Exploring Intrinsic Language-specific Subspaces in Fine-tuning Multilingual Neural Machine Translation

Anonymous ACL submission

Abstract

Multilingual neural machine translation models support fine-tuning hundreds of languages simultaneously. However, fine-tuning on full parameters solely is inefficient potentially leading to negative interactions among languages. In this work, we demonstrate that the fine-tuning for a language occurs in its intrinsic language-specific subspace with a tiny fraction of entire parameters. Thus, we propose language-specific LoRA to isolate intrinsic language-specific subspaces. Furthermore, we propose architecture learning techniques and introduce a gradual pruning schedule during fine-tuning to exhaustively explore the optimal setting and the minimal intrinsic subspaces for each language, resulting in a lightweight yet effective fine-tuning procedure. The experimental results on a 12-language subset and a 30-language subset of FLORES-101 show that our methods not only outperform full-parameter fine-tuning up to 2.25 spBLEU scores but also reduce trainable parameters to 0.4% for high and medium-resource languages and 1.6% for low-resource ones.

1 Introduction

Multilingual Neural Machine Translation (MNMT) aims to use a single model to translate among different languages (Ha et al., 2016; Johnson et al., 2017). Recent studies of MNMT (Fan et al., 2020; Team et al., 2022) achieved significant progress in training large-scale pre-trained models supporting hundreds of languages. Benefiting from cross-language learning, these pre-trained models offer the possibility of fine-tuning with limited data and show better performance in low-resource languages and non-English directions. However, multilingual fine-tuning still suffers from two limitations: (1) full-parameter fine-tuning becomes inefficient as the model size increases; (2) negative interactions among languages (Duh et al., 2012; Mohammadshahi et al., 2022; He et al., 2023; Chen et al., 2023; Huang et al., 2023) lower the performance of high-resource languages.

Recent studies have shown that the fine-tuning of pre-trained models can be re-parameterized in an intrinsic subspace, i.e., a low-rank subspace with tiny parameters (Li et al., 2018; Qin et al., 2022; Zhang et al., 2023b). This insight implies that language-specific fine-tuning in pre-trained MNMT models happens within intrinsic language-specific subspaces, thus overcoming the aforementioned limitations: (1) intrinsic subspaces significantly reduce the required trainable parameters; (2) isolating the intrinsic subspaces among languages alleviates the negative interference in the multilingual representations. Therefore, in this work, we propose Language-Specific LoRA (LSLo), consisting of multiple LoRA (Hu et al., 2021) modules with sparse language-specific activation, to model such intrinsic subspaces.

Moreover, prior works (Qu and Watanabe, 2022; Pfeiffer et al., 2022; Pires et al., 2023) allocate the same number of parameters to different languages, which can yield sub-optimal setup because pre-trained models have already learned a substantial amount of knowledge from high-resource languages given the imbalance distribution of training data. We hypothesize that fine-tuning of high-resource languages can be done in a smaller subspace compared to low-resource languages. To exhaustively explore the minimal intrinsic subspaces for each language, we first reduce the rank for high-resource languages and then introduce unstructured pruning with a Gradual Pruning Schedule (He et al., 2023) during fine-tuning.

However, determining the optimal structure of LSLo remains challenging. First, there are 2 cases when selecting the language-specific sub-module of each LSLo: selected by source language (source-indexed) and selected by target language (target-indexed). Furthermore, although we intuitively expect that high-resource languages require smaller subspaces, it's still insufficient for the complex multilingual setting. These lead to the exponential increase in the possible architectures with the increase of the number of model layers and supported languages. Therefore, in this work, we use two architecture Learning techniques to avoid the tedious manual trial-and-error. We applied Weight Learning (Elsken et al., 2019; Pires et al., 2023) to determine whether each LSLo module should be source-indexed or target-indexed, given its interpretability and ease of visualization. We also propose a Layer-wise Cross-Language Pruning method, which combines the LoRA modules of all languages at every layer for pruning to estimate the required subspace size for each language.

We conduct our experiments on a 12-language subset of FLORES-101 (Goyal et al., 2021). Results show that in a pre-trained MNMT model, the size of intrinsic language-specific subspace is highly correlated with the language's resource type. Specifically, High-resource languages can be fine-tuned within a very small parameter subspace. Our fine-tuning method outperforms full parameter fine-tuning by 1.3 spBLEU while only using 0.4% trainable parameters for high and medium languages, and 1.6% for low-resource ones. We further evaluate our method on a 30-language subset, achieving a 2.25 spBLEU improvement over full parameter fine-tuning with only 7% trainable parameters, which demonstrates the efficiency and effectiveness of our method.

2 Background

Given a set of n languages $\mathbb{L} = \{l_1, l_2, \cdots, l_n\}$, the multilingual translation task is defined as translating an input in source language $src \in \mathbb{L}$ into an output in target language $tgt \in \mathbb{L}$. To train an MNMT model, we need a parallel corpus including translations aligned at the sentence level for creating MNMT datasets. For instance, consider a collection with m sets of sentences $\mathbb{S} = \{\mathbb{S}_1, \mathbb{S}_2, \cdots, \mathbb{S}_m\}$, each sentence set includes sentences in different languages sharing the same semantics, $\mathbb{S}_k = \{\boldsymbol{s}^k_{l_1}, \boldsymbol{s}^k_{l_2}, \cdots, \boldsymbol{s}^k_{l_n}\}$. With a parallel corpus, we can conveniently construct MNMT datasets including different translation directions $src \to tgt$ by choosing source and target sentences pairs from $\mathbb{S}$, e.g., $\boldsymbol{s}^k_{src}$ as the input $\boldsymbol{x}$ and $\boldsymbol{s}^k_{tgt}$ as the output $\boldsymbol{y}$ of a single translation pair $(\boldsymbol{x}, \boldsymbol{y})$. Given a MNMT dataset with N translation pairs $\mathbb{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i \in 1 \cdots N\}$, the training loss is defined as:

$$\mathcal{L}_{MNMT} = -\sum_{\boldsymbol{x},\boldsymbol{y}\in\mathbb{D}} \sum_{j=1}^{J} \log\ p_\theta(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x}) \quad (1)$$

where $\boldsymbol{x} = x_1, x_2, \cdots, x_I$ is a source sentence with length I and $\boldsymbol{y} = y_1, y_2, \cdots, y_J$ is the corresponding target sentence with length J. We say an MNMT model is English-centric if all language pairs in its training data include English. Models without this limitation are classified as many-to-many models. In this work, we conduct experiments in a many-to-many setting.

3 Methodology

3.1 Language-specific LoRA

LoRA (Hu et al., 2021) is widely used in Parameter-efficient Fine-tuning (PEFT) for Large Language Models where fine-tuning is re-parameterized in a low-rank intrinsic subspace. For a weight matrix in a pre-trained model $W \in \mathbb{R}^{d\times k}$, LoRA forward pass can be calculated as:

$$h = Wx + BAx \quad (2)$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times d}$. During training, W will be frozen and the trainable parameters, i.e., A and B, will be reduced from $d\times k$ to $d\times r + r\times k$, where $r \ll min(d, k)$.

In this work, we propose Language Specific LoRA (LSLo), an instance of Mixture-of-LoRAs (Feng et al., 2024) but with a hard language-specific routing. Specifically, Each LSLo module contains multiple sparsely activated LoRA modules with different rank r_{l_i} for each language, instead of sharing a unified parameter space across all languages. The forward pass of LSLo is calculated as:

$$\begin{aligned} h &= Wx + LSLo(x, l_i) \\ &= Wx + B_{l_i} A_{l_i} x \end{aligned} \quad (3)$$

where $l_i \in \mathbb{L}$ is the selected language for this LSLo. Only the LoRA module of the selected language will be activated each time. Similar to LoRA, LSLo can be added to any weight matrices, e.g., projections for attention and fully connected layers. The number of trainable parameters for each LSLo module is $\sum_{i=1}^{n}(d \times r_{l_i} + r_{l_i} \times k)$, where r_{l_i} is the reduced dimension for each language l_i. We are allowed to flexibly adjust the size of intrinsic language-specific subspaces through r_{l_i}, thus achieving higher parameter efficiency.

3.2 Unstructured Pruning

The assumption that higher-resource languages have smaller intrinsic subspaces naturally leads to the following question: How small can these subspaces be? Therefore, we adopt unstructured pruning [1] to explore the minimal intrinsic language-specific subspaces exhaustively. Compared with structured pruning which directly removes entire rows or columns from a weight matrix, we choose unstructured pruning without the above limitation to achieve a higher pruning ratio. We also introduce a Gradual Pruning Schedule (Zhu and Gupta, 2017; He et al., 2023) during fine-tuning to avoid huge performance loss caused by pruning. We provide more details about Gradual Pruning Schedule (GPS) in Appendix A.

4 Architecture Learning

LSLo introduces additional hyperparameters: (1) each LSLo module can be selected by either the source or the target language; (2) each language can have a different rank r_{l_i}, leading to an exponential increase in possible architectures with the number of layers and languages. Therefore, we propose two architecture learning techniques in this section to avoid manual selection.

4.1 Weight Learning

Consider a translation from a source language $l_i \in \mathbb{L}$ to a target language $l_j \in \mathbb{L}$. We say an LSLo module is source-indexed if activated by the source language l_i and is target-indexed if activated by the target language l_j. Intuitively, we expect that the language information is transformed from the source side to the target side near the top layers of the encoder and the bottom layers of the decoder (Kudugunta et al., 2019), which motivates the assumption that a layer in an encoder or decoder might prefer either source or target language, e.g., top layers of the encoder require target-indexed LSLo for more target side information while bottom layers of the encoder require source-indexed LSLo for more source side information. However, finding an optimal setting remains tedious work. Inspired by Neural Architecture Search (Elsken et al., 2019; Pires et al., 2023), we introduce a weight learning method here to determine the activation strategy for each layer's LSLo modules. Given an LSLo module added to a pre-trained weight matrix W, let the layer index W located is i, and the module W belongs to is mo, we calculate weighted sum during forward pass as follows:

$$
\begin{aligned}
h_{mo}^i = & W_{mo}^i x \; + \\
& w_{src}^i \cdot LSLo_{mo}^i(x, l_{src}) \; + \\
& w_{tgt}^i \cdot LSLo_{mo}^i(x, l_{tgt}) \; .
\end{aligned} \quad (4)
$$

where w_{src}^i, w_{tgt}^i are shared among all LSLo modules in the same layer, and we use softmax to make sure the weights are non-negative and sum up to 1. We will simply choose the index strategy with the one having a larger weight.

4.2 Intrinsic Subspace Estimation

Intuitively, high-resource languages can be fine-tuned in smaller subspaces owing to the extensive knowledge learned during pre-training, while low-resource ones should preserve larger subspaces due to the limited resources. However, in practice, some medium-resource languages, such as Dutch, have data scales similar to high-resource languages, thus it is possible to reduce the size of subspaces. Additionally, some low-resource languages would benefit more from cross-lingual transfer thanks to their similarity to high-resource languages, e.g., the same language family, effectively allowing the reduction in the fine-tuning subspaces. Therefore, we propose an intrinsic subspace estimation technique using layer-wise cross-language pruning[2] to comprehensively analyze the fine-tuning space demands for each language.

We apply LSLo to all possible weight matrices and group B matrices from LSLo modules of all languages in the same layer for pruning. We use the same unstructured pruning in Section 3.2. Let $\#_B$ be the number of parameters in matrix B, P_{ISE} is the predefined pruning ratio, and $\#_{pruned_B}$ represents the actual number of parameters pruned from matrix B. We measure the intrinsic subspace using the importance score:

$$
Score(B) = \#_{pruned_B} - P_{ISE} \cdot \#_B \quad (5)
$$

If $Score(B)$ is positive, it means that matrix B was pruned more than the target rate, thus the fine-tuning can be done in a smaller subspace. Conversely, a negative one indicates the need for a

[1] We directly use the implementation from PyTorch. https://pytorch.org/docs/stable/generated/torch.nn.utils.prune.l1_unstructured.html

[2] We also use the implementation from PyTorch. https://pytorch.org/docs/stable/generated/torch.nn.utils.prune.global_unstructured.html

larger parameter space. By grouping all languages for pruning in each layer, we can estimate the size of each language's intrinsic subspace in different layers respectively.

We only focus our comparison among B matrices because, while the A matrices are randomly Gaussian initialized, B matrices are initialized by zero in LoRA, allowing us to compare more fairly.

5 Experimental Setup

Dataset FLORES-101 (Goyal et al., 2021) is a high-quality parallel dataset, including 3,001 sentences from English Wikipedia which are translated into 101 languages by human translators. Sentences are divided into three splits: dev (997 sentences), devtest (1,012 sentences), and test (992 sentences). Since the test set is not publicly available, we use the dev set for training and devtest set for evaluation. Languages are divided into four resource types: High (H), Medium (M), Low (L), and Very-Low (V), based on the available bitext data with English.

We first randomly selected four languages from each of the three resource types (high, medium, very-low) to form a small subset *lang12* of 12 languages. We conducted comprehensive analyses and tests on *lang12* to verify our proposed method. Then, we extend our method to a larger subset *lang30* to measure the impact when introducing more languages. Details for *lang12* and *lang30* are provided in Appendix B.

Model Setting We choose M2M-124 615M (Goyal et al., 2021) as our base model. This is a special version of M2M-100 (Fan et al., 2020) extended by supplementing OPUS data to support all languages in the FLORES-101 dataset.

Training We implemented LSLo using fairseq (Ott et al., 2019) based on Transformer architecture. All experiments were trained in a many-to-many setting in a single run. For full parameter fine-tuning, we trained the model for 15 epochs with a learning rate of 0.0001. For LSLo, we froze the parameters of the original model and trained for 15 epochs with a learning rate of 0.003. All models were trained on 4 RTX A6000 with automatic mixed precision.

Evaluation We choose the results of full parameter fine-tuning as the baseline to compare with the beam size of 5. We use the dev and devtest set mentioned above as our training and test sets respectively and report *spBLEU* score (SentencePiece BLEU) (Goyal et al., 2021) with the FLORES-101 tokenizer[3].

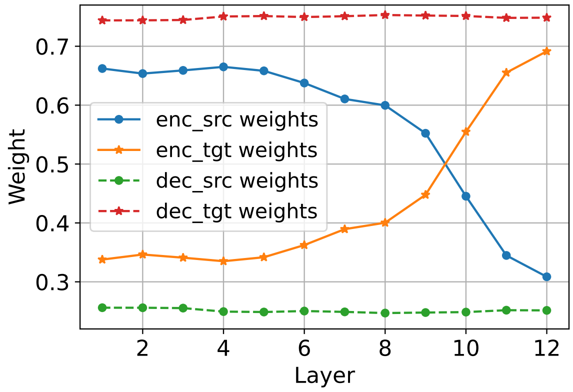


Figure 1: Source (src) and target (tgt) weights learned across layers in encoder (enc) and decoder (dec). The model's focus shifted from the source side to the target side near the top of the encoder.

6 Results

6.1 Weight Learning

As described in Section 4.1, we apply weight learning to the training data of *lang12* before conducting subsequent experiments to determine whether each LSLo module should be source-indexed or target-indexed. We build both source-indexed and target-indexed LSLo modules with the same rank $r_{l_i} = 8$ for all languages to all weight matrices in both encoder and decoder, including q, k, v, c-q, c-k, c-v, i.e., query, key and value matrices of attention and cross-attention respectively, and fc1, fc2, i.e., down and up matrices of MLP, respectively. In forward pass, we calculated the weighted sum of these two different indexed modules.

Figure 1 shows a clear tendency that the model's focus moves from the source side to the target side near the top of encoder, and in decoder, the model only focuses on target information. This is also mentioned by Tenney et al. (2019); Pires et al. (2023), where the bottom layers of encoder only learn some lower-level representation, and the top layers capture more output-related higher-level representation.

For the following experiments, we choose the indexed modules with the larger weights, which means the LSLo modules in the first 9 layers of encoder will be source-indexed and in other layers

[3]https://github.com/facebookresearch/flores/tree/main/previous_releases/flores101

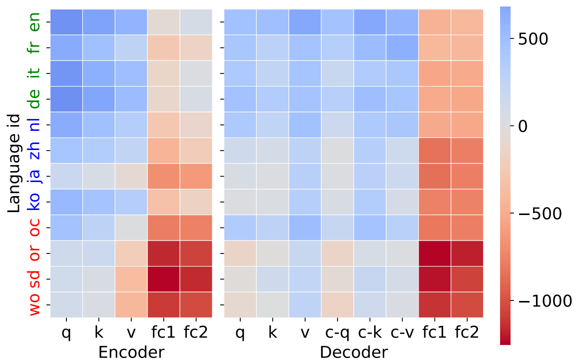


Figure 2: Illustration of the parameter space demands for each language, averaged across all layers. Color indicates the demands from low (blue) to high (red). Rows are organized by language resource type: high-resource (green), medium-resource (blue), and very-low-resource (red). Columns are organized by weight matrices in the encoder and decoder: query, key, and value matrices of attention (q, k, v) and cross-attention (c-q, c-k, c-v); down and up matrices of MLP (fc1, fc2).

of encoder and decoder will be target-indexed. We also analyze the impact of different index strategies in Appendix C.

6.2 Intrinsic Subspace Estimation

We performed layer-wise cross-language pruning as described in Section 4.2 on the training data of *lang12* to estimate the required parameter subspace (intrinsic subspace) for each language. We added LSLo with r_{l_i} for all languages to all weight matrices, allowing us to assess the parameter demands of different languages in each layer of encoder and decoder. See Appendix A for more details of layer-wise cross-language pruning. Figure 2 shows the demands of each language, averaged across all layers. 12 languages are organized by three resource types: high-resource (green), medium-resource (blue) and very-low-resource (red).

The results indicate that the intrinsic subspace for each language is highly correlated with the resource type. Very-low-resource languages need more parameters to learn the language-specific knowledge compared to high and medium-resource ones. This suggests there is no need to use the same architecture for all languages during fine-tuning. We observed similar tendencies in all layers, and the details are provided in Appendix D. Additionally, we also notice that compared to other languages in the same group, Dutch (nl) and Occitan (oc) require smaller parameter spaces. For Dutch (nl), it has much more bitext resources (82.4M) compared with the other three languages in the same group: Chinese (zh) (37.9M), Japanese (ja) (23.2M), Korean (ko) (7.46M). We think the resource type, which is close to high-resource languages, allows Dutch (nl) to have a smaller intrinsic subspace. For Occitan (oc), although it has only 5.11K bitext resources, it is the only language in the group that belongs to the same Language Family (Romance) as two high-resource languages, French (fr) and Italian (it). This suggests that similar languages can benefit more from cross-language learning, in line with Ma et al. (2023)'s approach of integrating similar languages into a single module.

For the following experiments, we further reduce the subspace size for high and medium-resource languages by lowering their rank and applying unstructured pruning with Gradual Pruning Schedule to further explore the minimal possible intrinsic subspace.

6.3 Main Results

In Table 1, we report the spBLEU scores of *lang12*, organized by languages' resource types: High (H), Medium (M), and Very-low (V). The first column shows the experimental settings. We use the format {H;M;V} to show the rank in LSLo for languages with different resource types. The notation WL means we use the architecture learned from Weight Learning in Section 6.1 and GPS(P_r) means we use the Gradual Pruning Schedule mentioned in Section 3.2 for high and medium-resource languages with the Pruning Ratio P_r. See Appendix A for more details of GPS. We choose the zero-shot (Pretrain) and full-parameter fine-tuning (Ft-all) results as our baselines. As shown in the first two rows, although the spBLEU of very-low-resource languages improved after full parameter fine-tuning (Ft-all), high-resource languages performed poorly due to the negative interference among languages, even worse than the zero-shot results (Pretrain).

We first experimented with the same subspace size for every language but varied ranks $r \in \{4, 8, 16, 64\}$. Results (LSLo+WL) show a trade-off between high-resource and low-resource languages, i.e., a smaller rank can alleviate the degradation of high-resource languages, e.g., 4;4;4+WL, but limits the performance of low-resource ones compared with higher rank settings, e.g., 64;64;64+WL. This indicates that sharing the same rank among languages with different resource types is suboptimal, improving low-resource performance requires a larger rank, which leads to

	Methods	#Params	Language Direction H2H	H2M	H2V	M2H	M2M	M2V	V2H	V2M	V2V	AVG
Baselines	Pretrain	-	31.76	20.06	5.56	20.71	17.12	3.47	9.24	5.03	0.52	12.26
	Ft-all	615M	29.29	20.46	12.53	19.28	17.14	8.95	15.23	11.02	6.66	15.43
LSLo +WL	4;4;4+WL	15.35M	30.15	20.35	11.76	19.49	17.04	8.13	14.58	10.02	5.66	15.03
	8;8;8+WL	30.7M	28.49	19.26	13.01	18.39	16.05	9.21	14.28	9.86	6.94	14.86
	16;16;16+WL	61.4M	25.90	17.82	14.32	16.86	14.93	10.57	13.80	9.71	8.46	14.55
	64;64;64+WL	245.6M	21.91	14.94	**14.91**	14.44	12.43	**11.47**	12.55	8.98	**9.96**	13.40
LSLo +WL +GPS	2;2;8+WL	15.3M	31.33	21.07	13.07	20.16	17.58	9.3	15.95	10.89	7.01	16.05
	2;2;8+WL+GPS(0.1)	15.3M	31.37	21.21	12.90	20.22	17.63	9.18	15.93	10.90	6.96	16.04
	2;2;8+WL+GPS(0.3)	15.3M	31.53	21.33	12.88	20.32	17.67	9.18	15.93	10.84	6.99	16.08
	2;2;8+WL+GPS(0.5)	15.3M	31.76	21.5	12.96	20.49	17.94	9.25	16.08	10.98	7.1	16.23
	2;2;8+WL+GPS(0.7)	15.3M	32.22	21.81	12.86	20.92	18.10	9.22	16.28	11.12	6.94	16.38
	2;2;8+WL+GPS(0.9)	15.3M	**33.13**	**22.33**	12.93	**21.49**	**18.58**	9.23	16.59	11.38	7.04	16.73
	2;2;16+WL+GPS(0.9)	25.6M	33.06	22.27	14.24	21.44	18.58	10.49	17.44	12.02	8.42	17.33
	2;2;64+WL+GPS(0.9)	86.9M	33.02	22.27	13.96	21.47	18.56	10.92	**18.67**	**12.98**	9.48	**17.70**

Table 1: The spBLEU scores on *lang12* organized by language resource type: High-resource (H), Medium-resource (M) and Very-low-resource (V), with the format {H;M;V} to show the rank we use for different languages in LSLo. WL means we follow the learned architecture of Weight Learning mentioned in Section 4.1. GPS(P_r) means we use the Gradual Pruning Schedule mentioned in Section 3.2 for High and Medium languages with the Pruning Ratio P_r. Our most efficient structure (2;2;8+WL+GPS(0.9)) outperforms full parameter fine-tuning across all language directions with a much smaller number of trainable parameters #Params.

greater degradation of high-resource performance. Although LSLo with $r = 64$ achieves the best performance on very-low-resource directions, it incurs a large number of trainable parameters and sacrifices high-resource performance.

Based on the findings of Section 6.2 that high and medium-resource languages can be fine-tuned in smaller subspaces, we set a lower rank $r = 2$ for high and medium-resource languages and $r = 8$ for very-low-resource languages (2;2;8+WL). Compared with the setting of 8;8;8+WL, reducing parameter space for high and medium-resource languages can effectively alleviate the degradation without compromising the performance of very-low-resource directions.

To further explore the minimal intrinsic subspace, we implemented the Gradual Pruning Schedule during fine-tuning mentioned in Section 3.2 for high and medium-resource languages. Based on the setting of 2;2;8+WL, we further reduce the parameter space for high and medium-resource languages by increasing P_r. We surprisingly find that, even after pruning 90% of the LSLo parameters for high and medium-resource languages (2;2;8+WL+GPS(0.9)), our method still achieves a 1.3 spBLEU improvement over the full parameter fine-tuning baseline, with only 2.5% trainable parameters. Furthermore, the degradation in high-resource languages has also been solved, with H2H performance improved from a decline of -2.47 spBLEU to an increase of +1.37 spBLEU. This suggests that language-specific fine-tuning for high and medium-resource languages actually occurs within tiny subspaces. Therefore, we can save more space for low-resource language learning. Simply increasing the rank for very-low-resource languages to 64 (2;2;64+WL+GPS(0.9)) can achieve a 2.26 spBLEU improvement and is more parameter-efficient.

We also expand our experiments to 30 languages *lang30* in Table 2 to assess our method's scalability. Languages are divided into four resource types: High (H), Medium (M), Low (L), and Very-low (V). Similar to Table 4, we use the format {H;M;L;V} to represent the rank setting in LSLo. Although the number of trainable parameters increases with the additional introduction of language-specific modules, our method (2;2;8;8+WL+GPS(0.9)) still achieved a 2.25 spBLEU improvement over full parameter fine-tuning with only 7% trainable parameters. This demonstrates our method's potential to support hundreds of languages while still keeping the number of trainable parameters near the original model.

7 Analysis and Discussion

7.1 What Causes the Degradation of High-resource Languages?

In previous experiments of Section 6.3, we discovered that merely isolating each language representation into different parameter spaces via LSLo did

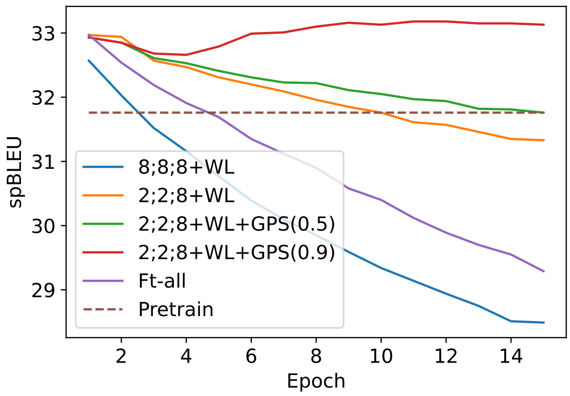
Methods	#Params	Language Direction																
		H2H	H2M	H2L	H2V	M2H	M2M	M2L	M2V	L2H	L2M	L2L	L2V	V2H	V2M	V2L	V2V	AVG
Pretrain	-	28.93	20.77	6.29	3.60	22.94	17.26	4.82	2.73	11.28	8.01	3.03	1.51	7.04	4.34	1.68	0.55	9.53
Ft-all	615M	24.48	19.80	9.76	7.94	19.44	16.72	8.55	6.72	12.53	11.17	6.76	5.15	11.11	9.72	6.05	4.04	11.61
8;8;8;8+WL	76.7M	22.94	17.91	**11.15**	**10.24**	18.07	15.00	**9.64**	8.74	12.33	10.46	8.15	7.34	11.07	9.30	7.47	6.11	11.83
16;16;16;16+WL	153.4M	19.58	15.29	11.08	10.47	15.53	12.95	**9.64**	**9.10**	11.18	9.56	**8.29**	**7.83**	10.10	8.59	**7.62**	**6.74**	10.98
2;2;8;8+WL+GPS(0.9)	46M	**29.92**	**22.90**	11.11	10.06	**23.60**	**19.20**	9.53	8.61	**15.34**	**12.70**	8.05	7.25	**13.75**	**11.31**	7.37	6.11	**13.86**

Table 2: The spBLEU scores on *lang30* organized by languages' resource type: High-resource (H), Medium-resource (M), Low-resource (L) and Very-low-resource (V), with the format {H;M;L;V} to show the rank we use for different languages in LSLo. Our most efficient structure (2;2;8;8+WL+GPS(0.9)) outperforms full parameter fine-tuning, demonstrating the effectiveness and scalability of our proposed method.

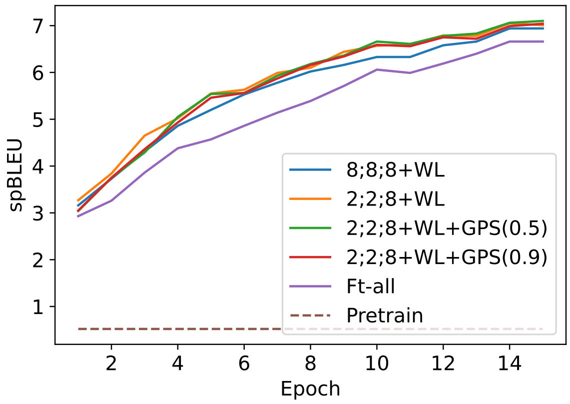
not mitigate the performance degradation of high-resource languages. This indicates that the trading or competing language representation might not be the only factor causing the decline.

We examined the spBLEU of H2H and V2V directions per epoch, as shown in Figure 3. We observed that the spBLEU of low-resource languages continuously improved during training, whereas high-resource languages' performance increased in the first epoch and then gradually declined. This suggests the pre-trained model has already acquired substantial knowledge of high-resource languages, making their subspace smaller compared to low-resource ones. When allocating same-sized parameter spaces for languages of different resource types, high-resource languages are more susceptible to over-fitting, which contributes to an over-fitting phenomenon leading to degradation.

This also explains why reducing the trainable parameter of high-resource languages can achieve better performance. As shown in Figure 3a, over-fitting is mitigated by continuously reducing the subspace size (2;2;8+WL+GPS(0.9)), without compromising the performance of low-resource languages.



(a) H2H Performance per epoch



(b) V2V Performance per epoch

Figure 3: We examined the performance of H2H and V2V directions per epoch. H2H performance declined during training.

7.2 Where Should We Apply LSLo to?

In this section, we want to discuss which weight matrices in the Transformer architecture are more crucial for LSLo. Similar to Section 4.2, we employ language-specific pruning on the training data of *lang12* to measure the demands for different weight matrices using Equation 5. Specifically, we add LSLos with a rank of 8 to all possible weight matrices and group the B matrix from all LoRA modules for each language into respective pruning groups. See Appendix A for more details of language-specific pruning. In this setting, we aim to examine which weight matrices are more important for different languages. The results averaged across all 12 languages are shown in Figure 4. Further details for each language respectively are shown in Appendix E. We observed a clear trend across all 12 languages: fc1 and fc2 play a more important role in both encoder and decoder compared to other weight matrices. This is in line with the observation by Geva et al. (2021) that feed-forward layers in Transformer architecture function as key-value memories for refining the final output, thus more crucial than other weight matrices. We empirically prove this in Appendix E, showing that placing LSLo in the fully connected layers is more

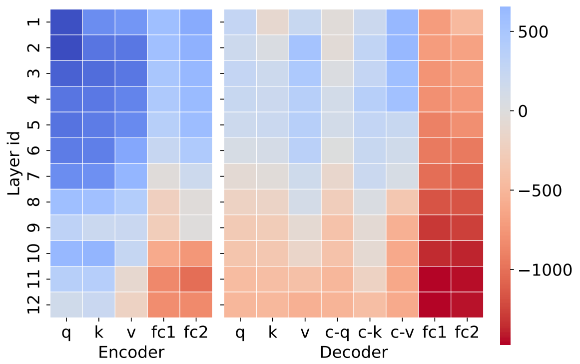


Figure 4: Illustration of the parameter space demands for each weight matrix, averaged across all languages. Color indicates the demands from low (blue) to high (red). Columns are organized by weight matrices in the encoder and decoder: query, key, and value matrices of attention (q, k, v) and cross-attention (c-q, c-k, c-v); down and up matrices of MLP (fc1, fc2).

efficient, given a similar parameter budget.

8 Related work

Intrinsic Subspace Intrinsic Subspace is the minimal parameter subspace required for models to learn new tasks. Li et al. (2018) first showed that intrinsic structures exist in deep neural networks through random subspace projection. Aghajanyan et al. (2021) further used this concept to explain the fine-tuning of Pre-trained Models. Following their works, Qin et al. (2022) found a universal task subspace that only includes hundreds of parameters through prompt-tuning (Brown et al., 2020; Li and Liang, 2021; Liu et al., 2022), and Zhang et al. (2023b) observe outlier dimensions during fine-tuning. However, their experiments do not include natural language generation (NLG) tasks. To bridge this gap, our work focuses on Multilingual Neural Machine Translation, a particularly challenging NLG task.

Low-Rank Adaptation (LoRA) LoRA (Hu et al., 2021) employs the product of two low-rank matrices to replace the original parameter matrix for fine-tuning. This method is parameter-efficient and widely used in Large Language Models. Recent works (Zhang et al., 2023a; Kopiczko et al., 2024) have focused on how to further enhance the efficiency of LoRA. Zhang et al. (2023a) modeled LoRA in the form of singular value decomposition and improved efficiency by pruning less important singular values. Kopiczko et al. (2024) reduced trainable parameters of LoRA by only leaning scaling vectors during training, fixed low-rank matrices are randomly initialized and shared for each layer. Inspired by these works, we propose LSLo, a LoRA based method, to model the intrinsic subspace of language-specific learning.

Language-specific Learning Multilingual models suffer from the negative interaction among languages (Duh et al., 2012; Chen et al., 2023; Huang et al., 2023). Introducing language-specific structures is a common strategy to address this issue. Sachan and Neubig (2018); Escolano et al. (2021); Pires et al. (2023) built language-specific encoder or decoder layers. Despite its effectiveness, a large number of trainable parameters are required for such architecture. Another line of work (Lin et al., 2021; Wang and Zhang, 2022; He et al., 2023) tried to extract sub-networks by first fine-tuning on all language pairs separately and then jointly training these sub-networks. However, the number for fine-tuning will increase quadratically with the number of languages, consuming significant computational resources. In this work, we propose a parameter-efficient method that maximizes the utilization of the substantial knowledge learned by Pre-trained Multilingual Models to improve the performance of all language pairs.

9 Conclusion

In this work, we studied the imbalance size distribution of intrinsic language-specific subspaces in a Pre-trained Multilingual Model. We modeled the intrinsic language-specific subspaces using LSLo. We further proposed an intrinsic subspace estimation method and found that the size of the intrinsic subspace for each language is highly correlated with its resource type. The required subspace size for higher-resource languages is much smaller than for lower-resource ones. Therefore, there is no need to set the same parameter budget for all languages when fine-tuning multilingual models. By fine-tuning languages in their respective intrinsic subspaces with different sizes using LSLo, we achieved significant improvements compared to full parameter fine-tuning while greatly reducing the number of trainable parameters. We also proposed methods to search for the optimal placement of LSLo. We showed that the model completes the transformation from the source side to the target side in the top layers of the encoder and that placing the LSLo module in the fully connected layers is most effective in the Transformer architecture.

Limitations

Despite the insights gained from our work, our research still has some limitations.

During the experiments, we categorized languages based on resource types, which is still a relatively coarse classification. We believe that setting individual ranks and pruning ratios for each language could further improve performance and efficiency. Although we did not conduct experiments for all the languages due to time constraints, our proposed optimal architecture search methods can support analysis for each language respectively.

Our experiments only used M2M124-615M Model. We believe that introducing more languages and larger-scale models would yield more interesting findings. However, due to resource and time constraints, it is challenging to use large language models for many-to-many training and conduct comprehensive analysis.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. 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 7319–7328, Online. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Liang Chen, Shuming Ma, Dongdong Zhang, Furu Wei, and Baobao Chang. 2023. On the pareto front of multilingual neural machine translation. *Preprint*, arXiv:2304.03216.

Kevin Duh, Katsuhito Sudoh, Xianchao Wu, Hajime Tsukada, and Masaaki Nagata. 2012. Learning to translate with multiple objectives. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10, Jeju Island, Korea. Association for Computational Linguistics.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21.

Carlos Escolano, Marta R. Costa-jussà, José A. R. Fonollosa, and Mikel Artetxe. 2021. Multilingual machine translation: Closing the gap between shared and language-specific encoder-decoders. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 944–948, Online. Association for Computational Linguistics.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. Beyond english-centric multilingual machine translation. *Preprint*, arXiv:2010.11125.

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. 2024. Mixture-of-LoRAs: An efficient multitask tuning method for large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 11371–11380, Torino, Italia. ELRA and ICCL.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzman, and Angela Fan. 2021. The flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Preprint*, arXiv:2106.03193.

Thanh-Le Ha, Jan Niehues, and Alex Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. In *Proceedings of the 13th International Conference on Spoken Language Translation*, Seattle, Washington D.C. International Workshop on Spoken Language Translation.

Dan He, Minh-Quang Pham, Thanh-Le Ha, and Marco Turchi. 2023. Gradient-based gradual pruning for language-specific multilingual neural machine translation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 654–670, Singapore. Association for Computational Linguistics.

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

Yichong Huang, Xiaocheng Feng, Xinwei Geng, Baohang Li, and Bing Qin. 2023. Towards higher Pareto frontier in multilingual machine translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3802–3818, Toronto, Canada. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2024. VeRA: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*.

Sneha Kudugunta, Ankur Bapna, Isaac Caswell, and Orhan Firat. 2019. Investigating multilingual NMT representations at scale. 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 1565–1575, Hong Kong, China. Association for Computational Linguistics.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. 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 4582–4597, Online. Association for Computational Linguistics.

Zehui Lin, Liwei Wu, Mingxuan Wang, and Lei Li. 2021. Learning language specific sub-network for multilingual machine translation. 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 293–305, Online. Association for Computational Linguistics.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Xinyu Ma, Xuebo Liu, and Min Zhang. 2023. Clustering pseudo language family in multilingual translation models with fisher information matrix. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13794–13804, Singapore. Association for Computational Linguistics.

Alireza Mohammadshahi, Vassilina Nikoulina, Alexandre Berard, Caroline Brun, James Henderson, and Laurent Besacier. 2022. SMaLL-100: Introducing shallow multilingual machine translation model for low-resource languages. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8348–8359, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Jonas Pfeiffer, Naman Goyal, Xi Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe. 2022. Lifting the curse of multilinguality by pre-training modular transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3479–3495, Seattle, United States. Association for Computational Linguistics.

Telmo Pires, Robin Schmidt, Yi-Hsiu Liao, and Stephan Peitz. 2023. Learning language-specific layers for multilingual machine translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14767–14783, Toronto, Canada. Association for Computational Linguistics.

Yujia Qin, Xiaozhi Wang, Yusheng Su, Yankai Lin, Ning Ding, Jing Yi, Weize Chen, Zhiyuan Liu, Juanzi Li, Lei Hou, Peng Li, Maosong Sun, and Jie Zhou. 2022. Exploring universal intrinsic task subspace via prompt tuning. *Preprint*, arXiv:2110.07867.

Zhi Qu and Taro Watanabe. 2022. Adapting to non-centered languages for zero-shot multilingual translation. *Preprint*, arXiv:2209.04138.

Devendra Sachan and Graham Neubig. 2018. Parameter sharing methods for multilingual self-attentional translation models. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 261–271, Brussels, Belgium. Association for Computational Linguistics.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation. *Preprint*, arXiv:2207.04672.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Qian Wang and Jiajun Zhang. 2022. Parameter differentiation based multilingual neural machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11440–11448.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023a. Adalora: Adaptive budget allocation for parameter-efficient finetuning. *Preprint*, arXiv:2303.10512.

Zhong Zhang, Bang Liu, and Junming Shao. 2023b. Fine-tuning happens in tiny subspaces: Exploring intrinsic task-specific subspaces of pre-trained language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1701–1713, Toronto, Canada. Association for Computational Linguistics.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *Preprint*, arXiv:1710.01878.

A Gradual Pruning Schedule

We introduce a Gradual Pruning Schedule (Zhu and Gupta, 2017; He et al., 2023) during fine-tuning to exhaustively explore the minimal intrinsic language-specific subspaces. The entire training process is divided into three stages as denoted in Equation 6. Given a predefined pruning ratio P and the total training process has T epochs. E is the starting epoch for pruning, and the pruning process will last for k epochs.

$$P = \begin{cases} 0 & e \leq E \\ P - P(1 - \frac{e-E}{k})^3 & E < e \leq (E+k) \\ P & (E+k) < e \leq T \end{cases} \tag{6}$$

During the first E epochs ($e \leq E$), no pruning is applied denoted by $P = 0$; for stage 2, the pruning ratio of the current e epoch is gradually increased until reaching the target ratio P for the next k epochs; for stage 3, the pruning ratio P is kept to the end. We use the L1 unstructured pruning.[4]

Experiments	P	E	k	T
ISE	0.7	2	8	15
LSP	0.7	2	8	15
LSLo	{0.1,0.3,0.5,0.7,0.9}	2	8	15

Table 3: Settings of Gradual Pruning Schedule in different experiments.

In Table 3, we show the settings of Gradual Pruning Schedule in different experiments, where ISE denotes Intrinsic Subspace Estimation mentioned in Section 6.2, LSP denotes Language-specific Pruning mentioned in Section 7.2 and LSLo denotes the Language-specific LoRA in Section 6.3. We empirically set the same E, k, and T for all experiments and the same P for ISE and LSP. For LSLo, as shown in Table 1, we experimented with different values of P to explore the possible minimal intrinsic subspace.

B Dataset Setting

The details of *lang12* and *lang30* are reported in Table 4 and Table 5. We follow the resource type classification from FLORES-101 (Goyal et al., 2021) based on available Bitext data through English (Bitext w/En). We use the language code of M2M-124 model. The Language Family information and available Bitext data through English are all from FLORES-101.

Resource Type	Language	Code	Family	Bitext w/ En
High	English	en	Germanic	-
	French	fr	Romance	289M
	German	de	Germanic	216M
	Italian	it	Romance	116M
Medium	Chinese	zh	Sino-Tibetan	37.9M
	Dutch	nl	Germanic	82.4M
	Japanese	ja	Japonic	23.2M
	Korean	ko	Koreanic	7.46M
Very Low	Occitan	oc	Romance	5.11K
	Oriya	or	Indo-Aryan	5K
	Sindhi	sd	Indo-Aryan	21.8K
	Wolof	wo	Nilotic+Other AC	86.9K

Table 4: Details for each language in *lang12*.

Resource Type	Language	Code	Family	Bitext w/ En
High	English	en	Germanic	-
	French	fr	Romance	289M
	German	de	Germanic	216M
	Italian	it	Romance	116M
	Portuguese	pt	Romance	137M
	Russian	ru	Balto-Slavic	127M
	Spanish	es	Romance	315M
Medium	Arabic	ar	Afro-Asiatic	25.2M
	Chinese	zh	Sino-Tibetan	37.9M
	Dutch	nl	Germanic	82.4M
	Hebrew	he	Afro-Asiatic	6.64M
	Hindi	hi	Indo-Aryan	3.3M
	Japanese	ja	Japonic	23.2M
	Korean	ko	Koreanic	7.46M
	Maltese	mt	Afro-Asiatic	5.82M
	Norwegian	no	Germanic	10.9M
Low	Afrikaans	af	Germanic	570K
	Amharic	am	Afro-Asiatic	339K
	Armenian	hy	Other IE	977K
	Hausa	ha	Afro-Asiatic	335K
	Nyanja	ny	Bantu	932K
	Shona	sn	Bantu	877K
	Yoruba	yo	Nilotic+Other AC	171K
	Zulu	zu	Bantu	123K
Very Low	Fula	ff	Nilotic+Other AC	71K
	Kamba	kam	Bantu	50K
	Occitan	oc	Romance	5.11K
	Oriya	or	Indo-Aryan	5K
	Sindhi	sd	Indo-Aryan	21.8K
	Wolof	wo	Nilotic+Other AC	86.9K

Table 5: Details for each language in *lang30*.

C Weight Learning

In this section, we analyze the improvements brought by the structure learned via Weight Learning from Section 6.1. Given the results in Figure 1 that the decoder always focuses on the target side information, we concentrate on comparing different encoder settings. We compared three different encoder settings in Table 6: (1) Weight Learning (WL) as described in Section 6.1, where LSLo modules in the first 9 layers of encoder are source-indexed and in the last 3 layers are

[4]We directly use the implementation from PyTorch. https://pytorch.org/docs/stable/generated/torch.nn.utils.prune.l1_unstructured.html

Methods	#Params	Language Direction									
		H2H	H2M	H2V	M2H	M2M	M2V	V2H	V2M	V2V	AVG
Pre-trained	-	31.76	20.06	5.56	20.71	17.12	3.47	9.24	5.03	0.52	12.26
Ft-all	615M	29.29	20.46	12.53	19.28	17.14	8.95	15.23	11.02	6.66	15.43
2;2;8+WL+GPS(0.9)	15.3M	**33.13**	22.33	12.93	**21.49**	18.58	**9.23**	**16.59**	**11.38**	**7.04**	**16.73**
2;2;8+SRC+GPS(0.9)	15.3M	33.06	**22.40**	12.42	21.41	**18.59**	8.76	16.41	11.24	6.59	16.52
2;2;8+TGT+GPS(0.9)	15.3M	32.97	22.34	**13.05**	21.40	18.53	**9.23**	11.91	7.69	5.05	15.52

Table 6: We compare the spBLEU of different index strategies on $lang12$.

target-indexed; (2) Source Encoder (SRC), where all LSLo modules in encoder are source-indexed; (3) Target Encoder (TGT), where all LSLo modules in encoder are target-indexed. We found that the structure selected through Weight Learning (2;2;8+WL+GPS(0.9) exhibited better overall performance, especially for very-low-resource languages.

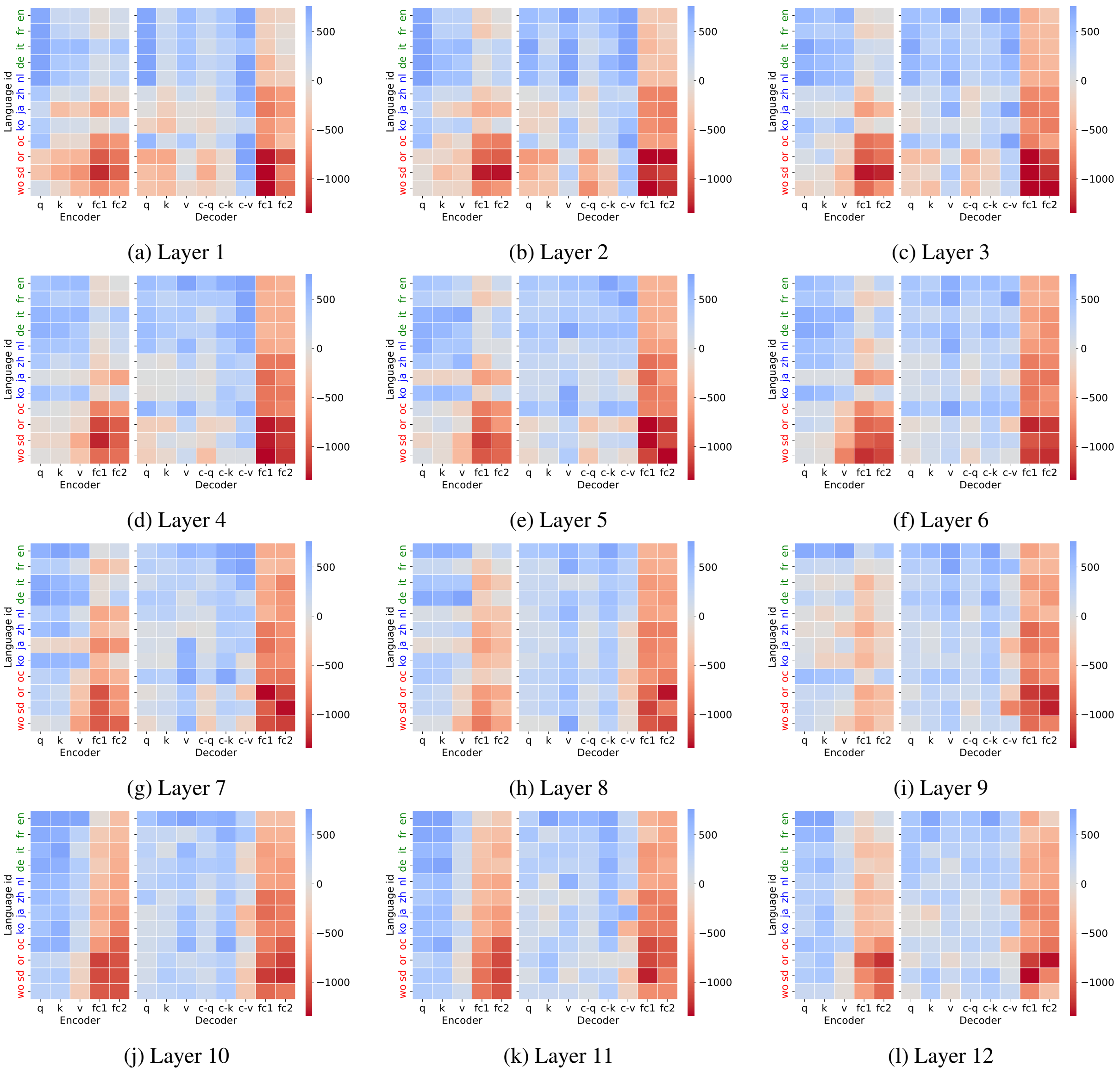
D Intrinsic Subspace Estimation

We present the results of Intrinsic Subspace Estimation in all 12 layers of encoder and decoder in Figure 5. The results show a clear tendency that the required subspace size for each language is highly correlated with its resource type. Very-low-resource languages require more parameters for fine-tuning compared to high and medium-resource languages.

E Language-specific Pruning

Language-specific pruning is applied to analyze the importance of different weight matrices for each language. We add LSLo with a rank of 8 to all weight matrices. Given n languages, each LSLo module will have n language-specific LoRA modules. All B matrices of LoRA are divided into n groups by language. By applying global pruning to each group, we can analyze which weight matrices are most important for each language. As shown in Figure 6, we can see a clear tendency among all languages that fc1 and fc2 play a more important role than other weight matrices.

In Table 7, we compared the results on $lang12$ of applying LSLo to all weight matrices versus only applying it to fc1 and fc2, given a similar parameter budget. We found that applying LSLo only to fc1 and fc2 consistently yields better results. This suggests that, under a limited parameter budget, concentrating parameters in the feed-forward layers are more effective than distributing them across all possible weight matrices.



(a) Layer 1 (b) Layer 2 (c) Layer 3

(d) Layer 4 (e) Layer 5 (f) Layer 6

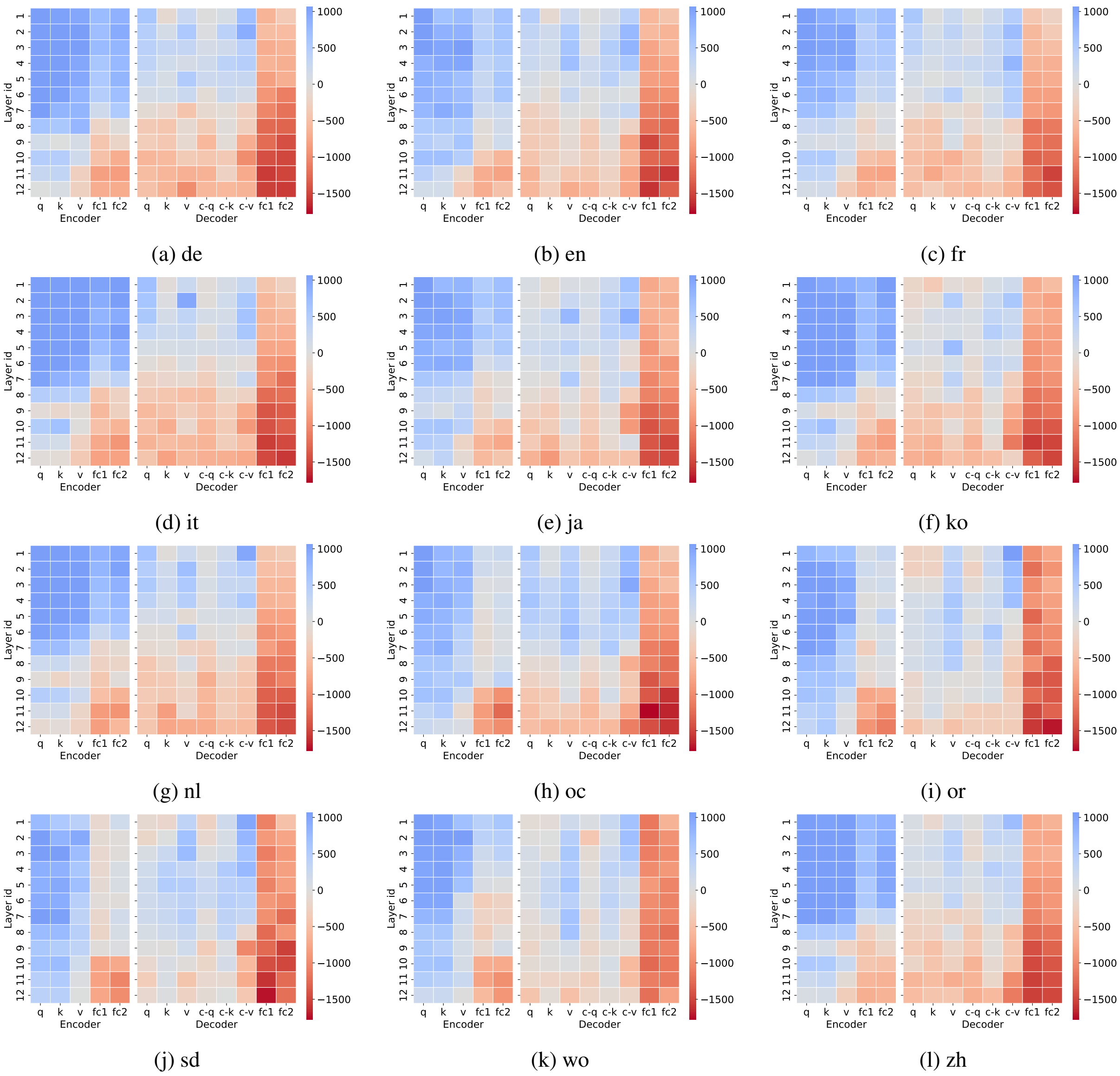
(g) Layer 7 (h) Layer 8 (i) Layer 9

(j) Layer 10 (k) Layer 11 (l) Layer 12

Figure 5: The parameter space demands for each language in all 12 layers of encoder and decoder respectively. Red color means a higher demand. We can see a clear tendency that very-low-resource languages require more parameters during fine-tuning.

Methods	#Params	Language Direction									AVG
		H2H	H2M	H2V	M2H	M2M	M2V	V2H	V2M	V2V	
Pre-trained	-	31.76	20.06	5.56	20.71	17.12	3.47	9.24	5.03	0.52	12.26
Ft-all	615M	29.29	20.46	12.53	19.28	17.14	8.95	15.23	11.02	6.66	15.43
2;2;8+WL+GPS(0.9)*	15.3M	33.13	**22.33**	12.93	21.49	**18.58**	9.23	16.59	11.38	7.04	16.73
2;2;16+WL+GPS(0.9)*	25.6M	33.06	22.27	14.24	21.44	**18.58**	10.49	17.44	12.02	8.42	17.33
2;2;16+WL+GPS(0.9)	7.7M	33.29	22.19	12.64	**21.60**	18.56	8.83	17.33	11.65	6.90	16.76
2;2;32+WL+GPS(0.9)	14.1M	33.24	22.31	14.11	21.50	18.46	10.12	18.19	12.47	8.19	17.44
2;2;64+WL+GPS(0.9)	26.7M	**33.27**	22.26	**14.86**	21.59	18.48	**10.95**	**18.97**	**12.97**	**9.79**	**17.91**

Table 7: We compare the performance on *lang12* of adding LSLo to all modules (with *) versus only adding it to fully connected layers. We found that, given a similar parameter budget, adding LSLo to fc1 and fc2 results in better performance.



(a) de (b) en (c) fr (d) it (e) ja (f) ko (g) nl (h) oc (i) or (j) sd (k) wo (l) zh

Figure 6: Parameter space demands of different languages in encoder and decoder respectively. Red color means a higher demand. We can see a clear trend across all languages that fc1 and fc2 in the top layers of the encoder are more important than other weight matrices.