1.867	**1.808**
HellaSwag (Acc.)	0-shot	38.8	46.2	49.1	50.5	**54.8**
PIQA (Acc.)	0-shot	66.8	68.4	70.5	70.6	**72.3**
ARC-easy (Acc.)	0-shot	41.0	45.3	45.9	43.9	**49.4**
ARC-challenge (Acc.)	0-shot	26.0	28.2	30.2	31.6	**34.3**
RACE-middle (Acc.)	5-shot	38.8	38.8	43.6	42.1	**44.0**
RACE-high (Acc.)	5-shot	29.0	30.0	30.9	30.4	**31.7**
HumanEval (Pass@1)	0-shot	0.0	1.2	2.4	3.7	**4.9**
MBPP (Pass@1)	3-shot	0.2	0.6	0.4	0.2	**2.2**
TriviaQA (EM)	5-shot	4.9	6.5	8.9	10.2	**16.6**
NaturalQuestions (EM)	5-shot	1.4	1.4	2.5	3.2	**5.7**

Table 1 | Evaluation results for validation experiments. **Bold** font indicates the best. Compared with other MoE architectures, DeepSeekMoE exhibits a substantial performance advantage.

and Switch Transformer achieve significantly stronger performance than the dense baseline with the same number of activated parameters. (2) Compared with Hash Layer and Switch Transformer, GShard has more activated parameters and achieves slightly better performance than Switch Transformer. (3) With the same number of total parameters and activated parameters, DeepSeekMoE demonstrates overwhelming advantages over GShard. These results showcase the superiority of our DeepSeekMoE architecture within the existing landscape of MoE architectures.

4.3. DeepSeekMoE Aligns Closely with the upper bound of MoE Models

We have demonstrated that DeepSeekMoE outperforms the dense baseline and other MoE architectures. In order to provide a more precise understanding of the performance of DeepSeekMoE, we compare it with larger baselines with more total parameters or activated parameters. The comparisons enable us to estimate the required model size of GShard or dense baselines to achieve equivalent performance to DeepSeekMoE.

Comparison with GShard×1.5. Table 2 shows the comparison between DeepSeekMoE and a larger GShard model with 1.5 times the expert size, which results in 1.5 times both expert parameters and expert computation. Overall, we observe that DeepSeekMoE achieves comparable performance with GShard×1.5, underscoring the significant advantage inherent in the DeepSeekMoE architecture. In addition to the comparison with GShard×1.5, we also show the comparison with GShard×1.2 in Appendix B.

Furthermore, we increase the number of total parameters of DeepSeekMoE to 13.3B and compare it with GShard×1.2 and GShard×1.5 with 15.9B and 19.8B total parameters, respectively. We find that at a larger scale, DeepSeekMoE can even outperform GShard×1.5 distinctly. These

Metric	# Shot	GShard×1.5	Dense×16	DeepSeekMoE
Relative Expert Size	N/A	1.5	1	0.25
# Experts	N/A	0 + 16	16 + 0	1 + 63
# Activated Experts	N/A	0 + 2	16 + 0	1 + 7
# Total Expert Params	N/A	2.83B	1.89B	1.89B
# Activated Expert Params	N/A	0.35B	1.89B	0.24B
FLOPs per 2K Tokens	N/A	5.8T	24.6T	4.3T
# Training Tokens	N/A	100B	100B	100B
Pile (Loss)	N/A	1.808	1.806	1.808
HellaSwag (Acc.)	0-shot	54.4	55.1	54.8
PIQA (Acc.)	0-shot	71.1	71.9	72.3
ARC-easy (Acc.)	0-shot	47.3	51.9	49.4
ARC-challenge (Acc.)	0-shot	34.1	33.8	34.3
RACE-middle (Acc.)	5-shot	46.4	46.3	44.0
RACE-high (Acc.)	5-shot	32.4	33.0	31.7
HumanEval (Pass@1)	0-shot	3.0	4.3	4.9
MBPP (Pass@1)	3-shot	2.6	2.2	2.2
TriviaQA (EM)	5-shot	15.7	16.5	16.6
NaturalQuestions (EM)	5-shot	4.7	6.3	5.7

Table 2 | Comparisons among DeepSeekMoE, larger GShard models, and larger dense models. In the line of "# Experts", $a + b$ denotes a shared experts and b routed experts. In the line of "# Activated Experts", $a + b$ denotes a activated shared experts and b activated routed experts. DeepSeekMoE achieves comparable performance with a GShard model containing 1.5 times expert parameters and computation. In addition, DeepSeekMoE nearly approaches the performance of a dense model with 16 times FFN parameters, which sets the upper bound for MoE models in terms of the model capacity.

results are also provided in Appendix B.

Comparison with Dense×16. Table 2 also shows the comparison between DeepSeekMoE and larger dense models. For a fair comparison, we do not use the widely used ratio (1:2) between the attention and FFN parameters. Instead, we configure 16 shared experts where each expert has the same number of parameters as a standard FFN. This architecture mimics a dense model with 16 times standard FFN parameters. From the table, we find that DeepSeekMoE nearly approaches the performance of Dense×16, which sets the strict upper bound of MoE models in terms of the model capacity. These results suggest that, ***at least at the scale of about 2B parameters and 100B training tokens, the performance of DeepSeekMoE aligns closely with the theoretical upper bound of MoE models***. Also, we provide additional comparisons with Dense×4 in Appendix B.

4.4. Ablation Studies

In order to substantiate the effectiveness of the fine-grained expert segmentation and shared expert isolation strategies, we conduct ablation studies for DeepSeekMoE and present the results in Figure 3. For a fair comparison, we ensure all models included in the comparison have the

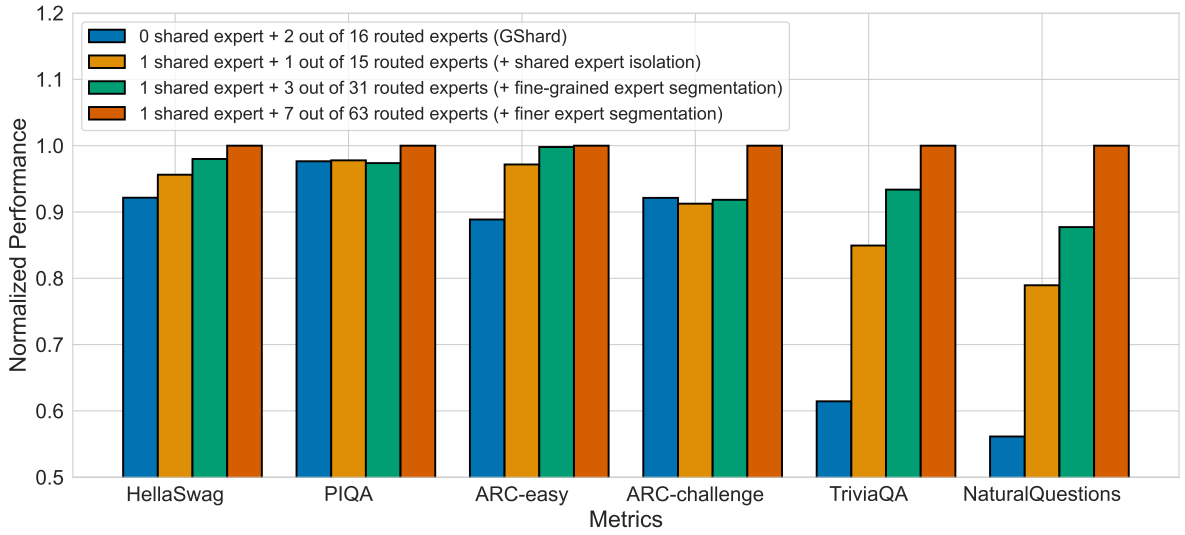

Figure 3 | Ablation studies for DeepSeekMoE. The performance is normalized by the best performance for clarity in presentation. All compared models have the same number of parameters and activated parameters. We can find that fine-grained expert segmentation and shared expert isolation both contribute to stronger overall performance.

same number of total parameters and activated parameters.

Shared Expert Isolation. In order to evaluate the influence of the shared expert isolation strategy, we isolate one expert as the shared one based on GShard. From Figure 3, we observe that compared with GShard, the intentional isolation of a shared expert yields improved performance across a majority of benchmarks. These results support the proposition that the shared expert isolation strategy contributes to a stronger model performance.

Fine-Grained Expert Segmentation. In order to assess the effectiveness of the fine-grained expert segmentation strategy, we conduct a more detailed comparison by further segmenting the experts into a finer grain. To be specific, we segment each expert into 2 or 4 smaller experts, resulting in a total of 32 (1 shared + 31 routed) or 64 (1 shared + 63 routed) experts. Figure 3 reveals a consistent trend that the continuous refinement of expert segmentation granularity corresponds to a continuous enhancement in overall model performance. These findings provide empirical substantiation for the effectiveness of the fine-grained expert segmentation strategy.

Ratios Between Shared and Routed Experts. In addition, we investigate the best ratio of shared experts and routed experts. Based on the finest granularity with 64 total experts and keeping the number of total experts and activated experts constant, we attempt to isolate 1, 2, and 4 experts as shared ones. We find that different ratios of the shared experts and routed experts do not significantly impact the performance, and 1, 2, and 4 shared experts achieve a Pile loss of 1.808, 1.806, and 1.811, respectively. Considering that the ratio of 1:3 yields a marginally better Pile loss, when scaling up DeepSeekMoE, we keep the ratio between shared experts and activated routed experts as 1:3.

4.5. Analysis on Expert Specialization

In this section, we conduct an empirical analysis on the expert specialization of DeepSeekMoE 2B. DeepSeekMoE 2B in this section refers to the model reported in Table 1, i.e., comprising 2.0B total parameters, with 1 shared expert and 7 out of 63 routed experts being activated.

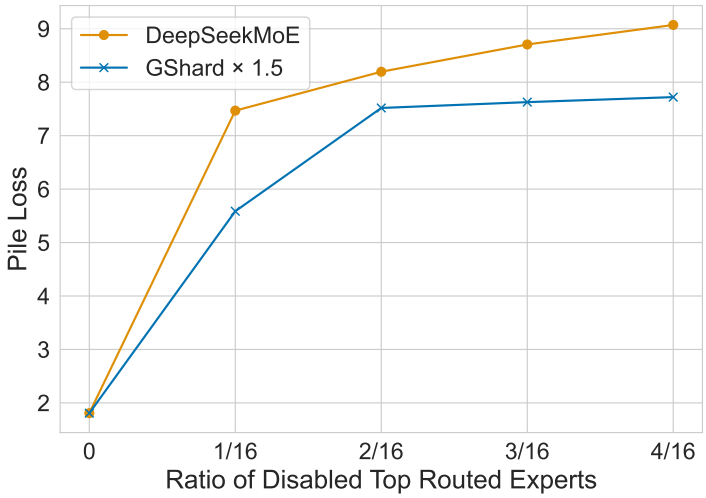

Figure 4 | Pile loss with regard to different ratios of disabled top routed experts. Notably, DeepSeekMoE exhibits greater sensitivity to the ratio of disabled top routed experts, indicating lower redundancy among routed experts in DeepSeekMoE.

DeepSeekMoE Exhibits Lower Redundancy Among Routed Experts. In order to assess the redundancy among routed experts, we disable varying ratios of top routed experts and evaluate the Pile loss. To be specific, for each token, we mask a certain ratio of experts with the highest routing probability, and then select top-K experts from the remaining routed experts. For fairness, we compare DeepSeekMoE with GShard×1.5 since they have the same Pile loss when no experts are disabled. As shown in Figure 4, compared with GShard×1.5, DeepSeekMoE is more sensitive to the disabling of top routed experts. This sensitivity suggests a lower level of parameter redundancy in DeepSeekMoE, since each routed expert is more irreplaceable. In contrast, GShard×1.5 exhibits greater redundancy among its expert parameters, so it can buffer the performance drop when top routed experts are disabled.

Shared Experts Are Irreplaceable by Routed Experts. In order to investigate the role of the shared expert in DeepSeekMoE, we disable it and activate one more routed expert. The evaluation on Pile shows a significant increase in the Pile loss, rising from 1.808 to 2.414, even though we maintain the same computational cost. This result highlights the crucial function of the shared expert and indicates that the shared expert captures fundamental and essential knowledge not shared with routed experts, making it irreplaceable by routed ones.

DeepSeekMoE Acquires Knowledge More Accurately. In order to validate our claim that higher flexibility in combining activated experts contributes to a more accurate and targeted knowledge acquisition, we investigate whether DeepSeekMoE can acquire requisite knowledge with fewer activated experts. To be specific, we vary the number of activated routed experts from 3 to 7 and evaluate the resulting Pile loss. As demonstrated in Figure 5, even with only

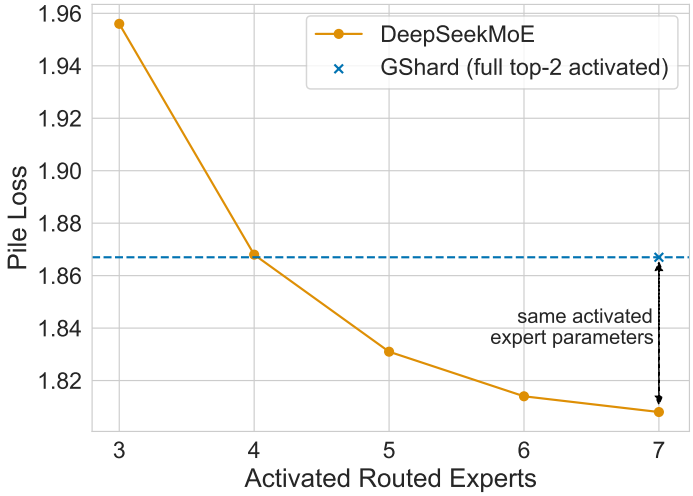

Figure 5 | Pile loss with regard to different numbers of activated routed experts in DeepSeekMoE. With only 4 routed experts activated, DeepSeekMoE achieves a Pile loss comparable with GShard.

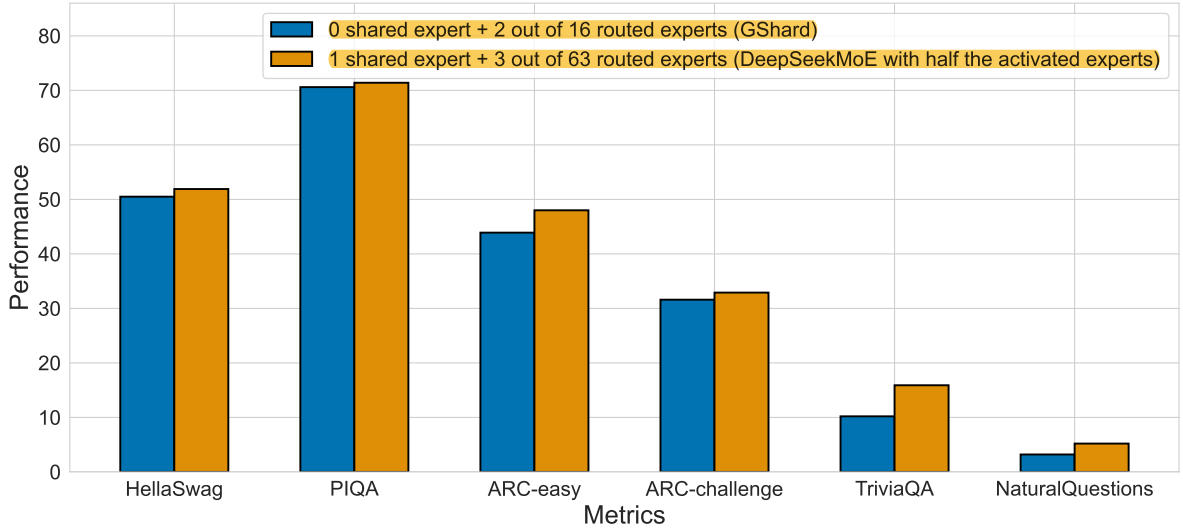

Figure 6 | Comparison between GShard and DeepSeekMoE with half the activated experts (trained from scratch). With the same total expert parameters and only half of the activated expert parameters, DeepSeekMoE still outperforms GShard.

4 routed experts activated, DeepSeekMoE achieves a Pile loss comparable with GShard. This observation supports the proposition that DeepSeekMoE can acquire requisite knowledge more accurately and efficiently.

Encouraged by these findings, in order to validate the expert specialization and accurate knowledge acquisition of DeepSeekMoE more rigorously, we train a new model from scratch. This model comprises 1 shared expert and 63 routed experts, where only 3 routed experts are activated. The evaluation results shown in Figure 6 demonstrate that, even with the same total expert parameters and only half of the activated expert parameters, DeepSeekMoE still outperforms GShard. This highlights the ability of DeepSeekMoE to leverage expert parameters

more efficiently, i.e., the proportion of effective parameters in the activated experts is much higher than that of GShard.

5. Scaling up to DeepSeekMoE 16B

With the DeepSeekMoE architecture, we scale up our MoE model to a larger scale with 16B total parameters and train it on 2T tokens. Our results demonstrate that compared with LLaMA2 7B, DeepSeekMoE 16B achieves superior performance with only about 40% of computations.

5.1. Experimental Setup

5.1.1. Training Data and Tokenization

We sample the training data from the same corpus as described in Section 4.1.1. Different from the validation experiments, we sample a larger amount of data with 2T tokens, aligning with the number of training tokens of LLaMA2 7B. We also use the HuggingFace Tokenizer tools to train a BPE tokenizer, but the vocabulary size is set to 100K for DeepSeekMoE 16B.

5.1.2. Hyper-Parameters

Model Settings. For DeepSeekMoE 16B, we set the number of Transformer layers to 28 and the hidden dimension to 2048. We employ the multi-head attention mechanism with a total of 16 attention heads, where each head has a dimension of 128. As for initialization, all learnable parameters are randomly initialized with a standard deviation of 0.006. We substitute all FFNs except for the first layer with MoE layers, since we observe that the load balance status converges especially slower for the first layer. Each MoE layer consists of 2 shared experts and 64 routed experts, where each expert is 0.25 times the size of a standard FFN. Each token will be routed to these 2 shared experts and 6 out of 64 routed experts. An even finer expert segmentation granularity is not employed due to the potential reduction in computational efficiency associated with excessively small expert sizes. At a larger scale over 16B, a finer granularity can still be employed. Under our configuration, DeepSeekMoE 16B has approximately 16.4B total parameters, with the number of activated parameters around 2.8B.

Training Settings. We employ the AdamW optimizer (Loshchilov and Hutter, 2019) with hyper-parameters set to $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight_decay = 0.1. The learning rate is also scheduled using a warmup-and-step-decay strategy. Initially, the learning rate linearly increases from 0 to the maximum value during the first 2K steps. Subsequently, the learning rate is multiplied by 0.316 at 80% of the training steps, and again by 0.316 at 90% of the training steps. The maximum learning rate for DeepSeekMoE 16B is set to 4.2×10^{-4}, and the gradient clipping norm is set to 1.0. The batch size is set to 4.5K, and with a maximum sequence length of 4K, each training batch contains 18M tokens. Correspondingly, the total number of training steps is set to 106,449 to achieve 2T training tokens. Due to the abundance of training data, we do not use dropout during training. We leverage pipeline parallelism to deploy different layers of a model on different devices, and for each layer, all the experts will be deployed on the same device. Therefore, we also do not drop any tokens during training and do not employ the device-level balance loss. In order to prevent routing collapse, we set a quite small expert-level balance factor of 0.001 because we find that under our parallelization strategy, a higher expert-level balance factor cannot increase the computation efficiency, but instead, it will compromise the model performance.

5.1.3. Evaluation Benchmarks

In addition to the benchmarks used in the validation experiments, we incorporate additional benchmarks for a more comprehensive evaluation. We introduce the distinctions from the benchmarks used in validation experiments as follows.

Language Modeling. For language modeling, we also evaluate the models on the test set of Pile (Gao et al., 2020). Since the tokenizer used in DeepSeekMoE 16B is different from that used in LLaMA2 7B. For a fair comparison, we use bits per byte (BPB) as the evaluation metric.

Reading Comprehension. For reading comprehension, we additionally consider DROP (Dua et al., 2019). The evaluation metric is the Exactly Matching (EM) rate.

Math Reasoning. For math reasoning, we additionally incorporate GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), using EM as the evaluation metric.

Multi-Subject Multiple-Choice. For multi-subject multiple-choice, we additionally evaluate the models on MMLU (Hendrycks et al., 2020). The evaluation metric is accuracy.

Disambiguation. For disambiguation, we additionally consider WinoGrande (Sakaguchi et al., 2019) and the evaluation metric is accuracy.

Chinese Benchmarks. Since DeepSeekMoE 16B is pretrained on a bilingual corpus, we also evaluate it on four Chinese benchmarks. CLUEWSC (Xu et al., 2020) is a Chinese disambiguation benchmark. CEval (Huang et al., 2023) and CMMLU (Li et al., 2023) are two Chinese multi-subject multiple-choice benchmarks with a similar form to MMLU. CHID (Zheng et al., 2019) is a Chinese idiom completion benchmark, aiming to evaluate the understanding of Chinese culture. The evaluation metrics for the aforementioned Chinese benchmarks are accuracy or EM.

Open LLM Leaderboard. We evaluate all of the aforementioned benchmarks based on our internal evaluation framework. In order to compare DeepSeekMoE 16B with open source models fairly and conveniently, we additionally evaluate DeepSeekMoE 16B on the Open LLM Leaderboard. The Open LLM Leaderboard is a public leaderboard supported by HuggingFace, it consists of six tasks: ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2020), TruthfulQA (Lin et al., 2022), Winogrande (Sakaguchi et al., 2019), and GSM8K (Cobbe et al., 2021).

5.2. Evaluations

5.2.1. Internal Comparison with DeepSeek 7B

We first conduct an internal comparison between DeepSeekMoE 16B and DeepSeek 7B (DeepSeek-AI, 2024), a dense language model with 6.9B parameters. Ensuring fairness, both models are trained on the same corpus with 2T tokens. This enables an accurate assessment of the effectiveness of our MoE architecture, independent of the influence of the training data.

Metric	# Shot	DeepSeek 7B (Dense)	DeepSeekMoE 16B
# Total Params	N/A	6.9B	16.4B
# Activated Params	N/A	6.9B	2.8B
FLOPs per 4K Tokens	N/A	183.5T	74.4T
# Training Tokens	N/A	2T	2T
Pile (BPB)	N/A	0.75	**0.74**
HellaSwag (Acc.)	0-shot	75.4	**77.1**
PIQA (Acc.)	0-shot	79.2	**80.2**
ARC-easy (Acc.)	0-shot	**67.9**	**68.1**
ARC-challenge (Acc.)	0-shot	48.1	**49.8**
RACE-middle (Acc.)	5-shot	**63.2**	61.9
RACE-high (Acc.)	5-shot	**46.5**	**46.4**
DROP (EM)	1-shot	**34.9**	32.9
GSM8K (EM)	8-shot	17.4	**18.8**
MATH (EM)	4-shot	3.3	**4.3**
HumanEval (Pass@1)	0-shot	26.2	**26.8**
MBPP (Pass@1)	3-shot	**39.0**	**39.2**
TriviaQA (EM)	5-shot	59.7	**64.8**
NaturalQuestions (EM)	5-shot	22.2	**25.5**
MMLU (Acc.)	5-shot	**48.2**	45.0
WinoGrande (Acc.)	0-shot	**70.5**	**70.2**
CLUEWSC (EM)	5-shot	**73.1**	72.1
CEval (Acc.)	5-shot	**45.0**	40.6
CMMLU (Acc.)	5-shot	**47.2**	42.5
CHID (Acc.)	0-shot	**89.3**	**89.4**

Table 3 | Comparison between DeepSeek 7B and DeepSeekMoE 16B. **Bold** font indicates the best or near the best. With only 40.5% of computations, DeepSeekMoE 16B achieves comparable performance with DeepSeek 7B.

The evaluation results are presented in Table 3, yielding the following observations: (1) On the whole, with about only 40% of the computations, DeepSeekMoE 16B achieves comparable performance with DeepSeek 7B. (2) DeepSeekMoE 16B exhibits notable strengths in language modeling and knowledge-intensive tasks such as Pile, HellaSwag, TriviaQA, and NaturalQuestions. Given that in an MoE model, FFN parameters are much heavier than attention parameters, these outcomes align with the proposition that FFNs in Transformers exhibit the capability for knowledge memorization (Dai et al., 2022a). (3) Compared with the excellent performance on other tasks, DeepSeekMoE exhibits limitations in addressing multiple-choice tasks. This inadequacy stems from the limited attention parameters in DeepSeekMoE 16B (DeepSeekMoE 16B has only about 0.5B attention parameters, while DeepSeek 7B has 2.5B attention parameters). Our earlier investigation on DeepSeek 7B reveals a positive correlation between the attention capacity and performance on multiple-choice tasks. For example, DeepSeek 7B MQA, which is equipped with the multi-query attention mechanism (Shazeer, 2019), also struggled in MMLU-like tasks. In addition, for a more comprehensive understanding of the training process of

DeepSeekMoE 16B, we also provide the benchmark curves of DeepSeekMoE 16B and DeepSeek 7B (Dense) during training in Appendix C for reference.

Critically, due to the modest number of parameters in DeepSeekMoE 16B, it enables single-device deployment on a GPU with 40GB of memory. With appropriate operator optimizations, it can achieve nearly 2.5 times the inference speed of a 7B dense model.

Metric	# Shot	LLaMA2 7B	DeepSeekMoE 16B
# Total Params	N/A	6.7B	16.4B
# Activated Params	N/A	6.7B	2.8B
FLOPs per 4K Tokens	N/A	187.9T	74.4T
# Training Tokens	N/A	2T	2T
Pile (BPB)	N/A	0.76	**0.74**
HellaSwag (Acc.)	0-shot	75.6	**77.1**
PIQA (Acc.)	0-shot	78.0	**80.2**
ARC-easy (Acc.)	0-shot	**69.1**	68.1
ARC-challenge (Acc.)	0-shot	49.0	**49.8**
RACE-middle (Acc.)	5-shot	60.7	**61.9**
RACE-high (Acc.)	5-shot	45.8	**46.4**
DROP (EM)	1-shot	**34.0**	32.9
GSM8K (EM)	8-shot	15.5	**18.8**
MATH (EM)	4-shot	2.6	**4.3**
HumanEval (Pass@1)	0-shot	14.6	**26.8**
MBPP (Pass@1)	3-shot	21.8	**39.2**
TriviaQA (EM)	5-shot	63.8	**64.8**
NaturalQuestions (EM)	5-shot	**25.5**	**25.5**
MMLU (Acc.)	5-shot	**45.8**	45.0
WinoGrande (Acc.)	0-shot	69.6	**70.2**
CLUEWSC (EM)	5-shot	64.0	**72.1**
CEval (Acc.)	5-shot	33.9	**40.6**
CMMLU (Acc.)	5-shot	32.6	**42.5**
CHID (Acc.)	0-shot	37.9	**89.4**

Table 4 | Comparison between LLaMA2 7B and DeepSeekMoE 16B. With only 39.6% of computations, DeepSeekMoE 16B outperforms LLaMA2 7B on the majority of benchmarks.

5.2.2. *Comparison with Open Source Models*

Internal Comparison with LLaMA2 7B. In the realm of open source models, we mainly compare DeepSeekMoE 16B with LLaMA2 7B (Touvron et al., 2023b), a well-known and strong open source language model with 6.7B parameters. Both DeepSeekMoE 16B and LLaMA2 7B are pre-trained on 2T tokens. Compared with LLaMA2 7B, DeepSeekMoE has 245% of total parameters but only needs 39.6% of computations. The results on our internal benchmarks are presented in Table 4, leading to the following observations. (1) Among the evaluated benchmarks, with only about 40% of computations, DeepSeekMoE 16B outperforms LLaMA2 7B on the majority of benchmarks. (2) The math reasoning and code generation capabilities of DeepSeekMoE 16B

are stronger than LLaMA2 7B, attributed to the enriched presence of mathematical and code-related text in our pretraining corpus. (3) Given the presence of Chinese texts in our pretraining corpus, DeepSeekMoE 16B exhibits a substantial performance advantage over LLaMA2 7B on Chinese benchmarks. (4) Despite being trained on fewer English texts, DeepSeekMoE 16B achieves comparable or better performance compared with LLaMA2 7B on English understanding or knowledge-intensive benchmarks, which demonstrates the exceptional capabilities of DeepSeekMoE 16B.

Evaluation on Open LLM Leaderboard. Beyond our internal evaluations, we also evaluate DeepSeekMoE 16B on the Open LLM Leaderboard and compare it with other open source models. In addition to LLaMA2 7B, we take a broader set of open source models into consideration, including LLaMA 7B (Touvron et al., 2023a), Falcon 7B (Almazrouei et al., 2023), GPT-J 6B (Wang and Komatsuzaki, 2021), RedPajama-INCITE 7B and 3B (Together-AI, 2023), Open LLaMA 7B and 3B (Geng and Liu, 2023), OPT 2.7B (Zhang et al., 2022), Pythia 2.8B (Biderman et al., 2023), GPT-neo 2.7B (Black et al., 2021), and BLOOM 3B (Scao et al., 2022). The evaluation results, as presented in Figure 1, show that DeepSeekMoE 16B consistently outperforms models with similar activated parameters by a large margin. Moreover, it achieves comparable performance with LLaMA2 7B, which has approximately 2.5 times the activated parameters.

6. Alignment for DeepSeekMoE 16B

Previous research indicates that MoE models typically do not emerge significant gains from fine-tuning (Artetxe et al., 2022; Fedus et al., 2021). However, Shen et al. (2023) present findings suggesting that MoE models can indeed benefit from instruction tuning. In order to assess whether DeepSeekMoE 16B can benefit from fine-tuning, we conduct supervised fine-tuning to construct a chat model based on DeepSeekMoE 16B. The experimental results reveal that DeepSeekMoE Chat 16B also achieves comparable performance with LLaMA2 SFT 7B and DeepSeek Chat 7B.

6.1. Experimental Setup

Training Data. For training the chat model, we conduct supervised fine-tuning (SFT) on our in-house curated data, comprising 1.4M training examples. This dataset spans a broad range of categories including math, code, writing, question answering, reasoning, summarization, and more. The majority of our SFT training data is in English and Chinese, rendering the chat model versatile and applicable in bilingual scenarios.

Hyper-Parameters. During supervised fine-tuning, we set the batch size to 1024 examples and conduct training over 8 epochs using the AdamW optimizer (Loshchilov and Hutter, 2019). We employ a maximum sequence length of 4K, and pack the training examples as densely as possible until reaching the sequence length limit. We do not use dropout for supervised fine-tuning, and simply set a constant learning rate of 10^{-5} without incorporating any learning rate scheduling strategy.

Evaluation Benchmarks. For the evaluation of the chat models, we employ benchmarks similar to those used in Section 5.1.3, with the following adjustments: (1) We exclude Pile (Gao et al., 2020) since chat models are seldom employed for pure language modeling. (2) We exclude

CHID (Zheng et al., 2019) due to the observed instability of results, hindering the derivation of solid conclusions. (3) We additionally include BBH (Suzgun et al., 2022) to provide a more comprehensive assessment of the reasoning ability of the chat models.

Metric	# Shot	LLaMA2 SFT 7B	DeepSeek Chat 7B	DeepSeekMoE Chat 16B
# Total Params	N/A	6.7B	6.9B	16.4B
# Activated Params	N/A	6.7B	6.9B	2.8B
FLOPs per 4K Tokens	N/A	187.9T	183.5T	74.4T
HellaSwag (Acc.)	0-shot	67.9	71.0	**72.2**
PIQA (Acc.)	0-shot	76.9	78.4	**79.7**
ARC-easy (Acc.)	0-shot	69.7	**70.2**	**69.9**
ARC-challenge (Acc.)	0-shot	**50.8**	50.2	50.0
BBH (EM)	3-shot	39.3	**43.1**	42.2
RACE-middle (Acc.)	5-shot	63.9	**66.1**	64.8
RACE-high (Acc.)	5-shot	49.6	**50.8**	**50.6**
DROP (EM)	1-shot	40.0	**41.7**	33.8
GSM8K (EM)	0-shot	**63.4**	62.6	62.2
MATH (EM)	4-shot	13.5	14.7	**15.2**
HumanEval (Pass@1)	0-shot	35.4	45.1	**45.7**
MBPP (Pass@1)	3-shot	27.8	39.0	**46.2**
TriviaQA (EM)	5-shot	60.1	59.5	**63.3**
NaturalQuestions (EM)	0-shot	**35.2**	32.7	**35.1**
MMLU (Acc.)	0-shot	**50.0**	49.7	47.2
WinoGrande (Acc.)	0-shot	65.1	68.4	**69.0**
CLUEWSC (EM)	5-shot	48.4	66.2	**68.2**
CEval (Acc.)	0-shot	35.1	**44.7**	40.0
CMMLU (Acc.)	0-shot	36.9	**51.2**	49.3

Table 5 | Comparison among LLaMA2 SFT 7B, DeepSeek Chat 7B and DeepSeekMoE Chat 16B, with all of these three models fine-tuned on the same SFT data. Compared with both 7B dense models, DeepSeekMoE Chat 16B still achieves comparable or better performance on the majority of benchmarks with only 40% of computations.

6.2. Evaluations

Baselines. In order to validate the potential of DeepSeekMoE 16B after alignment, we conduct supervised fine-tuning for LLaMA2 7B, DeepSeek 7B, and DeepSeekMoE 16B, where we utilize totally the same fine-tuning data to ensure fairness. Correspondingly, we construct three chat models, including LLaMA2 SFT 7B[3], DeepSeek Chat 7B, and DeepSeekMoE Chat 16B. Subsequently, we compare DeepSeekMoE Chat 16B with the other two dense chat models (with about 2.5 times the FLOPs) across a wide range of downstream tasks.

[3]We use LLaMA2 SFT to distinguish from the official LLaMA2 Chat (Touvron et al., 2023b) model.

Results. The evaluation results are presented in Table 5. Our key observations include: (1) DeepSeekMoE Chat 16B, while consuming nearly 40% of computations, achieves comparable performance with 7B dense models across language understanding and reasoning (PIQA, ARC, BBH), machine reading comprehension (RACE), mathematical (GSM8K, MATH), and knowledge-intensive tasks (TriviaQA, NaturalQuestions). (2) On code generation tasks, DeepSeekMoE Chat 16B significantly outperforms LLaMA2 SFT 7B, demonstrating notable improvements on HumanEval and MBPP. In addition, it also surpasses DeepSeek Chat 7B. (3) On multiple-choice question answering benchmarks including MMLU, CEval, and CMMLU, DeepSeekMoE Chat 16B still falls behind DeepSeek Chat 7B, consistent with the observations for the base model (Section 5.2.1). However, it is worth noting that, after supervised fine-tuning, the performance gap between DeepSeekMoE 16B and DeepSeek 7B is narrowed. (4) Benefiting from the pretraining on a bilingual corpus, DeepSeekMoE Chat 16B notably outperforms LLaMA2 SFT 7B on all Chinese benchmarks. These results demonstrate the balanced capabilities of DeepSeekMoE 16B in both Chinese and English, enhancing its versatility and applicability in diverse scenarios. In conclusion, the evaluation for the chat models highlights the potential of DeepSeekMoE 16B in benefiting from alignment, and validates its consistent advantages in achieving comparable performance with dense models while using only about 40% of computations.

7. DeepSeekMoE 145B Ongoing

Encouraged by the outstanding performance of DeepSeekMoE 16B, we further undertake a preliminary endeavor to scale up DeepSeekMoE to 145B. In this initial study, DeepSeekMoE 145B is trained on 245B tokens, but it has demonstrated consistent advantages over the GShard architecture and shown promise to match or exceed the performance of DeepSeek 67B (Dense). Furthermore, upon the completion of the final version and full training of DeepSeekMoE 145B, we also plan to make it publicly available.

7.1. Experimental Setup

Training Data and Tokenization. For DeepSeekMoE 145B, we employ exactly the same training corpus and tokenizer as DeepSeekMoE 16B, with the only difference being that DeepSeekMoE 145B is trained on 245B tokens for an initial study.

Model Settings. For DeepSeekMoE 145B, we set the number of Transformer layers to 62 and the hidden dimension to 4096. We employ the multi-head attention mechanism with a total of 32 attention heads, where each head has a dimension of 128. As for initialization, all learnable parameters are randomly initialized with a standard deviation of 0.006. As in DeepSeekMoE 16B, we also substitute all FFNs except for the first layer with MoE layers. Each MoE layer consists of 4 shared experts and 128 routed experts, where each expert is 0.125 times the size of a standard FFN. Each token will be routed to these 4 shared experts and 12 out of 128 routed experts. Under this configuration, DeepSeekMoE 145 has approximately 144.6B total parameters, with the number of activated parameters around 22.2B.

Training Settings. We employ the AdamW optimizer (Loshchilov and Hutter, 2019) with hyper-parameters set to $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight_decay = 0.1. For the preliminary study of DeepSeekMoE 145B, we employ a warmup-and-constant learning rate scheduler. Initially, the learning rate linearly increases from 0 to the maximum value during the first 2K steps.

Subsequently, the learning rate keeps constant during the remaining training process. The maximum learning rate for DeepSeekMoE 145B is set to 3.0×10^{-4}, and the gradient clipping norm is set to 1.0. The batch size is set to 4.5K, and with a maximum sequence length of 4K, each training batch contains 18M tokens. We train DeepSeekMoE 145B for 13,000 steps, achieving 245B training tokens. Also, we do not use dropout during training. We leverage pipeline parallelism to deploy different layers of a model on different devices, and for each layer, all the routed experts will be uniformly deployed on 4 devices (i.e., expert parallelism combined with data parallelism). Since we employ expert parallelism for DeepSeekMoE 145B, the device-level load balance should be considered to reduce the computational bottleneck. In response, we set the device-level balance factor to 0.05 to encourage balanced computation across devices. Also, we still set a small expert-level balance factor of 0.003 to prevent routing collapse.

Evaluation Benchmarks. We evaluate DeepSeekMoE 145B on exactly the same internal benchmarks as used for DeepSeekMoE 16B (see Section 5.1.3).

7.2. Evaluations

Baselines. Apart from **DeepSeekMoE 145B**, we consider three additional models for comparison. **DeepSeek 67B (Dense)** is a dense model with 67.4B total parameters (refer to DeepSeek-AI (2024) for the model and training details). **GShard 137B** shares the same hidden dimension and number of layers as DeepSeekMoE 145B, but follows the GShard architecture. Note that DeepSeekMoE 145B aligns the intermediate hidden dimension in each expert to a multiple of 64 for computation efficiency, so its model size is 6% larger than GShard 137B. **DeepSeekMoE 142B (Half Activated)** has a similar architecture to DeepSeekMoE 145B, but it contains only 2 shared experts, and only 6 out of 128 routed experts are activated. It is noteworthy that all compared models, including DeepSeekMoE 145B, share the same training corpus. In addition, all MoE models in the comparison are trained from scratch and share the same training hyper-parameters.

Results. From the evaluation results presented in Table 6, we have the following observations: (1) Despite having comparable total parameters and computations, DeepSeekMoE 145B significantly outperforms GShard 137B, highlighting the advantages of the DeepSeekMoE architecture again. (2) On the whole, with only 28.5% of computations, DeepSeekMoE 145B achieves comparable performance with DeepSeek 67B (Dense). Consistent with the findings from DeepSeekMoE 16B, DeepSeekMoE 145B exhibits remarkable strengths in language modeling and knowledge-intensive tasks, but with limitations in multiple-choice tasks. (3) At a larger scale, the performance of DeepSeekMoE 142B (Half Activated) does not lag behind too much from DeepSeekMoE 145B. In addition, despite having only a half of activated expert parameters, DeepSeekMoE 142B (Half Activated) still match the performance of DeepSeek 67B (Dense), with only 18.2% of computations. It also outperforms GShard 137B, which aligns with the conclusion from Section 4.5.

8. Related Work

The Mixture of Experts (MoE) technique is first proposed by Jacobs et al. (1991); Jordan and Jacobs (1994) to deal with different samples with independent expert modules. Shazeer et al. (2017) introduce MoE into language model training and build a large-scale LSTM-based (Hochreiter and Schmidhuber, 1997) MoE models. As Transformer become the most popular architecture

Metric	# Shot	DeepSeek 67B (Dense)	GShard 137B	DeepSeekMoE 145B	DeepSeekMoE 142B (Half Activated)
# Total Params	N/A	67.4B	136.5B	144.6B	142.3B
# Activated Params	N/A	67.4B	21.6B	22.2B	12.2B
Relative Expert Size	N/A	N/A	1	0.125	0.125
# Experts	N/A	N/A	0 + 16	4 + 128	2 + 128
# Activated Experts	N/A	N/A	0 + 2	4 + 12	2 + 6
FLOPs per 4K Tokens	N/A	2057.5T	572.7T	585.6T	374.6T
# Training Tokens	N/A	245B	245B	245B	245B
Pile (Loss.)	N/A	1.905	1.961	**1.876**	1.888
HellaSwag (Acc.)	0-shot	74.8	72.0	**75.8**	74.9
PIQA (Acc.)	0-shot	79.8	77.6	**80.7**	80.2
ARC-easy (Acc.)	0-shot	69.0	64.0	**69.7**	67.9
ARC-challenge (Acc.)	0-shot	**50.4**	45.8	48.8	49.0
RACE-middle (Acc.)	5-shot	**63.2**	59.2	62.1	59.5
RACE-high (Acc.)	5-shot	**46.9**	43.5	45.5	42.6
DROP (EM)	1-shot	**27.5**	21.6	**27.8**	28.9
GSM8K (EM)	8-shot	**11.8**	6.4	**12.2**	13.8
MATH (EM)	4-shot	2.1	1.6	**3.1**	2.8
HumanEval (Pass@1)	0-shot	**23.8**	17.7	19.5	23.2
MBPP (Pass@1)	3-shot	**33.6**	27.6	**33.2**	32.0
TriviaQA (EM)	5-shot	57.2	52.5	**61.1**	59.8
NaturalQuestions (EM)	5-shot	22.6	19.0	**25.0**	23.5
MMLU (Acc.)	5-shot	**45.1**	26.3	39.4	37.5
WinoGrande (Acc.)	0-shot	70.7	67.6	**71.9**	70.8
CLUEWSC (EM)	5-shot	69.1	65.7	**71.9**	72.6
CEval (Acc.)	5-shot	**40.3**	26.2	37.1	32.8
CMMLU (Acc.)	5-shot	**40.6**	25.4	35.9	31.9
CHID (Acc.)	0-shot	88.5	86.9	**90.3**	88.3

Table 6 | Comparison among DeepSeek 67B (Dense) and MoE models at the scale of about 140B total parameters. In the lines of "# Experts" and "# Activated Experts", $a + b$ denotes a shared experts and b routed experts, respectively. **Bold** font indicates the best or near the best performance excluding the last column. DeepSeekMoE 145B, and even DeepSeekMoE 142B (Half Activated) that has only a half of activated expert parameters, outperform GShard 137B by a large margin. Moreover, with 28.5% of computations, DeepSeekMoE 145B achieves comparable performance with DeepSeek 67B.

for NLP, many attempts extend FFNs in a Transformer as MoE layers to build MoE language models. GShard (Lepikhin et al., 2021) and Switch Transformer (Fedus et al., 2021) are pioneers which employ learnable top-2 or top-1 routing strategies to scale the MoE language models to an extremely large scale. Hash Layer (Roller et al., 2021) and StableMoE (Dai et al., 2022b) use fixed routing strategies for more stable routing and training. Zhou et al. (2022) propose an expert-choice routing strategy, where each token can be assigned to different numbers of experts. Zoph (2022) focus on the issues of training instability and fine-tuning difficulty in MoE models,

and propose ST-MoE to overcome these challenges. In addition to research on MoE architectures and training strategies, recent years have also witnessed the emergence of numerous large-scale language or multimodal models (Du et al., 2022; Lin et al., 2021; Ren et al., 2023; Xue et al., 2023) based on existing MoE architectures. By and large, most of the previous MoE models are based on conventional top-1 or top-2 routing strategies, leaving large room for improving expert specialization. In response, our DeepSeekMoE architecture aims to improve the expert specialization to the utmost extent.

9. Conclusion

In this paper, we introduce the DeepSeekMoE architecture for MoE language models, with the objective of achieving ultimate expert specialization. Through fine-grained expert segmentation and shared expert isolation, DeepSeekMoE achieves significantly higher expert specialization and performance compared with prevailing MoE architectures. Starting with a modest scale of 2B parameters, we validate the advantages of DeepSeekMoE, demonstrating its capability to approach the upper bound performance for MoE models. Furthermore, we provide empirical evidence to show that DeepSeekMoE has a higher level of expert specialization than GShard.

Scaling up to a larger scale of 16B total parameters, we train DeepSeekMoE 16B on 2T tokens and demonstrate its outstanding performance comparable with DeepSeek 7B and LLaMA2 7B, with only about 40% of computations. Additionally, supervised fine-tuning is conducted for alignment to construct an MoE chat model based on DeepSeekMoE 16B, further showing its adaptability and versatility. Further, we perform a preliminary exploration to scale DeepSeekMoE to 145B parameters. We find that DeepSeekMoE 145B still keeps substantial advantages over the GShard architecture, and demonstrates comparable performance with DeepSeek 67B, using only 28.5% (maybe even 18.2%) of computations.

For research purposes, we release the model checkpoint of DeepSeekMoE 16B to the public, which can be deployed on a single GPU with 40GB of memory. We aspire for this work to provide valuable insights for both academia and industry, and contribute to the accelerated advancement of large-scale language models.

References

E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, E. Goffinet, D. Heslow, J. Launay, Q. Malartic, B. Noune, B. Pannier, and G. Penedo. Falcon-40B: an open large language model with state-of-the-art performance, 2023.

M. Artetxe, S. Bhosale, N. Goyal, T. Mihaylov, M. Ott, S. Shleifer, X. V. Lin, J. Du, S. Iyer, R. Pasunuru, G. Anantharaman, X. Li, S. Chen, H. Akin, M. Baines, L. Martin, X. Zhou, P. S. Koura, B. O'Horo, J. Wang, L. Zettlemoyer, M. T. Diab, Z. Kozareva, and V. Stoyanov. Efficient large scale language modeling with mixtures of experts. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022, pages 11699–11732. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.804. URL https://doi.org/10.18653/v1/2022.emnlp-main.804.

J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, et al. Program synthesis with large language models. arXiv preprint arXiv:2108.07732, 2021.

S. Biderman, H. Schoelkopf, Q. G. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff, A. Skowron, L. Sutawika, and O. van der Wal. Pythia: A suite for analyzing large language models across training and scaling. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA, volume 202 of Proceedings of Machine Learning Research, pages 2397–2430. PMLR, 2023. URL https://proceedings.mlr.press/v202/biderman23a.html.

Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi. PIQA: reasoning about physical commonsense in natural language. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 7432–7439. AAAI Press, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

S. Black, L. Gao, P. Wang, C. Leahy, and S. Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, Mar. 2021. URL https://doi.org/10.5281/zenodo.5297715. If you use this misc, please cite it using these metadata.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating large language models trained on code. CoRR, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. CoRR, abs/1803.05457, 2018. URL http://arxiv.org/abs/1803.05457.

K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.

D. Dai, L. Dong, Y. Hao, Z. Sui, B. Chang, and F. Wei. Knowledge neurons in pretrained transformers. In S. Muresan, P. Nakov, and A. Villavicencio, editors, Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022, pages 8493–8502. Association for Computational Linguistics, 2022a. doi: 10.18653/V1/2022.ACL-LONG.581. URL https://doi.org/10.18653/v1/2022.acl-long.581.

D. Dai, L. Dong, S. Ma, B. Zheng, Z. Sui, B. Chang, and F. Wei. Stablemoe: Stable routing strategy for mixture of experts. In S. Muresan, P. Nakov, and A. Villavicencio, editors, Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022, pages 7085–7095. Association for Computational Linguistics, 2022b. doi: 10.18653/V1/2022.ACL-LONG.489. URL https://doi.org/10.18653/v1/2022.acl-long.489.

DeepSeek-AI. Deepseek llm: Scaling open-source language models with longtermism. arXiv preprint arXiv:2401.02954, 2024.

N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, B. Zoph, L. Fedus, M. P. Bosma, Z. Zhou, T. Wang, Y. E. Wang, K. Webster, M. Pellat, K. Robinson, K. S. Meier-Hellstern, T. Duke, L. Dixon, K. Zhang, Q. V. Le, Y. Wu, Z. Chen, and C. Cui. Glam: Efficient scaling of language models with mixture-of-experts. In International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA, volume 162 of Proceedings of Machine Learning Research, pages 5547–5569. PMLR, 2022. URL https://proceedings.mlr.press/v162/du22c.html.

D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In J. Burstein, C. Doran, and T. Solorio, editors, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 2368–2378. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1246. URL https://doi.org/10.18653/v1/n19-1246.

W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. CoRR, abs/2101.03961, 2021. URL https://arxiv.org/abs/2101.03961.

L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, et al. The Pile: An 800GB dataset of diverse text for language modeling. arXiv preprint arXiv:2101.00027, 2020.

X. Geng and H. Liu. Openllama: An open reproduction of llama, May 2023. URL https://github.com/openlm-research/open_llama.

A. Harlap, D. Narayanan, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, and P. B. Gibbons. Pipedream: Fast and efficient pipeline parallel DNN training. CoRR, abs/1806.03377, 2018. URL http://arxiv.org/abs/1806.03377.

D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. arXiv preprint arXiv:2009.03300, 2020.

D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset, 2021.

High-Flyer. Hai-llm: An efficient and lightweight tool for training large models, 2023. URL https://www.high-flyer.cn/en/blog/hai-llm.

S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computing, 9(8):1735–1780, 1997. URL https://doi.org/10.1162/neco.1997.9.8.1735.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre. Training compute-optimal large language models. CoRR, abs/2203.15556, 2022. doi: 10.48550/arXiv.2203.15556. URL https://doi.org/10.48550/arXiv.2203.15556.

Y. Huang, Y. Bai, Z. Zhu, J. Zhang, J. Zhang, T. Su, J. Liu, C. Lv, Y. Zhang, J. Lei, et al. C-Eval: A multi-level multi-discipline chinese evaluation suite for foundation models. arXiv preprint arXiv:2305.08322, 2023.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. Neural Computing, 3(1):79–87, 1991. URL https://doi.org/10.1162/neco.1991.3.1.79.

M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. Neural Computing, 6(2):181–214, 1994. URL https://doi.org/10.1162/neco.1994.6.2.181.

M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. arXiv e-prints, art. arXiv:1705.03551, 2017.

V. A. Korthikanti, J. Casper, S. Lym, L. McAfee, M. Andersch, M. Shoeybi, and B. Catanzaro. Reducing activation recomputation in large transformer models. Proceedings of Machine Learning and Systems, 5, 2023.

T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, M. Kelcey, J. Devlin, K. Lee, K. N. Toutanova, L. Jones, M.-W. Chang, A. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: a benchmark for question answering research. Transactions of the Association of Computational Linguistics, 2019.

G. Lai, Q. Xie, H. Liu, Y. Yang, and E. H. Hovy. RACE: large-scale reading comprehension dataset from examinations. In M. Palmer, R. Hwa, and S. Riedel, editors, Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017, pages 785–794. Association for Computational Linguistics, 2017. doi: 10.18653/V1/D17-1082. URL https://doi.org/10.18653/v1/d17-1082.

D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. In 9th International Conference on Learning Representations, ICLR 2021. OpenReview.net, 2021. URL https://openreview.net/forum?id=qrwe7XHTmYb.

H. Li, Y. Zhang, F. Koto, Y. Yang, H. Zhao, Y. Gong, N. Duan, and T. Baldwin. CMMLU: Measuring massive multitask language understanding in Chinese. arXiv preprint arXiv:2306.09212, 2023.

J. Lin, R. Men, A. Yang, C. Zhou, M. Ding, Y. Zhang, P. Wang, A. Wang, L. Jiang, X. Jia, J. Zhang, J. Zhang, X. Zou, Z. Li, X. Deng, J. Liu, J. Xue, H. Zhou, J. Ma, J. Yu, Y. Li, W. Lin, J. Zhou, J. Tang, and H. Yang. M6: A chinese multimodal pretrainer. CoRR, abs/2103.00823, 2021. URL https://arxiv.org/abs/2103.00823.

S. Lin, J. Hilton, and O. Evans. Truthfulqa: Measuring how models mimic human falsehoods. In S. Muresan, P. Nakov, and A. Villavicencio, editors, Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin,

Ireland, May 22-27, 2022, pages 3214–3252. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.ACL-LONG.229. URL https://doi.org/10.18653/v1/2022.acl-long.229.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro, et al. Efficient large-scale language model training on gpu clusters using megatron-lm. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pages 1–15, 2021.

OpenAI. GPT-4 technical report. CoRR, abs/2303.08774, 2023. doi: 10.48550/arXiv.2303.08774. URL https://doi.org/10.48550/arXiv.2303.08774.

S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He. Zero: memory optimizations toward training trillion parameter models. In C. Cuicchi, I. Qualters, and W. T. Kramer, editors, Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020, page 20. IEEE/ACM, 2020. doi: 10.1109/SC41405.2020.00024. URL https://doi.org/10.1109/SC41405.2020.00024.

S. Rajbhandari, C. Li, Z. Yao, M. Zhang, R. Y. Aminabadi, A. A. Awan, J. Rasley, and Y. He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation AI scale. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, editors, International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA, volume 162 of Proceedings of Machine Learning Research, pages 18332–18346. PMLR, 2022. URL https://proceedings.mlr.press/v162/rajbhandari22a.html.

X. Ren, P. Zhou, X. Meng, X. Huang, Y. Wang, W. Wang, P. Li, X. Zhang, A. Podolskiy, G. Arshinov, A. Bout, I. Piontkovskaya, J. Wei, X. Jiang, T. Su, Q. Liu, and J. Yao. Pangu-Σ: Towards trillion parameter language model with sparse heterogeneous computing. CoRR, abs/2303.10845, 2023. URL https://doi.org/10.48550/arXiv.2303.10845.

S. Roller, S. Sukhbaatar, A. Szlam, and J. Weston. Hash layers for large sparse models. CoRR, abs/2106.04426, 2021. URL https://arxiv.org/abs/2106.04426.

K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019.

T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilic, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, M. Gallé, J. Tow, A. M. Rush, S. Biderman, A. Webson, P. S. Ammanamanchi, T. Wang, B. Sagot, N. Muennighoff, A. V. del Moral, O. Ruwase, R. Bawden, S. Bekman, A. McMillan-Major, I. Beltagy, H. Nguyen, L. Saulnier, S. Tan, P. O. Suarez, V. Sanh, H. Laurençon, Y. Jernite, J. Launay, M. Mitchell, C. Raffel, A. Gokaslan, A. Simhi, A. Soroa, A. F. Aji, A. Alfassy, A. Rogers, A. K. Nitzav, C. Xu, C. Mou, C. Emezue, C. Klamm, C. Leong, D. van Strien, D. I. Adelani, and et al. BLOOM: A 176b-parameter open-access multilingual language model. CoRR, abs/2211.05100, 2022. doi: 10.48550/ARXIV.2211.05100. URL https://doi.org/10.48550/arXiv.2211.05100.

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers. The Association for Computer Linguistics, 2016. doi: 10.18653/V1/P16-1162. URL https://doi.org/10.18653/v1/p16-1162.

N. Shazeer. Fast transformer decoding: One write-head is all you need. CoRR, abs/1911.02150, 2019. URL http://arxiv.org/abs/1911.02150.

N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In 5th International Conference on Learning Representations, ICLR 2017. OpenReview.net, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

S. Shen, L. Hou, Y. Zhou, N. Du, S. Longpre, J. Wei, H. W. Chung, B. Zoph, W. Fedus, X. Chen, T. Vu, Y. Wu, W. Chen, A. Webson, Y. Li, V. Zhao, H. Yu, K. Keutzer, T. Darrell, and D. Zhou. Flan-moe: Scaling instruction-finetuned language models with sparse mixture of experts. CoRR, abs/2305.14705, 2023. doi: 10.48550/ARXIV.2305.14705. URL https://doi.org/10.48550/arXiv.2305.14705.

M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. arXiv preprint arXiv:1909.08053, 2019.

M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. Chi, D. Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. arXiv preprint arXiv:2210.09261, 2022.

P. Tillet, H. T. Kung, and D. Cox. Triton: An intermediate language and compiler for tiled neural network computations. In Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages, MAPL 2019, page 10–19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367196. doi: 10.1145/3315508.3329973. URL https://doi.org/10.1145/3315508.3329973.

Together-AI. Redpajama-data: An open source recipe to reproduce llama training dataset, April 2023. URL https://github.com/togethercomputer/RedPajama-Data.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models. CoRR, abs/2302.13971, 2023a. doi: 10.48550/arXiv.2302.13971. URL https://doi.org/10.48550/arXiv.2302.13971.

H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. Canton-Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models. CoRR, abs/2307.09288, 2023b. doi: 10.48550/arXiv.2307.09288. URL https://doi.org/10.48550/arXiv.2307.09288.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

B. Wang and A. Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

L. Xu, H. Hu, X. Zhang, L. Li, C. Cao, Y. Li, Y. Xu, K. Sun, D. Yu, C. Yu, Y. Tian, Q. Dong, W. Liu, B. Shi, Y. Cui, J. Li, J. Zeng, R. Wang, W. Xie, Y. Li, Y. Patterson, Z. Tian, Y. Zhang, H. Zhou, S. Liu, Z. Zhao, Q. Zhao, C. Yue, X. Zhang, Z. Yang, K. Richardson, and Z. Lan. CLUE: A chinese language understanding evaluation benchmark. In D. Scott, N. Bel, and C. Zong, editors, Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020, pages 4762–4772. International Committee on Computational Linguistics, 2020. doi: 10.18653/V1/2020.COLING-MAIN.419. URL https://doi.org/10.18653/v1/2020.coling-main.419.

F. Xue, Z. Zheng, Y. Fu, J. Ni, Z. Zheng, W. Zhou, and Y. You. Openmoe: Open mixture-of-experts language models. https://github.com/XueFuzhao/OpenMoE, 2023.

R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. HellaSwag: Can a machine really finish your sentence? In A. Korhonen, D. R. Traum, and L. Màrquez, editors, Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, pages 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1472. URL https://doi.org/10.18653/v1/p19-1472.

S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

C. Zheng, M. Huang, and A. Sun. Chid: A large-scale chinese idiom dataset for cloze test. In A. Korhonen, D. R. Traum, and L. Màrquez, editors, Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, pages 778–787. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1075. URL https://doi.org/10.18653/v1/p19-1075.

Y. Zhou, T. Lei, H. Liu, N. Du, Y. Huang, V. Zhao, A. M. Dai, Z. Chen, Q. V. Le, and J. Laudon. Mixture-of-experts with expert choice routing. In NeurIPS, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/2f00ecd787b432c1d36f3de9800728eb-Abstract-Conference.html.

B. Zoph. Designing effective sparse expert models. In IEEE International Parallel and Distributed Processing Symposium, IPDPS Workshops 2022, Lyon, France, May 30 - June 3, 2022, page 1044. IEEE, 2022. URL https://doi.org/10.1109/IPDPSW55747.2022.00171.

Appendices

A. Overview of Hyper-Parameters

We present the overview of hyper-parameters for DeepSeekMoE across various sizes in Table 7.

# Params	# Layers	Hidden Size	# Attn Heads	# Shared Experts	# Routed Experts	Relative Expert Size	Sequence Length	Batch Size (Sequence)	Learning Rate
2.0B	9	1280	10	1	63 (7 activated)	0.25	2048	2048	1.08e-3
16.4B	28	2048	16	2	64 (6 activated)	0.25	4096	4608	4.2e-4
144.6B	62	4096	32	4	128 (12 activated)	0.125	4096	4608	3.0e-4

Table 7 | Overview of hyper-parameters for DeepSeekMoE across various sizes. The relative expert size is in comparison to a standard FFN.

B. Comparing DeepSeekMoE with Larger Models

Comparisons among DeepSeekMoE, GShard×1.2, and GShard×1.5 are shown in Table 8. Comparisons among DeepSeekMoE, Dense×4, and Dense×16 are shown in Table 9.

Metric	**# Shot**	**GShard**×1.2	**GShard**×1.5	**DeepSeekMoE**
Relative Expert Size	N/A	1.2	1.5	0.25
# Experts	N/A	0 + 16	0 + 16	1 + 63
# Activated Experts	N/A	0 + 2	0 + 2	1 + 7
# Total Expert Params	N/A	2.3B	2.8B	1.9B
# Activated Expert Params	N/A	0.28B	0.35B	0.24B
# Training Tokens	N/A	100B	100B	100B
Pile (Loss)	N/A	1.824	**1.808**	**1.808**
HellaSwag (Acc.)	0-shot	53.7	54.4	**54.8**
PIQA (Acc.)	0-shot	71.8	71.1	**72.3**
ARC-easy (Acc.)	0-shot	46.8	47.3	**49.4**
ARC-challenge (Acc.)	0-shot	31.7	**34.1**	**34.3**
RACE-middle (Acc.)	5-shot	43.7	**46.4**	44.0
RACE-high (Acc.)	5-shot	31.9	**32.4**	31.7
HumanEval (Pass@1)	0-shot	3.7	3.0	**4.9**
MBPP (Pass@1)	3-shot	2.4	**2.6**	2.2
TriviaQA (EM)	5-shot	15.2	15.7	**16.6**
NaturalQuestions (EM)	5-shot	4.5	4.7	**5.7**

Table 8 | Comparison between DeepSeekMoE and larger GShard models.

At a larger scale of 13B total parameters, we also compare DeepSeekMoE with GShard×1.2 and GShard×1.5, and show results in Table 10. At a larger scale, DeepSeekMoE even outperforms GShard×1.5 distinctly.

Metric	# Shot	Dense×4	Dense×16	DeepSeekMoE
Relative Expert Size	N/A	1	1	0.25
# Experts	N/A	4 + 0	16 + 0	1 + 63
# Activated Experts	N/A	4 + 0	16 + 0	1 + 7
# Total Expert Params	N/A	0.47B	1.89B	1.89B
# Activated Expert Params	N/A	0.47B	1.89B	0.24B
# Training Tokens	N/A	100B	100B	100B
Pile (Loss)	N/A	1.908	**1.806**	**1.808**
HellaSwag (Acc.)	0-shot	47.6	**55.1**	**54.8**
PIQA (Acc.)	0-shot	70.0	71.9	**72.3**
ARC-easy (Acc.)	0-shot	43.9	**51.9**	49.4
ARC-challenge (Acc.)	0-shot	30.5	33.8	**34.3**
RACE-middle (Acc.)	5-shot	42.4	**46.3**	44.0
RACE-high (Acc.)	5-shot	30.7	**33.0**	31.7
HumanEval (Pass@1)	0-shot	1.8	4.3	**4.9**
MBPP (Pass@1)	3-shot	0.2	**2.2**	**2.2**
TriviaQA (EM)	5-shot	9.9	**16.5**	**16.6**
NaturalQuestions (EM)	5-shot	3.0	**6.3**	5.7

Table 9 | Comparison between DeepSeekMoE and larger dense baselines.

Metric	# Shot	GShard×1.2	GShard×1.5	DeepSeekMoE
Relative Expert Size	N/A	1.2	1.5	0.25
# Experts	N/A	0 + 16	0 + 16	1 + 63
# Activated Experts	N/A	0 + 2	0 + 2	1 + 7
# Total Expert Params	N/A	15.9B	19.8B	13.3B
# Activated Expert Params	N/A	2.37B	2.82B	2.05B
# Training Tokens	N/A	100B	100B	100B
HellaSwag (Acc.)	0-shot	66.6	67.7	**69.1**
PIQA (Acc.)	0-shot	75.6	**76.0**	**75.7**
ARC-easy (Acc.)	0-shot	56.8	56.8	**58.8**
ARC-challenge (Acc.)	0-shot	**39.9**	37.6	38.5
RACE-middle (Acc.)	5-shot	51.6	50.6	**52.4**
RACE-high (Acc.)	5-shot	37.4	36.3	**38.5**
HumanEval (Pass@1)	0-shot	6.1	6.1	**9.8**
MBPP (Pass@1)	3-shot	7.0	**11.6**	10.6
TriviaQA (EM)	5-shot	36.5	36.7	**38.2**
NaturalQuestions (EM)	5-shot	12.6	12.1	**13.7**

Table 10 | Comparison between DeepSeekMoE and larger GShard models at a larger scale.

C. Training Benchmark Curves of DeepSeekMoE 16B

We present the benchmark curves during training of DeepSeekMoE 16B and DeepSeek 7B (Dense) in Figure 7 for reference.

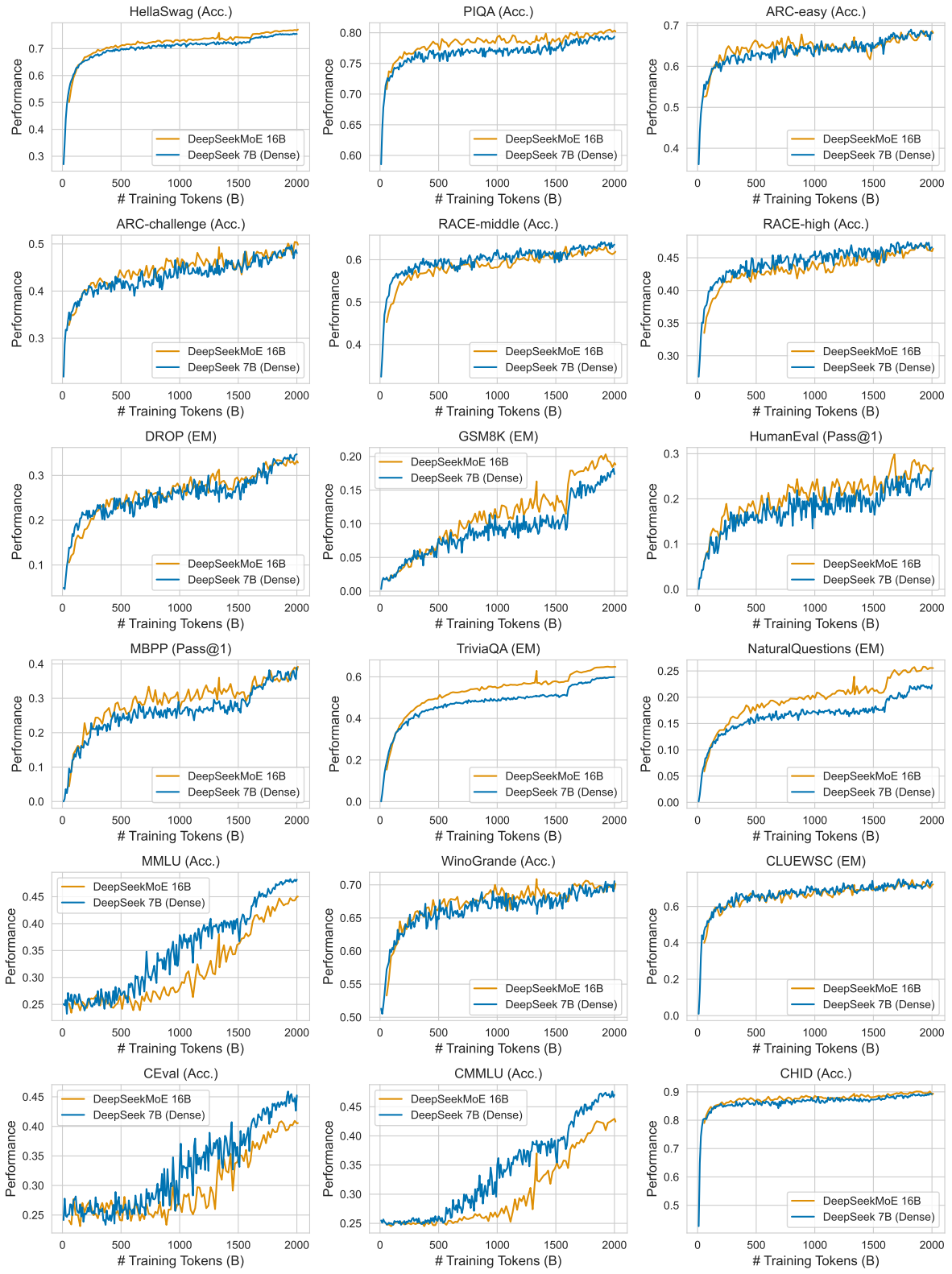

Figure 7 | Benchmark curves during training of DeepSeekMoE 16B and DeepSeek 7B (Dense).