Breaking ReLU Barrier: Generalized MoEfication for Dense Pretrained Models

Jaeseong Lee[1], **Seung-won Hwang**[1]*, **Wonpyo Park**[2], **Mingi Ji**[2]
[1]Computer Science and Engineering, Seoul National University
[2]Google
{tbvj5914,seungwonh}@snu.ac.kr
{wppark,mingiji}@google.com

Abstract

As the scale of language models (LMs) continues to grow, there is a heightened interest in reducing the inference cost associated with these models. Mixture-of-Experts (MoEs) present an efficient alternative to dense models, while the existing methods to convert pretrained dense models to MoEs is limited to ReLU-based models with natural sparsity. This paper introduces G-MoEfication, applicable to arbitrary dense models, where ReLU-based activation sparsity assumptions no longer hold. For generalizations, we encounter the dilemma of needing to zero-out deactivated experts, while also avoiding excessive zeroing-out to retain dense activation information. We publicly release our code[1] and report results conducted with mBERT, SantaCoder-1.1B, Phi-2-2.7B, and Falcon-7B demonstrating the efficacy of our approach in general scenarios: from multitask to multilingual, from fine-tuning to zero-shot evaluation.

1 Introduction

Despite the remarkable performance exhibited by recent language models (LMs), the increasing scale of these models, as highlighted in (Zeng et al., 2023), leads to a growing interest in mitigating the inference cost associated with LMs. However, as many emerging LMs are dense (Chowdhery et al., 2022; Touvron et al., 2023), the substantial inference cost incurred by these models arises from the necessity to activate all parameters, leading to significant communication overhead because weights are sharded across multiple accelerators.

The Mixture-of-Experts (Shazeer et al., 2017) (MoE) emerges as a promising alternative. Unlike dense models, MoE sparsely activates a subset of parameters, referred to as *experts*. Moreover, only the accelerator associated with the selected experts is utilized, resulting in a remarkable reduction in communication overhead.

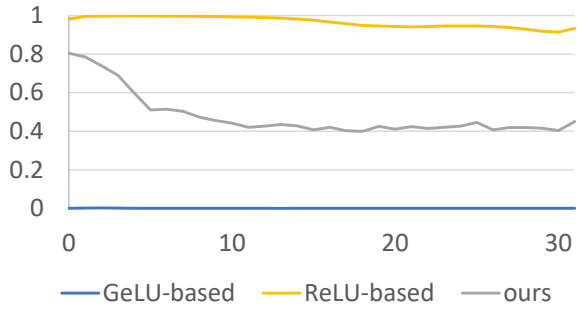

Figure 1: Sparsity[2] (y-axis) along the layer index (x-axis) of a GeLU-based PLM (Phi-2) and a ReLU-based PLM (OPT-2.7B).

Nevertheless, training a pretrained MoE model is computationally inefficient and environmentally detrimental (Zeng et al., 2023). Furthermore, MoE training presents unique challenges, such as the occurrence of *expert collapse*, where all experts function identically (Kaddour et al., 2023).

To avoid pretraining costs, this paper explores MoEfication, converting existing dense pre-trained language models into MoE versions while retaining the same parameters. This conversion, however, is targeted for ReLU-based models (Zhang et al., 2022). As ReLU is naturally sparse, as shown in the activation pattern of a ReLU-based model (Figure 1 yellow line), its conversion into an MoE is rather straightforward. Remarkably, this approach achieves 95% of the original performance while utilizing only a limited number of feed-forward network (FFN) parameters.

Unfortunately, there is a recent divergence in the trend of utilizing ReLU as an activation function. Instead, new models opt for smoother variants for better convergence, such as GeLU (Hendrycks and Gimpel, 2016), where the activation pattern is no longer sparse (Figure 1 blue line), thereby compli-

*Corresponding author
[1]https://github.com/thnkinbtfly/G-MoEfication

[2]We report (hard) sparsity for GeLU and ReLU, measuring the ratio of zero activations, which we later relax in Section 3.2.1 for ours (Figure 1 grey line).

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 10097–10107
November 12-16, 2024

cating MoEfication. A naïve approach would be, ReLUfying the LMs, by simply replacing the activation function with ReLU and further pretrain the model (Zhang et al., 2022; Piórczyński et al., 2023). However, this method incurs additional pretraining costs (Zhang et al., 2022), which are prohibitive given the prevailing trend of ever-growing LM scales (Zeng et al., 2023).

In this paper, we propose Generalized MoEfication (*G-MoEfication*) to **MoEfy arbitrary dense LMs, without pretraining**. Our primary challenge is addressing the conflicting goals associated with dense activations. With activations no longer sparse, we encounter the dilemma of needing to zero-out certain activations to deactivate specific experts, while also avoiding excessive zeroing-out to prevent substantial information loss in dense activations (Figure 1 blue line).

Our contribution lies in navigating this dilemma by introducing representative values for each feature (Figure 3b). Maintaining representative values instead of totally zeroing out would preserve information in activations. Simultaneously, our goal is to zero-out the residuals from the representative values, aiming to minimize information loss.

Desirably, representatives should avoid heavy multiplication costs, retaining optimality. First, to mitigate the heavy matrix multiplication, we ensure that the retained representatives remain static, and we generate the multiplication results as an embedding. During the inference step, this replaces the heavy matrix multiplicative operation with a lightweight vector additive operation (Figure 3c). Second, regarding optimality, we formulate a cost function to assess the sparsity of the residuals. Our aim is to calculate the optimal representative value that minimizes this measure.

Our contributions can be summarized as follows:

- We propose G-MoEfication, the first method to convert arbitrary dense PLMs into MoE without additional pretraining, to the best of our knowledge.
- We identify a fundamental dilemma of balancing the need to zero-out activations while preserving essential information. In response to this dilemma, we propose the retention of representatives for unselected experts.
- We evaluate on general scenarios –from multitask to multilingual, from fine-tuned to zero-shot evaluation, showing success in converting the given PLM into MoE. achieving performance levels exceeding the original MoEfication.
- Our selection of experts also aligns with human knowledge.
- We publicly release code.[1]

2 Related Work

2.1 MoEs

MoEs were initially introduced to develop multilayer networks with small modules, known as 'experts', each specialized in a subset of tasks (Jacobs et al., 1991). This approach has been promoted for scaling models with minimal computational overhead, treating different functional partitions of feed-forward networks (FFNs) as conditionally activated experts (Bengio, 2013). Then, it has been widely adopted in the NLP field from its first implementation for recurrent neural networks (Shazeer et al., 2017) to recent transformers (Lepikhin et al., 2021; Fedus et al., 2022). However, training MoE models from scratch can be unstable, often leading to expert imbalance or expert collapse (Shazeer et al., 2017; Kaddour et al., 2023), which poses challenges to their broader application.

2.2 Converting Dense Models to MoEs

To efficiently establish MoEs, researchers have probed methods to build MoEs from dense PLMs. Komatsuzaki et al. (2023) start from a given dense checkpoint, replicate the existing FFN parameters to build multiple experts, and continually pretrain the upcycled MoE. With less than 40% of the pretraining cost added, they obtain an MoE achieving better performance than the original dense model. However, the inference cost would not be reduced, since even using one expert, the upcycled MoE would need the same cost as the original dense model for inference. Llama-MoE (Zhu et al., 2024) continually pretrain the Llama to build an MoE with over 200B tokens. However, such continual pretraining would need tremendous cost considering the ever-growing scale of modern LMs (Zeng et al., 2023).

Alternatively, some researchers (Zhang et al., 2022; Piórczyński et al., 2023) investigated methods to convert the dense models to an MoE with the same scale. For example, Zhang et al. (2022) propose a method to cluster FFN parameters into several experts, and to train an expert selection module

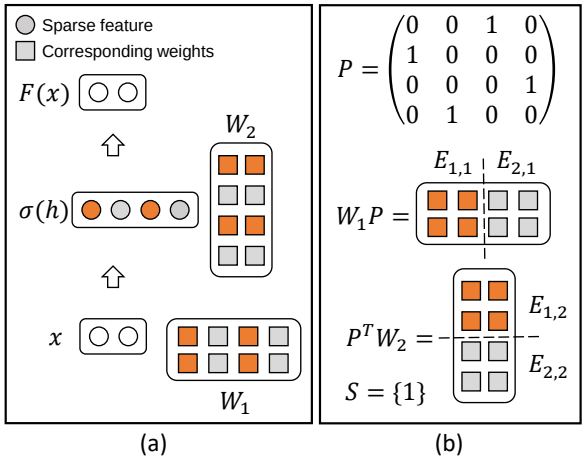

Figure 2: (a) When σ is ReLU, $\sigma(h)$ becomes sparse. The corresponding weights can be unactivated. (b) MoEfication aims to cluster such weights simultaneously unactivated. Adopted from Zhang et al. (2022).

to utilize a limited number of experts. However, these methods assume the PLM to be ReLU-based, thus to MoEfy the arbitrary PLM, they replace the activation function to ReLU and additionally pretrain the model. In contrast, our method is directly applicable to arbitrary models, without additional pretraining cost.

3 Proposed Method

We first formalize MoEfication under hard sparsity assumption (Section 3.1), then propose to overcome its limitation for generalization (Section 3.2).

3.1 Preliminaries: MoEfication

3.1.1 MoEfication under Hard Sparsity

Suppose the FFN layer in the given dense model is formulated as follows:

$$h = xW_1 + b_1 \tag{1}$$
$$F(x) = \sigma(h)W_2 + b_2 \tag{2}$$

where x is the input of the FFN layer, and σ is the activation function.

Meanwhile, an MoE is formulated as follows:

$$h_i = xE_{i,1} + e_{i,1} \tag{3}$$
$$F_{MoE}(x) = \left(\sum_{i \in S} \sigma(h_i)E_{i,2}\right) + b_2 \tag{4}$$

where $E_{i,1}, E_{i,2}, e_{i,1}$ are the weights and biases of experts. MoE typically utilizes a limited number of experts, and S denotes the indices of such selected experts.

The goal of MoEfication (Zhang et al., 2022) is approximating the given dense model (Eq. 1-2) as an MoE utilizing a limited number of experts (Eq. 3-4), while keeping the same total parameters. This goal is achieved through two stages. 1) *Expert construction*: W_1, W_2, b_1 are split into multiple experts, $E_{i,1}, E_{i,2}, e_{i,1}$ respectively. 2) *Expert selection*: Appropriately select S with a smaller size than the total number of experts.

3.1.2 Utilizing Sparsity to Construct Experts

The key idea of splitting the given dense FFN layer into multiple experts is grouping the neurons often activated simultaneously (Figure 2a orange dots). Since the neurons with similar vectors would show similar activation patterns, they utilize balanced K-means (Malinen and Fränti, 2014)[3] to cluster W_1 (Figure 2a orange weights). Figure 2b shows the MoEfication result with the clusters. Formally, with a permutation P representing clusters, $W_1P = \oplus E_{i,1}$, $b_1P = \oplus e_{i,1}$, and $P^TW_2 = \oplus E_{i,2}$, where $\oplus$ denotes the concatenation, which is defined as follows:

$$(\oplus(A, B))_{i,j} = \begin{cases} A_{i,j} & \text{if } j \le ncols(A) \\ B_{i,j} & \text{otherwise} \end{cases} \tag{5}$$

where $ncols$ denote the number of columns of given matrix. Concatenating several matrices are naturally generalized.

Then from Eq. 1-3, the real output $F(x)$ can be re-formulated as follows:

$$hP = xW_1P + b_1P \tag{6}$$
$$F(x) = \sigma(hP)P^TW_2 + b_2 \tag{7}$$
$$= (\sigma(x(\oplus E_{i,1}) + (\oplus e_{i,1}))) \oplus E_{i,2} + b_2 \tag{8}$$
$$= \left(\sum_i \sigma(h_i)E_{i,2}\right) + b_2 \tag{9}$$

Note that with Eq. 9, we do not need any permutation at the inference stage to make the same outputs as Eq. 2.

3.1.3 Expert Selection

To approximate the above formula into $F_{MoE}(x)$ (Eq. 4), how to determine S appropriately remains. With the intuition of unusing the experts with the sparsest activations (Figure 2 grey regions), they

[3]Their experiment with the best expert selection method reveals similar performance between various expert construction methods, thus we introduce only one of them.

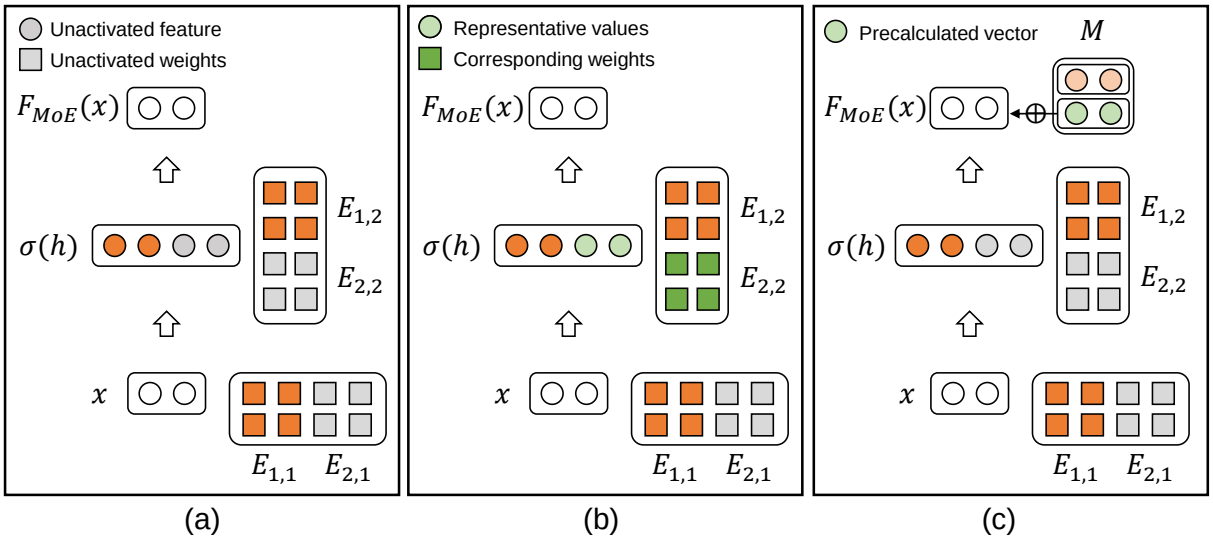

Figure 3: (a) Naïve MoEfication loses information. (b) Keeping representative values is beneficial, but hard to avoid the heavy computation of additional experts. (c) Saving precalculated vectors as embedding achieves sparse activation of experts, while recovering some information.

aim to minimize the difference between the real activation and the MoE activation as follows:

$$\min_S ||\sigma(h) - \oplus(\mathbb{1}(i \in S)\sigma(h_i))||_2 = \min_S ||\oplus(\mathbb{1}(i \notin S)\sigma(h_i))||_2 \quad (10)$$

where $\mathbb{1}$ denotes the indicator function, and $\oplus$ is concatenation as we defined above. To illustrate with Figure 2, if MLP selects $S = \{1\}$, $\mathbb{1}(i \in S)\sigma(h_i)$ denotes the features corresponding to the selected expert $i \in S = \{1\}$, the orange dots in Figure 2a. They train an MLP that selects S to minimize Eq. 10.[4] At the inference time, with the selected S from the MLP, Eq. 9 is approximated as Eq. 4.

When using σ as ReLU, $\sigma(h)$ becomes sparse (Figure 1 yellow line). Therefore, after minimization, the value of Eq. 10 would be small. Thus, the difference between the real output $F(x)$ (Eq. 9) and the MoE output $F_{MoE}(x)$ (Eq. 4).

$$F(x) - F_{MoE}(x) = \sum_{i \notin S} \sigma(h_i) E_{i,2} \quad (11)$$

would be marginal.

3.2 G-MoEfication

However, when we use arbitrary activation function σ, the assumption that $\sigma(h)$ is sparse no longer holds (Figure 1, blue line), in case of which zeroing-out activations would be detrimental to the given PLM.

To overcome this, we propose G-MoEfication in this section. We first relax the notion of sparsity for generalizing MoEfication (Section 3.2.1), based on which we propose the selection of representation value (Section 3.2.2) and expert (Section 3.2.3).

3.2.1 Soft Sparsity for MoEfication in General Activation Functions

According to hard sparsity (Figure 1 blue), GeLU would be hardly MoEfied. Meanwhile, sparsity s of vector $v \in \mathbb{R}^N$ can be straightforwardly generalized to treat activations with sufficiently small ϵ values as zero. Such softer sparsity and its significant benefits can also be found from prior work in a different problem context of network compression, which zeros out activations below a threshold (Kurtz et al., 2020), or removes entire layers with small activations (Yu et al., 2018; Lin et al., 2020; Tan and Motani, 2020). Activation quantization methods (Liu et al., 2024) can also be seen as a softer sparsity, quantizing near-zero activations as zero. We formally define soft sparsity as follows:

$$s(v) = \frac{\sum \mathbb{1}(|v_n| \leq \epsilon)}{N} \quad (12)$$

This notion subsumes hard sparsity when $\epsilon = 0$,[5] and we refer to generalized notion when we mention sparsity from this point on.

[4] We introduce only the best-performed selection method in Zhang et al. (2022).

[5] Figure 1 contrasts $\epsilon = 0$ (blue) and $\epsilon = 0.03$ (grey).

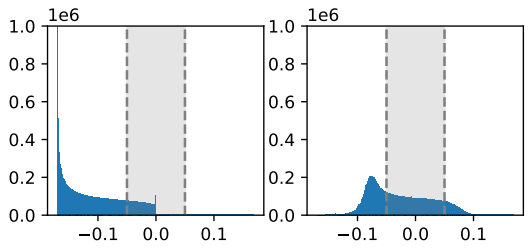

Figure 4: Activation distribution comparison of FFN in the 10th layer of mBERT. With the generalized notion of sparsity, we can consider whether the activation distribution is sparse or not, as the ratio of values in the grey region. We propose to shift the original distribution (left) using representative values (right).

3.2.2 Optimizing Sparsity with Representative Value

With this generalized notion of sparsity, Figure 4 (left) considers the ratio of activations contained in a grey region with ϵ boundary as sparse. If the ratio increases, the difference between real and MoE output, Eq. 11, would be more marginal.

For such increase, Figure 4 (right) suggests a distribution shift: Specifically, each activation value $\sigma(h)$ with a representative value $r(\sigma(h))$, is shifted by a mapping r, to be $\sigma(h) - r(\sigma(h))$. This can be implemented by keeping representative value $r(\sigma(h))$ while zeroing-out only the residual $\sigma(h) - r(\sigma(h))$, as in Figure 3b.

Maximal Sparsity of Residuals By keeping the representative value $r(\sigma(h_i))$ for unselected expert ($i \notin S$), the difference between real activation and the MoEfied activation changes as follows (cf. Eq. 10):

$$\begin{aligned}\min_S ||\sigma(h) - \oplus(\mathbb{1}(i \in S)\sigma(h_i) + \\ \mathbb{1}(i \notin S) r(\sigma(h_i)))||_2 \\ = \min_S || \oplus (\mathbb{1}(i \notin S)(\sigma(h_i) - r(\sigma(h_i))))||_2\end{aligned} \tag{13}$$

For ReLU-based models, as $\oplus\sigma(h_i)$ is assumed to be sparse, the value of Eq. 10 was assumed to be small (Zhang et al., 2022). Similarly, in our generalized notion of sparsity, we can make $\oplus(\sigma(h_i) - r(\sigma(h_i)))$ sparser, so that the assumption of small Eq. 13 value holds.

Toward this goal of optimizing sparsity of $\oplus(\sigma(h_i) - r(\sigma(h_i)))$, we first design a cost function l to measure how *sparse* the vector is. Since the value 0 means the maximal sparsity, $l(x)$ should be smaller as x gets closer to 0. We simply adopt $l(x) = ||x||_2^2$, which also aligns with Eq. 13.[6]

Therefore, $r(\sigma(h_i))$, the representative value as defined above, should be the value that minimizes the following formula:

$$\min_{\mathbf{r}} l(\oplus(\sigma(h_i) - r(\sigma(h_i)))) \tag{14}$$

$$= \min_{\mathbf{r}} \sum_i ||\sigma(h_i) - r(\sigma(h_i))||_2^2 \tag{15}$$

Computational Efficiency Although introducing representative values would reduce the information loss from MoEfying arbitrary PLMs, it incurs additional computation overhead (Figure 3b green regions) as follows. Introducing representative values change Eq. 4 as follows:

$$\begin{aligned}F_{MoE}(x) = \left(\sum_{i \in S} \sigma(h_i) E_{i,2}\right) + b_2 \\ + \left(\sum_{i \notin S} r(\sigma(h_i)) E_{i,2}\right)\end{aligned} \tag{16}$$

Eq. 16 introduces additional computation overhead, $\sum_{i \notin S} r(\sigma(h_i)) E_{i,2}$, compared with Eq. 4.

To avoid this heavy matrix-vector multiplication, we design $r(\sigma(h_i))$ to be a *static* vector r_i, and precalculate the multiplication results $m_i = r_i E_{i,2}$ (Figure 3c green dots). Then we introduce an embedding M that stores these results. In this way, Eq. 16 changes as follows:

$$\begin{aligned}F_{MoE}(x) = \left(\sum_{i \in S} \sigma(h_i) E_{i,2}\right) \\ + b_2 + \left(\sum_{i \notin S} m_i\right)\end{aligned} \tag{17}$$

which unactivates the heavy matrix multiplication (Figure 3c grey regions). Eq. 17 introduces only lightweight vector addition operations, $\sum_{i \notin S} m_i$, compared with Eq. 4 (Table 4).

Moreover, by designing r as a static vector, we can obtain a closed-form solution of Eq. 15 $r_i = \mu(\sigma(h_i))$, where $\mu(x)$ denotes the mean of the vector x.

To sum up, we collect the mean statistics of the activation $\sigma(h)$, and store the $\mu(\sigma(h_i))E_{i,2}$ vectors as an embedding. The compensation of unused experts could be calculated with a lightweight addition of vectors from the saved embedding. This

[6] Our preliminary experiment on multiPL-E Java shows choosing l2 norm (0.1377) outperforms l1 norm (0.0970).

	NER avg	POS avg	FLOPs
G-MoEfication (FFN 35%)	**83.85**	**89.77**	58%
G-MoEfication (FFN 50%)	**85.30**	**90.77**	68%
G-MoEfication (FFN 75%)	**86.46**	**91.55**	85%
MoEfication (FFN 75%)	82.79	89.63	85%
Oracle 100% (mBERT)	87.32	91.91	100%

Table 1: Averaged NER and POS F1 of 5 runs over total 42 languages. G-MoEfication outperforming the original MoEfication are bolded. We bolded the comparisons with the same number of FFN parameters in 1st column. At the last column, we report the relative number of FLOPs compared to the original mBERT.

simple yet effective solution is applicable to arbitrary activation functions– from static activation functions to adaptive activation functions (Farhadi et al., 2019).

3.2.3 Expert Selection

With r_i chosen as $\mu(\sigma(h_i))$, we can assume $\oplus(\sigma(h_i) - r_i)$ to be as sparse as possible, thus the minimum difference between the original output and output of MoE (Eq. 13) would be small. Now, we finally train an MLP that selects S to minimize Eq. 13.

4 Experiments

Models Drawing from prior research on MoEfying arbitrary PLMs (Zhang et al., 2022; Piórczyński et al., 2023), we adopt BERT (Devlin et al., 2019) architecture as our target model for MoEfication, which uses GeLU (Hendrycks and Gimpel, 2016) as the activation function. To explain expert selection aligns with linguistic genealogy, we opt for the multilingual variant of BERT. Moreover, we evaluate MoEficaiton on generation tasks, with SantaCoder-1.1B (Allal et al., 2023).

Unlike prior studies where experiments were invariably followed by a fine-tuning step (Zhang et al., 2022; Piórczyński et al., 2023), we disentangle the effect of fine-tuning from MoEfication in our study. This approach is crucial as parameter updates during fine-tuning might conceal potential performance drops from MoEfication, particularly in the context of recent trends favoring zero-shot evaluation scenarios. Therefore, we design an experiment without fine-tuning, and evaluate 0-shot performance on Phi-2-2.7B (Hughes, 2023) and Falcon-7B (Almazrouei et al., 2023).

Task Datasets and Languages for Evaluation We evaluate mBERT, on diverse in-language performance using XTREME benchmarks (Hu et al., 2020-07-13/2020-07-18; Ruder et al., 2021). We focus on the NER and POS tasks, since these are the tasks with the largest number of languages available in XTREME benchmarks. For reliable evaluation, we omit languages with an insufficient amount of train and test data.[7] We evaluate SantaCoder-1.1B on multiPL-E (Cassano et al., 2023) Java. We fine-tune with MegaCodeTraining set,[8], which we filter to contain Java codes only. To evaluate Phi-2 and Falcon-7B, we use the SuperGLUE benchmark (Wang et al., 2019), particularly with LM-EVALUATION-HARNESS (Gao et al., 2021).

Implementation Details For expert construction, we construct $K = 64$ experts, except for Falcon-7B, where we use $K = 128$. Following Zhang et al. (2022), for expert selection, we train 2-layered MLP with tanh activation function, with hidden and output units same as K, with Adam optimizer, learning rate of 10^{-2}, and batch size of 512, for 30 epochs.

To collect x and $\sigma(h)$ values for training MLP and calculating the representative value, we use Wikipedia articles extracted with WIKIEXTRACTOR.[9] For mBERT, we sample 200 examples with a sequence length of 128, spanning all languages on which mBERT was pretrained. For SantaCoder-1.1B, Phi-2 and Falcon-7B, with a sequence length of 1024, we sample 5000 examples from English Wikipedia for training MLP and to calculate the representative values.

The evaluation settings of mBERT largely follow Hu et al. (2020-07-13/2020-07-18). We fine-tune with a batch size of 32, learning rate of 2e-5, for 2 epochs, or at least 2500 iterations. For SantaCoder-1.1B, we use learning rate of 1e-5, batch size of 8. All fine-tuning and evaluation are conducted on RTX 3090.

[7] We choose languages with the number of train and test data examples larger than 100.

[8] huggingface.co/datasets/rombodawg/MegaCodeTraining

[9] https://github.com/attardi/wikiextractor

	SantaCoder		Phi-2		Falcon	
	Java	+params	S-GLUE	+params	S-GLUE	+params
G-MoEfication (ours)	**13.77%**	0.0041×	**0.632**	0.0021×	**0.512**	0.0029×
MoEficaion	8.44%	0.0038×	0.570	0.0019×	0.456	0.0028×
Oracle	17.81%	-	0.641	-	0.539	-

Table 2: SantaCoder evaluation on multiPL-E Java, Phi-2 and Falcon evaluation on zero-shot SuperGLUE. We also compare the added parameters for enabling MoEfication.

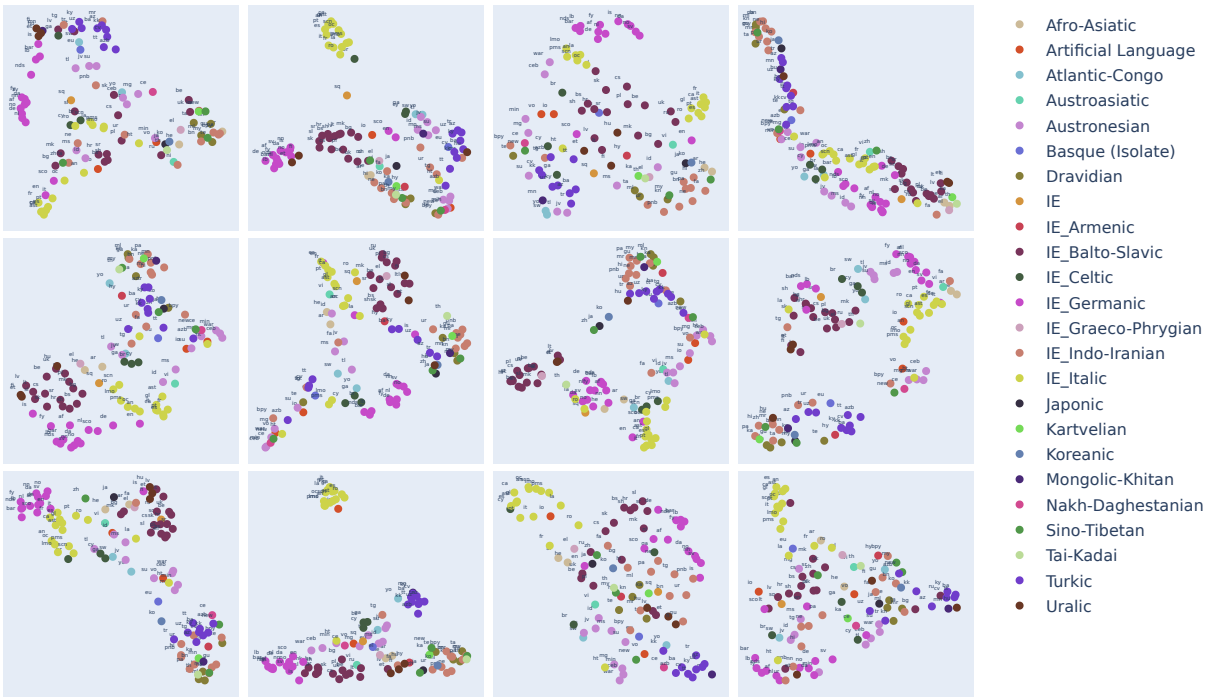

Figure 5: UMAP projection of expert selection pattern over 103 languages. Each color corresponds to a specific language family. We visualize expert selection of each layer; 1st layer (top left) to 12th layer (bottom right).

Baselines We compare G-MoEfication with the following baselines: a) *MoEfication:* We directly apply the conventional MoEfication (Zhang et al., 2022) with arbitrary activation functions intact. b) *Random expert:* We randomly choose and fix some experts, bypassing dynamic selection from the trained MLP. c) *Pruning (SNIP):* We utilize a popular training-free pruning method, SNIP (Lee et al., 2019) as an alternative to expert selection. d) *Oracle:* The original dense PLM serves as a reference point.

4.1 Experimental Results

RQ1: Effectiveness of G-MoEfication G-MoEfication successfully MoEfy the given PLM, mBERT (Table 1): We achieve above 96% of performance on both NER and POS, on average. For example, utilizing only 35% of FFN parameters, our MoEfied mBERT achieves 96% of performance on NER, and 97.7% of performance on POS, with only 58% of computation.

We emphasize that the original MoEfication (the fourth line in the tables) cannot achieve such performance. For example, even with more than twice our FFN parameters (FFN 75%), the conventional MoEfication is inferior to the performance of G-MoEfication (FFN 35%). SantaCoder-1.1B experiment with 85% of FFN (Table 2) shows that the trend is similar for code generation also.

Finally, we highlight that the complexity introduced by G-MoEfication is negligible compared to MoEfication, having similar number of FLOPs and added parameters (Table 1,2)

RQ2: G-MoEfication on Zero-Shot Scenarios In contrast to prior research (Zhang et al., 2022; Piórczyński et al., 2023), we isolate the impact of fine-tuning within MoEfication. Specifically, we apply MoEfication to Phi-2 and Falcon-7B, selecting experts with 85% of FFN parameters without any

	NER	POS
Random expert + rep. value	85.66	91.37
Pruning (SNIP) + rep. value	86.01	91.41
G-MoEfication	**86.46**	**91.55**

Table 3: Comparison of various FFN parameter selection methods on mBERT. We report the average scores over all languages.

	FLOPs
mBERT	100%
GMoE w/o static vector (FFN 35%)	101%
GMoE w/ static vector (FFN 35%)	58%

Table 4: mBERT FLOPs comparison of G-MoEfication with and without designing r as a static vector. We report the relative FLOPs compared with mBERT.

corpus	SuperGLUE
Wikipedia	0.632
Twitter	0.578
Ubuntu	0.578

(a)

corpus	Java
Wikipedia	13.8%
The Stack	13.4%

(b)

Table 5: Performance comparison varying corpus on (a) SuperGLUE evaluation on Phi-2-2.7B, and (b) multiPL-E Java evaluation on SantaCoder-1.1B.

additional parameter updates. Our MoEfied model consistently achieves over 95% performance on average, even with a limited number of experts (see Table 2). This performance surpasses that of the original MoEfication approach, which exhibits a significant decline in performance.

RQ3: Alignment of Expert Selection With Human Knowledge We explore whether the pattern of expert selection aligns with human knowledge, such as linguistic genealogy. We sample the selection frequencies of each expert given inputs of each language and project these vectors using UMAP (McInnes et al., 2018). Languages within the same linguistic family are colored identically.

As depicted in Figure 5, expert selection correlates with linguistic genealogy. Clusters of IE_Italic or Turkic languages in every layer. While we acknowledge that UMAP or visualization may have its limitations in representing human knowledge, this observation qualitatively suggests the alignment between expert clusters and human knowledge.

RQ4: Effectiveness of Expert Selection Method To validate the necessity of MoE architecture, we replace the expert selection module with two baselines: Random expert and SNIP Pruning (Lee et al., 2019). For SNIP, instead of dividing FFNs into multiple experts we directly pruned activations on selected channels. For a fair comparison, we apply our representative values to compensate the unused experts. We use 75% of FFN parameters.

As illustrated in Table 3, employing an expert selection module yields superior results compared to other baselines. This validates the necessity of MoE to attain such performance levels. Importantly, our approach also mitigated the occurrence of *expert collapse*, as evidenced by outperforming the random expert selection baseline.

RQ5: Computational Impact of r as Static Value A significant contribution of our work is the design of a representative value mapping r as a static vector, aimed at reducing computational costs. As illustrated in Table 4, this design choice has a substantial impact. Without such a design, the number of FLOPs slightly increases due to additional MLPs for expert selection. Conversely, by designing r as a static value, the number of FLOPs is nearly halved.

RQ6: Importance of General-Domain Corpus for r The precalculated static vectors and MLPs for expert selection are prepared once using the Wikipedia corpus. A natural question arises: Wouldn't it be better to utilize target domain-focused corpus? To address this question, we utilize 'The Stack' (Kocetkov et al., 2023), a collection of code corpora, to compare performance on a Java generation task. In addition, we consider Twitter and Ubuntu corpora (Xing et al., 2022).

The short answer is no. Table 5a shows that a utilizing domain-specific corpus hardly generalizes to tasks in different domains. In contrast, general-domain corpus such as Wikipedia shows decent performance. Moreover, even if the domain aligns well, such as 'The Stack' on Java generation task, the performance gain is not noticeable (Table 5b).

Overall, employing a general-domain corpus is

#samples	SuperGLUE
5000	0.632
1000	0.598
100	0.585

Table 6: SuperGLUE evaluation on Phi-2 varying the number of sample sizes to calculate r.

	NER avg	POS avg	FLOPs
G-MoEfication (FFN 35%)	83.68	88.57	58%
G-MoEfication (FFN 50%)	85.12	89.92	68%
Oracle (SiLU-mBERT)	86.87	91.15	100%

Table 7: Averaged NER and POS F1 of 5 runs over total 42 languages. At the last column, we report the relative number of FLOPs compared to the original SiLU-mBERT.

the most efficient solution, which generalizes to diverse tasks.

RQ7: Impact of Number of Samples for r Intuitively, as the number of samples to calculate r is increased, we would expect more accurate statistics, leading to further performance gains. Table 6 empirically validates such gains. Regarding the sample size, we empirically find using 5000 samples is satisfactory, though how to choose the number systematically is left as a future work.

RQ8: Experiments on Different Activation Functions The experiments above are done on GeLU-based models. We evaluate our method on another popular activation function, SiLU (Hendrycks and Gimpel, 2016; Elfwing et al., 2018). We change the activation function of mBERT to SiLU, then compare the G-MoEfication performance with original one. Table 7 shows that our method is successful on SiLU-based models.

5 Conclusion

This paper introduces G-MoEfication, a novel methodology applicable to arbitrary dense models, overcoming the limitations imposed by ReLU-based activation sparsity assumptions. Our technical contribution is identifying and overcoming the dilemma of zeroing out deactivated experts, while simultaneously preserving dense activation information to ensure optimal model performance.

Limitation

In this work, we focused on MoEfying given PLM as-is. However, the performance would more increase if we change the activation functions (Mirzadeh et al., 2024) or sparsification (Song et al., 2024). Nevertheless, this would require continual pretraining, needing much more computational cost than ours, which we leave as future work.

Acknowledgements

This research was partially supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2024-2020-0-01789) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation), MSIT/IITP grant (2022-0-00995, Automated reliable source code generation from natural language descriptions), and a gift grant from Google.

References

Loubna Ben Allal, Raymond Li, Denis Kocetkov, Chenghao Mou, Christopher Akiki, Carlos Munoz Ferrandis, Niklas Muennighoff, Mayank Mishra, Alex Gu, Manan Dey, Logesh Kumar Umapathi, Carolyn Jane Anderson, Yangtian Zi, Joel Lamy Poirier, Hailey Schoelkopf, Sergey Troshin, Dmitry Abulkhanov, Manuel Romero, Michael Lappert, Francesco De Toni, Bernardo García del Río, Qian Liu, Shamik Bose, Urvashi Bhattacharyya, Terry Yue Zhuo, Ian Yu, Paulo Villegas, Marco Zocca, Sourab Mangrulkar, David Lansky, Huu Nguyen, Danish Contractor, Luis Villa, Jia Li, Dzmitry Bahdanau, Yacine Jernite, Sean Hughes, Daniel Fried, Arjun Guha, Harm de Vries, and Leandro von Werra. 2023. SantaCoder: Don't reach for the stars!

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The Falcon Series of Open Language Models.

Yoshua Bengio. 2013. Deep learning of representations: Looking forward. In *International conference on statistical language and speech processing*, pages 1–37. Springer.

Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, Arjun Guha, Michael Greenberg, and Abhinav Jangda. 2023. MultiPL-E: A Scalable and Polyglot Approach to Benchmarking Neural Code Generation. *IEEE Transactions on Software Engineering*, 49(7):3675–3691.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. PaLM: Scaling Language Modeling with Pathways. *arXiv:2204.02311 [cs]*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. 2018. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11.

Farnoush Farhadi, Vahid Partovi Nia, and Andrea Lodi. 2019. Activation Adaptation in Neural Networks.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. 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, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. A framework for few-shot language model evaluation. Zenodo.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020-07-13/2020-07-18. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421. PMLR.

Alyssa Hughes. 2023. Phi-2: The surprising power of small language models.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and Applications of Large Language Models.

Denis Kocetkov, Raymond Li, Loubna Ben allal, Jia LI, Chenghao Mou, Yacine Jernite, Margaret Mitchell, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro Von Werra, and Harm de Vries. 2023. The Stack: 3 TB of permissively licensed source code. *Transactions on Machine Learning Research*.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. 2023. Sparse upcycling: Training mixture-of-experts from dense checkpoints. In *The Eleventh International Conference on Learning Representations*.

Mark Kurtz, Justin Kopinsky, Rati Gelashvili, Alexander Matveev, John Carr, Michael Goin, William Leiserson, Sage Moore, Nir Shavit, and Dan Alistarh. 2020. Inducing and Exploiting Activation Sparsity for Fast Inference on Deep Neural Networks. In *Proceedings of the 37th International Conference on Machine Learning*, pages 5533–5543. PMLR.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. 2019. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *International Conference on Learning Representations*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Zi Lin, Jeremiah Liu, Zi Yang, Nan Hua, and Dan Roth. 2020. Pruning Redundant Mappings in Transformer Models via Spectral-Normalized Identity Prior. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 719–730, Online. Association for Computational Linguistics.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2024. LLM-QAT: Data-Free Quantization Aware

Training for Large Language Models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 467–484, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Mikko I. Malinen and Pasi Fränti. 2014. Balanced K-Means for Clustering. In *Structural, Syntactic, and Statistical Pattern Recognition*, Lecture Notes in Computer Science, pages 32–41, Berlin, Heidelberg. Springer.

Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Seyed Iman Mirzadeh, Keivan Alizadeh-Vahid, Sachin Mehta, Carlo C del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. 2024. ReLU strikes back: Exploiting activation sparsity in large language models. In *The Twelfth International Conference on Learning Representations*.

Mikołaj Piórczyński, Filip Szatkowski, Klaudia Bałazy, and Bartosz Wójcik. 2023. Exploiting Transformer Activation Sparsity with Dynamic Inference.

Sebastian Ruder, Noah Constant, Jan Botha, Aditya Siddhant, Orhan Firat, Jinlan Fu, Pengfei Liu, Junjie Hu, Dan Garrette, Graham Neubig, and Melvin Johnson. 2021. XTREME-R: Towards More Challenging and Nuanced Multilingual Evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10215–10245, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Chenyang Song, Xu Han, Zhengyan Zhang, Shengding Hu, Xiyu Shi, Kuai Li, Chen Chen, Zhiyuan Liu, Guangli Li, Tao Yang, and Maosong Sun. 2024. ProSparse: Introducing and Enhancing Intrinsic Activation Sparsity within Large Language Models.

Chong Min John Tan and Mehul Motani. 2020. DropNet: Reducing Neural Network Complexity via Iterative Pruning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 9356–9366. PMLR.

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. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 294, pages 3266–3280. Curran Associates Inc., Red Hook, NY, USA.

Yujie Xing, Jinglun Cai, Nils Barlaug, Peng Liu, and Jon Atle Gulla. 2022. Balancing Multi-Domain Corpora Learning for Open-Domain Response Generation. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2104–2120, Seattle, United States. Association for Computational Linguistics.

Xin Yu, Zhiding Yu, and Srikumar Ramalingam. 2018. Learning Strict Identity Mappings in Deep Residual Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4432–4440, Salt Lake City, UT. IEEE.

Qingcheng Zeng, Lucas Garay, Peilin Zhou, Dading Chong, Yining Hua, Jiageng Wu, Yikang Pan, Han Zhou, Rob Voigt, and Jie Yang. 2023. GreenPLM: Cross-Lingual Transfer of Monolingual Pre-Trained Language Models at Almost No Cost. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 6290–6298, Macau, SAR China. International Joint Conferences on Artificial Intelligence Organization.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. MoEfication: Transformer Feed-forward Layers are Mixtures of Experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 877–890, Dublin, Ireland. Association for Computational Linguistics.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. 2024. LLaMA-MoE: Building Mixture-of-Experts from LLaMA with Continual Pre-training.