Towards Modular Fine-tuning of LLM-based Multilingual Neural Machine Translation

Zhe Cao[†,‡] , Yusuke Oda[†,‡] , Akiko Aizawa[‡], Taro Watanabe[†]
[†]Nara Institute of Science and Technology, [‡]NII LLMC
{cao.zhe.bw4,taro}@is.naist.jp, {odashi,aizawa}@nii.ac.jp

Abstract

As Large Language Models support more and more languages, they face increasing challenges in alleviating language inference and adapting to unseen languages. In this work, we propose a modular fine-tuning pipeline for multilingual neural machine translation, where adapters are trained separately for input and output languages. During translation, the parameters of the corresponding input and output language adapters are combined using weighted summation. Experiments on 5 languages show that our methods can reach 50% of full-parameter fine-tuning performance with only 0.5% to 1% trainable parameters. Moreover, under certain weight configurations, merging input and output language adapters outperforms using them individually in some language directions, highlighting the potential of our merging strategy.

1 Introduction

Multilingual Neural Machine Translation (MNMT) focuses on developing a unified model to translate among different languages. Although recent Large Language Models (LLMs) can handle dozens or even hundreds of languages, they also bring challenges related to training efficiency, language interference, and adaptation to unseen languages.

To address these issues, recent research has gradually focused on investigating the modular nature [1] of LLMs. One line of research leverages the sparsity of model parameters by projecting the parameter space into a low-rank, task-specific intrinsic subspace [2, 3, 4] such as Low-rank Adaptation (LoRA) [5]. [6] builds language-specific LoRAs to alleviate language interference in a parameter-efficient way. Another line of research [7] attempts to extract language-specific sub-networks within models and activate different sub-networks during training through masking. Despite effectiveness, these methods require training on all language pairs simultaneously, which limits the model's ability to extend to unseen languages.

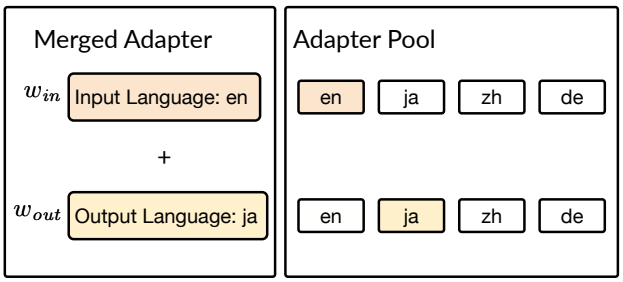


Figure 1: Example of our proposed modular fine-tuning pipeline on four languages: English (en), Japanese (ja), Chinese (zh), and German (de). We divided all adapters into two categories: adapters of input language and adapters of output language. During generation, we directly merge the corresponding two adapters (the colored ones) from the adapter pool.

Therefore, we propose modular fine-tuning of MNMT, a two-step pipeline which first trains adapters for different languages separately and then combines selected adapters directly without any further training. As shown in Figure 1, we categorize adapters into two types: adapters for input language l_{in} and adapters for output language l_{out}. Using these two items, we can describe a translation as $\mathcal{T}(l_{in}, l_{out})$. For each translation, we only choose two adapters from each category separately and then directly merge the parameters of these two adapters through weighting during generation. Given n languages, we can reduce the total number of required adapters from $n \times n$ to $n+n$ compared with fine-tuning for each language direction separately. Additionally, since no additional retraining or retrieval is required, our proposed method can be naturally extended to unseen languages without compromising the

performance of existing languages.

We conduct our experiments on a 5-language subset of FLORES-101 [8]. The results show that we can reach 50% of full-parameter fine-tuning performance with only 0.5% to 0.1% trainable parameters. We also find that under specific weight settings, introducing input language adapters can improve the performance of certain directions, demonstrating the potential of our merging strategy. We further introduce weight learning to analyze the impact of weights on performance, indicating that the output language often plays a more important role in translation tasks.

2 Related Work

Intrinsic Subspace Intrinsic Subspace is the minimal parameter subspace required for models to learn new tasks. [2, 9] showed that the fine-tuning of pre-trained models actually happened in a tiny subspace. Following their work, there is an increasing tendency to explore the modularity within pre-trained models [1] to improve translation performance. [6] built different sized language-specific adapters based on resource-level of languages to alleviate the interference among languages. Another line of work [7] try to locate language-specific neurons inside models and extract sub-network for different languages. Despite effectiveness, these methods require unified training of all langauge-specific modules, which hinders their flexibility in adapting to new languages.

Low-rank Adaptation (LoRA) LoRA [5] 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 [10, 11] have focused on how to further enhance the efficiency of LoRA. [10] modeled LoRA in the form of singular value decomposition and improved efficiency by pruning less important singular values. [11] reduced trainable parameters of LoRA by only leaning scaling vectors during training, fixed low-rank matrices are randomly initialized and shared for each layer. We choose LoRA as our adapter structure thanks to its efficiency and flexibility.

3 Methods

3.1 Multilingual Neural Machine Translation from a Modular Perspective

Given a set of n languages $\mathbb{L} = \{l_1, l_2, \cdots, l_n\}$, multilingual machine translation aims to translate an input sentence $\mathbf{x}$ in the source language $src \in \mathbb{L}$ into an output sentence $\mathbf{y}$ in the target language $tgt \in \mathbb{L}$. With an MNMT dataset including N sentence pairs $\mathbb{D} = \{(\mathbf{x}_i, \mathbf{y}_i), i \in 1 \cdots N\}$, the training loss is defined as:

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

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.

To enable modular fine-tuning, considering the MNMT task between input and output languages, we follow a two-step pipeline, which first train adapters for input languages $\mathbb{A}_{in} = \{\mathbf{A}^{in}_{l_1}, \mathbf{A}^{in}_{l_2}, \cdots, \mathbf{A}^{in}_{l_n}\}$ and output languages $\mathbb{A}_{out} = \{\mathbf{A}^{out}_{l_1}, \mathbf{A}^{out}_{l_2}, \cdots, \mathbf{A}^{out}_{l_n}\}$ separately, and then when translate from a soure language to a target langauge $src \rightarrow tgt$, we directly merge $\mathbf{A}^{in}_{l_{src}}$ and $\mathbf{A}^{out}_{l_{tgt}}$ during generation.

3.2 Training Modular LoRAs

LoRA [5] 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 $\mathbf{W} \in \mathbb{R}^{d\times k}$, LoRA forward pass can be calculated as:

$$\boldsymbol{h} = \mathbf{W}\boldsymbol{x} + \mathbf{B}\mathbf{A}\boldsymbol{x} \qquad (2)$$

where $\mathbf{B} \in \mathbb{R}^{d\times k}$ and $\mathbf{A} \in \mathbb{R}^{r\times d}$. 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 $r \ll min(d, k)$.

We divided LoRA into two categories: LoRAs for input language (LoRA_{in}), and LoRAs for output language (LoRA_{out}), and then we train LoRA_{in}s and LoRA_{out}s for each language separately in a English-centric way, e.g., when translating from Japanese to Chinese, the LoRA_{in} of Japanese will be trained by the data of Japanese $\rightarrow$ English, and the LoRA_{out} of Chinese will be trained by the data of English $\rightarrow$ Chinese. Since LLMs are trained on unbalanced, English-dominated corpora, English often serves as an internal pivot language. Therefore, we select English as the bridge language in our setting to maximize

cross-lingual transfer learning. This setting also makes our pipeline easier to extend to new languages, as introducing a new language requires only preparing data between the new language and English. For convenience, the LoRA_{in} and LoRA_{out} of English will be trained by the data of English → English.

3.3 Merging LoRAs

We compare two different strategies [12] when merging LoRA_{in} and LoRA_{out}.

Merge the outputs of LoRAs In this setting, we merge the outputs of LoRA_{in} and LoRA_{out} as:

$$\begin{aligned} \boldsymbol{h} &= w_{in}\text{LoRA}_{in}(\boldsymbol{x}) + w_{out}\text{LoRA}_{out}(\boldsymbol{x}) \\ &= (w_{in}\mathbf{B}_{in}\mathbf{A}_{in} + w_{out}\mathbf{B}_{out}\mathbf{A}_{out})\boldsymbol{x} \\ &= \text{LoRA}_{merged}(\boldsymbol{x}) \end{aligned} \tag{3}$$

Merge B and A separately In this setting, we merge LoRA_{in} and LoRA_{out} with the following equation:

$$\boldsymbol{h} = (\sqrt{w_{in}}\mathbf{B}_{in} + \sqrt{w_{out}}\mathbf{B}_{out})(\sqrt{w_{in}}\mathbf{A}_{in} + \sqrt{w_{out}}\mathbf{A}_{out})\boldsymbol{x} \tag{4}$$

The two settings have similar training and inference efficiency. The second setting, which separately merges the **A** and **B** matrices, offers finer granularity but requires all LoRAs to share the same rank. In contrast, the first setting directly merges the product of **A** and **B** matrices, potentially losing some information but offering greater flexibility in rank configuration.

3.4 Weight Learning

To further analyze the weight dynamics of LoRA_{in} and LoRA_{out}, we introducee a weight learning method inspired by neural architecture search [13, 6]. Given a pre-trained weight matrix **W** with LoRA_{in} and LoRA_{out}, we calculate a weighted sum during forward pass as follows:

$$\boldsymbol{h} = \mathbf{W}\boldsymbol{x} \quad + w_{in} \cdot \text{LoRA}_{in}(\boldsymbol{x}) \quad + w_{out} \cdot \text{LoRA}_{out}(\boldsymbol{x}) \tag{5}$$

where w_{in}, w_{out} are scalars shared among all LoRA modules in the same layer. We use softmax to make sure the weights are non-negative and sum up to 1. Specifically, $w_{in}, w_{out} = \text{softmax}(w_0, w_1)$, where w_0 and w_1 are initialized to 1.0.

4 Experimental Setup

Dataset FLORES-101 [8] 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. We choose five languages： English (en), French (fr), German (de), Japanese (ja) and Chinese (zh) in our following experiments.

Training We chose Qwen2.5-Instruct-0.5B [14] as our base model. We modified the Transformers [1)] and PEFT [2)] libraries to implement our LoRA settings in the experiments. We fine-tuned the model via Supervised Fine-tuning (SFT) [3)]. For all experiments, we trained the model for 2 Epochs with a learning rate of 0.00002. All models were trained with a single NVIDIA A100-40GB on the mdx [15] cluster.

Evaluation We choose full-parameter fine-tuning as our baseline and set the beam size to 5 during generation. We report the chrF++ score [16].

5 Results

Table 1 shows the chrF++ scores on selected 5 languages. We calculate the averaged score when translating to, e.g., →en, and translating from a specific langauge, e.g., en→. First, we compared the performance of using only LoRA_{in} or LoRA_{out} separately. We found that using LoRA_{out} achieved better results, indicating the target language plays a more important role in multilingual machine translation.

As mentioned in Section 3.3, we compared two merging strategies with different weight configurations: merging the outputs of LoRAs (Setting 1), and merging **A** and **B** separately (Setting 2). As shown in the Talbe 1, we achieved 50% performance of full parameter fine-tuning while using only 0.5% ∼ 1% trainable parameters. Although using more information from LoRA_{out} (1:9) generally yields better results for most language directions (except for →en), we found that under certain weight settings, introducing LoRA_{in} can achieve better performance (underlined scores).

As mentioned in Section 3.4, we showed the results of weight learning in Figure 2 of Appendix A to better understand the weight dynamics of LoRA_{in} and LoRA_{out}.

1) https://github.com/huggingface/transformers
2) https://github.com/huggingface/peft
3) https://huggingface.co/docs/trl/sft_trainer

Table 1: The chrF++ scores on 5 languages: English (en), French (fr), German (de), Japanese (ja), Chinese (zh). We calculate the average scores for a given language l_i as the input language and output language separately, denoted as $l_i \rightarrow$ and $\rightarrow l_i$%. Params means the ratio of trainable parameters compared with full-parameter fine-tuning. For LoRA$_{in}$ and LoRA$_{out}$, we only use the single LoRA module without merging. Setting 1 refers to merging the outputs of LoRA$_{in}$ and LoRA$_{out}$ and Setting 2 refers to merging **A** and **B** separately. The ratio behind shows the weights w_{in} and w_{out} we use for each setting. We underline the scores in Setting 1 and Setting 2 that outperform those achieved by using only LoRA$_{in}$ or LoRA$_{out}$ independently.

		Language Direction									
Methods	%Params	en→	fr→	de→	ja→	zh→	→en	→fr	→de	→ja	→zh
Pre-train	-	30.75	28.85	29.25	27.05	31.6	48.78	38.8	32.55	10.86	16.5
Full-parameter fine-tuning	100%	**34.25**	**33.23**	**33.38**	**30.9**	**33.5**	**51.08**	**42.13**	**34.78**	**16.1**	**21.1**
LoRA$_{in}$	0.5%	30.0	30.53	30.85	27.85	32.0	50.125	39.45	31.95	13.48	16.23
LoRA$_{out}$	0.5%	32.48	31.6	31.45	28.4	30.88	47.25	40.65	34.15	14.475	18.275
Setting 1 (1:9)	1%	32.25	31.7	31.65	28.53	31.5	48.58	40.48	34.075	14.4	18.1
Setting 1 (3:7)	1%	32.05	31.63	31.825	28.6	32.25	49.98	40.5	33.83	14.38	17.68
Setting 1 (7:3)	1%	31.2	31.08	31.45	28.3	32.23	50.4	39.95	32.95	13.9	17.05
Setting 1 (9:1)	1%	30.43	30.65	31.2	28.05	32.1	50.2	39.8	32.28	13.58	16.58
Setting 2 (1:9)	1%	32.78	31.48	31.38	28.03	30.33	45.98	40.6	34.23	14.6	18.58
Setting 2 (3:7)	1%	32.63	31.03	31.08	28.4	30.1	47.45	40.48	34.03	14.45	17.68
Setting 2 (7:3)	1%	30.6	29.93	30.5	28.1	32.28	49.9	39.85	33.05	14.05	14.55
Setting 2 (9:1)	1%	29.85	29.85	30.43	27.75	32.25	50.0	39.55	32.33	13.78	14.48

We observed that for language pairs involving English, the model focuses mainly on the target language information when translating from English to other languages (Figure 2a,2b, 2c, 2d), whereas it emphasizes the source language information when translating from other languages to English (Figure 2e,2i,2m,2q). We attribute this to the English-centric training process, where, for consistency, we included English-to-English data to train the English adapter. However, the results suggest that such data is unnecessary, leading the model to always prioritize directions involving other languages. Besides that, excluding language directions involving English, target information plays a dominant role in 8 out of the remaining 12 directions (Figure 2g,2h,2j,2k,2l,2r,2s,2t). For the remaining four directions (Figure 2f, 2n,2o,2p), we observed that the model's focus shifts from the source language to the target language in the intermediate layers. From the results of weight learning, we infer that the target language generally plays a more important role in machine translation. This observation also explains why assigning greater weight to LoRA$_{out}$ yields better performance.

6 Conclusion

In this work, we propose a modular fine-tuning pipeline for LLM-based Multilingual Neural Machine Translation, which first divide LoRAs into two groups: LoRAs for input languages (LoRA$_{in}$) and LoRAs for output languages (LoRA$_{out}$) and then train these LoRA adapters in an English-centric way. During translation, the corresponding LoRA$_{in}$ and LoRA$_{out}$ are directly merged without any additional training. These enables our pipeline to be easily extended to new languages by training only the LoRA$_{in}$ and LoRA$_{out}$ for the new language, without impacting the performance of existing languages. Our experiments on the FLORES dataset with five languages demonstrate that using only 0.5% to 1% of the trainable parameters achieves 50% performance of full-parameter fine-tuning. Additionally, under specific weight configurations, combining LoRA$_{in}$ and LoRA$_{out}$ yields better results than using them individually, highlighting the potential of this approach.

References

[1] Chaojun Xiao, Zhengyan Zhang, Chenyang Song, Dazhi Jiang, Feng Yao, Xu Han, Xiaozhi Wang, Shuo Wang, Yufei Huang, Guanyu Lin, Yingfa Chen, Weilin Zhao, Yuge Tu, Zexuan Zhong, Ao Zhang, Chenglei Si, Khai Hao Moo, Chenyang Zhao, Huimin Chen, Yankai Lin, Zhiyuan Liu, Jingbo Shang, and Maosong Sun. Configurable foundation models: Building llms from a modular perspective, 2024.

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

[3] 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. Exploring universal intrinsic task subspace via prompt tuning, 2022.

[4] Zhong Zhang, Bang Liu, and Junming Shao. Fine-tuning happens in tiny subspaces: Exploring intrinsic task-specific subspaces of pre-trained language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, **Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 1701–1713, Toronto, Canada, July 2023. Association for Computational Linguistics.

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

[6] Zhe Cao, Zhi Qu, Hidetaka Kamigaito, and Taro Watanabe. Exploring intrinsic language-specific subspaces in fine-tuning multilingual neural machine translation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, **Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing**, pp. 21142–21157, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[7] Shaomu Tan, Di Wu, and Christof Monz. Neuron specialization: Leveraging intrinsic task modularity for multilingual machine translation, 2024.

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

[9] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, **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)**, pp. 7319–7328, Online, August 2021. Association for Computational Linguistics.

[10] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023.

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

[12] Ziyu Zhao, Leilei Gan, Guoyin Wang, Wangchunshu Zhou, Hongxia Yang, Kun Kuang, and Fei Wu. LoraRetriever: Input-aware LoRA retrieval and composition for mixed tasks in the wild. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, **Findings of the Association for Computational Linguistics: ACL 2024**, pp. 4447–4462, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[13] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. **Journal of Machine Learning Research**, Vol. 20, No. 55, pp. 1–21, 2019.

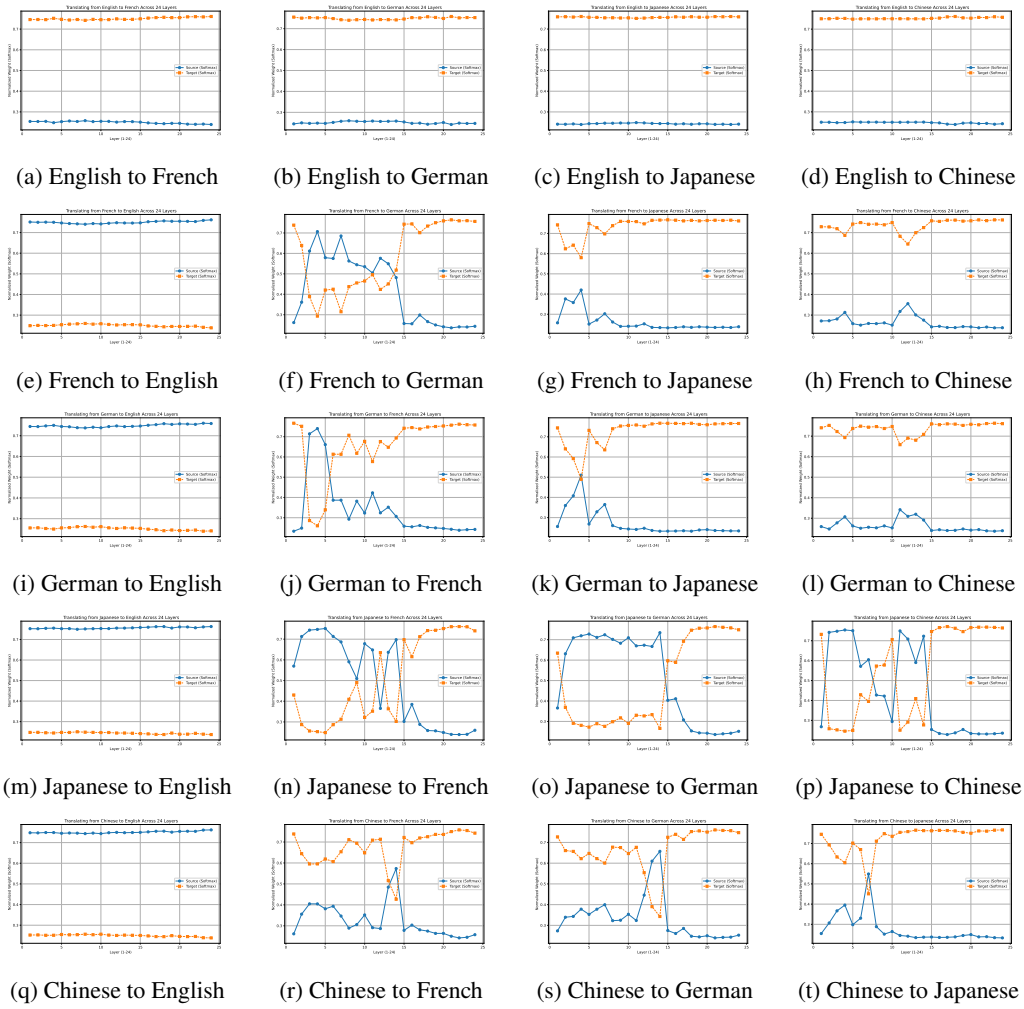
[14] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2024.

[15] Toyotaro Suzumura, Akiyoshi Sugiki, Hiroyuki Takizawa, Akira Imakura, Hiroshi Nakamura, Kenjiro Taura, Tomohiro Kudoh, Toshihiro Hanawa, Yuji Sekiya, Hiroki Kobayashi, Yohei Kuga, Ryo Nakamura, Renhe Jiang, Junya Kawase, Masatoshi Hanai, Hiroshi Miyazaki, Tsutomu Ishizaki, Daisuke Shimotoku, Daisuke Miyamoto, Kento Aida, Atsuko Takefusa, Takashi Kurimoto, Koji Sasayama, Naoya Kitagawa, Ikki Fujiwara, Yusuke Tanimura, Takayuki Aoki, Toshio Endo, Satoshi Ohshima, Keiichiro Fukazawa, Susumu Date, and Toshihiro Uchibayashi. mdx: A cloud platform for supporting data science and cross-disciplinary research collaborations. In **2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)**, pp. 1–7, 2022.

[16] Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In Ondřej Bojar, Rajan Chatterjee, Christian Federmann, Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, and Pavel Pecina, editors, **Proceedings of the Tenth Workshop on Statistical Machine Translation**, pp. 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

A Results of Weight Learning

As shown in Figure 2, we illustrate the trend of weight changes in the model for each language direction after weight learning.



(a) English to French (b) English to German (c) English to Japanese (d) English to Chinese

(e) French to English (f) French to German (g) French to Japanese (h) French to Chinese

(i) German to English (j) German to French (k) German to Japanese (l) German to Chinese

(m) Japanese to English (n) Japanese to French (o) Japanese to German (p) Japanese to Chinese

(q) Chinese to English (r) Chinese to French (s) Chinese to German (t) Chinese to Japanese

Figure 2: Results of Weight Learning