On the Sparsity of Neural Machine Translation Models

Yong Wang*
The University of Hong Kong
wangyong@eee.hku.hk

Longyue Wang
Tencent AI Lab
vinnylywang@tencent.com

Victor O.K. Li
The University of Hong Kong
vli@eee.hku.hk

Zhaopeng Tu
Tencent AI Lab
zptu@tencent.com

Abstract

Modern neural machine translation (NMT) models employ a large number of parameters, which leads to serious over-parameterization and typically causes the underutilization of computational resources. In response to this problem, we empirically investigate whether the redundant parameters can be reused to achieve better performance. Experiments and analyses are systematically conducted on different datasets and NMT architectures. We show that: 1) the pruned parameters can be rejuvenated to improve the baseline model by up to +0.8 BLEU points; 2) the rejuvenated parameters are reallocated to enhance the ability of modeling low-level lexical information.

1 Introduction

Modern neural machine translation (NMT) (Bahdanau et al., 2015; Gehring et al., 2017; Vaswani et al., 2017) models employ sufficient capacity to fit the massive data well by utilizing a large number of parameters, and suffer from the widely recognized issue, namely, over-parameterization. For example, See et al. (2016) showed that over 40% of the parameters in an RNN-based NMT model can be pruned with negligible performance loss. However, the low utilization efficiency of parameters results in a waste of computational resources (Qiao et al., 2019), as well as renders the model stuck in a local optimum (Han et al., 2017; Yu et al., 2019).

In response to the over-parameterization issue, network pruning has been widely investigated for both computer vision (CV) (Han et al., 2016; Luo et al., 2017) and natural language processing (NLP) tasks (See et al., 2016; Lan et al., 2020). Recent work has proven that such spare parameters can be reused to maximize the utilization of models in CV tasks such as image classification (Han et al., 2017; Qiao et al., 2019). The leverage of parameter rejuvenation in sequence-to-sequence learning, however, has received relatively little attention from the research community. In this paper, we empirically study the efficiency issue for NMT models.

Specifically, we first investigate the effects of weight pruning on advanced Transformer models, showing that 20% parameters can be directly pruned, and by continuously training the sparse networks, we can prune 50% with no performance loss. Starting from this observation, we then exploit whether these redundant parameters are able to be re-utilized for improving the performance of NMT models. Experiments are systematically conducted on different datasets (i.e. Zh⇒En, De⇒En and En⇒Fr) and NMT architectures (i.e. Transformer, RNNSearch and LightConv). Results demonstrate that the rejuvenation approach can significantly and consistently improve the translation quality by up to +0.8 BLEU points. Further analyses reveal that the rejuvenated parameters are reallocated to enhance the ability to model the source-side low-level information, lacking of which leads to a number of problems in NMT models (Tu et al., 2016; Dou et al., 2018; Emelin et al., 2019).

Contributions Our key contributions are:

- We try early attempts to empirically investigate parameter rejuvenation for NMT models across different datasets and architectures.
- We explore to interpret where the gains come from in two perspectives: learning dynamics and linguistic insights.

2 Approach

A standard NMT model directly optimizes the conditional probability of a target sentence $\mathbf{y} = y_1, \ldots, y_J$ given its corresponding source sentence $\mathbf{x} = x_1, \ldots, x_I$, namely $P(\mathbf{y}|\mathbf{x};\theta) =$

*Work was done when interning at Tencent AI Lab.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 1060–1066,
November 16–20, 2020.

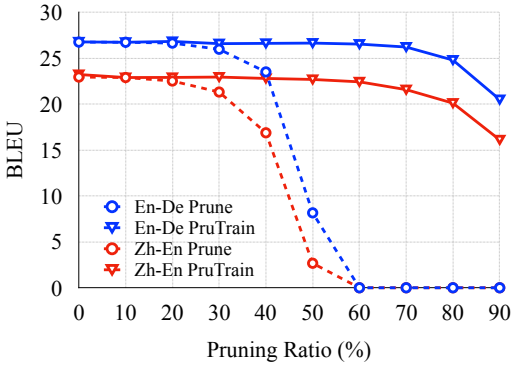

Figure 1: Effects of different pruning ratios on Transformer. "Prune" denotes directly pruning parameters while "PruTrain" indicates adding continuous training after the pruning phase.

$\prod_{j=1}^{J} P(y_j|\mathbf{y}_{<j}, \mathbf{x}; \theta)$, where θ is a set of model parameters and $\mathbf{y}_{<j}$ denotes the partial translation. The parameters of the NMT model are trained to maximize the likelihood of a set of training examples. Given a well-trained NMT model, we first prune its inactive parameters, and then rejuvenate them. Our implementation details are as follows.

Pruning The redundant parameters in neural networks can be pruned according to a certain criterion while the left ones are significant to preserve the accuracy of the model. Specifically, we mask weight connections with low magnitudes in the forward pass and these weights are not updated during optimization. Given the weight matrix W with N parameters, we rank the parameters according to their absolute values. Supposed that the pruning ratio is γ (i.e. $\gamma\%$ of parameters should be pruned), we keep the top n parameters ($n = N \times (1 - \gamma)$), and remove the others with a binary mask matrix, which is the same size of W. We denote the pruned parameters as θ_p, subject to $\theta_p \subset \theta$. There are two pruning strategies (Liu et al., 2019): 1) local pruning, which prunes $\gamma\%$ of parameters in each layer; and 2) global pruning, which compares the importance of parameters across layers. Following See et al. (2016), we retrain the pruned networks after the pruning phase. Specifically, we continue to train the remaining parameters, but maintain the sparse structure, that is we optimize $P(\mathbf{y}|\mathbf{x}; \theta)$ with the constraint: $a = 0, \forall a \in \theta_p$.

Rejuvenation After the pruning and retraining phases, we aim to restore the model capacity by rejuvenating the pruned parameters. This is a common method in optimization to avoid useless computations and further improve performances (Han et al., 2017; Qiao et al., 2019). Thus, we release the sparsity constraint ($a = 0, \forall a \in \theta_p$), which inversely recovers the pruned connections, and redense the whole networks. The recovered weight connections are then initialized by some strategy (e.g. zero or external). The entire networks are retrained with one order of magnitude lower learning rate since the sparse network is already at a good local optimum. As seen, the rejuvenation method contains three phases: 1) training a baseline model (BASE); 2) pruning $\gamma\%$ parameters and then retraining remaining ones (PruTrain); 3) restoring pruned parameters and training entire networks (RejTrain).

Pruning		Rejuvenation		BLEU
local	global	zero	external	
✓	×	✓	×	28.12
×	✓	✓	×	28.08
✓	×	✓	×	28.12
✓	×	×	✓	28.14

Table 1: Effects of different strategies on Transformer on WMT14 En⇒De. "zero" denotes using zero as initialization, while "external" denotes using corresponding parameters in the baseline model as initialization.

3 Experiments

3.1 Setup

Data We conduct experiments on English⇒German (En⇒De), Chinese⇒English (Zh⇒En), German⇒English (De⇒En) and English⇒French (En⇒Fr) translation tasks. For En⇒De task, we use WMT14 corpus which contains 4 million sentence pairs. The Zh⇒En task is conducted on WMT17 corpus, consisting of 21 million sentence pairs. We follow Dou et al. (2018) to select the development and test sets. Furthermore, we evaluate low-resourced translation on IWSLT14 De⇒En and IWSLT17 En⇒Fr corpora. We preprocess our data using byte-pair encoding (Sennrich et al., 2016) with 40K merge operations for En⇒De, 32K for Zh⇒En, and 10K for De⇒En and En⇒Fr, and keep all tokens in the vocabulary. We use 4-gram BLEU score (Papineni et al., 2002) as the evaluation metric and sign-test (Koehn, 2004) for statistical significance.

Models We implement our approach on top of three popular architectures, namely Transformer (Vaswani et al., 2017), RNNSearch (Luong et al., 2015) and LightConv (Wu et al., 2019) with

#	Model	# Para.	BLEU	Δ
1	BASE	108.6M	27.54	–
2	+ ConTrain	108.6M	27.74	+0.20
3	+ RejTrain	108.6M	28.12⇑	+0.58
4	+ RejTrain	108.6M	28.33⇑	+0.79
5	BIG	305.3M	28.55	–
6	+ ConTrain	305.3M	28.81	+0.26
7	+ RejTrain	305.3M	29.12⇑	+0.57

Table 2: Translation quality of Transformer model on WMT14 En⇒De. "# Para." denotes the trainable parameter size of each model. "+" denotes appending new features to the above row. "↑/⇑" indicates statistical significance ($p < 0.05/0.01$) over the baseline.

Data	Model	BLEU	Δ
Zh-En (21M)	BASE	24.18	–
	+ ConTrain	24.35	+0.17
	+ RejTrain	24.60↑	+0.42
De-En (0.16M)	SMALL	30.50	–
	+ ConTrain	30.50	+0.00
	+ RejTrain	30.87⇑	+0.37
En-Fr (0.22M)	SMALL	38.43	–
	+ ConTrain	38.43	+0.00
	+ RejTrain	38.97⇑	+0.54

Table 3: Translation quality of Transformer model on different datasets varied in language pair and size.

the open-source toolkit – fairseq (Ott et al., 2019). For Transformer, we investigate *big*, *base* and *small* settings. About RNNSearch and LightConv, we employ corresponding configurations in fairseq. The implementation is detailed in Appendix §A.1. All baseline models are trained for 100K updates using Adam optimizer (Kingma and Ba, 2015). Based on the baselines, the proposed pruning and rejuvenation methods are trained with additional 100K updates (i.e. 50K for each one). To rule out the circumstance that more training steps may bring improvements, we also conduct continuous training (ConTrain) as strong baselines and they employ the same training steps as our approach.

3.2 Results of Pruning

To study the effect of sparsity, we investigate the effects of different pruning ratios on Transformer *base* models. Experiments are conducted on WMT14 En⇒De and WMT17 Zh⇒En tasks. As shown in Figure 1, over 20% of parameters can be directly pruned without degrading the translation performance. When adding a simple continuous training phase after pruning, we are able to prune 50% with no performance loss. Compared with findings in See et al. (2016), Transformer is less over-parameterized than RNN-based NMT models (20% vs. 40% and 50% vs. 80%). This provides the evidence that different NMT models are over-parameterized to a different extent. Accordingly, we set the pruning threshold of 50% as a default in the following experiments (i.e. Tables 1−4).

3.3 Results of Rejuvenation

Ablation Study As shown in Table 1, we systematically compare different pruning and rejuvenation strategies on the translation task. As seen, the local pruning strategy performs better than the global one, especially with the rejuvenation counterpart (28.12 vs. 28.08 BLEU). However, See et al. (2016) found that the global pruning outperforms the local one without considering rejuvenation factors. Regarding the rejuvenation strategy, zero and external initialization perform similarly in terms of BLEU score. Therefore, we use local pruning and zero initialization strategies for the rest of the experiments (i.e. Tables 2−4).

Main Results We evaluate the rejuvenation approach on the Transformer using En⇒De dataset. As shown in Table 2 (Rows 1−4), our model (RejTrain) outperforms the baseline model and continuous training method (ConTrain) by +0.58 and +0.38 BLEU points, respectively. In addition, iterative rejuvenation can incrementally improve the baseline model up to 28.33 BLEU points (+0.79 and +0.59 over BASE and ConTrain). The results clearly demonstrate the effectiveness of rejuvenating redundant parameters for NMT models.

To verify the robustness, we evaluate different model sizes. As shown in Table 2 (Rows 5−7), the Transformer BIG model performs better than the *base* with an increase of 196.7M parameters. Surprisingly, the performance can be further improved by +0.57 BLEU points by our method. As seen, the continuous training can only slightly gain +0.2 BLEU over BIG, and RejTrain outperforms the strong baseline. This confirms that the rejuvenation method can consistently improve NMT models by alleviating the over-parameterization issue.

Different Datasets Table 3 shows results on three datasets: Zh⇒En, De⇒En and En⇒Fr, cov-

#	Model	Para.	BLEU	Δ
1	Transformer	108.6M	27.54	–
2	+ ConTrain	108.6M	27.74	+0.20
3	+ RejTrain	108.6M	28.12⇑	+0.58
4	RNNSearch	197.0M	22.98	–
5	+ ConTrain	197.0M	22.98	+0.00
6	+ RejTrain	197.0M	23.30	+0.32
7	LightConv	304.2M	28.01	–
8	+ ConTrain	304.2M	28.32	+0.31
9	+ RejTrain	304.2M	28.52⇑	+0.51

Table 4: Translation quality of different NMT models on WMT14 En⇒De.

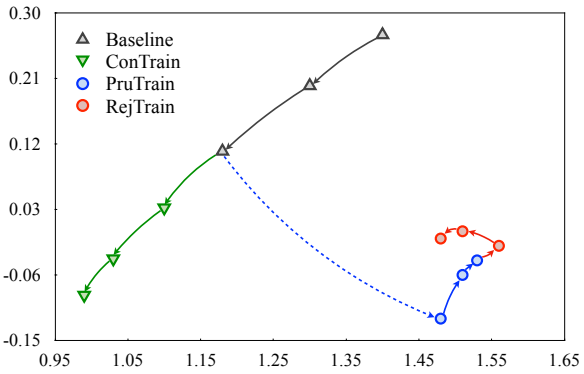

Figure 2: Visualization of encoder representations in different training phases. For each phase, we select sequentially three models. The solid arrow represents the changes in each phase. The dotted arrow represents the changes from the baseline to the pruning phase.

ering large-scale and small-scale training data (i.e. 21M, 0.16M and 0.22M). Trained with large-scale data (Zh⇒En), the continuous training achieves +0.17 BLEU point over the baseline while the rejuvenation approach obtains +0.42 improvement. For low-resource translation (De⇒En and En⇒Fr), ConTrain can not further improve the performance since it is easy to get stuck in a local optimum. However, RejTrain can jump out of local optimum with improved performances (+0.37 and +0.54 over De⇒En and En⇒Fr baselines, respectively). Compared with continuous training, the proposed method significantly and incrementally improves the translation quality in all cases. This again demonstrates the effectiveness of our method across different datasets varied in aspects of language and size.

Different Model Architectures As shown in Table 4, we conduct the experiments on WMT14 En⇒De translation task with RNNSearch, LightConv, Transformer models. Our approach achieves consistent and significant improvements over the baseline and ConTrain models across three architectures. For RNNSearch, continuous training cannot further improve the performance while our model achieves better performance (+0.32 over ConTrain). Furthermore, LightConv works better than the Transformer BASE model since it has 3× more parameters. However, RejTrain still outperforms the ConTrain model and achieves 28.52 BLEU scores. This demonstrates the effectiveness and universality of our approach.

4 Analysis

To better understand the effectiveness of the proposed method, the analyses are carried out in two ways: representation visualization and linguistic probing. Furthermore, we study the translation outputs in terms of adequacy and fluency.

Escaping from Local Optimum To study how our method help models to escape from local optimum, we analyze the change of source representations during different training phases. The analysis is conducted on the Transformer BASE model and En⇒De. Following Zeng et al. (2018), we feed source sentences in the development set into a checkpoint and output an element-wise averaged vector from representations of the last encoder layer. With the dimension-reduction technique of TruncatedSVD (Du et al., 2017), we can plot the dimensionally reduced values in Figure 2. Among the training phases (i.e. Baseline, ConTrain, PruTrain, RejTrain), we select checkpoints at which interval training updates are equal. As seen, within each phase, the representations change smoothly in direction and quantity. The continuous training still transforms the representations in the same direction as the baseline phase (i.e. grey vs. green lines). However, the pruning training dramatically changes the representations (i.e. blue vs. grey lines). Finally, the rejuvenation training jumps to a different place compared with the ConTrain (i.e. red vs. green lines). This demonstrates that our method can efficiently change the direction of optimization, thus providing more chances for the model to escape from the local optimum.

Linguistic Insights We follow Conneau et al. (2018) to conduct the linguistic probing task, which aims to measure the linguistic knowledge embedded in the encoder representations learned by the

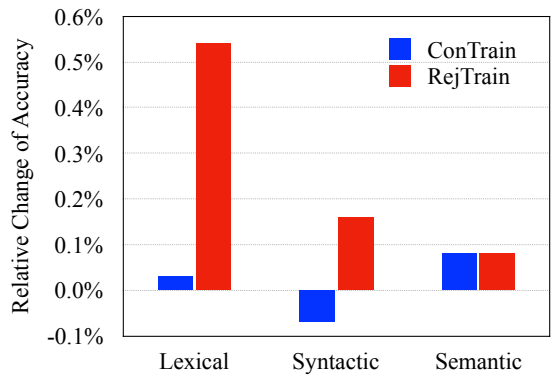

Figure 3: Relative change of performance on linguistic probing tasks compared with the baseline. "Lexical", "Syntactic", "Semantic" denote the averaged accuracy over corresponding tasks in each category.

model. Specifically, it contains 10 classification subtasks with 3 linguistic categories, including lexical, syntactic and semantic ones. We average the predicted accuracies of subtasks in the same category and calculate the relative changes of ConTrain and RejTrain over the baseline model. The analysis is conducted on the Transformer BASE model and En⇒De translation task. As shown in Figure 3, the RejTrain model performs better on the lower level of linguistic subtasks, especially on lexical ones (i.e. 0.6%). The details are listed in Appendix §A.2. The improvements are significant compared with those in Wang et al. (2019a). We hypothesize that better capturing lexical knowledge can improve the adequacy and fluency of translation, which is verified in next part.

Adequacy and Fluency Table 5 shows an example randomly selected from the test set in the Zh⇒En task. As seen, incorporating the rejuvenation approach into NMT can generate more fluent translation with higher adequacy. For instance, the Chinese word "奥运会" is under-translated by the baseline model, while the RejTrain model can correctly translate it into "olympics". Besides, the nominal modifier "21岁的" is mistranslated into a simple number by the baseline while RejTrain can fix the error. This confirms that the rejuvenation improves the adequacy of translation by enhancing the ability to understand the lexical information. To better evaluate the fluency of our models, we calculate the perplexity on the WMT14 En-De test set. As shown in Table 6, the RejTrain model can achieve lower perplexity than baseline and Contrain models (5.08 vs. 5.14/5.15). An interesting finding is that PruTrain increases the perplexity, which

Input	2000 年 悉尼奥运会 , 已经 21 岁 的 刘璇 已经 来到 了 运动员 生涯 的 末期 。
Reference	at the 2000 sydney olympic games , the already 21-year-old liu xuan came to the end of his athlete career .
Baseline	in sydney in 2000 , liu xuan , now 21 , has reached the end of his career as an athlete .
RejTrain	at the 2000 sydney olympics , the 21-year-old liu xuan has reached the end of his career .

Table 5: Example of Zh⇒En translation. Phrases colored in red and blue respectively denote adequacy and fluency problems in baseline but fixed by rejuvenation.

#	Model	PPL
1	BASE	5.14
2	+ ConTrain	5.15
3	+ PruTrain	5.25
4	+ RejTrain	5.08

Table 6: The perplexity of Transformer model on WMT14 En⇒De. "PruTrain" indicates retraining the remaining parameters after the pruning phase. "RejTrain" denotes using the rejuvenation approach.

may harm the fluency of translation outputs (5.25 vs. 5.14). This demonstrates that our rejuvenation approach improves the fluency of translation.

5 Conclusion

In this paper, we prove that existing NMT systems are over-parameterized and propose to improve the utilization efficiency of parameters in NMT models by introducing a rejuvenation approach. Empirical results on a variety of language pairs and architectures demonstrate the effectiveness and universality of the presented method. We also analyze the gains from perspectives of learning dynamics and linguistic probing, which give insightful research directions for future work.

Future directions include continuing the exploration of this research topic for large sequence-to-sequence pre-training models (Liu et al., 2020) and multi-domain translation models (Wang et al., 2019b). We will employ recent analysis methods to better understand the behaviors of rejuvenated models (He et al., 2019; Yang et al., 2020).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *ACL*.

Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. 2018. Exploiting deep representations for neural machine translation. In *EMNLP*.

Simon S Du, Yining Wang, and Aarti Singh. 2017. On the power of truncated svd for general high-rank matrix estimation problems. In *NIPS*.

Denis Emelin, Ivan Titov, and Rico Sennrich. 2019. Widening the representation bottleneck in neural machine translation with lexical shortcuts. In *WMT*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML*.

Song Han, Huizi Mao, and William J Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*.

Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, and John Tran. 2017. Dsd: Dense-sparse-dense training for deep neural networks. In *ICLR*.

Shilin He, Zhaopeng Tu, Xing Wang, Longyue Wang, Michael R Lyu, and Shuming Shi. 2019. Towards understanding neural machine translation with word importance. In *EMNLP*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2019. Rethinking the value of network pruning. In *ICLR*.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. 2017. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

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 *NAACL-HLT: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Siyuan Qiao, Zhe Lin, Jianming Zhang, and Alan Yuille. 2019. Neural rejuvenation: Improving deep network training by enhancing computational resource utilization. In *CVPR*.

Abigail See, Minh-Thang Luong, and Christopher D Manning. 2016. Compression of neural machine translation models via pruning. In *CoNLL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Xing Wang, Zhaopeng Tu, Longyue Wang, and Shuming Shi. 2019a. Self-attention with structural position representations. In *EMNLP*.

Yong Wang, Longyue Wang, Shuming Shi, Victor OK Li, and Zhaopeng Tu. 2019b. Go from the general to the particular: Multi-domain translation with domain transformation networks. In *AAAI*.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *ICLR*.

Yilin Yang, Longyue Wang, Shuming Shi, Prasad Tadepalli, Stefan Lee, and Zhaopeng Tu. 2020. On the sub-layer functionalities of transformer decoder. In *EMNLP*.

Hongfei Yu, Xiaoqing Zhou, Xiangyu Duan, and Min Zhang. 2019. Layer-wise de-training and re-training for convs2s machine translation. In *TALLIP*.

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 *EMNLP*.

A Supplemental Material

A.1 Experimental Setup

In the model configuration of Transformer, the BASE and BIG models differ in the hidden layer size (512 vs. 1024), filter size (2048 vs. 4096) and the number of attention heads (8 vs. 16). The encoder and decoder are composed of a stack of 6 layers. The best model parameters are determined based on the model performance on the development set. All the models are trained on 8 NVIDIA P40 GPUs where each is allocated with a batch size of 4,096 tokens. For IWSLT14 De⇒En and IWSLT17 En⇒Fr tasks, we use the SMALL model, where the encoder and decoder are composed of a stack of 2 layers respectively and which is trained on 1 GPU with a batch size of 4,096 tokens.

Model		BASE	ConTrain	RejTrain
Lexical	SeLen	91.35%	91.40%	91.54%
	WC	75.96%	75.98%	76.85%
	Avg.	83.66%	83.69%	84.20%
Syntactic	TeDep	44.58%	44.61%	44.67%
	ToCo	76.89%	76.71%	77.25%
	BShif	72.18%	72.11%	72.20%
	Avg.	64.55%	64.48%	64.71%
Semantic	Tense	87.61%	87.77%	88.04%
	SubN	85.25%	85.18%	85.03%
	ObjN	84.79%	84.67%	84.57%
	SoMo	53.60%	53.30%	53.26%
	CoIn	60.85%	61.58%	61.62%
	Avg.	74.42%	74.50%	74.50%

Table 7: Performance on the linguistic probing tasks of evaluating linguistics embedded in the encoder outputs. "BASE", "ConTrain" and "RejTrain" respectively denote the baseline model, continuous training and rejuvenation training. "Avg." denotes the average accuracy of each category.

A.2 Probing Task

In order to gain linguistic insights into the learned representations when carrying out the rejuvenation method, we conducted 10 probing tasks (Conneau et al., 2018) to evaluate linguistics knowledge embedded in the final encoding representation learned by the model, as shown in Table 7. From the table, we can see that RejTrain can capture more lexical (84.20% vs. 83.66%) and syntactic (64.71% vs. 64.55%) information.