LLaMA-MoE: Building Mixture-of-Experts from LLaMA with Continual Pre-training

Tong Zhu[1]*, **Xiaoye Qu**[2], **Daize Dong**[2], **Jiacheng Ruan**[3], **Jingqi Tong**[4],
Conghui He[2], **Yu Cheng**[5] ✉

[1] Soochow University [2] Shanghai AI Laboratory [3] Shanghai Jiao Tong University
[4] Fudan University [5] The Chinese University of Hong Kong
tzhu7@stu.suda.edu.cn, {quxiaoye,dongdaize.d,heconghui}@pjlab.org.cn,
jackchenruan@sjtu.edu.cn, jqtong23@m.fudan.edu.cn, chengyu@cse.cuhk.edu.hk

Abstract

Mixture-of-Experts (MoE) has gained increasing popularity as a promising framework for scaling up large language models (LLMs). However, training MoE from scratch in a large-scale setting still suffers from data-hungry and instability problems. Motivated by this limit, we investigate building MoE models from existing dense large language models. Specifically, based on the well-known LLaMA-2 7B model, we obtain an MoE model by: (1) *Expert Construction*, which partitions the parameters of original Feed-Forward Networks (FFNs) into multiple experts; (2) *Continual Pre-training*, which further trains the transformed MoE model and additional gate networks. In this paper, we comprehensively explore different methods for expert construction and various data sampling strategies for continual pre-training. After these stages, our LLaMA-MoE models could maintain language abilities and route the input tokens to specific experts with part of the parameters activated. Empirically, by training 200B tokens, LLaMA-MoE-3.5B models significantly outperform dense models that contain similar activation parameters. The source codes and models are available at https://github.com/pjlab-sys4nlp/llama-moe .

1 Introduction

Large language models (LLMs) (OpenAI, 2023; Touvron et al., 2023a; Cai et al., 2024) have presented remarkable understanding and reasoning capabilities on a wide range of tasks (Lu et al., 2024; Su et al., 2024a,b; Tao et al., 2024). Nowadays, scaling model size has become the de facto approach to further boost the overall performances (Wei et al., 2022; Hoffmann et al., 2022). However, the immense model size is unsustainable due to the computational costs (Fedus et al., 2022a). Motivated by the trade-off between scaling the model size and reducing the inference cost, sparse Mixture-of-Experts (MoE) is further proposed to activate a part of the model parameters (Shazeer et al., 2017; Lepikhin et al., 2020). Inspired by this, we focus on sparsely activated MoE models that decouple model size from computation costs.

However, training MoEs from scratch (Fedus et al., 2022b; Zoph et al., 2022; Xue et al., 2024; Dai et al., 2024) leads to a significant overall budget. In this work, we reduce the training costs by investigating building MoE models from existing dense LLMs. Moreover, starting from the dense model provides flexible structure design choices for MoE. In other words, we can place the MoE block in any transformer layers. In this paper, we dedicate to building a full MoE model from LLaMA (Touvron et al., 2023b), where each layer contains an MoE block.

---

*Work was done during an internship at Shanghai AI Laboratory.

To build strong LLaMA-MoE models, we identify two important challenges. First, **how to effectively construct experts** from the Feed-Forward Networks (FFNs) in the existing LLMs. There are works exploring splitting FFN parameters to construct experts on T5 (Zhang et al., 2021) or BERT (Zuo et al., 2022) models. Conversely, Komatsuzaki et al. (2022) directly copy the FFNs to form experts. However, there is no existing work exploring it for decoder-only models. Notably, the Swish-based FFN (Ramachandran et al., 2017; Shazeer, 2020) does not bring natural sparsity as ReLU (Nair and Hinton, 2010) in T5 and BERT. Second, **overcoming the performance decrease entailed by changing the network structure** from dense to sparse remains challenging. Due to the reduction in the amount of activated parameters and the newly introduced gate network for expert routing, we observe a significant performance decline between the LLaMA-MoE models and the original dense LLaMA models.

To solve the above issues, we comprehensively explore four different methods for expert construction. Among them, the non-overlapping randomly splitting method achieves the best performance. Subsequently, we continue training the transformed MoE models and additional gate networks. In this stage, we also carefully study both dynamic and static data sampling strategies for obtaining the fastest convergence and performance improvement. Finally, with a static domain weight proportion corresponding to the activated parameters, the LLaMA-MoE models can quickly converge to a decent level with 200B tokens.

In summary, our contributions are as follows:

- We propose a framework to develop mixture-of-experts from existing decoder-style LLMs by splitting FFNs and continual pre-training, which has never been explored before.
- We comprehensively explore different splitting methods for expert construction. Meanwhile, we comprehensively investigate both dynamic and static data sampling strategy for continual pre-training.
- Our extensive experiments on a variety of tasks validate the effectiveness of our proposed LLaMA-MoE series models. Notably, all our model construction processes and training data are transparent.

2 Related Work

Mixture-of-Experts (MoE). Traditionally, dense models feed all parameters to each input token. In this way, the growing model capacity brings increased computational cost. To alleviate this issue, sparse models attempt to activate a subset of parameters for each input and these activated parameters are referred as experts. In Shazeer et al. (2017), MoE was first proven effective in modern deep learning. This work added an MoE layer which was stacked between LSTM, resulting in state-of-the-art results in language modeling and machine translation benchmarks. Subsequently, the MoE layer is introduced to the transformer architecture as a substitute for the FFN layers. GShard (Lepikhin et al., 2020) applied the MoE to the Transformer and significantly improved machine translation across 100 languages. Switch Transformers (Fedus et al., 2022b) further scales the language model's size to the trillion-level parameter with a simple and effective MoE layer design. Naively trained MoE models is prone to load imbalance, e.g., only a few experts are frequently used while the others are scarcely activated. For optimizing the training, BASE layer (Lewis et al., 2021), HASH layer (Roller et al., 2021), and Expert Choice (Zhou et al., 2022) study how to build MoE models to fully utilize the model capacity. Recently, for model architecture, Xue et al. (2024) explore training a decoder-only MoE with a modified UL2 training objective. Mixtral is another decoder-style MoE model that selects two out of eight experts with token-choice routing (AI, 2023b).

Expert Construction. There are two lines of works constructing MoE from dense checkpoints. The first category splits the parameters of the FFNs and ensures that the total model parameters remain unchanged. MoEBERT (Zuo et al., 2022) propose an importance-based method to adapt the FFNs into experts. Considering that some neurons in the FFNs contribute more to the model performance, they share the most important neurons (i.e., the ones with the highest scores) among the experts, and the other neurons are distributed evenly. MoEfication (Zhang et al., 2021) study the activation patterns of FFNs in Transformer models and find a sparse activation phenomenon. Then, they discover the functional partitions (experts) in FFNs and build routers for selecting experts. It is worth noting that they only focus on the ReLU-based FFNs in T5 (Raffel et al., 2020) and BERT (Devlin et al., 2018). There is another type of work that expands the total model parameters while
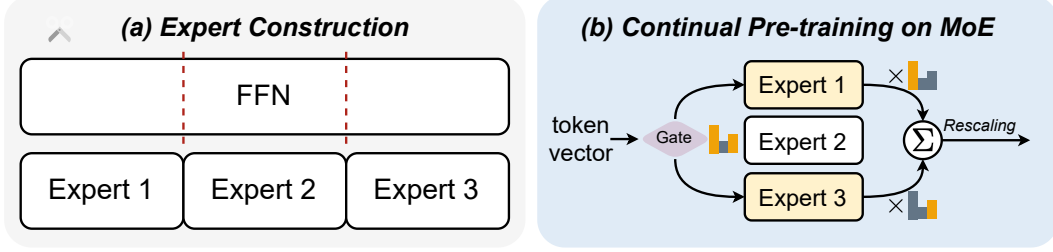
Figure 1: The main framework of building LLaMA-MoE models. (a) The original FFNs in the LLaMA are split into different experts. (b) In the transformed LLaMA-MoE, the hidden states are processed by partially chosen experts instead of all experts. We continue to train the LLaMA-MoE to improve the performance.

keeping the activation parameters as the original dense models. Sparse upcycling (Komatsuzaki et al., 2022) explore upgrading an existing dense model into a larger, sparsely activated MoE. In particular, the experts in the new MoE layer are identical copies of the original MLP layer that is replaced. In this paper, our work follows the first research line and decomposes the original FFNs into multiple small experts. Different from MoEBERT and MoEfication, our work focuses on a SwiGLU-based decoder-style models and continues training the MoE models.

3 Preliminary

A standard Mixture of Experts (MoE) layer comprises N expert networks $\{E_1, E_2, \ldots, E_N\}$ and a gating network G which activates the top-k experts and distributes input tokens to corresponding experts. In general, the number of selected experts k is fixed and much smaller than the total number of experts N, which presents the sparsely activated fashion of MoE models. Formally, given an input embedding x, $E_i(x)$ denotes the output of the i-th expert network, the MoE layer's output is the sum of outputs from k selected experts:

$$y = \sum_{i \in \mathcal{K}} G(x)_i \cdot E_i(x), \tag{1}$$

where the top-k indices are determined by $G(x)$, indicating which experts accept the input x. $\mathcal{K}$ is the set of selected top-k indices. Following Shazeer et al. (2017), we implement a token-level noisy top-k gating with load balancing in LLaMA-MoE.

4 Methodology

As illustrated in Figure 1, we construct LLaMA-MoE from LLaMA-2-7B by first partitioning FFNs into multiple experts and routing each token to top-k experts. Continual pre-training is subsequently applied to recover the MoE model's language ability. In the following sections, we first introduce the expert construction method from the original dense model, then present the data sampling and processing strategies in continual pre-training.

4.1 Expert Construction

In this section, we describe the construction of each expert network in the LLaMA-MoE. We start with the feed-forward network in LLaMA which uses SwiGLU (Shazeer, 2020) as the activation function. Each FFN layer in LLaMA consists of three parts: an up projection weight $W_{\text{up}} \in \mathbb{R}^{d \times d_h}$, a gate projection weight $W_{\text{gate}} \in \mathbb{R}^{d \times d_h}$ and a down projection weight $W_{\text{down}} \in \mathbb{R}^{d_h \times d}$. Given an input $x \in \mathbb{R}^d$, the output $y \in \mathbb{R}^d$ of the FFN is:

$$y = h W_{\text{down}}, \quad h = x W_{\text{up}} \odot \text{Swish}(x W_{\text{gate}}). \tag{2}$$

In LLaMA-MoE, each expert network is implemented as a feed-forward layer. Specifically, given the expert size m and the selection indices set S_j, the weights of the j-th expert network E_j is formulated

as:

$$W_{\text{up}}^{(j)} = W_{\text{up}_{:,S_j}} \in \mathbb{R}^{d \times m}, \quad W_{\text{gate}}^{(j)} = W_{\text{gate}_{:,S_j}} \in \mathbb{R}^{d \times m}, \quad W_{\text{down}}^{(j)} = W_{\text{down}_{S_j,:}} \in \mathbb{R}^{m \times d}, \quad (3)$$

where the selection indices set S_j is:

$$S_j = \{i_{j_1}, i_{j_2}, \ldots, i_{j_m} \mid 1 \le i_{j_1} \ne i_{j_2} \ne \cdots \ne i_{j_m} \le d_h\}. \tag{4}$$

Given an input $x \in \mathbb{R}^d$, the output $E_j(x) \in \mathbb{R}^d$ of the j-th expert network E_j is:

$$E_j(x) = h_j W_{\text{down}}^{(j)}, \quad h_j = x W_{\text{up}}^{(j)} \odot \text{Swish}(x W_{\text{gate}}^{(j)}). \tag{5}$$

Based on whether the intermediate neurons within the FFN are shared among different experts, we implement two groups of construction methods: *Neuron-Independent* and *Neuron-Sharing*.

Neuron-Independent. We formulate expert construction as a task of partitioning into equal-sized sets. Given a universal set U containing indices of all intermediate neurons $\{1, 2, \ldots, d_h\}$, we uniformly split U into n equal-sized indices set $S_1, S_2, \ldots, S_n$ and construct experts with size $m = \frac{d_h}{n}$ according to Equation 3, where we have:

$$\bigcup_{i=1}^{n} S_i = U \quad \text{and} \quad \bigcap_{i=1}^{n} S_i = \varnothing. \tag{6}$$

Specifically, we describe two kinds of partition methods:

- **Independent$_{\text{Random}}$**: We randomly partition U into n equal-sized subsets.
- **Independent$_{\text{Clustering}}$**: Following Zhang et al. (2021), we perform a balanced k-means clustering (Malinen and Fränti, 2014) with n centroids on the row vectors of W_{up} and partition U according to the clustering result.

Neuron-Sharing. According to Zuo et al. (2022), the representation ability of a model can be partially retained through a structured partition. Therefore, we treat the expert construction as a structured pruning problem, by measuring the first-order Taylor expansion on loss change ΔL for each intermediate neuron when it gets pruned. For each FFN layer, we maintain a vector $v \in \mathbb{R}^{d_h}$ initialized as zeros to record the importance of its intermediate neurons. Given batched data D, the importance vector v is updated as follows:

$$v := v + \sum_{(x,y) \in D} \left| h \odot \nabla_h L(x, y) \right|. \tag{7}$$

The indices sets $S_1, S_2, \ldots, S_n$ are then generated using certain algorithm for the experts with sizes $m = \frac{d_h}{n}$. Given the universal indices set $U = \{1, 2, \ldots, d_h\}$, we have:

$$\bigcup_{i=1}^{n} S_i \in U. \tag{8}$$

- **Sharing$_{\text{Inner}}$**: We obtain n importance vectors $v_1, v_2, \ldots, v_n$ through pre-clustered n groups of data. For each expert i, the corresponding S_i consists the indices of neurons with the largest m values in v_i.
- **Sharing$_{\text{Inter}}$**: Referencing the implementation in Rajbhandari et al. (2022), we set aside the neurons shared by most experts as independent residual blocks, while others are assigned according to the importance vectors $v_1, v_2, \ldots, v_n$.

Re-scaling After partitioning a dense FFN layer into multiple small experts, the activated expert parameters are much smaller than the original dense models. To preserve the representational capacity of the partitioned model, we introduce a scale factor and apply rescale operations to guarantee effective expert output. In particular, considering activating k out of N experts, we will scale the output of expert by a factor of $\frac{N}{k}$.

Model	#Activated Experts	#Total Experts	#Activated Params
OPT-2.7B	-	-	2.7B
Pythia-2.8B	-	-	2.8B
INCITE-Base-3B	-	-	2.8B
Open-LLaMA-3B-v2	-	-	3.4B
Sheared-LLaMA-2.7B	-	-	2.7B
LLaMA-MoE-3.0B (2/16)	2	16	3.0B
LLaMA-MoE-3.5B (4/16)	4	16	3.5B
LLaMA-MoE-3.5B (2/8)	2	8	3.5B

Table 1: The statistics for model parameters and activation parameters for sparse MoE models. All LLaMA-MoE-3.0B and LLaMA-MoE-3.5B models have the same parameters as LLaMA-2-7B. LLaMA-MoE-3.5B has two variants including 2/8 and 4/16. They have different numbers of experts but the same amount of activation parameters.

4.2 Continual Pre-training

Since the original LLaMA model structure is reorganized after converting to MoE, we continue pre-training the LLaMA-MoE model in Table 1 to recover their language modeling abilities. The training objective is the same as LLaMA-2 (Touvron et al., 2023a). To improve the training efficiency, we explore different data sampling strategies and data quality filtering methods as follows.

Data Sampling Weights. The data sampling weights are crucial to obtain a global optimum (Xie et al., 2023). LLaMA-v1 utilizes a set of static empirical sampling weights (Touvron et al., 2023a), while some of the domains (e.g. Wikipedia) have been proven to be less effective on downstream tasks (Shen et al., 2023b). This indicates it may be not appropriate to assign large weights when sampling these domains. Xie et al. (2023) employ additional models to obtain better static sampling weights. Although it is faster to get convergence, it brings additional training compute. Xia et al. (2023) introduce a dynamic weight sampling strategy in the training phase, which boosts performance on downstream tasks.

To obtain better performances, we investigate the following data sampling strategies for LLaMA-MoE continual pre-training. Data sampling weights are adjusted every 2.5B tokens in dynamic settings and the total training budget is 30B tokens.

- **Static$_{LLaMA}$**: Training with static LLaMA-1 sampling weights.
- **Static$_{Sheared}$**: Applying final static sampling weights of Sheared-LLaMA.
- **Dynamic$_{LLaMA}$**: Sheared-LLaMA dynamic sampling with LLaMA-v1 weights construction. We evaluate LLaMA-v2 on a subset of SlimPajama with all the training domains for obtaining the reference loss.
- **Dynamic$_{Uniform}$**: Sheared-LLaMA dynamic sampling with uniform weights construction.

Data Filtering. As our training budget is limited, we further explore two data filter strategies to speed up model convergence. Specifically, we filter out ∼50% advertisements and ∼15% non-fluent texts in CommonCrawl and C4 datasets.

5 Experiments

5.1 Training Dataset

The training dataset for LLaMA-MoE is SlimPajama (Soboleva et al., 2023), which cleans and deduplicates the RedPajama dataset. This dataset contains 627B tokens and encompasses data from seven domains, including CommonCrawl, C4, Github, Wikipedia, Books, arXiv, and StackExchange.

	Commonsense & Reading Comprehension					
Model	**SciQ**	**PIQA**	**WinoGrande**	**ARC-E**	**ARC-C (25)**	**HellaSwag (10)**
OPT-2.7B	78.9	74.8	60.8	54.4	34.0	61.4
Pythia-2.8B	83.2	73.6	59.6	58.8	36.7	60.7
INCITE-Base-3B	85.6	73.9	63.5	61.7	40.3	64.7
Open-LLaMA-3B-v2	88.0	**77.9**	63.1	63.3	40.1	71.4
Sheared-LLaMA-2.7B	87.5	76.9	65.0	63.3	41.6	71.0
LLaMA-MoE-3.0B	84.2	77.5	63.6	60.2	40.9	70.8
LLaMA-MoE-3.5B (4/16)	87.6	**77.9**	65.5	**65.6**	**44.2**	**73.3**
LLaMA-MoE-3.5B (2/8)	**88.4**	77.6	**66.7**	65.3	43.1	**73.3**

	Continued		**LM**	**World Knowledge**		
Model	**LogiQA**	**BoolQ (32)**	**LAMBADA**	**NQ (32)**	**MMLU (5)**	**Average**
OPT-2.7B	25.8	63.3	63.6	10.7	25.8	50.3
Pythia-2.8B	28.1	65.9	64.6	8.7	26.8	51.5
INCITE-Base-3B	27.5	65.8	65.4	15.2	27.2	53.7
Open-LLaMA-3B-v2	28.1	69.2	67.4	16.0	26.8	55.6
Sheared-LLaMA-2.7B	28.3	73.6	68.3	17.6	**27.3**	56.4
LLaMA-MoE-3.0B	**30.6**	71.9	66.6	17.0	26.8	55.5
LLaMA-MoE-3.5B (4/16)	29.7	**75.0**	**69.5**	**20.3**	26.8	**57.7**
LLaMA-MoE-3.5B (2/8)	29.6	73.9	69.4	19.8	27.0	57.6

Table 2: Main results on downstream tasks. We re-evaluate all the models on these datasets. LLaMA-MoE-3.5B significantly outperforms publicly available models of comparable size on most downstream tasks. The shot number used is noted in parentheses, with 0-shot if not specified.

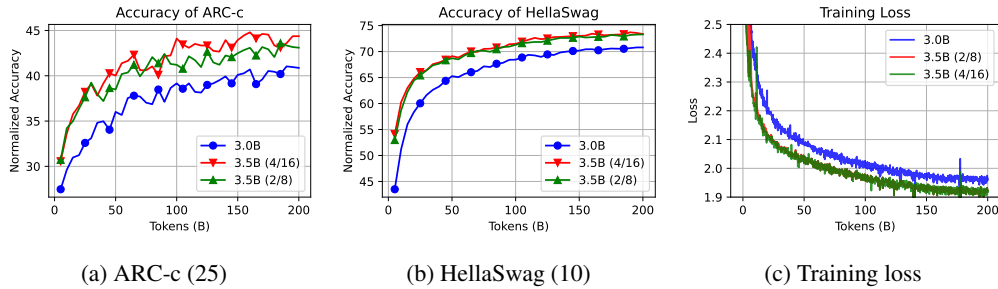(a) ARC-c (25)   (b) HellaSwag (10)   (c) Training loss

Figure 2: Model performances on ARC-c and HellaSwag dataset and the training loss for LLaMA-MoE-3.0B and LLaMA-MoE-3.5B. The two models are trained with 200B tokens.

5.2 Evaluation Datasets and Comparing Models

According to Wei et al. (2023) and AI (2023a), the performance on HellaSwag (Zellers et al., 2019) grows smoothly during pre-training. A similar trend is also found in ARC-c (Clark et al., 2018a), thus we utilize HellaSwag and ARC-c as the evaluation datasets for the analysis experiments.

For comprehensive model ability assessment, we follow Xia et al. (2023) and use the lm-evaluation-harness (Gao et al., 2023) to evaluate the following downstream tasks: 0-shot normalized accuracy (acc_norm) of ARC easy (Clark et al., 2018b), LAMBADA (Paperno et al., 2016), LogiQA (Liu et al., 2020), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), and WinoGrande Standard (Sakaguchi et al., 2021), 10-shot HellaSwag (Zellers et al., 2019), 25-shot ARC Challenge (Clark et al., 2018b), and 5-shot MMLU (Hendrycks et al., 2020). If there is no normalized accuracy, we use acc instead. Furthermore, we use OpenCompass (Contributors, 2023) to evaluate 32-shot NQ (Kwiatkowski et al., 2019). We compare LLaMA-MoE with strong pre-trained language models containing similar activation parameters, including OpenLLaMA-3B-v2 (Geng and Liu, 2023), OPT-2.7B (Zhang et al., 2022), Pythia-2.8B (Biderman et al., 2023), INCITE-Base-3B (TogetherAI, 2023), and Sheared-LLaMA (Xia et al., 2023).
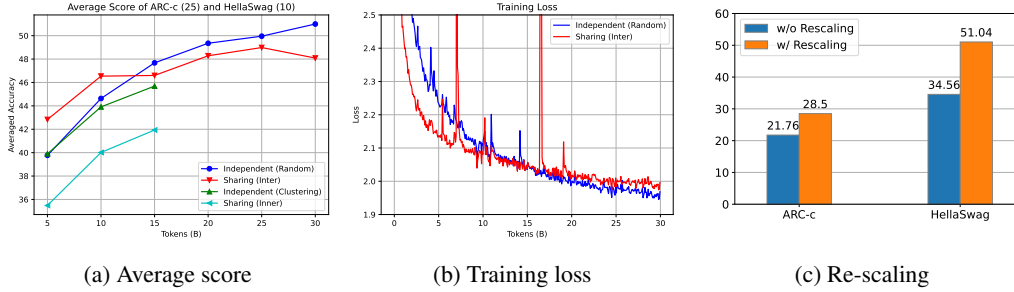
| (a) Average score | (b) Training loss | (c) Re-scaling |

Figure 3: Model performances with different expert construction methods. Among four kinds of construction methods, **Independent**$_{Random}$ obtains the best result. We also present the ablation study of expert output re-scaling after 5B tokens of continual pre-training.

5.3 Experiment Settings

We start from LLaMA-2-7B (Touvron et al., 2023b) and explore different MoE construction strategies. All models are trained on 112 A100 (80G) GPUs with a global batch size of 15M tokens. The context length is 4096. The maximum learning rate is 2e-4 with 100 warmup steps and the final learning rate decays to 2e-5 with cosine scheduling. Each LLaMA-MoE variant is expected to be trained on 200B tokens (13.6k steps). Our implementation is based on transformers (Wolf et al., 2020), ZeRO-1 (Rajbhandari et al., 2022), and FlashAttention v2 (Dao, 2023). The final LLaMA-MoE models are trained on **Independent**$_{Random}$ with **Static**$_{Sheared}$ data sampling weights and fluency-filtered SlimPajama datasets. More details can be found in our released code.

5.4 Main Results

As shown in Table 2, LLaMA-MoE-3.5B (2/8) and LLaMA-MoE-3.5B (4/16) achieve similar average results and the latter is slightly better. However, LLaMA-MoE-3.5B significantly surpasses open-source models with similar activation parameters. Specifically, LLaMA-MoE-3.5B (4/16) exceeds the most competitive model Sheared-LLaMA by 1.3 average points. Meanwhile, LLaMA-MoE-3.0B performs comparably with Open-LLaMA-3B-v2. To demonstrate the training progress and model capability changes. In Figure 2a and 2b, we present the model performances on both ARC-c and HellaSwag and find the results on these datasets grow gradually as the training process goes on. There are more fluctuations in ARC-c results, while HellaSwag provides smoother results. For the training loss, as shown in Figure 2c, LLaMA-MoE-3.0B and LLaMA-MoE-3.5B converges to about 1.95 and 1.90, respectively. The final loss are higher than LLaMA-2 7B as these two models activate relatively fewer parameters.

5.5 Expert Construction

In this section, we compare four types of expert construction methods as introduced in § 4.1. Interestingly, as presented in Figure 3, **Independent**$_{Random}$ achieves the best average score within the token budget. Since gates and experts are trained simultaneously, other partition methods may bring bias when construction, which introduces additional difficulties for recovering the model's language ability by continual pre-training. **Sharing**$_{Inter}$ has a good convergence trend at the first 5B tokens, but it struggles to get better loss performance. Actually, we have trained more tokens for **Sharing**$_{Inter}$ and the results are significantly lower than **Independent**$_{Random}$. We can also observe changes in the loss value as depicted in Figure 3b. However, We found that models should be trained for at least 15∼20B tokens to properly conclude. For **Sharing**$_{Inner}$, an average of ∼ 50% neurons are shared between each expert pair, thus the upper bound for this variant is quite low and the model would like to achieve lower performance. Finally, as **Independent**$_{Clustering}$ and **Sharing**$_{Inner}$ achieves much lower performances than other methods, so we only train those models for 15B tokens.

Besides the specific expert construction strategies, we conduct the ablation study on the re-scaling operation on the expert outputs. As shown in Figure 3c, we find that the re-scaling on the expert outputs provides significantly better performance for MoE models.
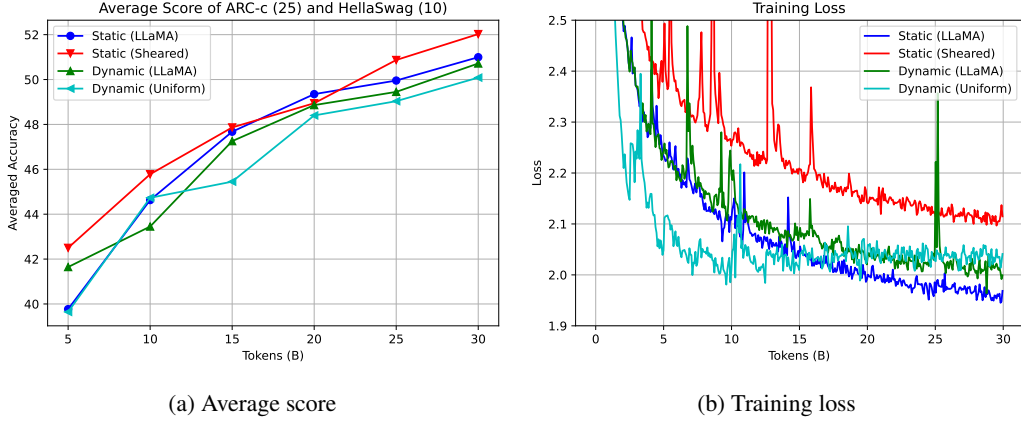
(a) Average score   (b) Training loss

Figure 4: Model performances with different data sampling strategies. Among four sampling ways, Static$_{Sheared}$ achieves the best performance. However, it does not achieve the lowest training loss.
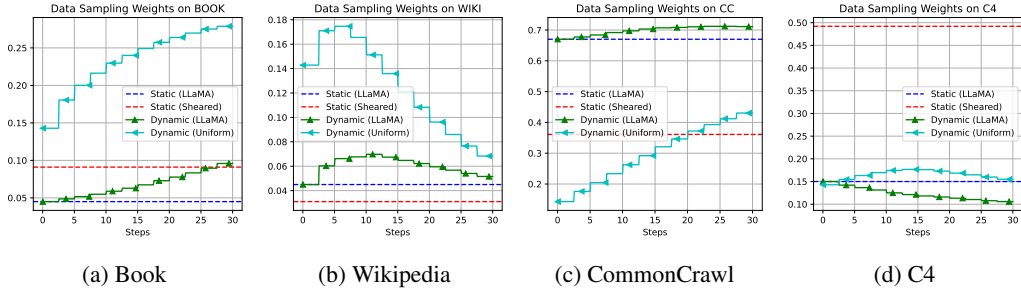

(a) Book   (b) Wikipedia   (c) CommonCrawl   (d) C4

Figure 5: Data sampling weights variation on four domains. For Static$_{Sheared}$ and Static$_{LLaMA}$, the sampling weight is fixed among the training process, while the domain importance gradually changes for Dynamic$_{Uniform}$ and Dynamic$_{LLaMA}$. Both Dynamic$_{Uniform}$ and Dynamic$_{LLaMA}$ are two dynamic weight sampling strategies from (Xia et al., 2023).

5.6 Data Sampling Weights

As Figure 4a shows, Static$_{Sheared}$ surpasses other methods within the token budget, and dynamic data sampling weights are worse than static weights. However, the Static$_{Sheared}$ loss in Figure 4b is greater than other methods, which indicates that the continual pre-training loss may be less relevant to downstream task performances. The loss of Dynamic$_{Uniform}$ drops down quickly, but it suffers from the instability problem and contains many fluctuations. From Figure 5, we find the sampling weight of C4 goes to the opposite directions compared to Static$_{Sheared}$ because the estimated Sheared LLaMA-2.7B reference loss is lower than LLaMA2-7B (2.033 vs. 2.075). It is very tricky to select the best reference loss, we leave it for future work.

5.7 Data Filtering

We further filter out data with advertisements and those in low fluency. The results are presented in Figure 6. Both fluency and advertisement filtering obtain lower training loss than the baseline. However, advertisement filtering perform worse in downstream tasks. We think the number of filtered advertisements are too large to bring more knowledge and information, and the filtering tagger should be improved with fine-grained thresholding adjustment. The fluency filtering method successfully removes texts in low-quality and improves the average score. Based on the results, we train our final model with the fluency-filtered dataset. It is worth noting that we do not introduce any new datasets but remove part of them considering model convergence speed.
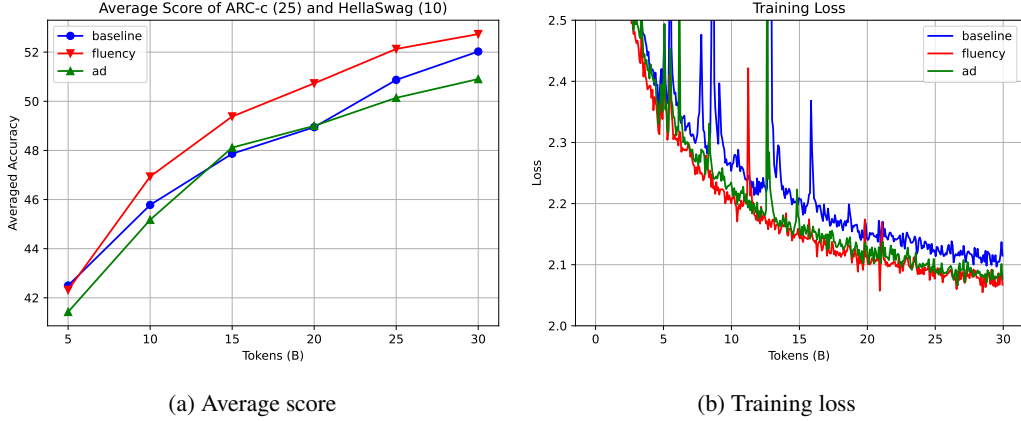
(a) Average score

(b) Training loss

Figure 6: Model performances with different data filtering strategies. Here, the "baseline" is **Independent**$_{Random}$ together with **Static**$_{Sheared}$ data sampling weights. "fluency" and "ad" means the baseline strategy is equipped with removing non-fluent texts or ads.

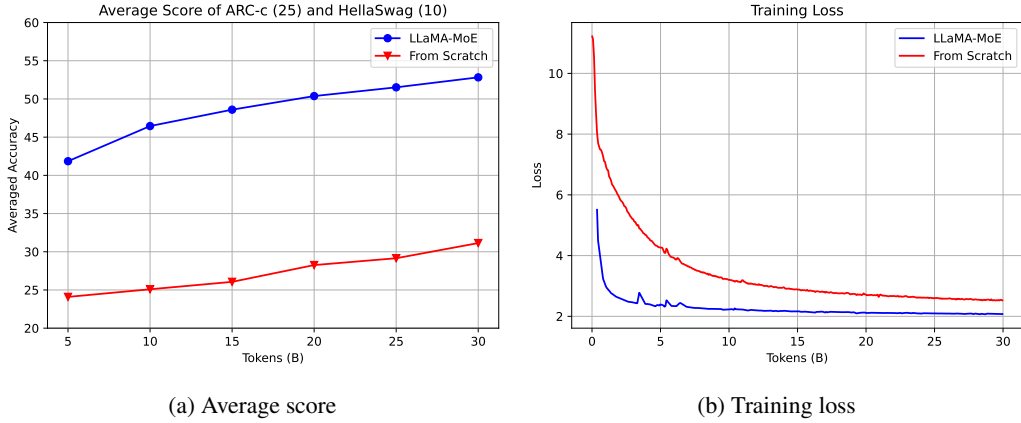(a) Average score

(b) Training loss

Figure 7: Model performances on LLaMA-MoE-3.5B (2/8) and MoE training from scratch. For comparing the initial losses, we set the y-axis limit to a larger scale.

5.8 LLaMA-MoE vs. Training from Scratch

To validate the effectiveness of our proposed LLaMA-MoE, we conduct experiments on training the MoE model from scratch. By comparing the training losses in Figure 7b and performances in Figure 7a, we find that constructing MoE models from dense models has significant advantages and could reduce huge computing resources.

5.9 Expert Specialization

As Figure 8 shows, deep layers have more routing preferences than shallow layers. This may indicate that the shallow layers may capture more common features, while deep layers focus more on task-specific features. Based on this finding, expert partition on the latter layers' FFNs may bring further improvements. We leave it for future exploration. In deeper layers, each expert has different domain preferences and some experts are shared across different domains. These shared experts may represent data similarities among different domains. We also find the imbalance problem at the first two layers, where some experts are seldom selected. These experts may be pruned for future MoE model compression.

To investigate the latent correlations among domains, we normalize the number of routed tokens and calculate the L2 distances to represent the expert selection differences. As illustrated in Figure 9a, CommonCrwal and C4 datasets have similar expert preferences, while GitHub has similar expert
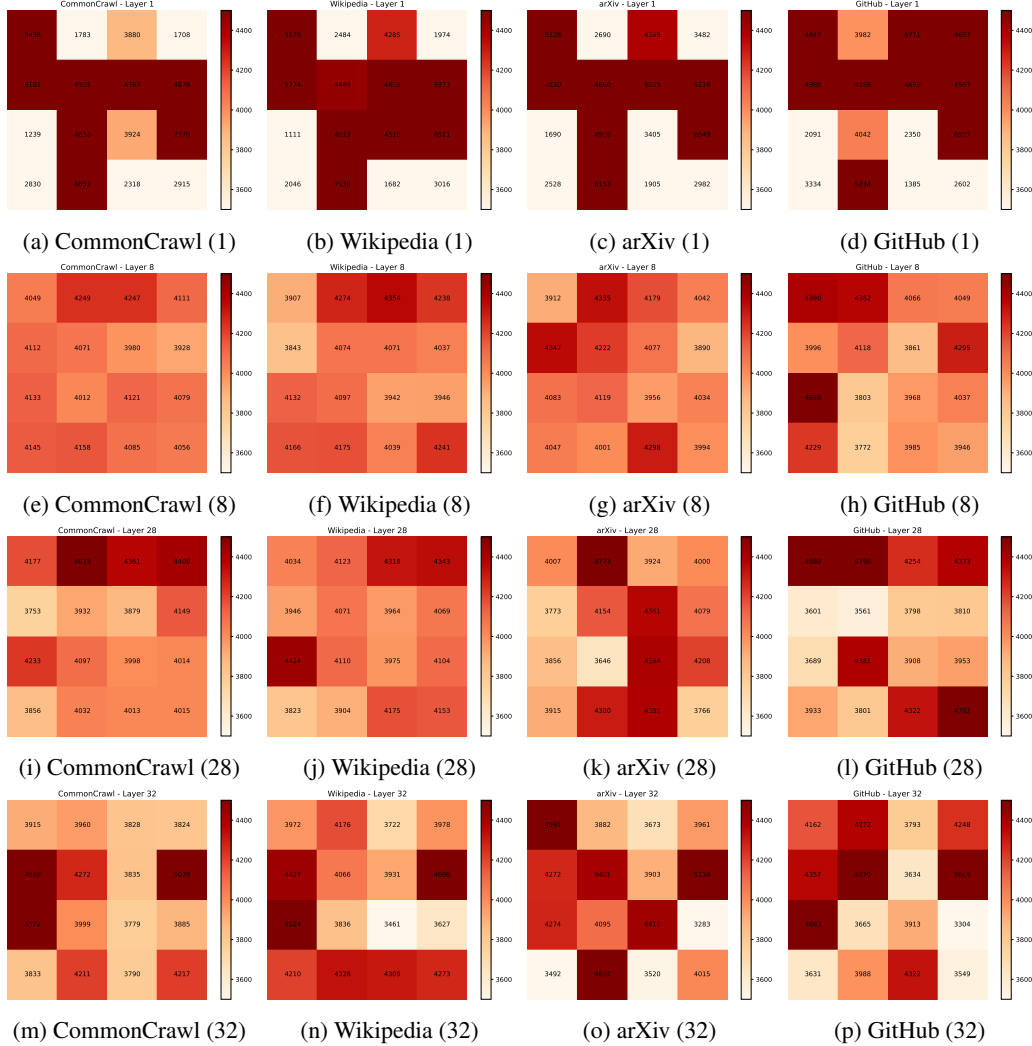
Figure 8: Expert routing statistics on the 1st, 8th, 28th, and 32nd layers for LLaMA-MoE-3.5B (4/16). Each cell represents the number of routed tokens to an expert. Our model has a total of 16 experts. We sample 65.5K tokens from each domain for this visualization.

preferences with arXiv and StackExchange. As to the Dev-to-Train differences in Figure 9b, we find HellaSwag and ARC-c share the most similar expert preferences with CommonCrawl and C4, and GSM-8K is similar to arXiv. This may provide some insights for continual pre-training to further improve downstream performances. For example, the model may consume more tokens from arXiv to improve GSM-8K results. However, expert selections on ARC-c and GSM-8K have greater distances with current pre-training data, which may involve new domains to deal with such tasks.

5.10 Instruction Tuning

MoE models have shown great prowess in instruction tuning with excellent task scaling properties (Shen et al., 2023a; Chen et al., 2024; Zhu et al., 2024). To evaluate the instructed MoE models' performances, we fine-tune LLaMA-MoE with 6k curated ShareGPT instruction data (Liu et al., 2023) for 2 epochs. As shown in Table 3, the instructed LLaMA-MoE-3.5B (4/16) outperforms the dense model on ARC-C (48.29 vs. 43.69) and HellaSwag (75.10 vs. 71.70) tasks. The overall performance on Open LLM Leaderboard [2] tasks surpasses the dense model (48.95 vs. 47.41). Besides, there is a

---

[2] https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard
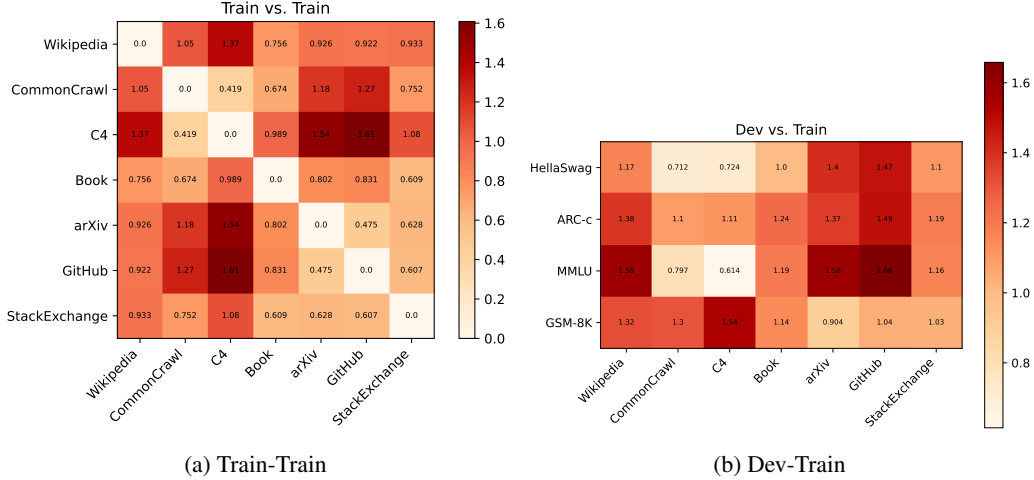
| (a) Train-Train | (b) Dev-Train |

Figure 9: Expert routing differences at the 32nd layer. Smaller numbers and lighter colors represent more similar expert routing preferences. 8.4M tokens per domain are sampled for this experiment.

Model	Open LLM Leaderboard					Alignment
	MMLU	ARC-C	HellaSwag	TruthfulQA	Average	MT-Bench
Sheared-LLaMA-2.7B-ShareGPT	**28.41**	41.04	71.21	47.65	47.08	3.79
Sheared-LLaMA-2.7B (Our Dataset)	25.24	43.69	71.70	**49.00**	47.41	4.06
LLaMA-MoE-v1-3.0B (2/16)	23.61	43.43	72.28	44.24	45.89	4.15
LLaMA-MoE-v1-3.5B (4/16)	26.49	**48.29**	**75.10**	45.91	**48.95**	4.60
LLaMA-MoE-v1-3.5B (2/8)	25.53	45.99	74.95	44.39	47.71	**4.72**

Table 3: Supervised fine-tuned model performances on Open LLM Leaderboard tasks and open-ended questions. Sheared LLaMA-2.7B-ShareGPT is a chat model created by Xia et al. (2023). We reimplement the chat model by instruction tuning on our dataset and provide fair comparisons.

large gap in alignment abilities, where LLaMA-MoE-3.5B (2/8) outperforms Sheared LLaMA-2.7B by 0.66 scores on MT-Bench.

6 Conclusion

In this technical report, we build LLaMA-MoE-3.0B and LLaMA-MoE-3.5B models based on pre-trained LLaMA 2. Specifically, we explore different expert construction methods and continual training strategies to obtain decent models under limited training budgets. Empirically, LLaMA-MoE-3.5B significantly outperforms open-source models with similar activation parameters, such as Sheared-LLaMA-2.7B and Open-LLaMA-3.0B. Meanwhile, LLaMA-MoE-3.0B achieves similar performance with Open-LLaMA-3B with less activated parameters.

From the ablation studies, we find the optimized static data sampling weights could achieve better results, and further data filtering on low-fluency texts also brings extra performance gain. LLaMA-MoE models also show the expert specialization phenomenon, where each expert has domain preferences. Based on this preference, we explore the expert selection similarities across pre-training datasets and downstream task datasets.

References

DeepSeek AI. 2023a. Deepseek llm: Let there be answers. https://github.com/deepseek-ai/DeepSeek-LLM.

Mistral AI. 2023b. Mixtral of experts: A high quality sparse mixture-of-experts.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff,

et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, Aijia Guo, Qipeng Guo, Conghui He, Yingfan Hu, Ting Huang, Tao Jiang, Penglong Jiao, Zhenjiang Jin, Zhikai Lei, Jiaxing Li, Jingwen Li, Linyang Li, Shuaibin Li, Wei Li, Yining Li, Hongwei Liu, Jiangning Liu, Jiawei Hong, Kaiwen Liu, Kuikun Liu, Xiaoran Liu, Chengqi Lv, Haijun Lv, Kai Lv, Li Ma, Runyuan Ma, Zerun Ma, Wenchang Ning, Linke Ouyang, Jiantao Qiu, Yuan Qu, Fukai Shang, Yunfan Shao, Demin Song, Zifan Song, Zhihao Sui, Peng Sun, Yu Sun, Huanze Tang, Bin Wang, Guoteng Wang, Jiaqi Wang, Jiayu Wang, Rui Wang, Yudong Wang, Ziyi Wang, Xingjian Wei, Qizhen Weng, Fan Wu, Yingtong Xiong, Chao Xu, Ruiliang Xu, Hang Yan, Yirong Yan, Xiaogui Yang, Haochen Ye, Huaiyuan Ying, Jia Yu, Jing Yu, Yuhang Zang, Chuyu Zhang, Li Zhang, Pan Zhang, Peng Zhang, Ruijie Zhang, Shuo Zhang, Songyang Zhang, Wenjian Zhang, Wenwei Zhang, Xingcheng Zhang, Xinyue Zhang, Hui Zhao, Qian Zhao, Xiaomeng Zhao, Fengzhe Zhou, Zaida Zhou, Jingming Zhuo, Yicheng Zou, Xipeng Qiu, Yu Qiao, and Dahua Lin. 2024. Internlm2 technical report.

Guanjie Chen, Xinyu Zhao, Tianlong Chen, and Yu Cheng. 2024. $\texttt{MoE-RBench}$: Towards building reliable language models with sparse mixture-of-experts. In *Forty-first International Conference on Machine Learning*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018a. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018b. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Tri Dao. 2023. FlashAttention-2: Faster attention with better parallelism and work partitioning.

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

William Fedus, Jeff Dean, and Barret Zoph. 2022a. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Xinyang Geng and Hao Liu. 2023. Openllama: An open reproduction of llama.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. 2022. Sparse upcycling: Training mixture-of-experts from dense checkpoints. *arXiv preprint arXiv:2212.05055*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

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

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

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2023. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *ArXiv*, abs/2312.15685.

Zhenyi Lu, Jie Tian, Wei Wei, Xiaoye Qu, Yu Cheng, Dangyang Chen, et al. 2024. Mitigating boundary ambiguity and inherent bias for text classification in the era of large language models. *arXiv preprint arXiv:2406.07001*.

Mikko I. Malinen and Pasi Fränti. 2014. Balanced k-means for clustering. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 32–41, Berlin, Heidelberg. Springer Berlin Heidelberg.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

OpenAI. 2023. Openai: Introducing chatgpt.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

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

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. 2022. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International Conference on Machine Learning*, pages 18332–18346. PMLR.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. 2017. Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. 2021. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 34:17555–17566.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

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

Sheng Shen, Le Hou, Yan-Quan Zhou, Nan Du, S. Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, Tu Vu, Yuexin Wu, Wuyang Chen, Albert Webson, Yunxuan Li, Vincent Zhao, Hongkun Yu, Kurt Keutzer, Trevor Darrell, and Denny Zhou. 2023a. Mixture-of-experts meets instruction tuning:a winning combination for large language models.

Zhiqiang Shen, Tianhua Tao, Liqun Ma, Willie Neiswanger, Zhengzhong Liu, Hongyi Wang, Bowen Tan, Joel Hestness, Natalia Vassilieva, Daria Soboleva, and Eric Xing. 2023b. Slimpajama-dc: Understanding data combinations for llm training.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama.

Zhaochen Su, Juntao Li, Jun Zhang, Tong Zhu, Xiaoye Qu, Pan Zhou, Yan Bowen, Yu Cheng, et al. 2024a. Living in the moment: Can large language models grasp co-temporal reasoning? *arXiv preprint arXiv:2406.09072*.

Zhaochen Su, Jun Zhang, Tong Zhu, Xiaoye Qu, Juntao Li, Min Zhang, and Yu Cheng. 2024b. Timo: Towards better temporal reasoning for language models.

Wei Tao, Yucheng Zhou, Wenqiang Zhang, and Yu Cheng. 2024. Magis: Llm-based multi-agent framework for github issue resolution. *arXiv preprint arXiv:2403.17927*.

TogetherAI. 2023. Redpajama: an open dataset for training large language models.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Tianwen Wei, Liang Zhao, Lichang Zhang, Bo Zhu, Lijie Wang, Haihua Yang, Biye Li, Cheng Cheng, Weiwei Lü, Rui Hu, Chenxia Li, Liu Yang, Xilin Luo, Xuejie Wu, Lunan Liu, Wenjun Cheng, Peng Cheng, Jianhao Zhang, Xiaoyu Zhang, Lei Lin, Xiaokun Wang, Yutuan Ma, Chuanhai Dong, Yanqi Sun, Yifu Chen, Yongyi Peng, Xiaojuan Liang, Shuicheng Yan, Han Fang, and Yahui Zhou. 2023. Skywork: A more open bilingual foundation model.

Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. 2023. Doremi: Optimizing data mixtures speeds up language model pretraining.

Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. 2024. Openmoe: An early effort on open mixture-of-experts language models. *arXiv preprint arXiv:2402.01739*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2021. Moefication: Transformer feed-forward layers are mixtures of experts. *arXiv preprint arXiv:2110.01786*.

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

Tong Zhu, Daize Dong, Xiaoye Qu, Jiacheng Ruan, Wenliang Chen, and Yu Cheng. 2024. Dynamic data mixing maximizes instruction tuning for mixture-of-experts. *arXiv preprint arXiv:2406.11256*.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*.

Simiao Zuo, Qingru Zhang, Chen Liang, Pengcheng He, Tuo Zhao, and Weizhu Chen. 2022. Moebert: from bert to mixture-of-experts via importance-guided adaptation. *arXiv preprint arXiv:2204.07675*.