Less, but Better: Efficient Multilingual Expansion for LLMs via Layer-wise Mixture-of-Experts

Xue Zhang[1,2*], **Yunlong Liang**[3], **Fandong Meng**[3], **Songming Zhang**[1,2], **Yufeng Chen**[1,2†], **Jinan Xu**[1,2], **Jie Zhou**[3]

[1]Key Laboratory of Big Data & Artificial Intelligence in Transportation, (Beijing Jiaotong University), Ministry of Education

[2]School of Computer Science and Technology, Beijing Jiaotong University, Beijing, China

[3]Pattern Recognition Center, WeChat AI, Tencent Inc, China

{zhang_xue,smzhang22,chenyf,jaxu}@bjtu.edu.cn

Abstract

Continually expanding new languages for existing large language models (LLMs) is a promising yet challenging approach to building powerful multilingual LLMs. The biggest challenge is to make the model continuously learn new languages while preserving the proficient ability of old languages. To achieve this, recent work utilizes the Mixture-of-Experts (MoE) architecture to expand new languages by adding new experts and avoid catastrophic forgetting of old languages by routing corresponding tokens to the original model backbone (old experts). Although intuitive, this kind of method is parameter-costly when expanding new languages and still inevitably impacts the performance of old languages. To address these limitations, we analyze the language characteristics of different layers in LLMs and propose a layer-wise expert allocation algorithm (**LayerMoE**) to determine the appropriate number of new experts for each layer. Specifically, we find different layers in LLMs exhibit different representation similarities between languages and then utilize the similarity as the indicator to allocate experts for each layer, *i.e.*, the higher similarity, the fewer experts. Additionally, to further mitigate the forgetting of old languages, we add a classifier in front of the router network on the layers with higher similarity to guide the routing of old language tokens. Experimental results show that our method outperforms the previous state-of-the-art baseline with 60% fewer experts in the single-expansion setting and with 33.3% fewer experts in the lifelong-expansion setting, demonstrating the effectiveness of our method.

1 Introduction

Although existing large language models (LLMs) have exhibited remarkable ability in high-resource languages, their performance in multilingual scenarios is still limited (Lai et al., 2023b; Zhang et al., 2024c; Huang et al., 2025). To enhance the multilingual ability of LLMs, continually expanding new languages for existing LLMs is a sustainable approach, which does not require excessive computation or data resources than re-training from scratch (Huang et al., 2024). During this process, the key challenge is updating the model with new language data while avoiding catastrophic forgetting of the ability in originally proficient languages.

As a promising solution, MoE-LPR (Zhou et al., 2024), a two-stage training strategy, utilizes the Mixture-of-Experts (MoE) architecture to achieve expansion of new languages and seeks to preserve the ability of the original languages by targeted routing. Specifically, at the first stage, it adds and trains several new experts beyond the dense LLM to learn new languages. Then, at the second stage, it tries to recover the ability of old languages by language priors routing training. Although this method shows the potential of MoE on balancing the capability of newly expanded languages and the original languages, there are still two problems. For one thing, the expansion of MoE-LPR is inefficient and leads to high resource usage after expansion (*e.g.*, the original Qwen1.5-1.8B has to be 3.2× larger to accommodate three additional languages). For another, although MoE-LPR designs the language priors routing to recover the abilities of the original languages, the corresponding performance of old languages still declines noticeably.

Towards these limitations, we first analyze the similarity between hidden states in different languages and find that some layers exhibit high similarity. On this basis, we propose a layer-wise expert allocation algorithm (**LayerMoE**) to determine the appropriate number of new experts for each layer, with the insight that higher representation similarity requires fewer experts. Additionally, we speculate that high similarity between old and new languages

[*] This work was done during the internship at Pattern Recognition Center, WeChat AI, Tencent Inc, China.

[†] Yufeng Chen is the corresponding author.

in some layers may confuse the router network, causing the forgetting of old languages. To further mitigate the forgetting, we add a classifier in front of the router network on the layers exhibiting higher similarity to guide the routing of tokens in old languages.

We conduct experiments under two settings: (1) single-expansion, where the model is enhanced once by incorporating a new language group, and (2) lifelong-expansion, where the model undergoes sequential adaptation to multiple new language groups. The corresponding experimental results on four benchmarks demonstrate the effectiveness of our LayerMoE. Specifically, in the single-expansion setting (please refer to Table 1), our method achieves superior performance with 60% fewer experts compared to the previous state-of-the-art method, *i.e.*, better expansion of new languages, and less forgetting of old languages. In the lifelong-expansion setting (please refer to Table 2), our method also outperforms baselines, requiring 33.3% fewer experts. These results indicate that our expert allocation algorithm offers a resource-efficient solution for continuous language learning, and the classifier added in front of the router network further prevents the forgetting of old languages.

In summary, the major contributions of this paper are as follows[1]:

- We devise a layer-wise expert allocation algorithm to assign the appropriate number of new experts for each layer, which is parameter-efficient and achieves better expansion performance of new languages.
- We add a classifier in front of the router network on the layers that exhibit higher similarity, further mitigating the forgetting of old languages.
- Experimental results in the single-expansion and lifelong-expansion settings demonstrate the effectiveness and the efficiency of our LayerMoE.

2 Background

In this section, we first introduce the two-stage expansion method MoE-LPR (Zhou et al., 2024) and discuss its limitations.

[1]The code is publicly available at `https://github.com/XZhang00/LayerMoE`.

Stage-1: Continual pre-training with MoE. The dense model is upcycled to an MoE model by adding new feed-forward networks (FFNs) and router networks. Formally, the MoE model includes N experts and a linear router network $W_r \in \mathbb{R}^{h \times N}$ for each layer, where h is the dimension of hidden states after the Attention module. During the forward process, the top-K experts $\mathcal{I}$ are dynamically selected by the router according to the router scores:

$$G(\mathbf{x}) = \text{Softmax}(\mathbf{x} \cdot W_r), \tag{1}$$

$$\mathcal{I} = \{i \mid G_i(\mathbf{x}) \in \text{TopK}(G(\mathbf{x}), K)\}, \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^h$ is the input token hidden state vector of the router network, $G(\mathbf{x}) \in \mathbb{R}^N$ represents the router scores of each expert, and K is the number of the activated experts ($K = 2$). Then we normalize the weights of the selected experts and weighted sum their outputs to obtain the final representation:

$$\mathbf{y} = \sum_{i \in \mathcal{I}} \frac{G_i(\mathbf{x})}{\Sigma_{j \in \mathcal{I}} G_j(\mathbf{x})} E_i(\mathbf{x}) + \mathbf{x}, \tag{3}$$

where $G_i(\mathbf{x})$ and $E_i(\mathbf{x})$ denote the router score and the output of the *i*-th expert respectively.

In stage-1, the parameters of the original dense model are frozen to avoid catastrophic forgetting and the newly added experts and routers are updated with the data of expanded languages. The training objective includes the next token prediction loss $\mathcal{L}_{NTP}$ and the load balance loss $\mathcal{L}_{balance}$ (Fedus et al., 2022). Given the MoE model $\mathcal{M}$ with the parameters of the newly added experts θ_e and the routers θ_r, the $\mathcal{L}_{NTP}$ of one input sequence $\mathbf{t}$ is $\mathcal{L}_{NTP}(\theta_e, \theta_r) = -\sum_i^{|\mathbf{t}|} \log p_{\mathcal{M}}(t_i \mid \mathbf{t}_{<i})$. The $\mathcal{L}_{balance}$ is an expert-level load balance loss (please refer to Appendix A for the detailed introduction) to mitigate the risk of routing collapse. The final training loss of stage-1 is:

$$\mathcal{L}_{stage1}(\theta_e, \theta_r) = \mathcal{L}_{NTP} + \alpha \mathcal{L}_{balance}, \tag{4}$$

where α is a hyper-parameter that controls the weight of the load balance loss ($\alpha = 0.01$).

Stage-2: Review with LPR. In this stage, only the routers are updated with the mixed data of the old languages and the new languages. The language priors routing (LPR) loss is designed to route the old language tokens to the original frozen experts (*i.e.*, the expert 0 in each layer). Formally, given a batch tokens $\mathbb{T}$ including the old language tokens set $\mathbb{T}_{old}$ and the indicator function $F(t)$, the LPR loss $\mathcal{L}_{LPR}$ is defined as:

$$\mathcal{L}_{LPR}(\theta_r) = -\sum_{t \in \mathbb{T}} F(t) \log G_0(t), \tag{5}$$

$$F(t) = \begin{cases} 1, & \text{if } t \in \mathbb{T}_{old}, \\ 0, & \text{if } t \notin \mathbb{T}_{old}, \end{cases} \tag{6}$$

where $G_0(t)$ is the routing score of the expert 0. The final training loss of stage-2 is:

$$\mathcal{L}_{stage2}(\theta_r) = \mathcal{L}_{NTP} + \beta \mathcal{L}_{LPR}, \tag{7}$$

where β is a hyper-parameter that controls the weight of the LPR loss ($\beta = 0.1$).

Discussions about MoE-LPR. During stage-1, MoE-LPR directly adds uniformly distributed experts for each layer, *i.e.*, each layer adds the same number of new experts. This expansion paradigm may lead to inefficient expansion and high resource usage, especially in lifelong learning settings. Inspired by the findings on language-specific and language-agnostic characteristics of different layers (Chen et al., 2023; Zhang et al., 2024b), we raise a question: do all layers need the same number of new experts? One intuition is that decreasing the number of experts in some layers may not hurt the overall performance, which motivates us to design an expert allocation algorithm. During stage-2, although the LPR loss is used to guide the routing of old language tokens, the performance of the old languages still declines noticeably. This suggests that the routing rules in some layers may be difficult for the router network to learn.

3 Methodology

To address the two limitations of MoE-LPR, we first analyze the similarity of hidden states between different languages across different layers. Then we introduce our layer-wise expert allocation algorithm based on the similarity. Lastly, we introduce the classifier ahead of the router network in the layers with high similarity to control the routing of the old languages.

3.1 Similarity Analysis

Considering the simplistic structure of the routing network, which essentially performs a linear transformation, the routing results are typically dependent on the input of the router network, i.e., the Hidden States obtained after the Attention module (HSAs). Therefore, we analyze the similarity of

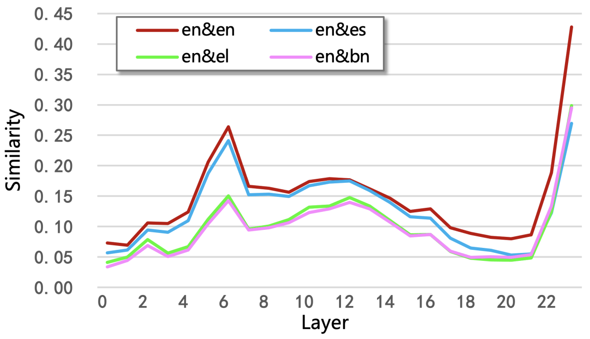


Figure 1: The similarity of HSAs in different languages across all layers within Qwen1.5-1.8B.

HSAs in different languages to investigate the potential differences of languages at different layers.

Given two languages ℓ_1 and ℓ_2, we randomly sample many examples for each language and perform forward propagation to obtain HSAs of each token at different layers. Then we randomly select Q tokens for each language and take their HSAs to build the candidate set $\mathbf{Q}_{\ell_1}$ and $\mathbf{Q}_{\ell_2}$ for calculating the similarity of these selected tokens in different languages. For example in layer i, the similarity of $\mathbf{Q}^i_{\ell_1}$ and $\mathbf{Q}^i_{\ell_2}$ are obtained by calculating the cosine similarity of any two HSA vectors:

$$\mathrm{sim}(\mathbf{Q}^i_{\ell_1}, \mathbf{Q}^i_{\ell_2}) = \frac{1}{Q^2} \sum_{\mathbf{u} \in \mathbf{Q}^i_{\ell_1}} \sum_{\mathbf{v} \in \mathbf{Q}^i_{\ell_2}} \cos(\mathbf{u}, \mathbf{v}), \tag{8}$$

where $\cos(\cdot)$ denotes the cosine similarity, and $\mathbf{u}/\mathbf{v} \in \mathbb{R}^h$. These HSAs of randomly selected tokens contain different contextual information (Ethayarajh, 2019), which simulates the actual input of the router network.

As a preliminary study, we calculate the similarity of HSAs between some languages in different layers. We select Qwen1.5-1.8B (Team, 2024) as the base model and choose four languages including English (*en*), Spanish (*es*), Bengali (*bn*), and Greek (*el*). For each language ℓ, we sample examples from CulturalX (Nguyen et al., 2024a) and then randomly select $Q = 100000$ tokens to obtain the candidate set $\mathbf{Q}_\ell$. Then we follow Eq.(8) to calculate the similarity between four language pairs, including *en&en*, *en&es*, *en&bn*, and *bn&hi*. The results in Figure 1 show that the similarity in different layers exhibits significant differences, with higher similarity in the middle and last few layers, and lower similarity in the 0~4 layers and 17~21 layers. *en&en* (the red line) displays the highest similarity across all layers among the four language pairs, while the other 3 language pairs show similar

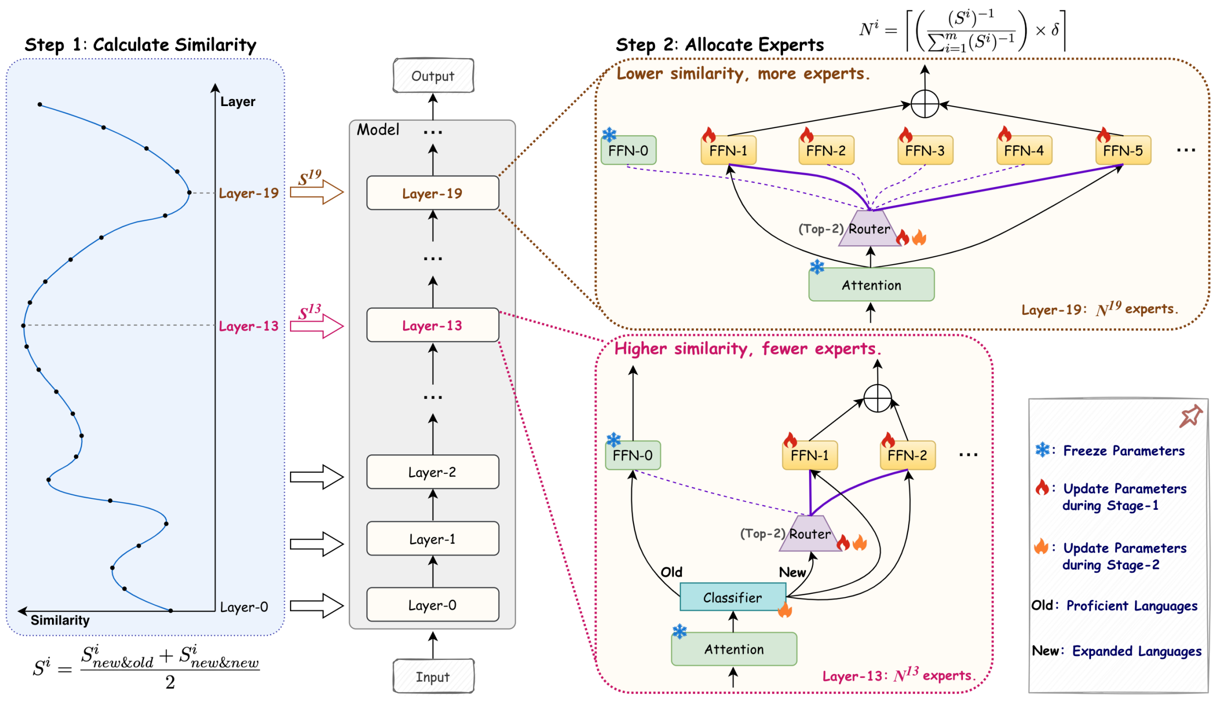

Step 1: Calculate Similarity
Layer
Layer-19
Layer-13
Layer-0
Similarity
$S^i = \frac{S^i_{new\&old} + S^i_{new\&new}}{2}$
S^{19}
S^{13}
Output
Model
Layer-19
Layer-13
Layer-2
Layer-1
Layer-0
Input
Step 2: Allocate Experts
$N^i = \left\lceil \left(\frac{(S^i)^{-1}}{\sum_{i=1}^m (S^i)^{-1}} \right) \times \delta \right\rceil$
Lower similarity, more experts.
FFN-0
FFN-1
FFN-2
FFN-3
FFN-4
FFN-5
(Top-2) Router
Attention
Layer-19: N^{19} experts.
Higher similarity, fewer experts.
FFN-0
FFN-1
FFN-2
(Top-2) Router
Old
New
Classifier
Attention
Layer-13: N^{13} experts.
Freeze Parameters
Update Parameters during Stage-1
Update Parameters during Stage-2
Old: Proficient Languages
New: Expanded Languages

Figure 2: The procedure of our LayerMoE. In Step 1, we first calculate the indicated similarity for each layer. In Step 2, we allocate experts based on the similarity. The layers with lower similarity are allocated more experts, while those with higher similarity are assigned fewer experts. Additionally, we add a classifier in front of the router in the layers with higher similarity to guide the routing of old language tokens.

trends with *en&en*.

According to the similarity difference across all layers, we speculate that the higher similarity indicates that HSAs capture more language-agnostic information, while the lower similarity suggests more language-specific characteristics, which aligns with previous studies (Zhang et al., 2024b; Tang et al., 2024) that have observed the existence of language-agnostic/specific neurons within LLMs. We hypothesize that a layer with higher similarity needs fewer new experts when expanding new languages, as the model can extract language-agnostic representations in this layer. Conversely, lower similarity implies that more experts are necessary to accommodate language-specific features for the new languages. On this basis, we propose our LayerMoE.

3.2 Layer-wise Expert Allocation

Figure 2 shows the procedure of our method. When expanding new languages, we first calculate the similarity between *new&new* and *new&old* languages for each layer. Then we allocate appropriate experts for each layer according to the similarity.

Step 1: Calculate Similarity. Given the dense model $\mathcal{M}_0$, and its proficient language group L_{old} and current expanded language group L_{new}, we first randomly select Q tokens to build the candidate token set $\mathbf{Q}_{old}$ and $\mathbf{Q}_{new}$. Then we calculate the similarity of $\mathbf{Q}_{new}\&\mathbf{Q}_{old}$ and $\mathbf{Q}_{new}\&\mathbf{Q}_{new}$, and compute their average as the indicated similarity of each layer. For example in layer i, the indicated similarity S^i is calucluated by:

$$S^i_{new\&old} = \frac{\sum_j^{|L_{new}|}\sum_k^{|L_{old}|}\text{sim}(\mathbf{Q}^i_{\ell_j}, \mathbf{Q}^i_{\ell_k})}{|L_{new}||L_{old}|},$$

$$S^i_{new\&new} = \frac{\sum_j^{|L_{new}|}\sum_{k,k\neq j}^{|L_{new}|}\text{sim}(\mathbf{Q}^i_{\ell_j}, \mathbf{Q}^i_{\ell_k})}{2|L_{new}|(|L_{new}|-1)},$$

$$S^i = \frac{S^i_{new\&old} + S^i_{new\&new}}{2}, \tag{9}$$

where $\text{sim}(\cdot,\cdot)$ follows Eq.(8).

Step 2: Allocate Experts. Given the dense model with m layers, we obtain the indicated similarity vector $\mathcal{S} = [S^1, S^2, \ldots, S^m]$ after Step 1. Then numbers of experts across m layers $\mathcal{N} = [N^1, N^2, \ldots, N^m]$ is calculated as follows:

$$N^i = \left\lceil \left(\frac{(S^i)^{-1}}{\sum_{i=1}^m (S^i)^{-1}} \right) \times \delta \right\rceil, \tag{10}$$

where δ is the target sum number of all experts such that $\sum_{i=1}^m N^i = \delta$, and $\lceil\cdot\rceil$ denotes round up to the nearest integer. As a result, fewer experts will be allocated to the layers with higher similarity, while more experts to the layers with lower similarity. After allocating experts to each layer, we conduct the post-training (stage-1) using the data of new languages to obtain the MoE model $\mathcal{M}_{stage1}$.

3.3 Routing Classification Network

During stage-2, the key objective is $\mathcal{L}_{LPR}$ that trains the router network to route the tokens in old languages to old experts. To reduce the confusion of the routers on old languages and new languages, we add a routing classification network W_c in front of the router network to judge whether the current token belongs to old languages, where $W_c \in \mathbb{R}^{h\times 2}$. After obtaining the model $\mathcal{M}_{stage1}$, we first calculate the similarities $\mathcal{S}_{new\&old} = [S^1_{new\&old}, S^2_{new\&old}, \dots, S^m_{new\&old}]$ between new and old languages following Eq.(9) for each layer. Then we add the classifier to the top-K layers according to $\{i|S^i_{new\&old} \in \mathrm{TopK}(\mathcal{S}_{new\&old}, K)\}$. The training objective of W_c is:

$$\mathcal{L}_{CLS}(\theta_c) = -\sum_{t\in\mathbb{T}} F(t)\mathrm{log\_softmax}(\mathbf{x}\cdot W_c), \tag{11}$$

where the indicator function $F(t)$ is the same as Eq.(6), $\mathbf{x}$ is the hidden state of the token t, and θ_c denotes the parameters of the classifiers. Therefore, the final training loss of stage-2 is:

$$\mathcal{L}_{stage2}(\theta_r, \theta_c) = \mathcal{L}_{NTP} + \beta\mathcal{L}_{LPR} + \gamma\mathcal{L}_{CLS}, \tag{12}$$

where β and γ are hyper-parameters that control the weights of $\mathcal{L}_{LPR}$ and $\mathcal{L}_{CLS}$.

During inference, the tokens classified to old languages are directly routed to the old expert, while others classified to new languages are normally routed following Eq.(3):

$$\mathbf{y} = \begin{cases} E_0(\mathbf{x}) + \mathbf{x}, & cls(\mathbf{x}) = 0, \\ \sum_{i\in\mathcal{I}} \frac{G_i(\mathbf{x})}{\Sigma_{j\in\mathcal{I}} G_j(\mathbf{x})} E_i(\mathbf{x}) + \mathbf{x}, & cls(\mathbf{x}) = 1, \end{cases} \tag{13}$$

where $cls(\mathbf{x}) = \mathrm{argmax}(\mathbf{x}\cdot W_r)$, “0/1” denotes the current token belonging to “old/new” languages, and E_0 represents the old expert.

4 Experiments

4.1 Experimental Setup

We conduct the experiments under single-expansion and lifelong-expansion settings. The single-expansion setting means expanding one group of new languages to enhance the original model once, while the lifelong-expansion setting means sequentially adapting the model on multiple groups of new languages.

Model and Languages. Following MoE-LPR (Zhou et al., 2024), we select Qwen1.5-1.8B (Team, 2024) as our base model, which has a powerful multilingual tokenizer and lower computation overhead for easily upcycling. Based on the ability of Qwen1.5-1.8B, we select three languages with high performance as the old language group (G0) to observe the catastrophic forgetting phenomenon: English (*en*), Chinese (*zh*), and Spanish (*es*). Additionally, we choose six low-resource languages[2] in which Qwen1.5-1.8B performs poorly as the expanded languages: Greek (*el*), Hungarian (*hu*), Turkish (*tr*), Bengali (*bn*), Hindi (*hi*), and Nepali (*ne*). Specifically, the selection of *el*, *hu*, and *tr* follows MoE-LPR (Zhou et al., 2024), while *bn*, *hi*, and *ne* are selected to observe the performance of different methods on morphologically rich non-Latin scripts, which pose greater modeling challenges and suffer from limited resources. Hence, we divide them into two groups (G1: *el*, *hu*, and *tr*, G2: *bn*, *hi*, and *ne*) for the single-expansion setting (G0$\rightarrow$G1, G0$\rightarrow$G2, and G0$\rightarrow$G1$+$G2) and the lifelong-expansion setting (G0$\rightarrow$G1$\rightarrow$G2, and G0 $\rightarrow$G2$\rightarrow$G1).

Training. During stage-1, we train the new experts and the router network with new language data. For each new language, we sample 2 billion tokens from CulturalX (Nguyen et al., 2024a), a substantial multilingual dataset with 6.3 trillion tokens in 167 languages. The detailed training setup follows MoE-LPR (Zhou et al., 2024): the batch size is 512, the sequence length is 1024, the learning rate is 5e-5, and the weight α of the load balancing loss is 0.01. During stage-2, we mix the data of old languages and new languages to train the router network and the classification network. The top-7 layers with high similarity for the single-expansion settings and top-5 layers for the lifelong-expansion settings are taken according to the best performance to add the routing classification network. We randomly sample 50K examples for each old language and 100K examples for each new language as the training data of stage-2. Following MoE-LPR (Zhou et al., 2024),

[2]The performance on six languages is shown in Table 1. More details of each language are listed in Appendix B.1.

the English/Chinese/Spanish data is sampled from Slimpajam (Soboleva et al., 2023), SkyPile-150B (Wei et al., 2023a), and CulturalX (Nguyen et al., 2024a), respectively. The 100k examples for each new language are sampled from the training data of stage-1. We set the batch size to 768 and the learning rate to 5e-5. The weight of $\mathcal{L}_{LPR}$ and $\mathcal{L}_{CLS}$ loss is set to 0.1. All experiments are conducted on 8×A100-40G GPUs.

Evaluation. We evaluate all methods on 4 benchmarks: ARC-Challenge (25-shot) (Clark et al., 2018), MMLU (5-shot) (Hendrycks et al., 2021), HellaSwag (10-shot) (Zellers et al., 2019), and Belebele (5-shot) (Bandarkar et al., 2024). We utilize the `lm-evaluation-hardness`[3] framework (Gao et al., 2024b) to conduct evaluation on their multilingual version. Please refer to Appendix B.2 for the detailed introduction.

4.2 Baselines

MoE-LPR (Zhou et al., 2024) adds the same number of new experts for each layer. Gao et al. proposes MOLA that experts in lower layers are more similar than those in higher layers, *i.e.*, higher layers need more experts. For G0→G1, G0→G2, we conduct three versions of MoE-LPR, adding 2/3/5 experts per layer as 3*24/4*24/6*24 (Qwen1.5-1.8B contains 24 layers). And MOLA (∇, **2468**) is employed following the best setting, allocating 2/4/6/8 experts for 0~5/6~11/12~17/18~23 layers. For G0 → G1 + G2, G0 → G1 → G2, and G0 → G2 → G1, we conduct MoE-LPR (7*24/4*24) as baselines. For a fair comparison, we reproduce all versions of MoE-LPR and MOLA using the same data as our method, *i.e.*, 2 billion tokens for each new language.

Additionally, we directly take five baselines of MoE-LPR (trained with 8 billion tokens for each new language) under G0 → G1 for a clear comparison, including LoRA (Hu et al., 2021), Full Fine-tuning, LLaMA-Pro (Wu et al., 2024), MoE, and LoRAMoE (Dou et al., 2024). **LoRA** (Hu et al., 2021) introduces Low-Rank Adaptation for efficiently fine-tuning large language models. In this paper, the LoRA targets include all linear modules. The LoRA rank is set to 8. **Full Fine-tuning** is fine-tuning all parameters on the dense model. **LLaMA-Pro** (Wu et al., 2024) insert new transformer blocks to replace the duplicate layers. Only

[3] `https://github.com/EleutherAI/lm-evaluation-harness`

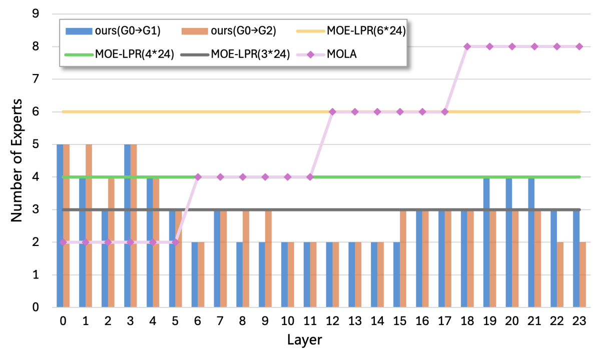


Figure 3: The expert allocation of our method and baselines for each layer under G0→G1, G0→G2.

the newly inserted blocks are trained to store new knowledge during post-pretraining. 12 new layers (the best setting) are inserted into the original dense model. **MoE** means that only the stage-1 of MoE-LPR is conducted. **LoRAMoE** (Dou et al., 2024) is a novel framework that combines multiple LoRAs with a router network. The router selects all LoRAs for each token. The LoRA rank is set to 180, and the number of LoRAs is 8 for each module. Due to the poor performance of the 5 baselines, we do not reproduce them and directly take the results trained with 8 billion tokens for each new language from the MoE-LPR paper (Zhou et al., 2024).

4.3 Allocation of Experts

After calculating the similarity following Step 1, we assign the number of new experts for each layer according to Eq.(10). For a fair comparison with MoE-LPR (3*24), we set the targeted number of new experts δ in Eq.(10) to 72. The allocation results of our LayerMoE and baselines (under G0 →G1 and G0→G2) are present in Figure 3. Our method allocates more experts for the shallow and deep layers and less experts for the middle layers. The detailed similarity and expert allocation results of our method under each setting are plotted in Figure 4 of Appendix B.3.

5 Results and Analysis

5.1 Main Results

Single-Expansion. The evaluation results of the single-expansion setting (G0→G1 and G0→G2) are listed in Table 1. The results show that our method surpasses all baselines in "*Old-avg*" and "*New-avg*" under the expansion of G1 and G2. Particularly, our method outperforms the previous SOTA method "MoE-LPR (6*24)" with 60% fewer parameter numbers of newly added experts

Methods	Expand-params	ARC-C		MMLU		Hellaswag		Belebele				
		Old	New	Old	New	Old	New	Old	New	*Old-avg*	*New-avg*	*avg*
Old (G0): ***en, es, zh*** **→New (G1):** ***el, hu, tr***												
Qwen1.5-1.8B	/	33.31	22.93	47.23	30.52	49.70	29.27	55.89	32.92	46.53	28.91	37.72
*LoRA	/	28.33	23.89	37.42	29.30	41.48	29.78	39.45	26.93	36.67	27.48	32.07
*Full Fine-tuning	/	31.72	25.98	43.51	33.18	47.38	35.28	45.26	33.70	41.98	32.04	37.00
*LLaMA-Pro	0.6B	31.77	24.35	44.06	34.02	48.36	33.85	48.78	31.52	43.24	30.94	37.09
*MoE	4B	32.51	26.43	44.16	35.07	48.54	37.01	45.37	32.74	42.65	32.81	37.73
*LoRAMoE	0.8B	32.43	26.63	45.41	34.17	48.61	37.17	47.74	32.81	43.55	32.70	38.12
MoE-LPR (6*24)	4B	32.26	26.55	46.51	34.95	49.75	37.53	52.92	39.04	45.36	34.52	39.94
MOLA (∇)	3.2B (↓ 20%)	32.49	27.00	46.38	33.98	49.63	37.03	53.85	38.37	45.59	34.10	39.84
MoE-LPR (4*24)	2.4B (↓ 40%)	32.29	26.69	46.51	34.11	49.60	37.12	52.33	37.74	45.18	33.92	39.55
MoE-LPR (3*24)	1.6B (↓ 60%)	32.77	26.43	46.54	34.59	49.63	37.08	52.07	37.15	45.25	33.81	39.53
LayerMoE (Ours)	**1.6B (↓ 60%)**	**33.03**	**26.66**	46.12	34.80	49.24	37.01	**54.82**	**40.37**	**45.80**	**34.71**	**40.26**
Old (G0): ***en, es, zh*** **→New (G2):** ***bn, hi, ne***												
Qwen1.5-1.8B	/	33.31	23.05	47.23	29.54	49.70	28.22	55.89	28.85	46.53	27.41	36.97
MoE-LPR (6*24)	4B	33.06	23.85	47.04	30.62	49.69	30.92	53.11	33.85	45.72	29.81	37.77
MOLA (∇)	3.2B (↓ 20%)	32.48	23.67	47.02	30.82	49.73	30.48	53.59	32.15	45.71	29.28	37.49
MoE-LPR (4*24)	2.4B (↓ 40%)	33.08	23.27	47.21	30.74	49.64	30.67	52.85	33.04	45.70	29.43	37.56
MoE-LPR (3*24)	1.6B (↓ 60%)	32.77	23.53	46.97	30.72	49.68	30.56	52.59	33.33	45.50	29.53	37.52
LayerMoE (Ours)	**1.6B (↓ 60%)**	32.52	**24.02**	**47.10**	**30.85**	49.46	30.47	**55.15**	**34.11**	**46.06**	**29.86**	**37.96**

Table 1: The results of the single-expansion settings (G0→G1, G0→G2). The results marked with * represent that they are directly taken from the MoE-LPR paper (Zhou et al., 2024), which utilizes 8 billion tokens for each new language during post-pretraining. Other experiments are trained with 2 billion tokens for each new language. "Expand-params" denotes the parameter number of newly added experts and (↓ *%) means the reduction percentage of parameters compared to MoE-LPR (6*24). The results in **bold** mean they are better than the previous SOTA method "MoE-LPR (6*24)". The detailed results for each language are reported in Table 7 and 8 of Appendix C.

(*i.e.*, from 4B to 1.6B), which suggests that our method is more parameter-efficient. On the contrary, MOLA and MoE-LPR (3*24/4*24) underperform MoE-LPR (6*24) when decreasing the number of expansion parameters. Although other baselines (LoRA, Full Fine-tuning, LLaMA-Pro, MoE, and LoRAMoE) utilize 8B tokens for each new language during post-pretraining, they still perform poorly due to the lack of special designs. Additionally, when keeping the same parameter numbers (1.6B) with "MoE-LPR (3*24)", our method exhibits a significant improvement over "MoE-LPR (3*24)" referring to "*New-avg*" (for G1 from 33.81 to 34.71 and for G2 from 29.53 to 29.86). This superiority indicates that allocating experts based on the language similarity in different layers has a positive contribution to the expansion of new languages. Furthermore, the results of "*Old-avg*' showcase that our method performs the best on preserving the ability of the old languages, with 45.80 for expanding G1 and 46.06 for G2. These results prove that appropriate correction of the router network can effectively mitigate the catastrophic forgetting of old languages.

The expansion performance exhibits differences among different languages according to "*New-avg*". When conducting G0→G1, the performance of new languages improves by 5.8 from 28.91 to 34.71, while only improves by 2.45 (from 27.41 to 29.86) on G0→G2. Although the amounts of training tokens (2B) for each new language are the same, the morphologically rich non-Latin languages in G2 (*bn*, *hi*, *ne*) obtain a smaller performance improvement compared to expanding G1. The difference can be attributed to their lower encoding efficiencies (Arnett and Bergen, 2024; Rust et al., 2021; Ahia et al., 2023; Petrov et al., 2023; Ali et al., 2024), *i.e.*, these languages typically employ UTF-8 encoding during the tokenization process, which results in longer encoded sequences. Consequently, the number of effective training tokens is reduced, limiting the potential for performance improvement. These results suggest that more tokens are needed for better expansion of G2. Furthermore, when expanding G1 or G2, there are different degrees of forgetting of old languages. Expanding G1 brings more forgetting (from 46.53 to 45.80) compared to expanding G2 (from 46.53 to 46.06), which may be caused by more token overlaps between G0 and G1 since the writing systems of *en/es* in G0 and *hu/tr* in G1 all contain the Latin alphabet (as shown in Table 6 of Appendix B.1).

Lifelong-Expansion. The results of G0→G1→G2 and G0→G2→G1 in Table 2 demonstrate that

Methods	Order	G1-E-p	G2-E-p	All-E-p	*G0-avg*	*G1-avg*	*G2-avg*	*avg*
Qwen1.5-1.8B	/	/	/	/	46.53	28.91	27.41	34.29
MoE-LPR (7*24)	G0→G1+G2	4.8B	4.8B	4.8B	45.15	34.17	29.98	36.43
MoE-LPR (4*24)	G0→G1+G2	2.4B	2.4B	2.4B (↓ 50.0%)	45.28	34.07	29.55	36.30
LayerMoE (Ours)	G0→G1+G2	1.6B	1.6B	**1.6B (↓ 66.7%)**	**45.56**	**34.73**	29.94	**36.74**
MoE-LPR (7*24)	G0→G1→G2	2.4B	2.4B	4.8B	45.88	34.40	30.26	36.85
LayerMoE (Ours)	G0→G1→G2	1.6B	1.6B	**3.2B (↓ 33.3%)**	**46.10**	**34.56**	**30.29**	**36.98**
MoE-LPR (7*24)	G0→G2→G1	2.4B	2.4B	4.8B	45.28	33.42	28.39	35.70
LayerMoE (Ours)	G0→G2→G1	1.6B	1.6B	**3.2B (↓ 33.3%)**	**46.25**	**33.76**	**29.80**	**36.60**

Table 2: The results of G0→G1+G2 and the lifelong-expansion settings (G0→G1→G2 and G0→G2 →G1). "G1/G2/All-E-p" represents the expansion number of newly added parameters for G1, G2, and the sum, respectively. The detailed results for each language are reported in Table 9 of Appendix C.

	Old-avg (↑)	*New-avg* (↑)	*avg* (↑)
LayerMoE (G0→G1)	**45.80**	**34.71**	**40.26**
w/ random	45.68 (↓ 0.12)	34.11 (↓ **0.60**)	39.90
w/o classifier	45.33 (↓ **0.47**)	34.61 (↓ 0.10)	39.97
LayerMoE (G0→G2)	**46.06**	**29.86**	**37.96**
w/ random	45.94 (↓ 0.12)	29.15 (↓ **0.71**)	37.54
w/o classifier	45.39 (↓ **0.67**)	29.53 (↓ 0.33)	37.46

Table 3: The ablation results under G0→G1 and G0→G2. "*w/* random" means randomly allocating numbers of new experts for each layer without calculating the similarity. "*w/o* classifier" denotes no classifier is added in front of the router network.

	Old	*New*	*avg*		*Old*	*New*	*avg*
top-1	44.90	34.30	39.60	**top-7**	45.80	34.71	**40.26**
top-2	44.81	34.69	39.75	top-8	45.79	34.54	40.17
top-3	45.10	34.41	39.76	top-9	45.95	34.42	40.19
top-4	45.41	34.67	40.04	top-10	**46.16**	34.28	40.22
top-5	45.28	34.66	39.97	top-24	46.04	34.32	40.18
top-6	45.74	**34.72**	40.23	top-23	46.15	33.23	39.69
last-7	44.96	34.60	39.78	random-7	45.32	34.60	39.96

Table 4: The results of adding classifiers in different layers under G0→G1. "top-*" denotes adding classifiers to * layers with the highest similarity. "random-7" means randomly selecting 7 layers, while "last-7" represents the 7 layers with the lowest similarity.

our method outperforms MoE-LPR (7*24) with fewer 33.3% new experts in terms of "*G0/G1/G2-avg*", proving the effectiveness of our method in lifelong-expansion setting. Additionally, we surprisingly observe that G0→G1→G2 brings better expansion performance for G2 ("*G2-avg*") and less forgetting of old languages ("*G0-avg*") compared to G0→G1+G2. But G0→G2→G1 does not show similar improvement for G1. This difference indicates that the learning order of different languages may have non-trivial influences on the final performance of each language.

5.2 Ablation Study

To verify the effectiveness of our expert allocation algorithm and the added classifier, we conduct the ablation experiments under G0→G1 and G0→G2 settings and report the results in Table 3. These results show that randomly allocating experts for each layer without calculating the similarity significantly decreases the expansion performance (↓ 0.60 in G1, ↓ 0.71 in G2). Additionally, the forgetting of old languages becomes worse (↓ 0.47 in G1, ↓ 0.67 in G2) when no classifiers are added in front of the router network. Overall, these results prove the necessity of our layer-wise expert allocation algorithm and the added classifier.

5.3 Classifiers in Different Layers

The top-K layers with the higher similarity between new and old languages are selected according to $\{i|S^i_{new\&old} \in \text{TopK}(\mathcal{S}_{new\&old}, K)\}$ to add the classifier. To investigate the best value of K, we conduct the experiments from top-1 to top-11 and top-24 (all layers) under G0→G1 and list the results in Table 4. The results show that as more classifiers are added, the performance of the new language expansion first increases and then decreases, and the performance of the old language preservation increases constantly. The best overall performance peaks on the top-7 setting. Additionally, we also report the results of "random-7" and "last-7" to verify the effectiveness of adding classifiers in layers with higher similarity. Randomly selecting layers or opting for those with lower similarity consistently yields poor performance in preserving the capability of old languages.

5.4 Generalization Study

To evaluate the generalization of our method on different models, we conduct a brief verification of our method on G0→G1 with Llama-3.2-3B

	ARC-C		MMLU		Hellaswag		Belebele		FLORES				
Methods	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**	***Old-avg***	***New-avg***	***avg***
Llama-3.2-3B	44.64	33.39	49.84	42.15	63.56	43.62	70.48	61.37	84.88	83.96	62.68	52.90	57.79
MoE-LPR (3*24)	43.70	37.58	49.99	43.98	64.17	51.19	70.74	64.41	84.58	84.35	62.64	56.30	59.47
LayerMoE (Ours)	**44.10**	**37.64**	49.96	**44.08**	64.07	**51.33**	**71.22**	**64.63**	**84.72**	**84.69**	**62.81**	**56.47**	**59.64**

Table 5: The results of Llama-3.2-3B under the single-expansion setting (G0→G1). The detailed results for each language are reported in Table 10 and 11 of Appendix C.

(Grattafiori et al., 2024) and evaluate the performance on 5 benchmarks. Particularly, we add the machine translation task FLORES (Team, 2022) to prove the effectiveness of our method on generation tasks. Specifically, we set the following directions: *en* → *zh*, *zh* → *en*, *en* → *es*, *es* → *en*, *en* → *el*, *el* → *en*, *en* → *hu*, *hu* → *en*, *en* → *tr*, and *tr* → *en*. And we utilize COMET (Rei et al., 2020) as the evaluation metric. The results in Table 5 demonstrate the effectiveness of our method across different LLMs and generative tasks, which further proves the practical application value of our method.

6 Related Work

There are two categories (Huang et al., 2024) in expanding new languages to build powerful multilingual LLMs: (1) utilizing the collected multilingual corpus to train the multilingual LLMs from scratch and (2) continually training LLMs on top of existing LLMs only with new language data. The first category generally necessitates meticulous design in terms of the data ratio between different languages (AI et al., 2024; Zhang et al., 2024a; Nguyen et al., 2024b) and the implementation of curriculum learning strategies (Wei et al., 2023b; Üstün et al., 2024) during the training process to ensure a balanced improvement in proficiency across various languages. Training from scratch typically requires excessive computational resources and training durations, which are often challenging to implement and environmentally unfriendly.

The second category is to continually train the original model with new language data to enhance the ability of new languages while avoiding the catastrophic forgetting of the ability in originally proficient languages. MoE-LPR (Zhou et al., 2024) and MoE-CT (Li et al., 2024) upcycle the dense model to the MoE model for improving the multilingual ability of LLMs. Similarly, some methods also have introduced external parameters to improve the ability of the original LLMs and preserve the original capabilities by freezing the old parameters. LLaMA-Pro (Wu et al., 2024) insert new transformer blocks to replace the duplicate layers to further enhance the original LLM. LoRAMoE (Dou et al., 2024) combines multiple low-rank adapters with a router network to achieve better performance on downstream tasks. In this work, we only focus on the multilingual expansion and propose LayerMoE to address the limitations of existing methods, *e.g.*, inefficient expansion, high resource usage, and accumulated forgetfulness of old languages. Additionally, the concurrent work AlphaLoRA (Qing et al., 2024) similar to ours allocates LoRA experts based on Heavy-Tailed Self-Regularization Theory to mitigate redundancy of LoRA experts and improve the general ability of LLMs. Differently, we calculate the representation similarity of different languages in different layers as the indicator to allocate new experts for each layer, which is intuitive and effective for the expansion of new languages.

7 Conclusion

In this work, we propose a layer-wise expert allocation algorithm to assign the appropriate number of new experts for each layer when expanding new languages for existing LLMs. We first calculate the similarity of different languages for each layer as the indicator and allocate more experts to the layers exhibiting lower similarity. Additionally, we add a classifier in front of the router network for the layers with higher similarity to further mitigate the forgetting of old languages. Experimental results demonstrate the effectiveness of our method, which is parameter-efficient and achieves good expansion of new languages and less forgetting of old languages.

Limitations

Although our method achieves better improvements in terms of "*Old-avg*" and "*New-avg*" than MoE-LPR (6*24) with 60% fewer experts, the improvements in different benchmarks and languages are inconsistent. The inherent reasons for the inconsistency are not yet clear and need further investigation. Additionally, we set the total number of new experts according to the baseline "MoE-LPR (3*24)" for a fair comparison rather than exploring the optimal total number. In the future, we will continually solve these limitations for a better expert allocation solution.

Acknowledgments

The research work described in this paper has been supported by the National Nature Science Foundation of China (No. 62476023, 61976016, 62376019, 61976015) and Henan Provincial Science and Technology Research Project (No. 252102210102), and the authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve this paper.

References

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David Mortensen, Noah Smith, and Yulia Tsvetkov. 2023. Do all languages cost the same? tokenization in the era of commercial language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9904–9923, Singapore. Association for Computational Linguistics.

01. AI, :, Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Tao Yu, Wen Xie, Wenhao Huang, Xiaohui Hu, Xiaoyi Ren, Xinyao Niu, Pengcheng Nie, Yuchi Xu, Yudong Liu, Yue Wang, Yuxuan Cai, Zhenyu Gu, Zhiyuan Liu, and Zonghong Dai. 2024. Yi: Open foundation models by 01.ai. *Preprint*, arXiv:2403.04652.

Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, Charvi Jain, Alexander Weber, Lena Jurkschat, Hammam Abdelwahab, Chelsea John, Pedro Ortiz Suarez, Malte Ostendorff, Samuel Weinbach, Rafet Sifa, Stefan Kesselheim, and Nicolas Flores-Herr. 2024. Tokenizer choice for LLM training: Negligible or crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3907–3924, Mexico City, Mexico. Association for Computational Linguistics.

Catherine Arnett and Benjamin K. Bergen. 2024. Why do language models perform worse for morphologically complex languages? *Preprint*, arXiv:2411.14198.

Lucas Bandarkar, Davis Liang, Benjamin Muller, Mikel Artetxe, Satya Narayan Shukla, Donald Husa, Naman Goyal, Abhinandan Krishnan, Luke Zettlemoyer, and Madian Khabsa. 2024. The belebele benchmark: a parallel reading comprehension dataset in 122 language variants. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 749–775. Association for Computational Linguistics.

Yuheng Chen, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2023. Journey to the center of the knowledge neurons: Discoveries of language-independent knowledge neurons and degenerate knowledge neurons. *Preprint*, arXiv:2308.13198.

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

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Loramoe: Alleviate world knowledge forgetting in large language models via moe-style plugin. *Preprint*, arXiv:2312.09979.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Preprint*, arXiv:2101.03961.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. 2024a. Higher layers need more lora experts. *Preprint*, arXiv:2402.08562.

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. 2024b. A framework for few-shot language model evaluation.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

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

Kaiyu Huang, Fengran Mo, Hongliang Li, You Li, Yuanchi Zhang, Weijian Yi, Yulong Mao, Jinchen Liu, Yuzhuang Xu, Jinan Xu, Jian-Yun Nie, and Yang Liu. 2024. A survey on large language models with multilingualism: Recent advances and new frontiers. *Preprint*, arXiv:2405.10936.

Kaiyu Huang, Fengran Mo, Xinyu Zhang, Hongliang Li, You Li, Yuanchi Zhang, Weijian Yi, Yulong Mao, Jinchen Liu, Yuzhuang Xu, Jinan Xu, Jian-Yun Nie, and Yang Liu. 2025. A survey on large language models with multilingualism: Recent advances and new frontiers. *Preprint*, arXiv:2405.10936.

Viet Lai, Chien Nguyen, Nghia Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan Rossi, and Thien Nguyen. 2023a. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 318–327, Singapore. Association for Computational Linguistics.

Viet Dac Lai, Nghia Ngo, Amir Pouran Ben Veyseh, Hieu Man, Franck Dernoncourt, Trung Bui, and Thien Huu Nguyen. 2023b. ChatGPT beyond English: Towards a comprehensive evaluation of large language models in multilingual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13171–13189, Singapore. Association for Computational Linguistics.

Tianhao Li, Shangjie Li, Binbin Xie, Deyi Xiong, and Baosong Yang. 2024. Moe-ct: A novel approach for large language models training with resistance to catastrophic forgetting. *Preprint*, arXiv:2407.00875.

Thuat Nguyen, Chien Van Nguyen, Viet Dac Lai, Hieu Man, Nghia Trung Ngo, Franck Dernoncourt, Ryan A. Rossi, and Thien Huu Nguyen. 2024a. CulturaX: A cleaned, enormous, and multilingual dataset for large language models in 167 languages. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 4226–4237, Torino, Italia. ELRA and ICCL.

Xuan-Phi Nguyen, Wenxuan Zhang, Xin Li, Mahani Aljunied, Zhiqiang Hu, Chenhui Shen, Yew Ken Chia, Xingxuan Li, Jianyu Wang, Qingyu Tan, Liying Cheng, Guanzheng Chen, Yue Deng, Sen Yang, Chaoqun Liu, Hang Zhang, and Lidong Bing. 2024b. Seallms – large language models for southeast asia. *Preprint*, arXiv:2312.00738.

Aleksandar Petrov, Emanuele La Malfa, Philip H. S. Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. *Preprint*, arXiv:2305.15425.

Peijun Qing, Chongyang Gao, Yefan Zhou, Xingjian Diao, Yaoqing Yang, and Soroush Vosoughi. 2024. AlphaLoRA: Assigning LoRA experts based on layer training quality. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20511–20523, Miami, Florida, USA. Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online. Association for Computational Linguistics.

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.

Tianyi Tang, Wenyang Luo, Haoyang Huang, Dongdong Zhang, Xiaolei Wang, Xin Zhao, Furu Wei, and Ji-Rong Wen. 2024. Language-specific neurons: The key to multilingual capabilities in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5701–5715, Bangkok, Thailand. Association for Computational Linguistics.

NLLB Team. 2022. No language left behind: Scaling human-centered machine translation.

Qwen Team. 2024. Introducing qwen1.5.

Ahmet Üstün, Viraat Aryabumi, Zheng Yong, Wei-Yin Ko, Daniel D'souza, Gbemileke Onilude, Neel Bhandari, Shivalika Singh, Hui-Lee Ooi, Amr Kayid, Freddie Vargus, Phil Blunsom, Shayne Longpre, Niklas Muennighoff, Marzieh Fadaee, Julia Kreutzer, and Sara Hooker. 2024. Aya model: An instruction finetuned open-access multilingual language model. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15894–15939, Bangkok, Thailand. Association for Computational Linguistics.

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. 2023a. Skywork: A more open bilingual foundation model. *Preprint*, arXiv:2310.19341.

Xiangpeng Wei, Haoran Wei, Huan Lin, Tianhao Li, Pei Zhang, Xingzhang Ren, Mei Li, Yu Wan, Zhiwei Cao, Binbin Xie, Tianxiang Hu, Shangjie Li, Binyuan Hui, Bowen Yu, Dayiheng Liu, Baosong Yang, Fei Huang, and Jun Xie. 2023b. Polylm: An open source polyglot large language model. *Preprint*, arXiv:2307.06018.

Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ying Shan, and Ping Luo. 2024. Llama pro: Progressive llama with block expansion. *Preprint*, arXiv:2401.02415.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Shaolei Zhang, Kehao Zhang, Qingkai Fang, Shoutao Guo, Yan Zhou, Xiaodong Liu, and Yang Feng. 2024a. Bayling 2: A multilingual large language model with efficient language alignment. *Preprint*, arXiv:2411.16300.

Xue Zhang, Yunlong Liang, Fandong Meng, Songming Zhang, Yufeng Chen, Jinan Xu, and Jie Zhou. 2024b. Multilingual knowledge editing with language-agnostic factual neurons. *Preprint*, arXiv:2406.16416.

Yuanchi Zhang, Yile Wang, Zijun Liu, Shuo Wang, Xiaolong Wang, Peng Li, Maosong Sun, and Yang Liu. 2024c. Enhancing multilingual capabilities of large language models through self-distillation from resource-rich languages. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11189–11204, Bangkok, Thailand. Association for Computational Linguistics.

Hao Zhou, Zhijun Wang, Shujian Huang, Xin Huang, Xue Han, Junlan Feng, Chao Deng, Weihua Luo, and Jiajun Chen. 2024. Moe-lpr: Multilingual extension of large language models through mixture-of-experts with language priors routing. *Preprint*, arXiv:2408.11396.

A Load Balance Loss

The load balance loss (Fedus et al., 2022) generally mitigates the risk of routing collapse by constraining an even distribution of tokens across different experts. Given a batch tokens $\mathbb{T}$ and N experts for each layer, the load balance loss $\mathcal{L}_{balance}$ (Zhou et al., 2024) is calculated following:

$$\mathcal{L}_{balance} = \sum_{i=1}^{N} f_i P_i, \tag{14}$$

$$f_i = \frac{N}{K|\mathbb{T}|} \sum_{t \in \mathbb{T}} \mathbb{1}\{\text{Token } t \text{ selects expert } i\}, \tag{15}$$

$$P_i = \frac{1}{|\mathbb{T}|} \sum_{t \in \mathbb{T}} G_i(t), \tag{16}$$

where t represents a token, $G_i(t)$ is the router score of the expert i, and $\mathbb{1}(\cdot)$ is a indicator function when the token t selecting the expert i equals 1.

B Experimental Details

B.1 Detailed Introduction of Different Languages

The language families and writing systems of our nine selected languages are listed in Table 6. Specifically, *bn*, *hi*, and *ne* all belong to the Indo-Aryan branch of the Indo-European family and use non-Latin scripts, making them more challenging to model effectively and less well-resourced compared to Latin-script languages.

Languages	**Language Family**	**Writing System**
English (*en*)	Indo-European (Germanic)	Latin alphabet (26 letters)
Chinese (*zh*)	Sino-Tibetan (Sinitic)	Chinese characters
Spanish (*es*)	Indo-European (Romance)	Latin alphabet (27 letters, including ñ)
Greek (*el*)	Indo-European (Hellenic)	Greek alphabet (24 letters)
Hungarian (*hu*)	Uralic (Finno-Ugric)	Latin alphabet with diacritics (40 letters)
Turkish (*tr*)	Turkic	Latin alphabet (29 letters)
Bengali (*bn*)	Indo-European (Indo-Aryan)	Bengali script
Nepali (*ne*)	Indo-European (Indo-Aryan)	Devanagari script
Hindi (*hi*)	Indo-European (Indo-Aryan)	Devanagari script

Table 6: The detailed language families and writing systems.

B.2 Evaluation Details

The `lm-evaluation-hardness` framework has included the multilingual version of ARC-Challenge, MMLU, and HellaSwag translated by Okapi (Lai et al., 2023a). For unsupported languages in new languages, we manually integrate the Turkish version from `OpenLLM Turkish leaderboard`[4] and the Greek version translated by ILSP[5] following the official guide[6].

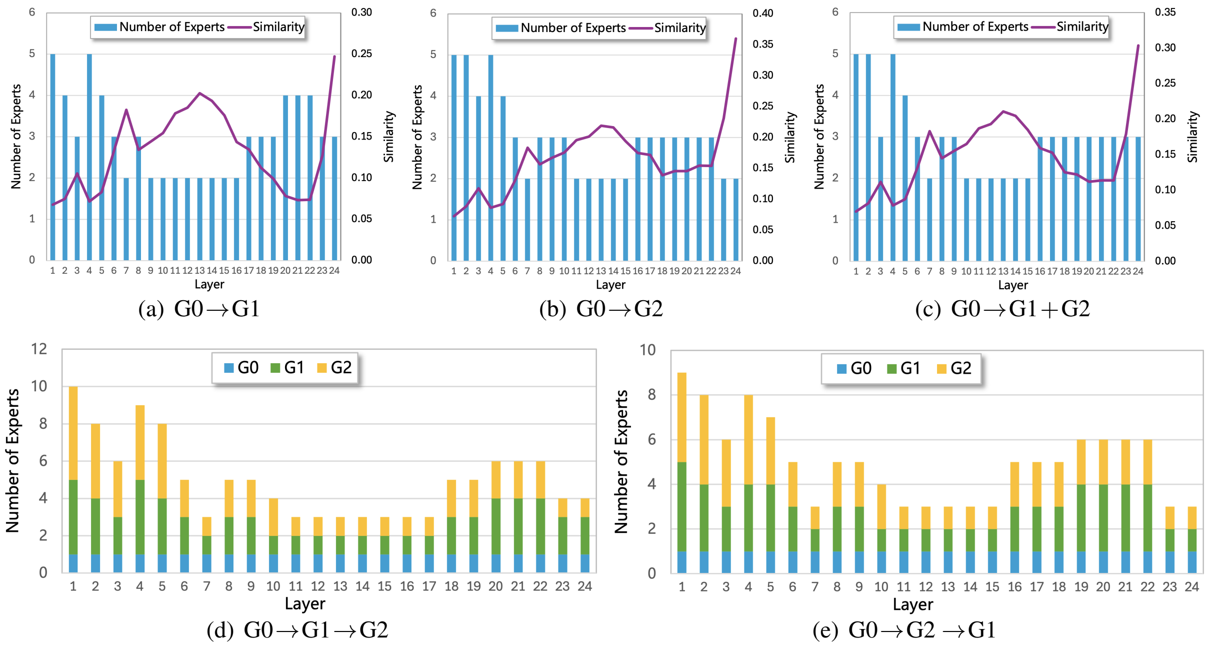
B.3 Similarity and Allocation

Figure 4 presents the detailed similarity and expert allocation of our method for each layer under the five settings.

[4] https://huggingface.co/spaces/malhajar/OpenLLMTurkishLeaderboard

[5] https://huggingface.co/ilsp

[6] https://github.com/EleutherAI/lm-evaluation-harness/blob/main/docs/new_task_guide.md



(a) G0→G1 (b) G0→G2 (c) G0→G1+G2

(d) G0→G1→G2 (e) G0→G2 →G1

Figure 4: The detailed similarity and expert allocation of our method for each layer of Qwen1.5-1.8B under the five settings.

C Detailed Results

Qwen1.5-1.8B: The detailed results for each language under the single-expansion settings (G0→G1, G0→G2) are listed in Table 7 and 8. The detailed results of G0→G1+G2 and the lifelong-expansion settings (G0→G1→G2 and G0→G2 →G1) are reported in Table 9.

	ARC-Challenge			**MMLU**			**Hellaswag**			**Belebele**			
Methods	**en**	**es**	**zh**	**en**	**es**	**zh**	**en**	**es**	**zh**	**en**	**es**	**zh**	***G0-avg***
Qwen1.5-1.8B	37.97	27.86	34.10	45.67	38.81	57.21	61.62	39.35	48.14	60.78	47.44	59.44	46.53
MoE-LPR (6*24)	38.05	26.84	31.88	45.77	38.26	55.50	61.34	39.41	48.49	58.44	45.89	54.44	45.36
MOLA	37.88	26.84	32.74	45.81	38.30	55.02	61.36	39.27	48.26	60.11	47.11	54.33	45.59
MoE-LPR (4*24)	36.69	27.35	32.82	46.09	38.57	54.87	61.17	39.37	48.27	58.33	44.22	54.44	45.18
MoE-LPR (3*24)	37.46	27.35	33.50	46.00	38.67	54.94	61.20	39.43	48.25	59.44	44.44	52.33	45.25
LayerMoE (Ours)	38.57	26.58	33.93	45.81	37.97	54.59	60.90	38.98	47.84	60.56	46.89	57.00	**45.80**
Methods	**el**	**hu**	**tr**	**el**	**hu**	**tr**	**el**	**hu**	**tr**	**el**	**hu**	**tr**	***G1-avg***
Qwen1.5-1.8B	22.95	23.20	22.63	28.94	32.25	30.37	29.15	29.63	29.04	33.33	32.00	33.44	28.91
MoE-LPR (6*24)	25.26	27.14	27.24	32.06	36.30	36.49	36.09	38.63	37.87	38.33	36.56	42.22	34.52
MOLA	25.94	27.40	27.67	31.43	35.80	34.71	35.85	37.79	37.46	35.22	39.00	40.89	34.10
MoE-LPR (4*24)	25.34	27.57	27.16	31.58	35.79	34.97	35.70	38.26	37.39	34.44	40.22	38.56	33.92
MoE-LPR (3*24)	24.32	27.05	27.92	31.90	36.27	35.59	35.55	38.18	37.51	32.00	39.56	39.89	33.81
LayerMoE (Ours)	25.34	26.80	27.84	31.96	36.76	35.68	35.61	38.10	37.31	37.44	41.56	42.11	**34.71**

Table 7: The detailed results of Qwen1.5-1.8B under the G0→G1 setting for each language.

Llama-3.2-3B: Figure 5 presents the detailed similarity and expert allocation of our method for each layer of Llama-3.2-3B under the G0→G1 setting. The detailed results for each language under the G0→G1 setting are listed in Table 10 and 11.

	ARC-Challenge			MMLU			Hellaswag			Belebele			
Methods	**en**	**es**	**zh**	**en**	**es**	**zh**	**en**	**es**	**zh**	**en**	**es**	**zh**	***G0-avg***
Qwen1.5-1.8B	37.97	27.86	34.10	45.67	38.81	57.21	61.62	39.35	48.14	60.78	47.44	59.44	46.53
MoE-LPR (6*24)	37.29	27.26	34.62	46.05	38.46	56.60	61.73	39.27	48.07	59.67	46.33	53.33	45.72
MOLA	37.37	27.26	32.82	45.88	38.71	56.48	61.76	39.09	48.34	59.56	46.22	55.00	45.71
MoE-LPR (4*24)	37.80	27.18	34.27	46.00	38.88	56.76	61.65	39.03	48.24	59.67	45.22	53.67	45.70
MoE-LPR (3*24)	37.80	27.09	33.42	45.52	38.62	56.78	61.59	39.13	48.32	59.89	45.33	52.56	45.50
LayerMoE (Ours)	36.86	27.44	33.25	46.15	39.05	56.09	61.50	39.11	47.77	61.89	46.67	56.89	**46.06**
Methods	**bn**	**hi**	**ne**	**bn**	**hi**	**ne**	**bn**	**hi**	**ne**	**bn**	**hi**	**ne**	***G2-avg***
Qwen1.5-1.8B	22.50	23.46	23.18	29.59	29.89	29.13	28.23	28.48	27.95	29.67	30.00	26.89	27.41
MoE-LPR (6*24)	24.64	25.26	21.64	30.29	30.57	31.01	29.90	32.78	30.08	33.22	35.22	33.11	29.81
MOLA	24.29	25.09	21.64	31.01	30.67	30.77	29.44	32.54	29.46	31.00	33.33	32.11	29.28
MoE-LPR (4*24)	23.52	24.91	21.39	30.73	30.54	30.94	30.02	32.55	29.45	32.11	34.33	32.67	29.43
MoE-LPR (3*24)	24.98	24.57	21.04	31.59	30.87	29.70	29.55	32.68	29.44	32.44	34.33	33.22	29.53
LayerMoE (Ours)	24.21	25.26	22.58	31.10	30.43	31.01	29.31	32.42	29.67	33.44	33.44	35.44	**29.86**

Table 8: The detailed results of Qwen1.5-1.8B under the G0→G2 setting for each language.

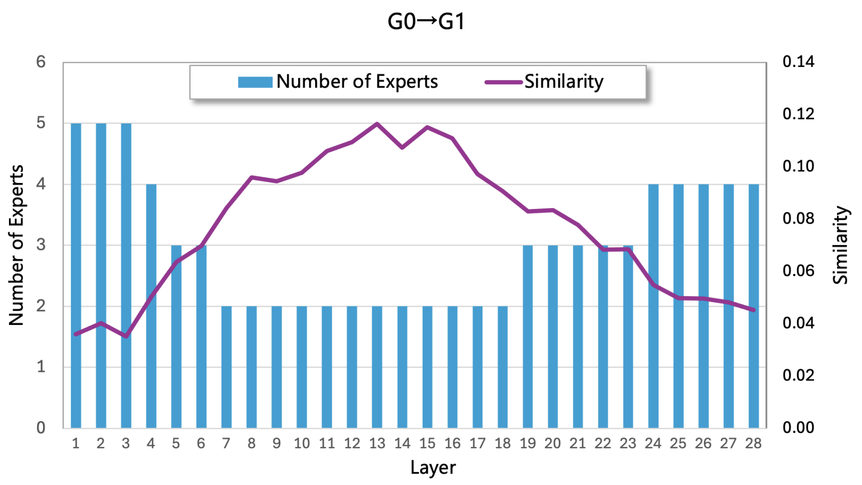


Figure 5: The detailed similarity and expert allocation of our method for each layer of Llama-3.2-3B under the G0 →G1 setting.

		ARC-Challenge			MMLU			Hellaswag			Belebele			
Methods	**Order**	**en**	**es**	**zh**	**en**	**es**	**zh**	**en**	**es**	**zh**	**en**	**es**	**zh**	***G0-avg***
Qwen1.5-1.8B	/	37.97	27.86	34.10	45.67	38.81	57.21	61.62	39.35	48.14	60.78	47.44	59.44	46.53
MoE-LPR (7*24)	G0→G1+G2	37.88	26.92	33.85	45.26	37.61	54.51	61.43	39.29	48.14	59.33	44.56	53.00	45.15
MoE-LPR (4*24)	G0→G1+G2	37.80	27.69	32.82	45.29	38.12	54.39	61.45	39.11	48.20	59.89	44.33	54.22	45.28
LayerMoE (Ours)	G0→G1+G2	37.46	27.01	34.27	45.63	38.14	54.18	61.37	39.08	47.86	60.11	46.22	55.44	**45.56**
MoE-LPR (7*24)	G0→G1→G2	37.54	26.84	33.16	45.71	38.29	55.42	61.43	39.21	48.12	61.22	47.33	56.33	45.88
LayerMoE (Ours)	G0→G1→G2	37.29	26.58	33.93	45.69	38.86	55.40	61.23	39.23	48.05	62.22	48.00	56.67	**46.10**
MoE-LPR (7*24)	G0→G2→G1	38.14	27.18	33.42	46.07	38.26	55.28	61.36	39.44	48.26	57.44	44.78	53.67	45.28
LayerMoE (Ours)	G0→G2→G1	38.31	27.09	34.70	46.19	38.79	55.85	61.19	39.05	48.05	61.89	46.44	57.44	**46.25**
Methods	**Order**	**el**	**hu**	**tr**	**el**	**hu**	**tr**	**el**	**hu**	**tr**	**el**	**hu**	**tr**	***G1-avg***
Qwen1.5-1.8B	/	22.95	23.20	22.63	28.94	32.25	30.37	29.15	29.63	29.04	33.33	32.00	33.44	28.91
MoE-LPR (7*24)	G0→G1+G2	26.46	28.00	27.92	30.38	35.14	34.64	36.40	39.01	37.92	35.33	39.00	39.78	34.17
MoE-LPR (4*24)	G0→G1+G2	26.20	27.65	27.92	31.01	35.08	34.37	35.60	38.14	37.11	34.11	41.11	40.56	34.07
LayerMoE (Ours)	G0→G1+G2	25.34	27.83	27.92	31.51	36.43	35.32	35.50	38.00	37.27	36.89	42.11	42.67	**34.73**
MoE-LPR (7*24)	G0→G1→G2	25.51	26.20	27.75	30.51	35.29	34.46	35.20	37.92	37.08	37.44	43.44	42.00	34.40
LayerMoE (Ours)	G0→G1→G2	25.34	25.77	27.75	31.94	36.58	35.09	35.17	37.92	36.24	37.34	43.34	42.18	**34.56**
MoE-LPR (7*24)	G0→G2→G1	25.34	26.97	26.47	30.76	34.41	34.47	35.11	37.42	37.09	33.67	38.44	40.89	33.42
LayerMoE (Ours)	G0→G2→G1	24.57	26.80	26.30	30.86	35.36	35.26	34.54	37.07	36.12	36.44	39.00	42.78	**33.76**
Methods	**Order**	**bn**	**hi**	**ne**	**bn**	**hi**	**ne**	**bn**	**hi**	**ne**	**bn**	**hi**	**ne**	***G2-avg***
Qwen1.5-1.8B	/	22.50	23.46	23.18	29.59	29.89	29.13	28.23	28.48	27.95	29.67	30.00	26.89	27.41
MoE-LPR (7*24)	G0→G1+G2	24.29	24.91	20.96	30.94	32.26	31.31	30.03	33.24	30.03	31.56	34.00	36.22	**29.98**
MoE-LPR (4*24)	G0→G1+G2	24.38	25.00	21.47	29.64	31.81	30.09	29.59	32.72	29.73	33.11	34.33	32.78	29.55
LayerMoE (Ours)	G0→G1+G2	23.27	25.26	21.47	31.23	32.05	30.60	29.67	32.34	29.66	32.67	36.00	35.00	29.94
MoE-LPR (7*24)	G0→G1→G2	23.44	25.34	21.81	31.66	32.31	31.41	29.55	32.60	29.54	34.22	35.44	35.78	30.26
LayerMoE (Ours)	G0→G1→G2	23.78	25.43	22.33	31.42	31.92	31.11	29.71	32.49	29.88	34.02	35.62	35.82	**30.29**
MoE-LPR (7*24)	G0→G2→G1	25.32	24.40	21.81	30.41	31.89	29.92	28.79	30.73	29.00	27.78	29.67	31.00	28.39
LayerMoE (Ours)	G0→G2→G1	24.98	25.17	22.67	31.75	29.95	30.41	29.25	32.49	29.46	32.56	35.11	33.78	**29.80**

Table 9: The detailed results of Qwen1.5-1.8B under the G0→G1+G2, G0→G1→G2, and G0→G2 →G1 settings for each language.

	ARC-Challenge			MMLU			Hellaswag			Belebele			
Methods	**en**	**es**	**zh**	**en**	**es**	**zh**	**en**	**es**	**zh**	**en**	**es**	**zh**	***G0-avg***
Llama-3.2-3B	51.11	43.59	39.23	56.42	49.09	44.00	76.32	61.50	52.87	74.22	68.56	68.67	57.13
MoE-LPR (3*24)	50.00	42.31	38.80	56.25	49.63	44.08	76.35	62.23	53.94	74.89	69.22	68.11	57.15
LayerMoE (Ours)	49.91	42.56	39.83	56.05	49.77	44.07	76.28	62.18	53.74	75.33	69.44	68.89	**57.34**
Methods	**el**	**hu**	**tr**	**el**	**hu**	**tr**	**el**	**hu**	**tr**	**el**	**hu**	**tr**	***G1-avg***
Llama-3.2-3B	31.76	34.85	33.56	41.08	43.01	42.37	43.37	44.65	42.85	65.22	59.67	59.22	45.13
MoE-LPR (3*24)	36.64	39.21	36.89	43.96	44.51	43.46	52.49	51.33	49.74	68.44	63.00	61.78	49.29
LayerMoE (Ours)	36.64	39.73	36.55	43.88	44.86	43.51	52.72	51.66	49.60	67.89	64.22	61.78	**49.42**

Table 10: The results of Llama-3.2-3B on ARC-Challenge, MMLU, Hellaswag, and Belebele under the G0→G1 setting for each language.

Methods	***en → zh***	***zh → en***	***en → es***	***es → en***	**old-avg**	***en → el***	***el → en***	***en → hu***	***hu → en***	***en → tr***	***tr → en***	**new-avg**
Llama-3.2-3B	84.36	83.95	84.79	86.4	84.88	82.47	85.86	83.81	86.15	78.97	86.53	83.96
MoE-LPR (3*24)	84.19	83.19	84.67	86.27	84.58	84.18	85.89	83.24	85.6	81.95	85.26	84.35
LayerMoE (Ours)	84.32	83.67	84.63	86.26	**84.72**	85.14	86.18	83.7	85.65	82.02	85.44	**84.69**

Table 11: The detailed results of Llama-3.2-3B on the FLORES benchmark under the G0→G1 setting for each language.