# On the Representation Collapse of Sparse Mixture of Experts

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Sparse mixture of experts provides larger model capacity while requiring a constant computational overhead. It employs the routing mechanism to distribute input tokens to the best-matched experts according to their hidden representations. However, learning such a routing mechanism encourages token clustering around expert centroids, implying a trend toward representation collapse. In this work, we propose to estimate the routing scores between tokens and experts on a low-dimensional hypersphere. We conduct extensive experiments on cross-lingual language model pre-training and fine-tuning on downstream tasks. Experimental results across seven multilingual benchmarks show that our method achieves consistent gains. We also present a comprehensive analysis on the representation and routing behaviors of our models. Our method alleviates the representation collapse issue and achieves more consistent routing than the baseline mixture-of-experts methods.

## 1 Introduction

Scaling up model capacities has shown to be a promising way to achieve better performance on a wide range of problems such as language model pre-training (Radford et al., 2019; Raffel et al., 2020), and visual representation learning (Dosovitskiy et al., 2021; Bao et al., 2022). Despite the effectiveness, increasing the number of parameters leads to larger computational cost, which motivates recent studies to explore Sparse Mixture-of-Experts (SMoE) models (Shazeer et al., 2017; Fedus et al., 2021; Lepikhin et al., 2021). SMoE increases the model capacity by building several sparsely-activated neural networks. With nearly constant computational overhead, SMoE models achieve better performance than dense models on various tasks, including machine translation (Lepikhin et al., 2021), image classification (Riquelme et al., 2021), and speech recognition (Kumatani et al., 2021).

The routing mechanism plays an important role in SMoE models. Given an input token, the router measures the similarity scores between each token and experts. Then we distribute tokens to the best-matched experts according to the routing scores. Recent studies explored various token assignment algorithms to improve SMoE training. For instance, Lewis et al. (2021) formulate SMoE routing as a linear assignment problem that globally maximizes token-expert similarities. Zhou et al. (2022) have experts selecting top tokens rather than assigning tokens to top experts. Roller et al. (2021) and Dai et al. (2022) propose to keep routing choices consistent. Many studies in recent years focus on how to design the token-expert assignment algorithm. In this paper, we present that current routing mechanisms tend to push hidden representations clustering around expert centroids, implying a trend toward representation collapse, which in turn harms model performance.

In order to alleviate the representation collapse issue, we introduce a simple yet effective routing algorithm for sparse mixture-of-experts models. More specifically, rather than directly using the hidden vectors for routing, we project the hidden vectors into a lower-dimensional space. Then, we

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

apply $L_2$ normalization to both token representations and expert embeddings, i.e., measuring routing scores on a low-dimensional hypersphere. Besides, we propose a soft expert gate with learnable temperature, which learns to control the activation of experts.

We evaluate the proposed method on cross-lingual language model pre-training and fine-tuning on downstream tasks. Experimental results show that our model consistently outperforms the baseline SMoE models in terms of both language modeling and fine-tuning performance. Moreover, analysis indicates that our method alleviates the representation collapse issue compared with the SMoE baseline. Our method also achieves more consistent routing behaviors during both pre-training and fine-tuning, which confirms the effectiveness of the proposed routing algorithm.

Our contributions are summarized as follows:

- We point out the representation collapse issue in sparse mixture-of-experts models, which is under-explored in previous work.
- We propose to estimate routing scores between tokens and experts on a low-dimensional hypersphere in order to alleviate representation collapse.
- We conduct extensive experiments on cross-lingual language model pre-training and fine-tuning on downstream tasks.
- We present a detailed analysis of routing behaviors and representation properties, which shows that our method improves performance and achieves more consistent routing.

# 2 Background

## 2.1 Sparse Mixture of Experts

Sparse Mixture-of-Experts (SMoE) models take advantage of conditional computation, and have shown to be a promising way to scale up the number of parameters. In this work, we consider SMoE for Transformers, where SMoE layers are inserted into neighboring Transformer blocks. Each SMoE layer consists of a router and several expert networks. Following most previous work (Fedus et al., 2021), we use feed-forward networks as experts, instead of self-attention modules.

For the input token $x$ with its hidden representation $\boldsymbol{h} \in \mathbb{R}^d$, the router computes the routing score between $\boldsymbol{h}$ and the $i$-th expert by a dot-product similarity metric $s_i = \boldsymbol{h} \cdot \boldsymbol{e}_i$, where $\boldsymbol{e}_i \in \mathbb{R}^d$ is a learnable expert embedding, and $d$ is the hidden size of the model. Then, the router utilizes a sparse gating function $g(r)$ to make the expert network conditionally activated.

In this paper, we use top-1 routing, i.e., only the expert with the largest routing score is activated. Formally, considering a SMoE layer with $N$ experts, the forward function of SMoE can be written as:

$$k = \arg\max_i s_i = \arg\max_i \boldsymbol{h} \cdot \boldsymbol{e}_i \tag{1}$$

$$f^{\text{SMoE}}(\boldsymbol{h}) = \boldsymbol{h} + g(s_k) f_k^{\text{FFN}}(\boldsymbol{h}) \tag{2}$$

where $f_k^{\text{FFN}}(\cdot)$ stands for the $k$-th expert network that is implemented as stacked feed-forward networks. Moreover, we explore both softmax gating (Lepikhin et al., 2021; Fedus et al., 2021) and sigmoid gating (Lewis et al., 2021; Dai et al., 2022) for the function $g(s_k)$:

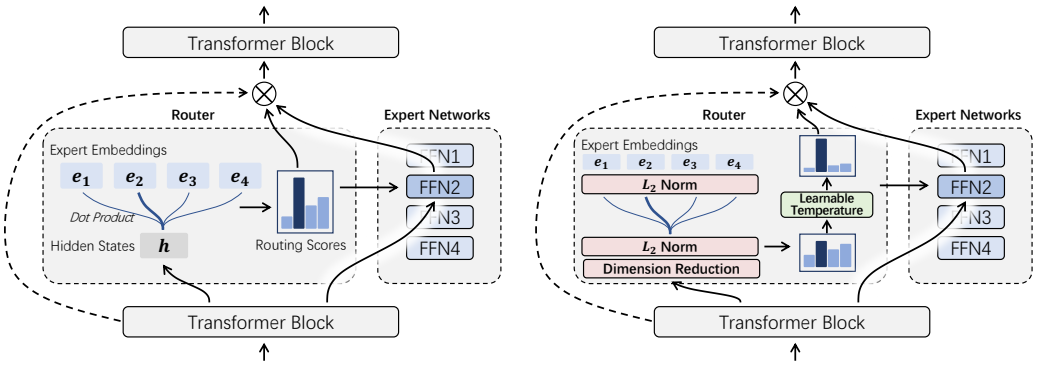
$$g(s_k) = \begin{cases} \exp(s_k) / \sum_{j=1}^{N} \exp(s_j), & \textit{softmax gating} \\ \sigma(s_k), & \textit{sigmoid gating} \end{cases}, \tag{3}$$

where $\sigma(\cdot)$ is the $\mathrm{sigmoid}$ function.

## 2.2 Representation Collapse of Sparse Mixture-of-Experts

We present how representation collapse happens in sparse mixture-of-experts models. For convenience, we use $\boldsymbol{h}' = f^{\text{SMoE}}(\boldsymbol{h})$ to denote the output of the SMoE layer as in Equation (2), $S_k = g(s_k)$ to denote the $k$-th output of the softmax function, and $\boldsymbol{h}^{\text{FFN}} = f_k^{\text{FFN}}(\boldsymbol{h})$ to denote the output of the $k$-th expert network. The Jacobian matrix with respect to $\boldsymbol{h}$ is given by:

$$\boldsymbol{J} = \boldsymbol{J}_1 + \boldsymbol{J}_2 = (\boldsymbol{I} + S_k \boldsymbol{J}^{\text{FFN}}) + \sum_{j=1}^{N} S_k (\delta_{kj} - S_j) \boldsymbol{h}^{\text{FFN}} \boldsymbol{e}_j^\top, \tag{4}$$




(a) Sparse Mixture-of-Experts (SMoE) Layer

(b) X-MoE Layer (Ours)

Figure 1: Illustration of a typical SMoE layer and the proposed X-MoE layer. (a) An SMoE layer consists of a router and expert networks, where the experts are sparsely activated according to dot-product token-expert routing scores. (b) X-MoE improves the routing algorithm via dimension reduction, $L_2$ normalization, and gating temperature.

where $\delta_{kj}$ is a Kronecker delta. The equation means that the Jacobian matrix can be decomposed into two terms. The first term $\boldsymbol{J}_1$ represents producing a better token representation given the current activation $S_k$. The second term $\boldsymbol{J}_2$ means to learn better gating function for appropriate activation score $S_k$. After back-propagation, the gradient is received from the above two paths, written as $\nabla_{\boldsymbol{h}}\mathcal{L} = \boldsymbol{J}_1^\top \nabla_{\boldsymbol{h}'}\mathcal{L} + \boldsymbol{J}_2^\top \nabla_{\boldsymbol{h}'}\mathcal{L}$. The second term can be expanded as:

$$\boldsymbol{J}_2^\top \nabla_{\boldsymbol{h}'}\mathcal{L} = \sum_{j=1}^{N} S_k(\delta_{kj} - S_j)(\boldsymbol{h}^{\text{FFN}\top} \nabla_{\boldsymbol{h}'}\mathcal{L})\boldsymbol{e}_j = \sum_{j=1}^{N} c_j \boldsymbol{e}_j, \tag{5}$$

where $c_j = S_k(\delta_{kj} - S_j)(\boldsymbol{h}^{\text{FFN}\top} \nabla_{\boldsymbol{h}'}\mathcal{L})$. The above equation indicates that *the token representation $\boldsymbol{h}$ tends to be updated toward a linear combination of the expert embeddings*.

We consider that such behavior potentially harms the representation capacity of Transformers. Firstly, consider that the $N$ expert vectors can span a $N$-dimensional space at most via linear combinations. As $N$ is much smaller than the hidden size $d$ in practice, the spanning subspace does not fully utilize the entire available capacity. Thus, the mechanism renders the Transformer hidden vector $\boldsymbol{h}$ collapsed to an $N$-dimensional subspace, implying a trend toward representation collapse from $\mathbb{R}^d$ to $\mathbb{R}^N$ where $N \ll d$ in practice. Secondly, Equation (5) indicates that the hidden vector $\boldsymbol{h}$ tends to be similar to the expert embedding that it is routed to. If the hidden states were routed to the same expert, they are going to be pushed closer. However, we would like to encourage the representations more diverse, so that they can be more expressive and discriminative. The phenomenon possibly restricts the expressibility of hidden states, especially when an expert is inclined to dominate routing.

# 3 Methods

We introduce the routing algorithm for sparse mixture of experts, which measures the routing scores between tokens and experts on a low-dimensional hypersphere. As shown in Figure 1b, we address the representation collapse issue of SMoE by applying dimensionality reduction and $L_2$ normalization for the token representations and expert embeddings. Then, we describe how to incorporate the routing algorithm into an SMoE model under the pre-training-then-fine-tuning paradigm.

## 3.1 Routing Algorithm

**Dimension Reduction** In order to alleviate the representation collapse issue mentioned in Section 2.2, we represent the expert embedding $\boldsymbol{e}_i$ and the token vector $\boldsymbol{h}$ in a low-dimensional space instead of the original high-dimensional hidden space. Specifically, we first parameterize the experts with lower-dimensional embeddings $\boldsymbol{e}_i \in \mathbb{R}^{d_e}$ such that $d_e$ is much smaller than the Transformer hidden size $d$. Next, we conduct a projection over the hidden states $f^{\text{proj}}(\boldsymbol{h})$, which projects $\boldsymbol{h}$ to the expert embedding space. We use a linear projection $f^{\text{proj}}(\boldsymbol{h}) = \boldsymbol{W}\boldsymbol{h}$ such that $\boldsymbol{W} \in \mathbb{R}^{d_e \times d}$.

Thus, the routing scoring function between the tokens and experts can be written as $s_i = (\boldsymbol{W}\boldsymbol{h}) \cdot \boldsymbol{e}_i$. Typically we set $d_e = N/2$ (i.e., half of the number of experts) in our implementation.

Inspired by Jing et al. (2022), dimension reduction mitigates the issues described in Section 2.2 from two perspectives. First, linear projection $\boldsymbol{W}\boldsymbol{h}$ isolates the direct interaction between hidden vector $\boldsymbol{h}$ and expert embedding $\boldsymbol{e}_i$, which tends to relieve cascaded collapse for representations. Second, it is natural to apply a low-rank projector for hidden vectors, as the number of experts is usually much smaller than the hidden size of Transformers. Hence the reduced dimension better fits in with the low-rank nature of routing.

**$L_2$ Normalization** After dimension reduction, we apply $L_2$ normalization to both token representations and expert embeddings. Our routing score is defined as:

$$s_i = \frac{(\boldsymbol{W}\boldsymbol{h}) \cdot \boldsymbol{e}_i}{\|\boldsymbol{W}\boldsymbol{h}\| \|\boldsymbol{e}_i\|}, \tag{6}$$

where $\| \cdot \|$ is $L_2$ normalization. Thus, the resulting representations are transformed into a certain scale with stabilized routing scoring.

As described in Section 2.2, if an expert dominated a set of hidden states, the representations were pushed toward the expert embedding. In order to fully utilize the space, we favor larger uniformity of representations while avoiding dominated experts. Given a hidden vector $\boldsymbol{h}$, the dot-product routing $s_i = (\boldsymbol{W}\boldsymbol{h}) \cdot \boldsymbol{e}_i$ is affected by both $\|\boldsymbol{e}_i\|$ and $\cos(\boldsymbol{W}\boldsymbol{h}, \boldsymbol{e}_i)$. So some experts are allocated with more tokens because of larger values of $\|\boldsymbol{e}_i\|$. In contrast, $L_2$ normalization projects vectors on the unit hypersphere, which suppresses the undesired effect of $\|\boldsymbol{e}_i\|$. The visualization in Figure 2b also confirms that our method improves the uniformity of learned representations.

**Gating with Learnable Temperature** In addition, we add a learnable temperature scalar $\tau$ in the SMoE gating function $g(s_k)$. Because $L_2$ normalization rescales the routing scores $s_k$ to the range $[-1, 1]$, directly using the scores for SMoE gating tends to make expert activation too conservative. The introduced temperature enables the router to adjust the gating $g(s_k)$ accordingly. To be more specific, our gating function is:

$$g(s_k) = \begin{cases} \frac{\exp(s_k/\tau)}{\sum_{j=1}^{N} \exp(s_j/\tau)}, & \textit{softmax gating} \\ \sigma(s_k/\tau), & \textit{sigmoid gating} \end{cases}, \tag{7}$$

where $\sigma(\cdot)$ is the $\mathrm{sigmoid}$ function, and the temperature scalar $\tau$ is learnable.

### 3.2 Training Objective

The training objective is jointly minimizing the loss of the target task and an auxiliary load balancing loss (Fedus et al., 2021). The load balancing loss is separately computed for each router. For each router, given the frequency $t_i$ of how many tokens are routed to the $i$-th expert and the routing score $s_i$, the load balancing loss is computed via:

$$\mathcal{L}^{\text{balance}} = \frac{N}{|\mathcal{B}|} \sum_{i=1}^{N} \sum_{\text{token} \in \mathcal{B}} t_i \frac{\exp(s_i/\tau_0)}{\sum_{j=1}^{N} \exp(s_j/\tau_0)}, \tag{8}$$

where $N$ is the number of the experts, $\mathcal{B}$ is a batch of training examples, $|\mathcal{B}|$ is the number of tokens, and $\tau_0$ stands for a constant temperature. Different from the learnable $\tau$ in Equation (7), $\tau_0$ is kept fixed during training. The overall training objective is to minimize:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \alpha \mathcal{L}^{\text{balance}}, \tag{9}$$

where $\alpha$ is a coefficient for load balancing . The term $\mathcal{L}_{\text{task}}$ is determined by the specific task that Transformer learns. For example, we employ the masked language modeling loss (Devlin et al., 2019) for pre-training, and the sequence-to-sequence learning objective for neural machine translation.

### 3.3 Frozen Routing During Fine-tuning

We evaluate SMoE under the pre-training-then-fine-tuning paradigm in our work. During fine-tuning, we freeze all the parameters of experts, including both the router and expert networks. Because

Table 1: Evaluation results on the cross-lingual XTREME benchmark. The models are fine-tuned on the English training data and directly evaluated in all target languages. SMoE models are grouped according to the choice of gating function. The results are averaged over five runs.

| **Model** | **Structured Prediction** | | **Question Answering** | | | **Classification** | | **Avg** |
|---|---|---|---|---|---|---|---|---|
| | POS | NER | XQuAD | MLQA | TyDiQA | XNLI | PAWS-X | |
| Metrics | F1 | F1 | F1 / EM | F1 / EM | F1 / EM | Acc. | Acc. | |
| Dense (without SMoE) | 70.0 | 61.1 | 67.3 / 51.1 | 58.7 / 41.1 | 42.1 / 28.3 | 70.1 | 84.1 | 61.4 |
| *With* softmax *gating* | | | | | | | | |
| SMoE Baseline | 70.1 | 60.9 | 71.3 / 55.2 | 62.8 / 44.8 | 50.9 / 34.5 | 71.5 | 84.6 | 63.8 |
| X-MoE (Ours) | **70.8** | **63.2** | **72.4 / 56.2** | **64.5 / 46.3** | **53.7 / 38.1** | **72.0** | **85.2** | **65.3** |
| *With* sigmoid *gating* | | | | | | | | |
| SMoE Baseline | 70.6 | 61.2 | 71.5 / 55.7 | 63.2 / 45.3 | 50.1 / 35.1 | 71.2 | 85.1 | 64.1 |
| X-MoE (Ours) | **71.1** | **62.7** | **72.3 / 56.3** | **64.3 / 46.0** | **51.5 / 36.6** | **72.2** | **85.2** | **65.0** |

the fine-tuning datasets are usually small compared with pre-training corpora. We find that SMoE models tend to overfit downstream tasks, which often leads to inconsistent routing. Freezing SMoE parameters helps to relieve the above issues. Notice that we still use load balancing loss although the routers are kept fixed, which empirically improves fine-tuning performance in our experiments.

# 4 Experiments

We conduct experiments on cross-lingual language model pre-training (Devlin et al., 2019). We evaluate the performance by fine-tuning the pretrained models on various downstream benchmarks. We also compare validation losses of the masked language modeling task. Our method is named as X-MoE in the following sections.

## 4.1 Experimental Setup

**Pre-training Data** Following (Chi et al., 2021), we use the combination of CCNet (Wenzek et al., 2019) and Wikipedia dump as pre-training corpora. We sample sentences in 94 languages from the corpora, and employ a re-balanced distribution introduced by Conneau and Lample (2019), which increases the probability of low-resource languages.

**Model Architecture and Hyperparameters** We construct our X-MoE models using the Transformer (Vaswani et al., 2017) encoder (L = 12, H = 768, A = 12) with the vocabulary provided by Conneau et al. (2020) as the backbone architecture. Following Lewis et al. (2021), we build a 32-expert sparse layer with 3 FFN sub-layers, and insert it after the 6-th Transformer layer. The routing dimension $d_e$ is set as 16. The gating temperature $\tau_0$ is set as 0.3 and 0.07 for the softmax gate and sigmoid gate, respectively. The detailed hyperparameters of X-MoE models can be found in Appendix A. X-MoE models are pretrained with the Adam optimizer ($\beta_1$ = 0.9, $\beta_2$ = 0.98) using a batch size of $2,048$ for 125K steps. The pre-training procedure takes 2 days on 2 Nvidia DGX-2 Stations. Appendix B and Appendix C provide the detailed hyperparameters for X-MoE pre-training and fine-tuning.

**Baselines** We consider two baselines in our experiments. (1) **Dense** is a dense Transformer encoder without sparsely-activated modules. (2) **SMoE** is our implementation of Switch Transformers (Fedus et al., 2021). The SMoE baseline is built with the same setting with X-MoE. In addition to its original softmax-gating implementation, we also implement a sigmoid-gating (Lewis et al., 2021; Dai et al., 2022) variant of Switch Transformers as a baseline approach. Notice that the baseline models are pretrained with the same training data as X-MoE for a fair comparison.

## 4.2 Downstream Evaluation

We conduct a downstream evaluation on seven widely-used cross-lingual understanding benchmarks from XTREME (Hu et al., 2020). Specifically, we conduct experiments on Universal Dependencies v2.5 part-of-speech tagging (Zeman et al., 2019), WikiAnn named entity recognition (Pan et al., 2017;

Table 2: Results of upstream evaluation. We report the validation perplexities on masked language modeling.

| Model | Perplexity |
|---|---|
| Dense (without SMoE) | 23.51 |
| *With* softmax *gating* | |
| SMoE Baseline | 19.02 |
| X-MoE (Ours) | **18.72** |
| *With* sigmoid *gating* | |
| SMoE Baseline | 19.59 |
| X-MoE (Ours) | **19.12** |

Table 3: Ablation studies of X-MoE components. The models employ various combinations of dimension reduction, $L_2$ normalization, and frozen routing. Average fine-tuning results of five random seeds are reported.

| Dim. Red. | $L_2$ Norm | Frozen | XNLI | MLQA |
|---|---|---|---|---|
| ✓ | ✓ | ✓ | **72.2** | **64.3 / 46.0** |
| ✗ | ✓ | ✓ | 71.5 | 63.4 / 45.2 |
| ✓ | ✗ | ✓ | 71.4 | 63.0 / 45.2 |
| ✓ | ✓ | ✗ | 71.7 | 63.9 / 45.8 |
| ✗ | ✗ | ✓ | 71.6 | 63.6 / 45.5 |
| ✗ | ✗ | ✗ | 71.2 | 63.2 / 45.3 |

Rahimi et al., 2019), natural language inference (XNLI; Conneau et al. 2018), paraphrase adversaries from word scrambling (PAWS-X; Yang et al. 2019), and question answering on MLQA (Lewis et al., 2020), XQuAD (Artetxe et al., 2020), and TyDiQA-GoldP (Clark et al., 2020). Among the benchmarks, we adopt the cross-lingual transfer setting, where the models are fine-tuned with the training data in English and evaluated in all target languages.

Table 1 presents the evaluation results on the seven downstream tasks from the XTREME benchmark. For each task, the results are first averaged among the test languages and then averaged over five random seeds. Overall, the softmax-gating X-MoE model obtains the best performance, achieving an average score of 65.3. Comparing SMoE models with the dense model, SMoE models show notable improvement, indicating that SMoE models benefit from the large model capacity. Comparing X-MoE with the two SMoE baselines, it shows that X-MoE models provide consistent gains on downstream tasks, demonstrating the effectiveness of our proposed routing algorithm.

### 4.3 Upstream Evaluation

We compare the pretrained models for the upstream performance by the validation perplexity on masked language modeling (MLM). We sample multilingual sentences from mC4 (Xue et al., 2020), and construct an MLM validation dataset that contains $65,536$ sequences with lengths around $512$.

The results are shown in Table 2. Similar to the downstream results, we observe that SMoE models perform better than the dense model. In terms of the SMoE models, X-MoE models with both softmax and sigmoid gating achieve lower masked language modeling perplexities than their counterparts. Among all the pretrained models, the softmax-gating X-MoE the achieves the lowest validation perplexity. The results show that our method not only works well for learning transferable text representations for downstream tasks, but also brings improvements to the upstream masked language modeling task. Comparing the upstream results with the downstream results, it shows that achieving a lower upstream perplexity does not promise better downstream performance. For instance, the sigmoid-gating X-MoE model has larger perplexity than the softmax-gating SMoE baseline has, but outperforms the fine-tuning performance of the baseline on the downstream tasks.

### 4.4 Ablation Studies

**Routing Algorithm** To better understand our routing algorithm, we pretrain several variants of sigmoid-gating X-MoE models with various combinations of dimension reduction (Dim. Red.), $L_2$ normalization ($L_2$ Norm), and routing frozen (Frozen). For a fair comparison, all the models are pretrained and fine-tuned under the same setup, i.e., training data, steps, and the random seeds. We evaluate the models on XNLI and MLQA, and report the results in Table 3. Jointly using the three routing methods achieves the best performance. When ablating one of the three routing methods, the model performs less well, demonstrating that X-MoE benefits from all the three components.

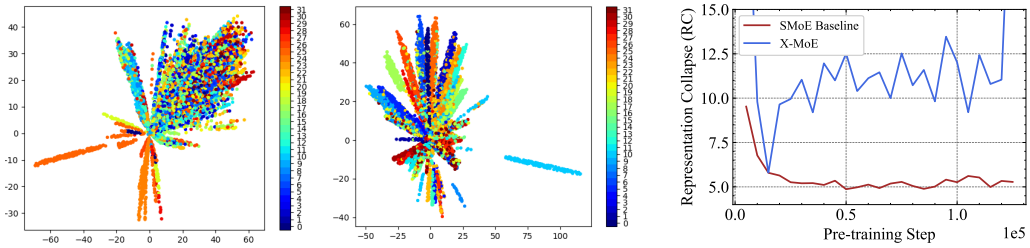
**Dimension of Expert Embedding** We conduct experiments by adjusting the routing dimension for dimensionality reduction. Specifically, we compare sigmoid-gating X-MoE models with routing

Table 4: Comparison of routing dimensions for dimensionality reduction. '$N$' stands for the number of experts. We report the average results of five random seeds on the XNLI and MLQA benchmarks.

| Routing Dimension | XNLI | MLQA |
|---|---|---|
| $N/4$ | 71.4 | **64.3 / 46.4** |
| $N/2$ | **72.2** | **64.3** / 46.0 |
| $N$ | 71.7 | 63.8 / 45.9 |
| $2N$ | 71.7 | 62.7 / 44.8 |
| $4N$ | 71.2 | 63.1 / 45.0 |

Table 5: Effects of load balancing during fine-tuning. The models are fine-tuned with various weights for the auxiliary load balancing loss. We report the validation accuracy for XNLI and F1 for MLQA with five random seeds.

| Weight | XNLI | MLQA |
|---|---|---|
| 0 | 71.71 | 64.57 |
| $10^{-3}$ | 71.65 | 64.40 |
| $10^{-2}$ | **71.93** | **64.59** |
| $10^{-1}$ | 71.68 | 64.50 |




(a) SMoE Baseline (b) X-MoE (Ours) (c) RC Through Pre-training

Figure 2: Analysis on the representation collapse of the Transformer hidden states. Figure (a) and (b) visualize the spatial structure of the experts. Each data point represents a token to be routed, and its color stands for the expert that it is assigned to. Figure (c) presents the curves of representation collapse (RC), which measures the within-class variability of hidden states. Larger RC values indicate less collapse.

dimensions of $N/4$, $N/2$, $N$, $2N$, and $4N$, where $N$ is the number of the experts. Table 4 shows the downstream performance. It shows that using the routing dimension of $N/2$ provides the best performance for XNLI and $N/4$ is the best for MLQA. The results also confirm that dimension reduction better fits in with the low-rank nature of SMoE routing.

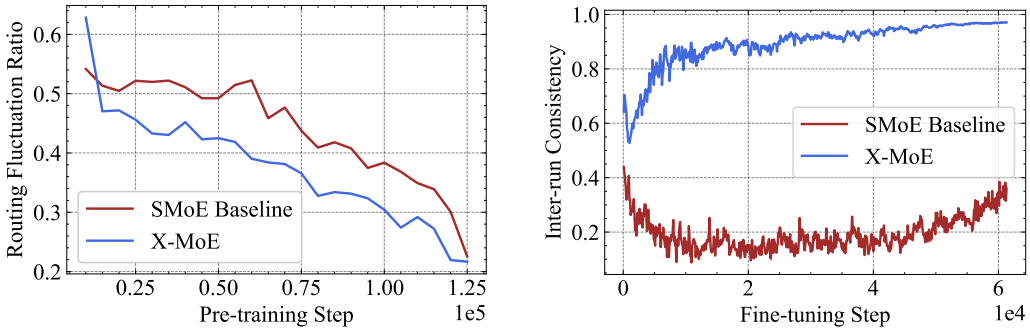
**Load Balancing During Fine-tuning** We explore whether load balancing is beneficial for fine-tuning SMoE models. To this end, we add load balancing loss to the total loss with various weights when fine-tuning X-MoE models on XNLI and MLQA. Table 5 shows the average validation scores where we search the load balancing coefficient $\alpha$ ranging from 0 to $10^{-1}$. We observe that using balance loss during fine-tuning is slightly beneficial for X-MoE. When removing the balance loss, X-MoE still remains comparable results on both XNLI and MLQA.

### 4.5 Analysis

**Representation Collapse** We qualitatively analyze the representation collapse issue by visualizing the experts. Figure 2a and 2b illustrate the spatial structure of the experts of SMoE baseline and X-MoE in hyperbolic space, which is produced by Uniform Manifold Approximation and Projection (UMAP; McInnes et al. 2018) with n-neighbor of 100 and min-dist of 1. Each data point represents a token to be routed, where we use the hidden states for SMoE baseline and the projected token representations for X-MoE. Each color stands for an expert that the tokens are assigned to.

Figure 2a shows that most of the points are mixed together with a large amount of available room unused, which suggests a representation collapse in the expert embedding space. In contrast, X-MoE in Figure 2b shows a well-organized feature space with clear distinctions between clusters. It indicates that our routing methods successfully project the tokens to the expert embedding space with routing features preserved.

Additionally, we conduct quantitative analysis on the degree of representation collapse for the learned Transformer hidden states that are fed into SMoE routing. We use the representation collapse metric




(a) Routing Fluctuation Ratio Through Pre-training

(b) Inter-run Consistency Through Fine-tuning

Figure 3: The routing behaviors of SMoE baseline and X-MoE. (a) Routing fluctuation (RF) ratio measures the ratio of the tokens that change their target experts between two checkpoints. Smaller RF values indicate more stable routing. (b) Inter-run consistency measures the correlation among the token assignments of various fine-tuning runs. Larger values indicate more consistent routing.

proposed in (Zhu et al., 2021). Given the representations to be measured, we use $\mathbf{\Sigma}_W$ and $\mathbf{\Sigma}_B$ to denote the within-class and between-class covariance matrices, respectively. The representations collapse (RC) metric is calculated via:

$$\text{RC} = \text{Tr}(\mathbf{\Sigma}_W \mathbf{\Sigma}_B^{\dagger}), \tag{10}$$

where $\mathbf{\Sigma}_B^{\dagger}$ is the pseudo inverse of $\mathbf{\Sigma}_B$. Smaller RC values indicate representation collapse to a greater extent. Figure 2c illustrates the metrics during pre-training, where the data is sampled from the validation set mentioned in Section 4.3. SMoE baseline is unlike unconstrained feature models that can empirically collapse to almost zero RC, but still shows a consistent descending trend through pre-training, implying a trend toward representation collapse. Differently, X-MoE obtains larger RC scores than SMoE baseline with uptrend through pre-training.

**Routing Consistency Through Pre-training** We examine whether our proposed routing algorithm achieves more consistent routing through training. We measure the routing consistency via the routing fluctuation (RF) ratio metric. Routing fluctuation is defined as the change of the target expert of an input token. Correspondingly, the RF ratio measures the ratio of RF between the current and the last checkpoints for the same input. A lower RF ratio indicates better routing consistency. As shown in Figure 3a, we present the RF ratio on the MLM validation set mentioned in Section 4.3. After the 15K step, X-MoE shows a much lower RF ratio than the SMoE baseline, indicating that our model produces more consistent routing behaviors.

**Inter-run Consistency Through Fine-tuning** In the experiments of the downstream evaluation, we find that the routing behaviors of SMoE baseline models can be sensitive to random seeds. As the learned token assignments are various for different training data orders, the final downstream performance can be diverse among runs. Therefore, we study the routing behaviors of the SMoE baseline and X-MoE models through fine-tuning. To achieve this, we develop a metric, named *inter-run consistency*, which measures how closely the token assignments converge among the runs with different seeds. Considering a model with $N$ experts, let $\boldsymbol{l} = [n_1, ..., n_N]$ denote the total load of the experts, where $n_i$ stands for the number of the tokens that are assigned to the $i$-th expert. Given two loads $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ from two runs with different seeds, the similarity between $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ is defined as the Pearson correlation coefficient (PCC) between them, which is denoted as $\rho_{\boldsymbol{l}_1,\boldsymbol{l}_2}$. Here PCC only serves as a similarity metric rather than measuring linear correlation between variables. By extending it to $m$ runs with different seeds for each run, we define the inter-run consistency as the average of correlation matrix $\text{IC} = \sum_{i,j\in\{1...m\}} \rho_{\boldsymbol{l}_i,\boldsymbol{l}_j}/m^2$.

We fine-tune X-MoE and SMoE baseline models on XNLI for 12 runs separately. Then we compute the inter-run consistency for every 100 mini-batches, i.e., the expert loads are accumulated for 100 steps. Figure 3b illustrates the inter-run consistency. The SMoE baseline converges toward different routing solutions across multiple runs of fine-tuning, even though the only difference between runs is the random seed. In comparison, X-MoE obtains substantially better inter-run consistency than the

SMoE baseline. The curve of X-MoE indicates that the models have various routing behaviors at the beginning of the fine-tuning, but finally converge to almost the same routing behaviors.

## 5 Related Work

**SMoE for Large-Scale Models** Sparse Mixture-of-Experts (SMoE) models are introduced by Shazeer et al. (2017), which extends mixture of experts (Jacobs et al., 1991; Jordan and Jacobs, 1994) with conditional computation (Bengio et al., 2013; 2015) techniques. Taking advantage of computational computation, SMoE enables a massive increase in model capacity while maintaining computational efficiency. To explore the potential of SMoE, recent studies apply SMoE in a wide range of machine learning problems such as machine translation (Lepikhin et al., 2021), image classification (Riquelme et al., 2021), speech recognition (Kumatani et al., 2021). In addition to the supervised learning scenario, there has been work on exploring SMoE under the pre-training-fine-tuning paradigm, and observing discrepancies between strong pre-training quality and poor fine-tuning performance (Fedus et al., 2021; Artetxe et al., 2021; Zoph et al., 2022). Besides, the scaling behaviors of SMoE are also studied (Clark et al., 2022; Du et al., 2021).

**SMoE Routing Algorithms** Many recent studies explore the token assignment algorithms for SMoE routing. BASE layers (Lewis et al., 2021) formulate the token routing problem as a linear assignment problem. Hash Layers (Roller et al., 2021) employ a parameter-free assignment algorithm that routes tokens by hashing. Zhou et al. (2022) let each expert select top-k tokens rather than distribute tokens to experts. Dai et al. (2022) propose to freeze the routing function in order to relieve routing fluctuation. These methods focus on the assignment algorithm in routing, but our routing algorithm focuses on improving the underlying routing scoring metric, which is still under-explored.

**Representation Collapse** Representation collapse, also termed neural collapse, is the degeneration of the representations during the training of neural networks. Several studies observe that the within-class variation of the representations in classification networks becomes negligible at the terminal phase of training (Papyan et al., 2020; Zhu et al., 2021; Tirer and Bruna, 2022). Besides, this phenomenon has also been observed in language model fine-tuning (Aghajanyan et al., 2021), and visual representation learning (Chen and He, 2021; Ermolov et al., 2021; Jing et al., 2022). These studies focus on densely-activated neural networks. In this work, we point out the representation collapse issue in SMoE models.

## 6 Conclusion

In this work, we point out the representation collapse issue in sparse mixture-of-experts (SMoE) models, and propose a routing algorithm that estimates the routing scores on a low-dimensional hypersphere. We conduct extensive experiments on cross-lingual language model pre-training. Experimental results across various benchmarks demonstrate that our method brings consistent improvements over SMoE baselines in terms of both language modeling and fine-tuning performance. Besides, our method alleviates the trend toward representation collapse and achieves more consistent routing. We are going to improve the work from the following perspectives. First, most current X-MoE experiments are conducted on language tasks, such as multilingual language model pre-training, and machine translation. We will also evaluate the proposed method on the pre-training of vision Transformers (Bao et al., 2022). Second, we would like to report the results of scaling up model size. The performance gain tends to be greater with a larger number of experts.

**Ethical Considerations** One of the negative societal impacts of training large-scale models is the high computational and environmental cost. Our paper focuses on improving SMoE, which is usually more efficient than dense model training with the same number of parameters. So better SMoE algorithms potentially save required computation and lessen CO2 emissions from computing. Moreover, X-MoE improves multilingual pre-training and fine-tuning, so that we can better transfer cross-lingual knowledge from high- to low-resource languages. The bless of larger model size brought by SMoE reduces the parameter conflicts of multilinguality, while keeping the computation cost manageable.

## References


Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. Better fine-tuning by reducing representational collapse. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=OQ08SN70M1V.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.421. URL https://www.aclweb.org/anthology/2020.acl-main.421.

Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, et al. Efficient large scale language modeling with mixtures of experts. *arXiv preprint arXiv:2112.10684*, 2021.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=p-BhZSz59o4.

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

Zewen Chi, Li Dong, Bo Zheng, Shaohan Huang, Xian-Ling Mao, Heyan Huang, and Furu Wei. Improving pretrained cross-lingual language models via self-labeled word alignment. 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 3418–3430, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.265. URL https://aclanthology.org/2021.acl-long.265.

Aidan Clark, Diego de las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. *arXiv preprint arXiv:2202.01169*, 2022.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470, 2020. doi: 10.1162/tacl_a_00317. URL https://www.aclweb.org/anthology/2020.tacl-1.30.

Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 7057–7067, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/c04c19c2c2474dbf5f7ac4372c5b9af1-Abstract.html.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1269. URL https://www.aclweb.org/anthology/D18-1269.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, pages 8440–8451. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.747. URL https://doi.org/10.18653/v1/2020.acl-main.747.

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. StableMoE: Stable routing strategy for mixture of experts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7085–7095, Dublin, Ireland, May 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.acl-long.489.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, NAACL-HLT 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. *arXiv preprint arXiv:2112.06905*, 2021.

Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3015–3024. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/ermolov21a.html.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. *arXiv preprint arXiv:2003.11080*, 2020.

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

Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=YevsQ05DEN7.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://www.aclweb.org/anthology/D18-2012.

Kenichi Kumatani, Robert Gmyr, Felipe Cruz Salinas, Linquan Liu, Wei Zuo, Devang Patel, Eric Sun, and Yu Shi. Building a great multi-lingual teacher with sparsely-gated mixture of experts for speech recognition. *arXiv preprint arXiv:2112.05820*, 2021.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. {GS}hard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qrwe7XHTmYb.

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

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. MLQA: Evaluating cross-lingual extractive question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7315–7330, Online, July 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.acl-main.653.

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

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1178. URL https://www.aclweb.org/anthology/P17-1178.

Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40): 24652–24663, 2020.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019. URL http://www.persagen.com/files/misc/radford2019language.pdf.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Afshin Rahimi, Yuan Li, and Trevor Cohn. Massively multilingual transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1015. URL https://www.aclweb.org/anthology/P19-1015.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34, 2021.

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

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Tom Tirer and Joan Bruna. Extended unconstrained features model for exploring deep neural collapse. *arXiv preprint arXiv:2202.08087*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzman, Armand Joulin, and Edouard Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. *arXiv preprint arXiv:1911.00359*, 2019.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. 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 3687–3692, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1382. URL https://www.aclweb.org/anthology/D19-1382.

Daniel Zeman, Joakim Nivre, Mitchell Abrams, and et al. Universal dependencies 2.5, 2019. URL http://hdl.handle.net/11234/1-3105. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Yanqi Zhou, Tao Lei, Han-Chu Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing. *arXiv preprint arXiv:2202.09368*, 2022.

Zhihui Zhu, Tianyu DING, Jinxin Zhou, Xiao Li, Chong You, Jeremias Sulam, and Qing Qu. A geometric analysis of neural collapse with unconstrained features. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=KR0DJAa6pzE.

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

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code will be made publicly available upon the publication of the paper.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See the experimental setup and the appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

# A Model Hyperparameters

Table 6 presents the model hyperparameters of X-MoE. The gating temperature $\tau_0$ is initialized as 0.3 and 0.07 for the softmax gating and sigmoid gating, respectively. We use the same vocabulary as XLM-R (Conneau et al., 2020) with 250K subwords tokenized by SentencePiece (Kudo and Richardson, 2018).

Table 6: Model hyperparameters of X-MoE.

| Hyperparameters | Value |
|---|---|
| FFNs in X-MoE layer | 3 |
| Number of experts | 32 |
| Expert embedding dimension | 16 |
| Initialized gating temperature $\tau_0$ | 0.3 / 0.07 |
| Transformer blocks | 12 |
| Hidden size | 768 |
| FFN inner hidden size | 3,072 |
| Attention heads | 12 |

# B Hyperparameters for Pre-training

Table 7 presents the hyperparameters for pre-training.

Table 7: Hyperparameters for pre-training.

| Hyperparameters | Value |
|---|---|
| Optimizer | Adam |
| Training steps | 125,000 |
| Batch tokens per task | 1M |
| Adam $\epsilon$ | 1e-6 |
| Adam $\beta$ | (0.9, 0.98) |
| Maximum learning rate | 5e-4 |
| Learning rate schedule | Linear decay |
| Warmup steps | 10,000 |
| Weight decay | 0.01 |
| Transformer dropout | 0.1 |
| X-MoE dropout | 0 |
| Load balancing coefficient | 1e-2 |

# C Hyperparameters for Fine-tuning

Table 8 presents the hyperparameters for fine-tuning.

Table 8: Hyperparameters for fine-tuning on the XTREME downstream tasks.

| Hyperparameters | POS | NER | XQuAD | MLQA | TyDiQA | XNLI | PAWS-X |
|---|---|---|---|---|---|---|---|
| Batch size | 8 | 8 | 32 | 32 | 32 | 32 | 32 |
| Learning rate | {1,2,3}e-5 | {5,...,9}e-6 | {2,3,4}e-5 | {2,3,4}e-5 | {2,3,4}e-5 | 2e-5 | {1,2}e-5 |
| LR schedule | Linear decay | Linear decay | Linear decay | Linear decay | Linear decay | Linear decay | Linear decay |
| Warmup | 10% | 10% | 10% | 10% | 10% | 20% | 10% |
| Weight decay | 0 | 0 | 0 | 0 | 0 | 1e-2 | 0 |
| Epochs | 10 | 10 | {2,3,4} | {2,3,4} | {10,20,40} | 5 | 10 |
| Load balancing | 0,1e{-3,-2,-1} | 0,1e{-3,-2,-1} | 0,1e{-3,-2,-1} | 0,1e{-3,-2,-1} | 0,1e{-3,-2,-1} | 0,1e{-3,-2,-1} | 0,1e{-3,-2,-1} |