A Lightweight Mixture-of-Experts Neural Machine Translation Model with Stage-wise Training Strategy

Fan Zhang[1,2] and **Mei Tu**[2] and **Song Liu**[2] and **Jinyao Yan**[1*]
[1]State Key Laboratory of Media Convergence and Communication, Communication University of China
[2]Samsung R&D Institute China-Beijing
{zhang.fan, mei.tu, s0101.liu}@samsung.com, jyan@cuc.edu.cn

Abstract

Dealing with language heterogeneity has always been one of the challenges in neural machine translation (NMT). The idea of using mixture-of-experts (MoE) naturally excels in addressing this issue by employing different experts to take responsibility for different problems. However, the parameter-inefficiency problem in MoE results in less performance improvement when boosting the number of parameters. Moreover, most of the MoE models are suffering from the training instability problem. This paper proposes MoA (Mixture-of-Adapters), a lightweight MoE-based NMT model that is trained via an elaborately designed stage-wise training strategy. With the standard Transformer as the backbone model, we introduce lightweight adapters as experts for easy expansion. To improve the parameter efficiency, we explicitly model and distill the language heterogeneity into the gating network with clustering. After freezing the gating network, we adopt the Gumbel-Max sampling as the routing scheme when training experts to balance the knowledge of generalization and specialization while preventing expert over-fitting. Empirical results show that MoA achieves stable improvements in different translation tasks by introducing much fewer extra parameters compared to other MoE baselines. Additionally, the performance evaluations on a multi-domain translation task illustrate the effectiveness of our training strategy.

1 Introduction

In recent years, neural machine translation (NMT), a key component of natural language processing (NLP), has been studied extensively with significant progress (Vaswani et al., 2017; Dabre et al., 2020). Texts from various domains often exhibit unique expression styles. Domain diversity leads to heterogeneous data distribution of a large multi-source dataset. When training an NMT model with the global optimization strategy, data from diverse domains tend to adjust model parameters to fitting their respective distributions, which harms the convergence of the model. In literature, some works (Kobus et al., 2017; Britz et al., 2017; Zeng et al., 2018; Bapna and Firat, 2019; Pham et al., 2020) regarded this problem as domain shift and tried to address it by transfer learning. However, domain knowledge is required in these works, which introduces a new data collection problem. How to deal with the heterogeneity of language in NMT tasks remains challenging.

The core concept of MoE is using multiple experts to divide a problem space into homogeneous regions (Baldacchino et al., 2016), which has a natural advantage in solving the problem of language heterogeneity. Recently, previous works (Shazeer et al., 2017; Fedus et al., 2021; Dai et al., 2022) explored the mixture-of-experts (MoE) structure in NMT tasks. These studies demonstrate the impressive capacity of MoE in handling various data distributions. They boost the number of parameters from million to billion while maintaining low computational requirements. However, MoE is reported to be parameter-inefficient (Hoffmann et al., 2022; Jawahar et al., 2023; Xu et al., 2023a,b) i.e., a huge number of parameters only brings a small performance improvement. As an illustration, compared with a dense model, an MoE model only offers an average improvement of 0.3 BLEU with 20 times more parameters(Costa-jussà et al., 2022).

Meanwhile, training the gating network implicitly by an overall optimization makes most of the MoE models suffer from the training instability problem. It is crucial to meticulously design a training strategy to prevent instability. For instance, expert load imbalance may occur during training of an MoE model: the gating network may route most data to a small number of experts, meanwhile many other experts do not get sufficiently trained at all (Lepikhin et al., 2020). Moreover, the routing fluctuation (Dai et al., 2022) issue, i.e. the gating

Findings of the Association for Computational Linguistics: NAACL 2024, pages 2381–2392

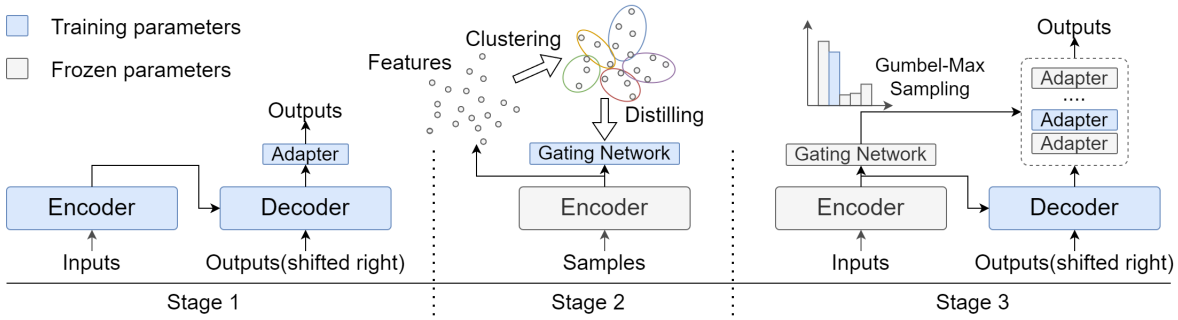


Figure 1: The stage-wise training strategy. MoA is composed of three components: an encoder-decoder based backbone model, a gating network and a set of adapters. The language heterogeneity is modeled explicitly using clustering and distilled into the gating network through a multi-classification task in stage 2 to improve parameter efficiency and ensure training stability, while the Gumbel-Max sampling routing scheme is adopted in stage 3 to balance the knowledge of generalization and specialization and avoid over-fitting. With this training strategy, MoA achieves stable improvements in different translation tasks by introducing very few extra parameters.

network may route the same data to different experts along with training, is also one of the factors leading to training instability.

In this paper, we propose MoA (Mixture-of-Adapters), a lightweight MoE-based NMT model that is trained using a stage-wise training strategy. Our model is composed of three components: (i) an encoder-decoder based backbone model; (ii) a gating network responsible for routing data to suitable experts by their encoded features; (iii) a set of lightweight adapters (Bapna and Firat, 2019) as the experts transplanted in every decoder layer of the backbone model. With the stage-wise training strategy, these three components are trained sequentially. Specifically, the backbone model is trained using a standard machine translation task. Meanwhile, we pre-inject an adapter in every decoder layer in this training stage, and use these adapters for parameter initialization of the other adapters in the adapter training stage. In the training stage of the gating network, the language heterogeneity is modeled explicitly using clustering and distilled into the gating network through a multi-classification task. Such an explicit learning strategy improves the parameter efficiency and ensures the training stability of our model. Moreover, to balance the knowledge of generalization and specialization and prevent the over-fitting problem, we employ the Gumbel-Max sampling as the routing scheme when training the adapters. Empirical results show that MoA achieves stable improvement in different translation tasks by introducing much fewer extra parameters compared to the other MoE baselines. Additionally, the performance evaluations and the ablation studies on the multi-domain translation task illustrate the effectiveness of our training strategy.

2 Related Works

The MoE structure (Jacobs et al., 1991) has been widely studied in the machine translation area (Shazeer et al., 2017; Lepikhin et al., 2020; Dai et al., 2022; Xu et al., 2023b). With the same core concept, different MoE models draw attention to different design strategies.

One difference lies in what to use as experts. Most of the MoE models (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2021) adopt feed-forward networks (FFN) as experts. Based on Transformer (Vaswani et al., 2017), many works (Lepikhin et al., 2020; Fedus et al., 2021; Jawahar et al., 2023) inject extra MoE layers or substitute the FFN layers with MoE layers. Instead of using FFN layers, Zhang et al. (2022) use the attention heads as experts to achieve stronger performance than the standard multi-head attention layer.

Another difference is the training strategy. Shazeer et al. (2017) activate two or more experts to obtain nonzero derivatives for the gating networks in back-propagation. Fedus et al. (2021) only activate one expert per time, they train the gating network by auxiliary losses. Dai et al. (2022) use a two-stage training strategy to address the routing fluctuation problem. Different from the above works that use load balancing loss to prevent expert load imbalance, Lewis et al. (2021) formulate token-to-expert allocation as a linear assignment problem that requires no auxiliary load balancing loss. Liu et al. (2022) propose gating dropout to reduce cross-machine communication and speed

up the training process.

Moreover, according to the granularity of different routing schemes, MoE models can be divided into three levels: token-level, sentence-level, and task-level. Most of the above works adopt token-level schemes, where different experts will be activated for different tokens. The sentence-level routing scheme refers to tokens from a sentence that share the same gating result. When selecting experts by task boundaries as opposed to making input-level decisions, e.g., for multilingual machine translation tasks, the routing scheme is regarded as task-level (Kudugunta et al., 2021).

3 Model Architecture

MoA consists of three components: (i) an encoder-decoder based **backbone model**; (ii) a **gating network** that responsibility is to route data to suitable experts by their encoded features; and (iii) a set of lightweight **adapters** as experts transplanted in the end of every decoder layer of the backbone model.

The **backbone model** is based on the encoder-decoder structure, where the encoder/decoder block is composed of a stack of several identical layers. It theoretically can be any encoder-decoder based model, we use the powerful Transformer (Vaswani et al., 2017) in our experiment. Given a source sentence $x = (x_1, ..., x_n)$, the encoder block maps it to a sequence of hidden states $h = (h_1, ..., h_n)$. Then, h is fed to the decoder block to generate an output sequence $y = (y_1, ..., y_m)$ with an auto-regressive process.

The **gating network** makes use of the hidden states h to discriminate different data distributions. First, $h \in \mathbb{R}^{n \times d}$ is condensed to $\hat{h} \in \mathbb{R}^d$ by mean pooling on the sequence length dimension n,

$$\hat{h} = Pooling(h) \tag{1}$$

Then two linear transformations are introduced with a $tanh$ activation in between to compute adapter scores s,

$$s = tanh(\hat{h}W_1 + b_1)W_2 + b_2 \tag{2}$$

where $W_1 \in \mathbb{R}^{d \times d}$, $W_2 \in \mathbb{R}^{d \times K}$, $b_1 \in \mathbb{R}^d$ and $b_2 \in \mathbb{R}^K$ are the parameter matrices of the linear transformations and K is the predefined adapter number.

The **adapters** transfer the decoded hidden states from generic to specific. Different from previous MoE models using original feed-forward network (FFN) (Vaswani et al., 2017) with large inner dimensions as experts, introducing extra lightweight adapters makes the size of experts can be more flexibly controlled. In each adapter, the output z_i of the i-th decoder layer is first normalized with layer normalization,

$$\tilde{z}_i = LN(z_i) \tag{3}$$

Then $\tilde{z}_i$ is fed to an FFN with a small inner dimension, followed by a residual connection, to obtain the adapter output,

$$o_i = FFN(\tilde{z}_i) + z_i \tag{4}$$

In inference, only the adapter with the biggest score in each decoder layer is activated. Unlike inference, the Gumbel-Max sampling is adopted as the routing scheme in the adapter training stage, which will be discussed in the next section.

4 Stage-wise Training

Most of the MoE models train their gating network along with an overall optimization of the final task. Although some auxiliary losses are introduced to avoid potential risks such as expert load imbalance, this implicit learning approach introduces another discrete latent variable learning problem and increases the training difficulty of the gating networks on how to distinguish different data distributions, which leads to the parameter-inefficiency problem in MoE. In this paper, we train MoA with a stage-wise training strategy. Each training stage is elaborately designed to improve model performance with as few extra parameters as possible. Next, we will discuss our training process in detail.

4.1 Backbone model

The backbone model is trained through a standard machine translation task. Specifically, in this training stage, we inject an adapter in every decoder layer in advance and train them with the backbone model. These pre-injected adapters are used for parameter initialization of the other adapters in the adapter training stage.

Given a dataset of parallel text $D^{mt} = \{(x, y^*)\}_{i=1}^{N_t}$, the training objective is varying the trainable parameters θ to minimize the cross-entropy loss:

$$\mathcal{L}_{mt}(\theta) = -\sum_{i=1}^{N_t}\sum_{t=1}^{m} logP(y_t^*|y_{1:t-1}^*, x; \theta) \tag{5}$$

At this training stage, θ refers to the parameters of the backbone model and the pre-injected adapters.

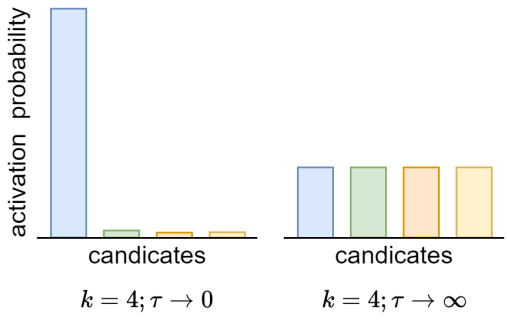


Figure 2: Activation probability controlled by candidate number k and temperature $\tau > 0$. k controls the boundary width while τ controls the probability distribution.

4.2 Gating network

To guide the gating network to explicitly learn the language heterogeneity, it is necessary to model the language heterogeneity first and distill it into the gating network in a supervised manner. Features from the same data distribution are usually closer than those from different distributions (Aharoni and Goldberg, 2020), so the data distribution differences can be modeled by unsupervised clustering. Meanwhile, the encoder in the backbone model can be adopted as the data feature extractor after the previous training stage. Following the clues above, in practice, we first sample a set of source sentences from D^{mt} at random. Then we use the encoder to convert these sentences into the condensed hidden states $\hat{h}$ (Eq. 1) as the sentence features and then cluster them into K groups, where K is the adapter number we pre-defined according to the training data scale. In the end, we distill the clustering results into the gating network through a multi-classification task.

Let $D^d = \{(x, c)\}_{i=1}^{N_d}$ be the training set we construct above where c is the one-hot vector of the data category label, the goal in this training stage is minimizing the multi-classification loss:

$$\mathcal{L}_d(\theta) = -\sum_{i=1}^{N_d}\sum_{j=1}^{K} c_j log(p_j) \tag{6}$$

where

$$p_j = \frac{e^{s_j}}{\sum_{k=1}^{K} e^{s_k}} \tag{7}$$

and θ refers to the parameters in Eq. 2.

4.3 Adapters

To train the adapters, a straightforward scheme is routing data to the adapters with the top-1 highest scores. Since there is a balance between the knowledge of generalization and specialization, this routing scheme is reckless. After freezing the gating network, only choosing the highest-scored adapters makes them trained on a restricted subset of the whole training data, which may result in the overfitting problem. In the adapter training stage, we first use the pre-injected adapters to initialize all other adapters in the same layer. Then we propose routing sentences with the Gumbel-Max sampling scheme (Gumbel, 1954; Maddison et al., 2014). While ensuring the specialization of knowledge, this routing scheme further improves the knowledge generalization of the adapters.

Formally, given the adapter scores s, we focus on k ($k \leq K$) candidates with the highest scores and compute their relative probabilities,

$$p = softmax(topk(s)/\tau) \tag{8}$$

Then the activated adapter is chosen as:

$$e = arg\ max(G(p)) \tag{9}$$

where

$$G(p) = log(p) + g \tag{10}$$

and g is a set of i.i.d samples that are drawn from Gumbel(0,1) distribution (Gumbel, 1954). In Eq. 8, the temperature $\tau > 0$ is introduced to control the probability distribution. The higher the τ, the closer the probability distribution to the discrete uniform distribution, which means candidates will be activated with more similar probabilities. On the contrary, it is closer to the one-hot distribution, which means candidates with the highest scores will be activated with very high probabilities.

The training objective in this stage is the same as the backbone model (Eq. 5), except that θ refers to the parameters of the decoder and the adapters.

5 Experimental Settings

To evaluate the effectiveness of our method, we conduct a set of experiments on both several standard machine translation tasks and a multi-domain machine translation task. The translation quality is measured by the BLEU-4 (Papineni et al., 2002) score. Next, we will provide a comprehensive description of our experimental settings.

5.1 Datasets

For the standard machine translation, we test our method on the German-to-English, the English-to-

	$K = 24$				$K = 12$		$avg.S$
	$Param.$	en-de	de-en	zh-en	$Param.$	en-th	
Backbone	$85M$	28.24	34.31	26.40	$86M$	17.10	26.51
SGMoE	$[+289]M$	28.67	34.49	26.77	$[+138]M$	18.00	26.98
SGMoE-SL	$[+289]M$	**29.17**	34.65	27.60	$[+138]M$	18.00	27.36
Switch	$[+289]M$	28.66	33.92	26.76	$[+138]M$	17.30	26.66
Switch-SL	$[+289]M$	28.83	34.54	27.47	$[+138]M$	18.10	27.24
BASE	$[+302]M$	29.10	34.77	27.53	$[+151]M$	**18.65**	27.51
MoA (Ours)	$[+19]M$	29.13	**34.82**	**27.66**	$[+10]M$	18.50	**27.53**
Backbone-big	$[+165]M$	29.64	35.38	27.29	$[+165]M$	24.00	29.08
MoA-big (Ours)	$[+203]M$	**29.85**	**35.41**	**27.75**	$[+184]M$	**24.10**	**29.28**

Table 1: Performance evaluation over standard machine translation tasks. The average BLEU scores of the four translation tasks are listed in $avg.S$ column. The best values of the same backbone model are shown in bold.

German, the Chinese-to-English, and the English-to-Thai translation tasks. We collect the sentence pairs of the full WMT-2014 German-English (about 36.0 million), the WMT-2019 Chinese-English (about 25.2 million) and the OPUS English-Thai (about 3.3 million, provided by Lowphansirikul et al. (2020)) for corresponding translation tasks, and test the translation tasks of the German-English, the Chinese-to-English and the English-to-Thai by WMT-14, WMT-19 and IWSLT-14 testsets, respectively.

For the multi-domain machine translation, we test our method on the German-to-English multi-domain translation task. We collect two datasets and mix them up as the training set. One is the standard WMT-2014 German-English sentence pairs (about 4.6 million), which can be seen as a large generic domain (*WMT*). Another one is the multi-domain sentence pairs (about 1.5 million) from Aharoni and Goldberg (2020) which is originally provided by Koehn and Knowles (2017), including textual data in five diverse domains: IT-related text (*IT*, manuals and localization files of open-source software), translations of the Koran (*KOR*), legal text (*LAW*, legislative text of the European Union), medical text (*MED*, PDF documents from the European Medicines Agency), and subtitles (*SUB*).

In the data processing phase, the English and the German sentences are first tokenized by Moses tokenizer (Koehn et al., 2007) and then split into subwords by Byte-Pair Encoding (BPE) (Sennrich et al., 2016), where the BPE is learned jointly on the English and German sentences and the merge operation is set to 30,000 during learning. Meanwhile, the Chinese and the Thai sentences are split by SentencePiece (Kudo and Richardson, 2018) with a vocabulary size of 30,000.

5.2 Implementations

We use the Transformer (Vaswani et al., 2017) implemented in Fairseq (Ott et al., 2019) as the backbone model structure. All baseline models are implemented with the backbone model structure, and the experts are only introduced in each decoder layer. According to the data scale of the training set, the expert number K of the German-to-English, the English-to-German, and the Chinese-to-English translation task is set to 24, while that of the English-to-Thai and the multi-domain translation task is set to 12. Next, we will introduce these models briefly.

SGMoE: The Sparsely-gated mixture-of-experts (SGMoE) (Shazeer et al., 2017) is originally based on the LSTM structure (Hochreiter and Schmidhuber, 1997). It introduces sparsely gated MoE layers with the noisy top-k token-level gating scheme, which activates $k > 1$ experts per time to obtain nonzero derivatives in back-propagation. It introduces auxiliary losses to deal with the expert load imbalance. In practice, k is set to 2, and FFN layers are adopted as the experts.

SGMoE-SL: The SGMoE with Sentence-Level routing scheme. The sentence-level routing scheme means we use the condensed hidden states of the encoder (w.r.t. Eq. 1) to compute the overall gating scores and route data in all layers with these scores.

Switch: Switch Transformer (Fedus et al., 2021) is another MoE method with a token-level routing scheme that activates only one expert per time to keep efficiency. It introduces both a capacity factor and an auxiliary load balancing loss to avoid the expert load imbalance.

	$Param.$	*WMT*	*KOR*	*IT*	*MED*	*LAW*	*SUB*	$avg.M$
Backbone	$84M$	32.08	19.65	44.79	51.47	54.34	30.60	38.82
ADPT	$[+9]M$	32.22	**22.48**	45.88	53.58	56.36	**31.97**	40.42
SGMoE	$[+138]M$	32.23	21.54	46.45	53.48	56.85	31.40	40.33
SGMoE-SL	$[+138]M$	32.29	21.76	46.44	53.47	57.09	31.85	40.48
Switch	$[+138]M$	32.05	20.54	46.53	51.51	55.34	30.67	39.44
Switch-SL	$[+138]M$	32.26	20.68	46.43	52.87	56.72	30.94	39.98
BASE	$[+151]M$	**32.59**	22.24	46.36	53.67	57.58	31.55	40.67
MoA (Ours)	$[+10]M$	32.58	22.08	**46.88**	**54.48**	**57.68**	31.50	**40.87**
Backbone-big	$[+165]M$	32.29	22.31	48.11	56.35	59.13	32.08	41.71
MoA-big (Ours)	$[+184]M$	**32.66**	**22.70**	**48.60**	**56.89**	**59.83**	**32.31**	**42.17**

Table 2: Multi-domain translation performance. The average BLEU scores of the six domains are listed in the $avg.M$ column. The best values of the same backbone model are shown in bold. The expert number K is set to 6 for ADPT and 12 for the other MoE models.

Switch-SL: The Switch Transformer with Sentence-Level routing scheme that is the same as SGMoE-SL.

BASE: The Balanced Assignment of Sparse Experts (BASE) layer (Lewis et al., 2021) formulates token-to-expert allocation as a linear assignment problem, which requires no auxiliary load balancing loss. Instead of replacing the original FFN layers, it introduces extra FFN layers after each decoder layer as the experts.

ADPT: Since the domain labels are accessible in multi-domain machine translation tasks, we train a set of adapters (Bapna and Firat, 2019) for every domain by injecting an adapter in every encoder and decoder layer using the same backbone model. All parameters of the backbone model are frozen when training these adapters.

MoA: Our proposed method. In the training stage of the gating network, we sample 200,000 sentences from the NMT training set at random. We choose the Gaussian Mixture Model (GMM) as our clustering approach. The inner dimension of the adapters is set to 128 for both ADPT and MoA in the standard backbone settings, and that is set to 256 for MoA in the big backbone settings. In the training stage of the adapters, the adapter candidate number k and the temperature τ are set to 4 and 1.0, respectively. The Gumbel-Max routing scheme is shut down in inference with $k = 1$.

6 Results and Discussion

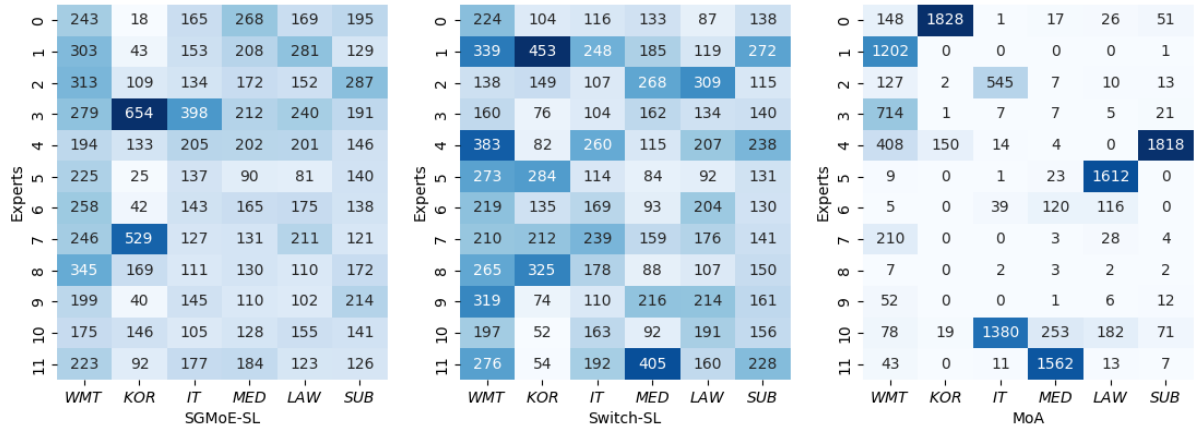
6.1 Standard machine translation

We evaluate the performance of the MoE models over the four standard machine translation tasks and report their BLEU scores in Table 1. For the baseline MoE models, we use Transformer under standard settings as the backbone model. For our method, we evaluate it on the Transformer settings of both standard and big. To show the differences in model size, we present the number of parameters ($Param.$) in Table 1. The $Param.$ number in the setting of $k = 24$ is the average parameter number of the three models.

As shown in Table 1, compared to other MoE models, MoA achieves the highest performance improvement while introducing much fewer parameters. When applying MoA on the big backbone model, it also achieves stable performance improvements. Although other MoE models introduce a huge amount of parameters, even much higher than the backbone model, their performance improvements are limited. Meanwhile, compared to the big backbone model, the parameter-inefficiency problem results in worse model performance for these MoE models with even more parameters. Moreover, methods based on the sentence-level routing scheme (methods with -SL flag) show better model performance than token-level in our experimental settings. It demonstrates that the more gating networks that require implicit training, the more challenging the discrete latent variable learning problem becomes. The discrimination ability of language heterogeneity of these gating networks will be discussed in the next sections.

6.2 Multi-domain machine translation

We further evaluate these MoE models on a multi-domain machine translation task, which has domain labels so that we can analyze the ability of the gating networks to distinguish different data distributions. With the multi-domain machine trans-



(a) SGMoE-SL (b) Switch-SL (c) MoA

Figure 3: Routing statistics of sentence-level MoEs on the test sets of the six domains.

lation task, we test the translation performance of these MoE models in this section and analyze the sentence-level gating networks in the next section. The BLEU scores are reported in Table 2.

As shown in Table 2, the conclusion on average translation performance is consistent with Table 1. Different from the other MoE models, since ADPT is trained with in-domain data per domain, it also requires domain labels in inference to manually route data to the corresponding adapter. Given domain labels, ADPT can be regarded as an MoE model with a label-guided routing scheme. Although ADPT is parameter-efficient, such a routing scheme requires extra data information and introduces the data collection problem. Furthermore, the expert number of ADPT is limited by the known domain number (i.e., ADPT can only introduce $k = 6$ experts to be consistent with the domain number), and the small domains will not take benefits from the big generic dataset (i.e. the *WMT* training set in our experiments), which makes its model performance on some small domains is poorer than MoA.

6.3 Routing results

To analyze the discrimination ability of language heterogeneity of these gating networks through the accessible domain labels, we count the routing results of these sentence-level MoE models on the test sets. Since SGMoE-SL uses top-2 experts for each sentence, we only count the expert with the highest gating score.

Based on the statistics, we roughly measure the discrimination ability by two metrics. One is the category purity score PUR,

$$PUR = \frac{1}{U} \sum_{i=1}^{K} u_i^{max} \tag{11}$$

where U is the total number of the test cases, K is the number of the categories (NOT the number of the test domains), and u_i^{max} is the maximum number of i-th category. The other one is the normalized mutual information NMI (Danon et al., 2005) score between true domain labels and the predicted category labels, as implemented in scikit-learn (Pedregosa et al., 2011). The two metrics measure the mixing degree of different domains in a category. The higher the PUR and the NMI, the better the domain discrimination ability.

	PUR	NMI
SGMoE-SL	0.2855	0.0395
Switch-SL	0.2706	0.0321
MoA	**0.8498**	**0.6480**

Table 3: Measurements of the domain discrimination ability on the test sets of the six domains.

Results in Table 3 show that the domain discrimination ability of our gating network is significantly higher than the other two MoE models. In SGMoE-SL and Switch-SL, the auxiliary load balancing loss makes their routing results relatively balanced. However, the challenging discrete latent variable learning problem is not just a load balancing problem, the domain discrimination results of the two MoE models illustrate that their performance in modeling language heterogeneity is very weak. Their routing decisions result in a very high overlap of their expert knowledge, thus

	$avg.S$	$avg.M$
Backbone	26.51	38.82
Naive MoA	26.92	40.32
+Pre-A	27.35	40.72
++Unf-D	**27.53**	**40.87**

Table 4: Ablation study on different translation tasks. $avg.S$ and $avg.M$ indicate the average BLEU scores of the standard machine translation tasks on different directions and the multi-domain machine translation task on different domains, respectively.

leading to the parameter-inefficiency problem. In contrast, MoA models the language heterogeneity well, different experts are in charge of different domains, which allows it to achieve better translation performance with much fewer parameters. Because the expert (adapter) number is bigger than the known domain number (12 vs. 6), some experts (e.g. expert 8 and 9) are activated very few times by the test sets, they are responsible for the other data distributions beyond the six known domains.

6.4 Ablation study

To further analyze the impact of our training strategy on the model performance, we further conduct a set of ablation experiments on these machine translation tasks.

We conduct experiments of training with/without the pre-injected adapter in stage 1 and freeze/unfreeze decoder parameters in stage 3. We first train a naive MoA without the pre-injected adapter in stage 1 and freeze decoder parameters in stage 3. Then we pre-inject the adapter (+Pre-A) and unfreeze decoder parameters (++Unf-D) step by step. The experimental results are presented in Table 4. After pre-injecting an adapter in every decoder layer and using it for parameter initialization of the other adapters in the same layer, the information gap between newly injected adapters and the backbone model is eliminated. It brings significant performance improvements. After unfreezing the decoder parameters and training them with adapters, MoA achieves a higher average BLEU score. These ablation studies demonstrate the effectiveness of the two training tricks.

In the adapter training stage, we also adopt the Gumbel-Max routing scheme to balance the knowledge of generalization and specialization and avoid the over-fitting problem. The two hyper-parameters, the adapter candidate number k and the temperature τ, control the activation probability between different adapters (w.r.t. Eq. 8). We experiment with adjusting them in the expert training stage. Experimental results are reported in Table 5 and Table 6, respectively.

	$avg.M$
Backbone	38.82
$\tau \to 0.0$	40.40
$\tau = 0.1$	40.58
$\tau = 1.0$	**40.87**
$\tau = 10.0$	40.42

Table 5: Hyper-parameter τ analysis on the multi-domain translation task. $\tau \to 0.0$ is equivalent to shutting down the Gumbel-Max routing scheme.

	$avg.M$
Backbone	38.82
$k = 1$	40.40
$k = 2$	40.64
$k = 4$	**40.87**
$k = 8$	40.67
$k = 12$	40.46

Table 6: Hyper-parameter k analysis on the multi-domain translation task. $k = 1$ is equivalent to shutting down the Gumbel-Max routing scheme.

In Table 5, we fix k to 4 and vary τ to analyze the difference. Meanwhile, the experimental settings in Table 6 are that τ is fixed to 1.0 and k is varied. Both $\tau \to 0.0$ and $k = 1$ are equivalent to shutting down the Gumbel-Max routing scheme, i.e., the routing scheme of only choosing the top-1 highest-scored adapters. It means every adapter is trained with a restricted subset of the whole training set, leading to the over-fitting problem, the model performance is not as good as those with the Gumbel-Max routing scheme. Meanwhile, the moderate values $\tau = 1.0$ and $k = 4$ perform better than the other settings. It demonstrates that there is a balance in the domain knowledge of each expert in specialization and generalization.

7 Conclusion

This paper proposes MoA, a lightweight MoE-based NMT model that is trained via an elaborately designed stage-wise training strategy. The lightweight adapters are introduced as experts for easy expansion. By modeling the language heterogeneity with clustering and distilling the knowledge into the gating network explicitly, MoA improves

the parameter efficiency and avoids training instability. The Gumbel-Max sampling is adopted as the routing scheme when training the adapters to balance the knowledge of generalization and specialization and avoid over-fitting. Empirical results show the effectiveness of the proposed method.

Limitations

The proposed MoA method shows stable improvements in different translation tasks by introducing only a few parameters. However, due to the computational complexity limitation, modeling the language heterogeneity through clustering approaches limits the data scale used for training the gating network. When the data distribution of the sampling sentences deviates from that of the whole dataset, the language heterogeneity may not be modeled very well. Exploring alternative methods to clustering for modeling language heterogeneity should be an interesting direction. Additionally, the Gumbel-Max sampling scheme has been shown to enhance model performance, but its two hyper-parameters are fixed empirically in the current version. In future work, adjusting these two hyper-parameters automatically according to the number of experts and the characteristics of the training set may be better.

References

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7747–7763.

Tara Baldacchino, Elizabeth J Cross, Keith Worden, and Jennifer Rowson. 2016. Variational bayesian mixture of experts models and sensitivity analysis for nonlinear dynamical systems. *Mechanical Systems and Signal Processing*, 66:178–200.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. 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 1538–1548.

Denny Britz, Quoc Le, and Reid Pryzant. 2017. Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 118–126.

Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. 2020. A survey of multilingual neural machine translation. *ACM Computing Surveys (CSUR)*, 53(5):1–38.

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Stablemoe: Stable routing strategy for mixture of experts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7085–7095.

Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. 2005. Comparing community structure identification. *Journal of statistical mechanics: Theory and experiment*, 2005(09):P09008.

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

Emil Julius Gumbel. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. An empirical analysis of compute-optimal large language model training. *Advances in Neural Information Processing Systems*, 35:30016–30030.

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

Ganesh Jawahar, Subhabrata Mukherjee, Xiaodong Liu, Young Jin Kim, Muhammad Abdul-Mageed, VS Laks Lakshmanan, Ahmed Hassan, Sebastien Bubeck, and Jianfeng Gao. 2023. Automoe: Heterogeneous mixture-of-experts with adaptive computation for efficient neural machine translation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9116–9132.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Catherine Kobus, Josep M Crego, and Jean Senellart. 2017. Domain control for neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 372–378.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *EMNLP 2018*, page 66.

Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. 2021. Beyond distillation: Task-level mixture-of-experts for efficient inference. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3577–3599.

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*.

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

Rui Liu, Young Jin Kim, Alexandre Muzio, and Hany Hassan. 2022. Gating dropout: Communication-efficient regularization for sparsely activated transformers. In *International Conference on Machine Learning*, pages 13782–13792. PMLR.

Lalita Lowphansirikul, Charin Polpanumas, Attapol T Rutherford, and Sarana Nutanong. 2020. scb-mt-en-th-2020: A large english-thai parallel corpus. *arXiv preprint arXiv:2007.03541*.

Chris J Maddison, Daniel Tarlow, and Tom Minka. 2014. A* sampling. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 3086–3094.

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.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. 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.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Minh Quang Pham, Josep-Maria Crego, François Yvon, and Jean Senellart. 2020. A study of residual adapters for multi-domain neural machine translation. In *Conference on Machine Translation*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

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*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Haoran Xu, Maha Elbayad, Kenton Murray, Jean Maillard, and Vedanuj Goswami. 2023a. Towards being parameter-efficient: A stratified sparsely activated transformer with dynamic capacity. *arXiv preprint arXiv:2305.02176*.

Haoran Xu, Weiting Tan, Shuyue Stella Li, Yunmo Chen, Benjamin Van Durme, Philipp Koehn, and Kenton Murray. 2023b. Condensing multilingual knowledge with lightweight language-specific modules. *arXiv preprint arXiv:2305.13993*.

Jiali Zeng, Jinsong Su, Huating Wen, Yang Liu, Jun Xie, Yongjing Yin, and Jianqiang Zhao. 2018. Multi-domain neural machine translation with word-level domain context discrimination. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 447–457.

Xiaofeng Zhang, Yikang Shen, Zeyu Huang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2022. Mixture of attention heads: Selecting attention heads per token. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4150–4162.

8 Appendices

8.1 Training details

In any training phase, we use the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. For translation optimization, we use the Noam decay as the learning rate scheduler with 4000 warmup steps and a learning rate of 0.0007. With a batch size of $32k$ in the token level and the update frequency of 5 on 2 A100 GPUs, the maximum update number of training is set to $300k$, while that of fine-tuning is set to $30k$ with the early stopping strategy. The maximum update number of the gating network training stage is set to $10k$ with a batch size of $8k$ in the token level. In inference, the beam size is set to 5 for all models.

8.2 Clustering details

We choose the Gaussian Mixture Model (GMM) in scikit-learn (Pedregosa et al., 2011) as the clustering approach. The convariance type of GMM is set to 'full', while all other settings are set by default. Before clustering, we perform dimensionality reduction with Principal Components Analysis (PCA) to reduce the vector dimension of the sentence representations from 512 to 64.

8.3 Gumbel-Max sampling

We implement the Gumbel-Max sampling strategy with PyTorch (Paszke et al., 2019) of version 1.10.1+cu102. Implementation details are shown in Algorithm 1. It is worth noting that the adapter scores S and the activated adapter indices E are at batch-level compared with that at element-level in subsection 4.3.

8.4 Detailed BLEU scores

We report the detailed BLEU scores of the ablation studies in Table 7, Table 8 and Table 9, respectively.

Algorithm 1: Gumbel-Max sampling of PyTorch version

params : Adapter scores S; Adapter candidate number k; Temperature τ; Activated adapter indices E.

import $torch$;
import $torch.nn.functional$ as F;
if $k <= 1$ **then**
  $E = torch.argmax(S)$;
  return E;
end
$topk\_val, topk\_idx = torch.topk(S, k = k, dim = 1)$;
$topk\_val\ /= \tau$;
$log\_probs = torch.log(F.softmax(topk\_val, dim = 1))$;
$g = F.gumbel\_softmax(log\_probs, dim = 1)$;
$sampled = torch.argmax(g, dim = 1, keepdim = True)$;
$E = torch.gather(topk\_idx, 1, sampled).squeeze()$;
return E;

	en-de	de-en	zh-en	en-th	$avg.S$
Backbone	28.24	34.31	26.40	17.10	26.51
Naieve MoA	28.78	34.45	26.95	17.50	26.92
+Pre-A	29.04	34.62	27.33	18.40	27.35
++Unf-D	**29.13**	**34.82**	**27.66**	**18.50**	**27.53**

Table 7: Ablation study of the two training tricks on the standard translation tasks.

	WMT	*KOR*	*IT*	*MED*	*LAW*	*SUB*	$avg.M$
Backbone	32.08	19.65	44.79	51.47	54.34	30.60	38.82
Naieve MoA	32.45	20.97	46.20	53.96	56.91	31.40	40.32
+Pre-A	32.36	21.87	46.82	54.32	57.39	**31.53**	40.72
++Unf-D	**32.58**	**22.08**	**46.88**	**54.48**	**57.68**	31.50	**40.87**

Table 8: Ablation study of the two training tricks on the multi-domain machine translation task.

		WMT	*KOR*	*IT*	*MED*	*LAW*	*SUB*	AVG
Backbone		32.08	19.65	44.79	51.47	54.34	30.60	38.82
$k = 4$	$\tau \to 0.0$	32.16	22.05	46.25	53.78	56.84	31.32	40.40
	$\tau = 0.1$	32.43	**22.21**	46.30	54.01	57.11	31.43	40.58
	$\tau = 1.0$	**32.58**	22.08	**46.88**	**54.48**	**57.68**	**31.50**	**40.87**
	$\tau = 10.0$	32.56	21.47	46.41	53.96	56.64	31.46	40.42
$\tau = 1.0$	$k = 1$	32.16	22.05	46.25	53.78	56.84	31.32	40.40
	$k = 2$	32.43	**22.45**	46.46	54.37	56.93	31.22	40.64
	$k = 4$	32.58	22.08	**46.88**	**54.48**	**57.68**	31.50	**40.87**
	$k = 8$	32.58	22.22	46.55	54.12	57.01	**31.55**	40.67
	$k = 12$	**32.64**	21.96	46.34	53.67	56.71	31.42	40.46

Table 9: Ablation study of the Gumbel-Max sampling routing scheme on the multi-domain machine translation task.