Mixture of Languages: Improved Multilingual Encoders Through Language Grouping

João Maria Janeiro[1,2,*] Belen Alastruey[1,3,*] Francisco Massa[1]
Maha Elbayad[1] Benjamin Piwowarski[2] Patrick Gallinari[2,4] Loïc Barrault[1]
[1]FAIR at Meta [2]Sorbonne Université, CNRS, ISIR, F-75005 Paris, France
[3]Université Paris Dauphine - PSL, Paris [4]Criteo AI Lab, Paris, France
{joaojaneiro,alastruey}@meta.com

Abstract

We propose Mixture of Languages (MoL), a new strategy to pretrain largely multilingual encoders. Recent work in this field has relied on training transformer encoders on a large amount of multilingual data, with all parameters shared across all languages, without studying how to optimally balance language transfer and interference to achieve better performance. To address this, MoL proposes to group languages based on their similarity, and add parallel, sparsely activated layers that process each group independently. This architecture allows MoL to boost language transfer while minimizing interference, without increasing the active parameter count. We show that MoL largely outperforms a dense counterpart trained with the same configuration, as well as MoE models and public multilingual encoders such as XLM-R or mBERT on downstream tasks.

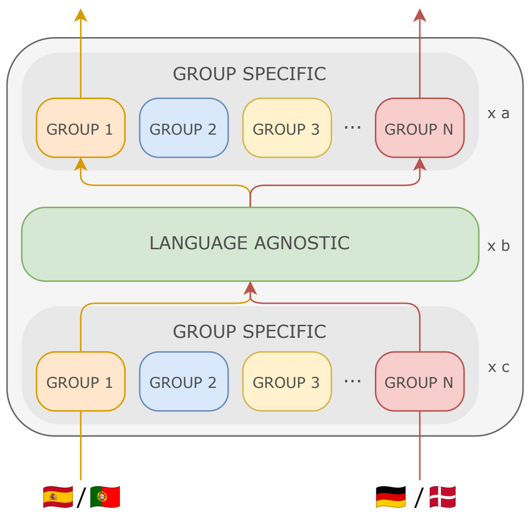


Figure 1: Architecture of MoL, a multilingual encoder that combines language agnostic layers with group specific layers. Each group consists of a set of similar languages, that are processed together in a parallel layer.

1 Introduction

Learning contextualized text representations has been extensively explored, notably by BERT (Devlin et al., 2019), which introduced the idea of training Transformer encoders using the masked language modeling (MLM) objective. Some variants of this strategy have been proposed, most notably ELECTRA (Clark et al., 2020) or XLM (Lample and Conneau, 2019), however, MLM has prevailed as the standard pretraining strategy for encoders.

Recently, monolingual encoders have received increasing interest, marked by the releases of ModernBERT (Warner et al., 2024) and NeoBERT (Breton et al., 2025). However, largely multilingual encoders have not received the same attention, with XLM-R (Conneau et al., 2020) and mBERT (Devlin et al., 2019) still being the most widely used encoders in this category.

Significant progress has been made in finetuning large multilingual encoders, with examples including MEXMA (Janeiro et al., 2025), mE5 (Wang et al., 2024), and Jina-Embeddings-V3 (Sturua et al., 2024). These efforts have explored various architectures, improved data filtering methods, and loss variants during the finetuning stage.

In stark contrast, recent research has not explored different pretraining strategies specifically designed for large multilingual encoders. Instead, the predominant approach in previous work has been to aggregate vast amounts of multilingual data and train a dense transformer model, often without explicitly addressing the challenges and implications of language sharing and interference.

Understanding these language interactions is crucial for effective multilingual representation learning. Studies investigating multilingual models have provided valuable insight in this direction. For instance, research on encoder-decoder models for Machine Translation (MT) (Shaham et al., 2023), as well as studies on various encoder architec-

[*] Equal contribution.

tures (Conneau et al., 2020; Alastruey et al., 2025), have consistently shown that while beneficial transfer occurs between similar languages, detrimental interference between dissimilar ones can hinder learning, especially in low-resource scenarios.

Previous research has explored the use of modular and sparse architectures and applied them to different NLP tasks. This includes the use of language-specific modules or adapters for multilingual MT architectures (Bapna et al., 2019; Purason and Tättar, 2022; Pfeiffer et al., 2023), the well-established Mixture-of-Experts (MoE) approach (Shazeer et al., 2017; Lepikhin et al., 2020) to train MT systems (Elbayad et al., 2023; NLLB Team, 2024) or decoder-only LLMs (Du et al., 2022; Jiang et al., 2024), and a hybrid of the two with language-aware routing (Zhang et al., 2021). To the best of our knowledge, none of these techniques has been studied in the context of pretraining encoder-only models.

In this work, we develop a novel modular and sparse design for pretraining largely multilingual encoders that aims for a balance between language interference and language transfer. Instead of a single multilingual model prone to interference, or a collection of independent monolingual models that scale poorly, we propose a middle ground solution with Mixture of Languages (MoL). As depicted in Figure 1, MoL is an encoder featuring a combination of shared and group-specific layers. Languages are grouped based on similarity criteria, detailed in Section 3.4. Languages within the same group are processed together in their dedicated specific layers. This architecture allows MoL to maintain the same active parameter count as a standard dense model during inference, while benefiting from language sharing and minimizing harmful interference.
Our main contributions include:

- A novel approach to pretrain large multilingual encoders using parallel layers with language grouping, proven more effective in balancing language transfer and interference (Section 3).

- A comprehensive exploration of language grouping methods and a systematic framework for architecture selection guided by language identification accuracy (Sections 3.4 and 3.5).

- The proposal of MoL, a new architecture leveraging parallel group-specific layers. MoL achieves state-of-the-art performance compared to dense, MoE, and existing public models trained on similar pretraining tasks, while maintaining a constant active parameter count (Section 4).

- Evidence demonstrating that MoEs are unsuitable for large-scale multilingual encoder pretraining (Section 5).

2 Related Work

The field of multilingual NLP has seen significant advancements with the development of large, fully shared multilingual encoders like mBERT (Devlin et al., 2019) and XLM-RoBERTa (Conneau et al., 2020). These models establish a successful recipe for learning multilingual representations by pretraining a fully shared encoder on large, mixed-language corpora using self-supervised objectives. However, a key challenge in such *all-is-shared* multilingual models is the phenomenon known as the "curse of multilinguality" (Conneau et al., 2020), where performance starts to degrade as more languages are added to a fixed-size model due to increased interference.

Although simply increasing the size of the model can sometimes mitigate this issue, it leads to higher computational costs and slower inference. This limitation motivates the exploration of alternative architectures that can handle a growing number of languages more efficiently, often through modularity or sparsity. These alternatives aim to increase total model capacity while keeping the number of active parameters for any given input fixed (thus, inference FLOPs remain the same). Our proposed MoL falls into this category with the use of explicit modular and group-specific layers. Below, we discuss related work that explores different facets of modularity and sparsity, particularly in the context of multilinguality.

Implicit Sparsity and Mixture-of-Experts (MoE). The most common way to keep the number of active parameters fixed with increased model capacity is MoE (Shazeer et al., 2017), typically employing top-k (often k=2 (Lepikhin et al., 2020) or k=1 (Fedus et al., 2022)) sparsely activated experts. While superficially similar to our MoL, MoE relies on learned routing mechanisms that often require auxiliary losses, such as load balancing (Lepikhin et al., 2020), to ensure balanced expert utilization and prevent collapse to a single expert. Training MoE models effectively is known to be challenging due to stability issues and difficulties in enforcing balanced expert exploration (Zoph et al., 2022; Cai et al., 2025; Lepikhin et al., 2020). In contrast, our approach achieves expert exploration naturally through language grouping.

Moreover, standard MoE routing is typically done at token-level, directing tokens within a sentence to different experts. Our approach, however, routes all tokens within a given sentence to the same expert, responsible for handling the group the language of the sentence belongs to. This fundamental difference, coupled with the load balancing requirement, which does not easily accommodate the language resource variability in a multilingual context, makes standard MoEs less suitable for explicit language separation.

MoE have been used extensively for Machine Translation (Elbayad et al., 2023; Zhao et al., 2024; Li et al., 2023). However, to the best of our knowledge, MoEs have seen limited success or exploration in the pretraining of text encoder-only models; Nussbaum and Duderstadt (2025) proposed the first encoder-only MoE, but MoE was only used during contrastive finetuning, while Videau et al. (2024) showed MoE performing poorly for encoder-only models in vision.

Explicit Language-Specific layers. Another line of research introduces explicit modular components dedicated to specific languages, tasks or capabilities. Adapters (Houlsby et al., 2019; He et al., 2021) are a common example in encoder-only models, that add a small number of parameters typically on top of an existing, commonly frozen, pretrained model. Adapters are usually finetuned for specific downstream tasks or languages. This differs from MoL, which explores optimizing performance through better pretraining of the core encoder architecture itself. Additionally, adapters do increase the active parameter count, whereas our design keeps the active parameter count fixed.

In MT, there has been extensive exploration of parallel language-specific layers (Pfeiffer et al., 2023; Qu et al., 2025; Purason and Tättar, 2022), with some exploration on the encoder side (Qu et al., 2025). These works demonstrate the benefits of language-specific or group-specific parameters for improving MT performance. However, much of this exploration has centered on the decoder and on language-specific modules without grouping, which scales poorly. There has been less work exploring parameters specific to groups of languages or focusing specifically on the pretraining of a general-purpose multilingual encoder.

Our contribution in context. Building on the insights from prior work on mitigating language interference and exploring modular and sparse architectures, our work investigates the use of explicit parallel layers dedicated to groups of languages for the pretraining of a large multilingual encoder-only model. We address the scalability challenge of language-specific parameters by focusing on optimized groups and offer a compelling alternative to the training complexities and routing limitations of MoEs.

3 Mixture of Languages (MoL)

In Mixture-of-Languages (MoL), languages are partitioned into specific groups. Each group has its own dedicated layers that exclusively process inputs from its language group (see Figure 1). This is similar to language-specific layers, but differs in that it groups languages for efficiency.

By forming these groups in a guided manner, MoL is able to promote language sharing by grouping together languages that benefit one another, while simultaneously reducing interference by splitting dissimilar languages into different groups. Furthermore, this approach inherently maintains a reasonable memory footprint and a constant number of active parameters, making it more scalable than traditional dense methods.

In this section, we first examine how language interference affects the performance of encoder-only models and how grouping similar languages together improves results (3.3). We then study different criteria for grouping languages (3.4) and finally explore various architectures with different distributions of shared and group-specific layers (3.5).

3.1 Experimental Configuration

All models in the paper are trained on the NLLB data (NLLB Team, 2024), on the selected languages, for 3 epochs. We limit the number of sentences per language to 25M maximum, 5M minimum and every language in between is left as is. The synthetic data from NLLB is used for languages with human data below 5M sentences.

We use the XLM-R tokenizer and fix a maximum sequence length for our input data to 256 tokens.

	Swapped Languages					
Setting	**heb_Hebr**	**dan_Latn**	**deu_Latn**	**spa_Latn**	**zsm_Latn**	**arb_Arab**
Multilingual	37.76	44.67	49.02	53.90	48.25	36.41
Family Groups	**39.45**	**45.86**	**52.39**	**55.41**	**49.75**	**39.18**
Family Groups w/Interference	38.25	43.73	50.63	54.28	48.72	37.48

Table 1: Classification results on different groups of languages. Multilingual model has all parameters shared, Family-based Groups uses linguistic family groups, and Family-based Groups w/Interference has languages swapped across linguistic groups. The swapped languages are used to evaluate and are specified in the column title. Results show that grouping in a linguistically-aware way works best.

The training objective is Masked Language Modeling (MLM). We apply a masking ratio of 15% to the input data, as in the original mBERT (Devlin et al., 2019). The architecture is similar to XLM-R, with the memory efficient attention from xformers (Lefaudeux et al., 2022) to remove padding.

For the group specific layers, we parallelize the full transformer block, with all linear layers and layer norms now specific per group. To do so, we change all language-specific linear layers to a 3D tensor instead of a 2D tensor, of shape (`#groups`, `input_dimension`, `output_dimension`), and all 1D parameters (e.g. bias) from 1D to 2D tensors (`#groups`, `dimension`). All tokens in the same sentence are routed to the same group-specific layer.

We utilize the AdamW optimizer (Loshchilov and Hutter, 2017) and the learning rate is set to $4 \cdot 10^{-5}$, with linear warmup for 2.5k steps and cosine decay. The batch size per GPU is set to 1200 sentences, and the models were trained on 4 or 16 nodes, of 8 A100 GPUs each, for 26k and 17k steps, in the experiments with 30 and 90 languages respectively.

Due to the large size of our linear layers, doing the full communication of FSDP becomes very costly. To solve this, we train the model with Hybrid Sharding Data Parallel (HSDP) from PyTorch's (Paszke et al., 2019) FSDP2, where we only split the model per node instead of across all GPUs, with the forward pass done in bf16 and the reduction in fp32. To perform efficient multiplication of the tokens for each expert and the expert weight, we develop custom CUDA kernels.

3.2 Evaluation Methodology

All models are evaluated on the MTEB (Muennighoff et al., 2023) tasks and datasets detailed in Table 11 of the appendix, divided in three tasks.

Single Sentence Classification. This task consists of standard classification of the output embeddings into different classes. The reported metric is classification accuracy.

Pairwise Sentence Classification. This task consists in classifying two sentences, e.g. determining if a pair of sentences are duplicates or not. The reported metric, from MTEB, is the Average Precision (AP) based on the distance between sentence representations.

Semantic Textual Similarity (STS). The STS task evaluates the model's ability to replicate human judgments on sentence similarity. The reported metric, from MTEB, is the Spearman correlation based on distances.

The sentence representations to use for evaluation are obtained by average pooling the output token embeddings from the last layer.

Unless otherwise specified, we report the averaged results per language across datasets of each task. We then average across languages, and then across the three tasks.

3.3 Language Interference and Transfer

We posit that grouping similar languages in an encoder-only model enhances transfer, whereas combining dissimilar ones causes interference. To test this hypothesis, we design an experiment using the stacked architecture described in Section 3.5, with 30 languages, initially divided into five groups of six based on their linguistic family (see Table 8 of the appendix). We then swap two languages between groups and measure the resulting performance change.

To investigate how specific language characteristics influence performance when swapped between groups, we perform targeted swaps according to criteria designed to isolate the effect of: (1) **Language subgroup.** Swapping languages differing

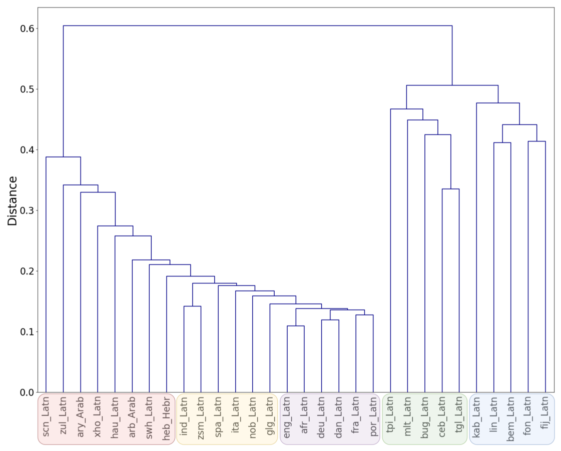


Figure 2: Dendrogram of embedding similarity between the 30 analysed languages. We build 5 groups of languages based on their distances.

Strategy	**Average**
Dense Multilingual Baseline	52.65
Language-Specific Layers (topline)	**53.86**
Linguistic Knowledge	**53.32**
Token Overlap	52.84
Token Overlap - Dendrogram	53.01
Embedding Similarity	52.85
Embedding Similarity - Dendrogram	**53.58**
Embedding Similarity - OP	**53.49**
Balanced Data	53.08
Random	52.80

Table 2: Averaged results for all grouping strategies. We see that linguistic knowledge and balanced embedding similarity achieve almost the same performance as a model with language-specific layers (with 5x the parameters).

only in linguistic subgroup (within same family, resource level and script), e.g., *spa_Latn* / *deu_Latn*. (2) **Script.** Swapping with a language of the same resource level (both high-resourced) but with a different script, e.g., *heb_Hebr* / *dan_Latn*, and (3) **Resource-level.** Swapping with a language using a different script and with a different resource level, e.g., *zsm_Latn* / *arb_Ara*.

The results in Table 1 show that language interference does occur in encoder-only models when languages are not grouped by similarity. Notably, even groups with targeted interference (family-based groups w/ interference) perform slightly better on average than the fully multilingual model. This suggests that the multilingual model suffers greater interference due to its exposure to a larger number of dissimilar languages, confirming that interference scales with the number of languages. Conversely, the best performing setup is the one that places languages in groups with similar languages (family-based groups), confirming our initial hypothesis. Additional results in other language pairs are provided in Table 14.

3.4 Language Grouping

Building on our finding that language grouping matters for downstream performance (Section 3.3), this section explores different criteria for creating these language groups.

We use the same 30 languages and stacked architecture as in the previous experiments and investigate groups based on the following criteria:

Linguistic Knowledge. Groups are defined according to language families, as done in the experiments in Section 3.3, and are detailed in Table 8.

Embedding Similarity. We embed FLORES-200, an n-way parallel translation dataset (NLLB Team, 2024), with MEXMA (Janeiro et al., 2025). The distance between two languages is calculated by averaging the cosine similarities of the embedded translation pairs. Languages are then clustered based on these pairwise distances using an agglomerative hierarchical clustering algorithm (UPGMA). However, this clustering does not inherently force balanced clusters in terms of number of languages. To correct the disparity between groups, we employ two rebalancing strategies: (1) manual adjustment after inspecting the dendrogram (Figure 2), and (2) solving an optimization problem to create equally sized groups while minimizing the average pairwise distance within each cluster.

Token Overlap. This grouping method is based on the lexical similarities of the languages. For this, we encode all tokens in FLORES-200 with the XLM-R tokenizer, and cluster the languages (again using UPGMA) based on the number of overlapping tokens. Similarly to the embedding strategy, the resulting groups are initially unbalanced, so we use the dendrogram approach to create balanced groups.

Balanced Data. We build language groups such that each group-specific layer is trained on an equal amount of data.

Random. As a baseline, we build random language groups, each consisting of 6 languages.

We compare our proposed group architectures, with 1B total parameters, to a 560M-parameter dense multilingual baseline, and to a 5B-parameter model with 30 language-specific layers. Our group architectures and the language-specific layers model both have 560M active parameters matching the dense backbone. Table 2 presents the results of each strategy. It is possible to see that the model with 30 language-specific layers performs the best. However, using only five parallel layers with groups formed by linguistic knowledge or embedding similarity (after balancing) yields comparable results, with a performance decrease of less than 0.2%. These configurations are significantly more memory efficient (one-fifth the total size) and offer greater scalability to more languages.

3.5 Model Architecture

Finally, we explore different distributions of shared (language-agnostic) and group-specific layers across the encoder, maintaining a constant depth of 24 layers as in the baseline model. We study three different architectures:

Stacked. Previous studies on MT have established that the first and last layers typically contain more language information (Kudugunta et al., 2019). Drawing on this finding, we design an architecture with six language-group specific layers at the beginning and end of the encoder, sandwiching 12 shared language-agnostic layers. This results in a 6-12-6 configuration.

Interleaved. Inspired by the common pattern in MoE models that alternate dense and sparse layers, we configure an architecture where language-agnostic and group-specific layers alternate throughout the encoder's depth, starting with a group-specific layer.

LID-Based. For this approach, we train a language identification (LID) model on top of the frozen dense multilingual baseline. We hypothesize that layers where the accuracy of the LID model is high will benefit more from having language-group specific layers. We evaluate on FLORES200 and show the results in Figure 3. Based on the observed U-shaped pattern, we set the decision threshold at 92.3%. This results in layers 0 to 5 and 15 to 23 being language-group specific sandwiching 9 language-agnostic layers (i.e., a 6-9-9 configuration). While the previous two architectures feature a 12-12 split, this method has more parameters with

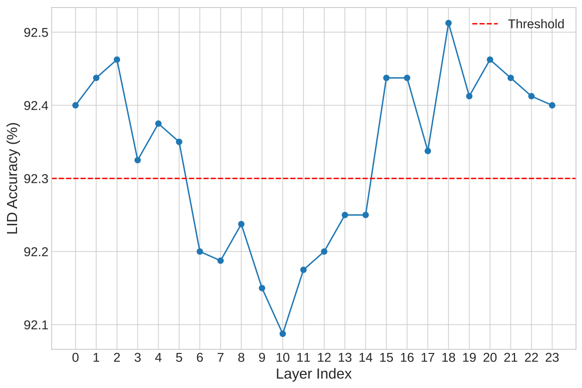


Figure 3: Accuracy on FLORES200 of the LID model trained on the output embeddings of each layer of the dense multilingual baseline.

Architecture	Average
Stacked	53.32
Interleaved	53.31
LID-Based	**53.49**

Table 3: Averaged results for different architectures.

only 9 shared layers, which can be a deciding factor for improved performance.

In Table 3, we see that the stacked and interleaved architectures have similar performances. Additionally, we see that the LID based architecture outperforms the remaining architectures. This indicates that the decision of which layers to make language specific is well informed by the LID model, yielding the best performing architecture.

4 Scaling Mixture-of-Languages

After performing different analyses to better understand how to group languages effectively and how to build an optimal architecture, we scale our experiments to 90 languages. We select the LID-based architecture detailed in Section 3.5, with a=9, b=9, c=6 in Figure 1 as our optimal design. For language grouping, we utilize linguistic groups. This strategy was chosen due to its strong performance (Section 3.4) and its better interpretability compared to embedding similarity-based grouping. We create 10 groups out of 90 languages, obtaining the groups detailed in Table 9 of the appendix. Our final MoL model has 2B total parameters, with 560M active parameters, matching XLM-R.

4.1 Results

The evaluation results for our scaled MoL model are presented in Table 4. Our model significantly

Model	Average	Classification	Pair Classification	STS
XLM-R	45.61	43.40	54.81	38.63
mBERT	53.39	45.40	56.25	58.54
Multilingual	51.05	39.96	56.83	56.36
MoL	**56.91**	**48.54**	**59.11**	**63.08**

Table 4: Results for MoL and the baselines on MTEB.

Model	Low Resource	High Resource
XLM-R	43.27 $_{\downarrow 7.44}$	47.84 $_{\downarrow 4.27}$
mBERT	43.49 $_{\downarrow 7.23}$	51.19 $_{\downarrow 0.92}$
Multilingual	44.70 $_{\downarrow 6.02}$	44.86 $_{\downarrow 7.26}$
MoL	**50.72**	**52.12**

Table 5: Results of the model in high vs low resource scenarios. MoL outperforms all baselines in both settings.

outperforms a fully dense multilingual model trained on the exact same training setup. Furthermore, it surpasses publicly available models trained on different datasets, as XLM-R (trained on the same languages with the same tokenizer and with the same number of active parameters), and mBERT (trained on the same pretrainig task but with a different tokenizer).

Additionally, revisiting the experiments with 30 languages (Table 2), we note that the gap between the dense multilingual baseline and MoL with linguistic groups is only 0.7%. However, when scaling to 90 languages, this gap widens to 5.9%, highlighting the significantly better scalability of MoL.

Overall, MoL establishes a new SOTA for largely multilingual pretrained encoder-only models trained with the MLM objective. This is achieved through parallelization of layers and grouping languages in a way that boosts transfer and reduces interference.

4.2 Performance in Low vs High Resource Languages

To assess performance across different resource settings, we compile average results per language in Table 12. The low and high resource classification in the table is based on the classifications made in NLLB (NLLB Team, 2024). The average results per resource level are shown in Table 5. MoL substantially outperforms other models in low resource scenarios, with up to 7.5% performance improvements, while still being the best in high resource scenarios. Interestingly, when compared to the multilingual baseline (a fairer comparison with the same training setup), MoL shows significant advantage in both settings. However, the difference is more pronounced in high resource languages. This is likely attributable to (1) the lower quality of data in low-resource languages which has a more detrimental effect on high-resource languages, and (2) the high-resource languages being able to benefit more from the additional model capacity.

5 Mixture-of-Languages vs. Mixture-of-Experts

In this section we directly compare MoL against MoE architectures. Both designs increase model capacity while maintaining the number of active parameters. For a direct comparison, we train an MoE model using the setup of Section 3 with a comparable stacked design (6 sparse layers, 12 shared layers, then 6 sparse layers). Typical MoE only uses sparse feedforward networks, but for a direct comparison with MoL we sparsify the whole Transformer block including self-attention and layer norms. Routing occurs once at the beginning of each Transformer block, directing tokens to their chosen expert. A linear layer performs the gating based on the normalized output embedding of the preceding Transformer block. During training, Gaussian noise is added the gate's output before softmax normalization to promote expert exploration.

We train two MoE models, one with the typical token-level routing, and a second with sentence-level routing. Sentence-level routing makes MoE closer to MoL with its language-based group-specific layers where all the tokens of a given sentence are handled by the same expert. Our MoE models use top-1 routing and are trained with the Gshard load balancing loss (Lepikhin et al., 2020).

Table 6 shows results comparing the two MoE models to MoL (optimal linguistic knowledge and random grouping) and the dense multilingual base-

Strategy	Average
Dense Multilingual	52.65
MoL- Random	52.80
MoL- Linguistic Knowledge	**53.32**
MoE - Token-level Routing	49.53
MoE - Sentence-level Routing	47.94

Table 6: Averaged results across 30 languages for all grouping strategies, comparing MoE, MoL and its dense counterpart.

line (all shared). We observe that MoE underperforms compared to both the dense model and MoL, despite all models having the same number of active parameters, with MoE and MoL also having the same total number of parameters. Interestingly, even random grouping of languages with MoL outperforms MoE. This suggests MoE's failure to capture the inherent language prior that MoL benefits from through its predefined grouping.

This underperformance of MoE models in encoder-only architectures has also been noted by Videau et al. (2024), where dense vision models outperform their MoE counterparts, with a much smaller total number of parameters.

Although MoEs have shown great performance in generative tasks with LLMs (Cai et al., 2025), the linguistic prior of MoL proves to be a much more effective method for increasing the total model capacity at fixed inference FLOPs in multilingual encoder-only models.

5.1 Analysis of MoE Architecture

To further understand the behavior of our MoE models, we perform an additional analysis on experts utilization.

A common issue in MoEs is routing collapse (Shazeer et al., 2017; Lepikhin et al., 2020), where only one expert dominates the routing gate and is used to process all tokens, falling back to the capacity of a dense model. To check for a similar collapse, we analyze the router distribution in our trained MoE models.

During training, Gaussian noise is added to the router's gate to systematically push the gate towards uniform allocation of tokens. At inference time, no noise is added, and we use the top-1 prediction of the router. We thus check the router's decisions at test-time averaged over 500 random batches during the evaluation on the *MassiveIntent-*

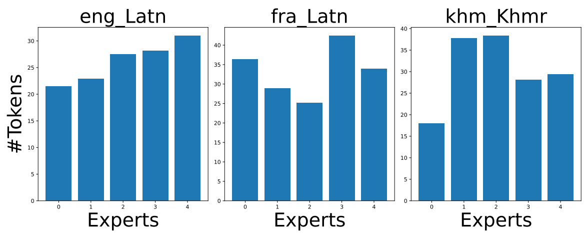


Figure 4: Histogram of expert routing in testing, averaged over 500 batches in English (highest resource), French (high resource) and Khmer (low resource).

Classification MTEB dataset in French, English and Khmer. These three languages have different levels of resource, with English being the highest resource language of all, French a high resource and Khmer a low resource language.

We plot the histogram of expert routing in Figure 4. We observe that all experts are used in all languages, and there is no token collapse. As a final check, we also analyze the effectiveness of the learned routing, by comparing our learned routing with random routing. This analysis shows that our learned routing is effective and performance degrades without it. Full details on this additional analysis can be found in Appendix A. Additional train and test loss curves are provided in Figure 5.

6 Conclusion

In this paper, we introduce MoL, a new multilingual encoder that uses language group-specific layers to enhance language transfer and reduce interference, keeping the active parameters constant.

Through a set of experiments on 30 languages, we analyze different language grouping strategies and architectural configurations, to identify an optimal setup. Subsequently, we scale our best approach to 90 languages.

Our scaled MoL largely outperforms a dense counterpart trained with the same active parameters and training setup, as well as public multilingual encoders such as XLM-R or mBERT on downstream tasks. MoL particularly boosts the performance in low resource languages while still being best in high resource. MoL also exhibits better scalability when increasing the number of languages.

Finally, we perform a comparison against MoE models, and establish that MoL is more effective in the context of multilingual encoder-only models, thus positioning MoL as a prime candidate for scaling model capacity at a constant number of active parameters (inference FLOPs).

Limitations

In this work, we show that our model with group specific parameters and language based routing is able to outperform a dense equivalent model, an MoE model, and public SOTA models as XLM-R and mBERT. The hyperparameters in this work were optimized for the dense model and also used for both our MoL and the MoE model. More careful tuning of the parameters could have been done for both to achieve higher performance.

We perform the comparison against publicly available models, however those models are trained on different data, so the direct comparison is hard to make. It would be ideal to compare to both XLM-R and mBERT trained on the same data. The comparison to mBERT is also harder, since the tokenizer and active number of parameters are also different.

We train our models on the Masked Language Modeling (MLM) task only, it would also be interesting to use different pretraining tasks proposed since the release of mBERT.

We train our models on both 30 and 90 languages, and show that the gap at 90 languages is larger compared to the dense model, which suggests a better scalability of our approach. It would be interesting to scale even further to more languages, such as the 200 languages in NLLB (NLLB Team, 2024), however that would require more compute, new language groups, and a different tokenizer, which makes the comparisons and training harder.

For linguistic groups we use language families, it would be interesting to explore other properties, and other groups inside language families.

References

Belen Alastruey, João Maria Janeiro, Alexandre Allauzen, Maha Elbayad, Loïc Barrault, and Marta R. Costa-jussà. 2025. Interference matrix: Quantifying cross-lingual interference in transformer encoders. *Preprint*, arXiv:2508.02256.

Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. *arXiv preprint arXiv:1909.08478*.

Lola Le Breton, Quentin Fournier, Mariam El Mezouar, and Sarath Chandar. 2025. Neobert: A next-generation bert. *Preprint*, arXiv:2502.19587.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. 2025. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*, page 1–20.

Daniel Cer, Mona Diab, Eneko Agirre, I
nigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *Preprint*, arXiv:2003.10555.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. *Preprint*, arXiv:1911.02116.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

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.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*, pages 5547–5569. PMLR.

Maha Elbayad, Anna Sun, and Shruti Bhosale. 2023. Fixing MoE over-fitting on low-resource languages in multilingual machine translation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 14237–14253, Toronto, Canada. Association for Computational Linguistics.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem

Natarajan. 2022. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *Preprint*, arXiv:2204.08582.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

João Maria Janeiro, Benjamin Piwowarski, Patrick Gallinari, and Loic Barrault. 2025. MEXMA: Token-level objectives improve sentence representations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23960–23995, Vienna, Austria. Association for Computational Linguistics.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Sneha Reddy Kudugunta, Ankur Bapna, Isaac Caswell, Naveen Arivazhagan, and Orhan Firat. 2019. Investigating multilingual nmt representations at scale. *arXiv preprint arXiv:1909.02197*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *Preprint*, arXiv:1901.07291.

Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. 2022. xformers: A modular and hackable transformer modelling library. `https://github.com/facebookresearch/xformers`.

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

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2021. MTOP: A comprehensive multilingual task-oriented semantic parsing benchmark. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2950–2962, Online. Association for Computational Linguistics.

Shangjie Li, Xiangpeng Wei, Shaolin Zhu, Jun Xie, Baosong Yang, and Deyi Xiong. 2023. MMNMT: Modularizing multilingual neural machine translation with flexibly assembled MoE and dense blocks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4978–4990, Singapore. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

NLLB Team. 2024. Scaling neural machine translation to 200 languages. *Nature*, 630(8018):841–846.

Zach Nussbaum and Brandon Duderstadt. 2025. Training sparse mixture of experts text embedding models. *Preprint*, arXiv:2502.07972.

James O'Neill, Polina Rozenshtein, Ryuichi Kiryo, Motoko Kubota, and Danushka Bollegala. 2021. I wish I would have loved this one, but I didn't – a multilingual dataset for counterfactual detection in product review. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7092–7108, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32.

Jonas Pfeiffer, Francesco Piccinno, Massimo Nicosia, Xinyi Wang, Machel Reid, and Sebastian Ruder. 2023. mmt5: Modular multilingual pre-training solves source language hallucinations. *Preprint*, arXiv:2305.14224.

Taido Purason and Andre Tättar. 2022. Multilingual neural machine translation with the right amount of sharing. In *Proceedings of the 23rd Annual Conference of the European Association for Machine Translation*, pages 91–100, Ghent, Belgium. European Association for Machine Translation.

Zhi Qu, Chenchen Ding, and Taro Watanabe. 2025. Languages transferred within the encoder: On representation transfer in zero-shot multilingual translation. *Preprint*, arXiv:2406.08092.

Uri Shaham, Maha Elbayad, Vedanuj Goswami, Omer Levy, and Shruti Bhosale. 2023. Causes and cures for interference in multilingual translation. In *Proceedings of the 61st Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 15849–15863, Toronto, Canada. Association for Computational Linguistics.

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

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, and Han Xiao. 2024. jina-embeddings-v3: Multilingual embeddings with task lora. *Preprint*, arXiv:2409.10173.

Ankit Kumar Upadhyay and Harsit Kumar Upadhya. 2023. Xnli 2.0: Improving xnli dataset and performance on cross lingual understanding (xlu). In *2023 IEEE 8th International Conference for Convergence in Technology (I2CT)*, pages 1–6. IEEE.

Mathurin Videau, Alessandro Leite, Marc Schoenauer, and Olivier Teytaud. 2024. Mixture of experts in image classification: What's the sweet spot? *arXiv preprint arXiv:2411.18322*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. *Preprint*, arXiv:2402.05672.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *Preprint*, arXiv:2412.13663.

Biao Zhang, Ankur Bapna, Rico Sennrich, and Orhan Firat. 2021. Share or not? learning to schedule language-specific capacity for multilingual translation. In *International Conference on Learning Representations*.

Xinyu Zhao, Xuxi Chen, Yu Cheng, and Tianlong Chen. 2024. Sparse moe with language guided routing for multilingual machine translation. In *The Twelfth International Conference on Learning Representations*.

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

A Analysis of Routing in Mixture of Experts

As an additional analysis, we assess the effectiveness of the learned routing by we comparing it to random routing. To do so, we load the trained MoE model and evaluate it using both approaches. Initially, we test the model on the pretraining task, MLM, using the FLORES200 dataset across all languages the model was trained on. The results, presented in Table 7, show that the learned routing significantly reduces the loss, suggesting its effectiveness. To determine if the learned routing is only beneficial for the pretraining task or if it generalizes well, we also evaluate the model on the *MassiveIntentClassification* task from MTEB in English. The results indicate that accuracy is nearly halved with random routing, demonstrating that the learned routing is both effective and well learned.

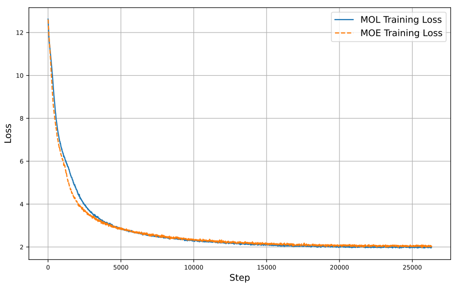
Routing	Val Loss	Accuracy
Learned	2.823	41.42
Random	6.130	22.22

Table 7: Comparison of learned routing vs random routing. Evaluated on FLORES in all languages covered by the model for val loss, and MassiveIntentClassification in English for the accuracy column.

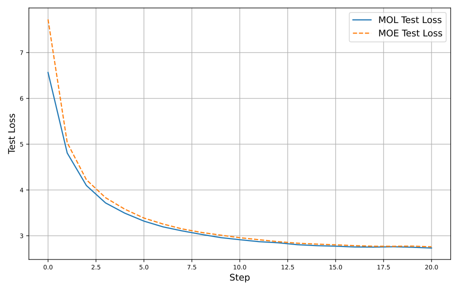
B Additional Tables and Plots

In this Appendix we collect additional tables and plots, in particular:

- Table 8: Contains the language family based groups created with 30 languages.
- Table 9: Contains the language family based groups created with 90 languages. The groups are designed to group similar languages together while keeping a balanced amount of languages in each group.
- Table 10: List of all languages covered in the 90 languages models.
- Table 11: MTEB datasets and tasks used for evaluation.
- Table 12: Results by language (averaged across datasets and tasks) for MoL and all baselines, categorized by amount of resources of the languages.



(a) Train curves.



(b) Test curves.

Figure 5: Loss curves for MoL and MoE.

- Table 14: Additional results performing the same analysis thar Table 1 but in other languages.
- Figure 5: Train and test loss curves for both MoL and MoE.
- Table 13: Execution time of MoL, MoE and dense models at inference time.

1	arb_Arab heb_Hebr mlt_Latn	hau_Latn ary_Arab kab_Latn
2	swh_Latn xho_Latn zul_Latn	lin_Latn bem_Latn fon_Latn
3	ind_Latn bug_Latn ceb_Latn	fij_Latn zsm_Latn tgl_Latn
4	fra_Latn ita_Latn por_Latn	spa_Latn glg_Latn scn_Latn
5	eng_Latn deu_Latn dan_Latn	afr_Latn nob_Latn tpi_Latn

Table 8: Groups based on language family for 30 languages.

1	als_Latn hye_Armn cym_Latn gla_Latn	gle_Latn ell_Grek kmr_Latn prs_Arab	pbt_Arab est_Latn fin_Latn hun_Latn
2	bel_Cyrl bos_Latn bul_Cyrl ces_Latn srp_Cyrl	hrv_Latn lvs_Latn lit_Latn mkd_Cyrl ukr_Cyrl	pol_Latn rus_Cyrl slk_Latn slv_Latn
3	afr_Latn dan_Latn deu_Latn	eng_Latn isl_Latn nld_Latn	nno_Latn swe_Latn ydd_Hebr
4	asm_Beng ben_Beng guj_Gujr hin_Deva	mar_Deva npi_Deva ory_Orya pan_Guru	san_Deva sin_Sinh snd_Arab urd_Arab
5	cat_Latn fra_Latn glg_Latn	ita_Latn por_Latn ron_Latn	spa_Latn
6	amh_Ethi arb_Arab hau_Latn	heb_Hebr gaz_Latn som_Latn	
7	ind_Latn jav_Latn plt_Latn	zsm_Latn sun_Latn	
8	kan_Knda mal_Mlym tam_Taml	tel_Telu	
9	azj_Latn kaz_Cyrl kir_Cyrl	tur_Latn uig_Arab uzn_Latn	
10	lao_Laoo tha_Thai mya_Mymr zho_Hans kat_Geor	zho_Hant swh_Latn xho_Latn khm_Khmr kor_Hang	vie_Latn eus_Latn epo_Latn jpn_Jpan khk_Cyrl

Table 9: Linguistic groups of the 90 languages.

Languages				
afr_Latn	fra_Latn	kir_Cyrl	por_Latn	tur_Latn
amh_Ethi	gla_Latn	kmr_Latn	prs_Arab	uig_Arab
arb_Arab	gle_Latn	kor_Hang	pbt_Arab	ukr_Cyrl
asm_Beng	glg_Latn	lao_Laoo	ron_Latn	san_Deva
azj_Latn	guj_Gujr	lvs_Latn	rus_Cyrl	sin_Sinh
bel_Cyrl	hau_Latn	lit_Latn	slk_Latn	vie_Latn
ben_Beng	heb_Hebr	mal_Mlym	slv_Latn	xho_Latn
bos_Latn	hin_Deva	mar_Deva	snd_Arab	ydd_Hebr
bul_Cyrl	hrv_Latn	mkd_Cyrl	som_Latn	zho_Hans
cat_Latn	hun_Latn	plt_Latn	spa_Latn	zho_Hant
ces_Latn	hye_Armn	khk_Cyrl	als_Latn	
cym_Latn	ind_Latn	zsm_Latn	srp_Cyrl	
dan_Latn	isl_Latn	mya_Mymr	urd_Arab	
deu_Latn	ita_Latn	nld_Latn	uzn_Latn	
ell_Grek	jav_Latn	nno_Latn	sun_Latn	
eng_Latn	jpn_Jpan	npi_Deva	swe_Latn	
epo_Latn	kan_Knda	gaz_Latn	swh_Latn	
est_Latn	kat_Geor	ory_Orya	tam_Taml	
eus_Latn	kaz_Cyrl	pan_Guru	tel_Telu	
fin_Latn	khm_Khmr	pol_Latn	tha_Thai	

Table 10: List of languages covered by our model.

Task Type	**Task Name**
Classification	MassiveIntentClassification (FitzGerald et al., 2022) MassiveScenarioClassification (FitzGerald et al., 2022) MTOPDomainClassification (Li et al., 2021) MTOPIntentClassification (Li et al., 2021) AmazonCounterfactualClassification (O'Neill et al., 2021)
STS	STS17 (Cer et al., 2017)
Pair Classification	XNLI (Conneau et al., 2018) XNLIV2 (Upadhyay and Upadhya, 2023)

Table 11: Full list of MTEB (Muennighoff et al., 2023) tasks used for evaluation.

Language	**Resource**	**XLM-R**	**mBERT**	**Multilingual**	**MoL**
afr_Latn	High	34.49	46.29	40.43	48.59
amh_Ethi	Low	28.86	5.26	33.42	40.04
ara_Arab	High	36.77	48.32	47.08	55.13
asm_Beng	Low	55.19	55.07	57.46	59.65
ben_Beng	High	53.27	55.48	56.81	53.42
bul_Cyrl	High	45.72	49.87	47.42	58.70
cym_Latn	Low	30.85	40.58	38.05	43.83
dan_Latn	High	48.45	50.69	39.19	48.40
deu_Latn	High	54.39	58.65	53.01	58.20
ell_Grek	High	47.50	51.98	46.37	50.75
eng_Latn	High	51.78	62.23	57.14	63.40
fin_Latn	High	46.44	48.40	36.85	43.76
fra_Latn	High	50.69	57.19	50.79	56.50
guj_Gujr	Low	55.54	58.93	57.51	60.12
heb_Hebr	High	43.05	44.92	33.21	41.58
hin_Deva	High	51.14	54.92	49.99	56.47
hun_Latn	High	44.93	47.35	37.09	45.67
hye_Armn	Low	38.96	47.50	37.67	44.66
ind_Latn	High	53.08	50.47	44.11	54.20
isl_Latn	High	38.28	41.15	36.13	44.85
ita_Latn	High	44.40	53.20	41.02	49.37
jav_Latn	Low	35.97	40.10	38.80	46.77
jpn_Jpan	High	56.74	53.80	46.20	53.87
kan_Knda	Low	46.88	49.30	46.40	52.79
kat_Geor	Low	32.66	40.01	33.58	41.61
khm_Khmr	Low	32.58	07.01	39.28	44.14
mal_Mlym	Low	42.74	43.92	37.20	47.35
mar_Deva	Low	54.26	56.60	55.21	61.11
mya_Mymr	Low	34.73	36.86	40.81	48.34
nld_Latn	High	43.69	51.34	38.54	50.03
ory_Orya	Low	54.91	50.29	57.43	59.18
pan_Guru	Low	55.45	57.46	58.03	60.41
pol_Latn	High	47.48	50.52	37.83	46.97
por_Latn	High	44.77	51.50	40.60	48.37
ron_Latn	High	44.18	46.51	35.88	44.80
rus_Cyrl	High	51.03	54.71	47.21	53.66
san_Deva	Low	54.19	55.65	55.35	56.08
slv_Latn	High	44.46	50.31	39.75	47.55
spa_Latn	High	47.15	60.84	56.58	61.40
swe_Latn	High	51.33	48.33	41.52	51.34
tam_Taml	Low	47.98	52.18	45.75	52.58
tel_Telu	Low	40.40	42.13	34.99	47.49
tha_Thai	High	51.37	34.50	52.78	56.75
tur_Latn	High	51.83	51.12	46.16	54.76
urd_Arab	Low	36.77	43.90	37.67	46.78
vie_Latn	High	53.11	53.16	50.64	55.19
zho_Hant	High	55.92	56.90	50.51	57.75

Table 12: Results per language averaged over all tasks.

Model	Latency (seconds)
Dense	1.2
MoE	3.4
MoL	1.6

Table 13: Execution time of MoL, MoE and dense model processing 1k sentences from FLORES200 in English, averaged over 10 executions.

	Swapped Languages					
	Script		Subgroup		Resource	
Setting	**ara_Arab**	**por_Latn**	**deu_Latn**	**por_Latn**	**nob_Latn**	**ara_Arab**
Multilingual	36.41	46.04	49.02	46.04	43.54	36.41
Family Groups	**39.18**	**47.30**	**52.39**	**47.30**	**45.51**	**39.18**
Family Groups w/Interference	36.68	45.05	52.39	46.36	42.58	35.82

Table 14: Classification results on different groups of languages. Multilingual model has all parameters shared, Family-based Groups uses linguistic family groups, and Family-based Groups w/Interference has languages swapped across linguistic groups. The swapped languages are used to evaluate and are specified in the column title. Results show that grouping in a linguistically-aware way works best.