Pruning-then-Expanding Model for Domain Adaptation of Neural Machine Translation

Shuhao Gu[1,2], **Yang Feng**[1,2]*, **Wanying Xie**[1,3]
[1] Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS)
[2] University of Chinese Academy of Sciences
[3] Beijing Language and Culture University, China
{gushuhao19b,fengyang}@ict.ac.cn, xiewanying07@gmail.com

Abstract

Domain Adaptation is widely used in practical applications of neural machine translation, which aims to achieve good performance on both general domain and in-domain data. However, the existing methods for domain adaptation usually suffer from catastrophic forgetting, large domain divergence, and model explosion. To address these three problems, we propose a method of "divide and conquer" which is based on the importance of neurons or parameters for the translation model. In this method, we first prune the model and only keep the important neurons or parameters, making them responsible for both general-domain and in-domain translation. Then we further train the pruned model supervised by the original whole model with knowledge distillation. Last we expand the model to the original size and fine-tune the added parameters for the in-domain translation. We conducted experiments on different language pairs and domains and the results show that our method can achieve significant improvements compared with several strong baselines.

1 Introduction

Neural machine translation (NMT) models (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015; Gehring et al., 2017; Vaswani et al., 2017) are data-driven and hence require large-scale training data to achieve good performance (Zhang et al., 2019a). In practical applications, NMT models usually need to produce translation for some specific domains with only a small quantity of in-domain data available, so domain adaptation is applied to address the problem. A typical domain adaptation scenario as discussed in Freitag and Al-Onaizan (2016) is that an NMT model have been trained with large-scale general-domain data and then is adapted to specific domains, hoping the model can fit in-domain data well meanwhile the performance will not degrade too much on the general domain.

Towards this end, many researchers have made their attempts. The fine-tuning method (Luong and Manning, 2015) performs in-domain training based on the general-domain model by first training the model on general-domain data and then continuing to train on in-domain data. Despite its convenience for use and high-quality for in-domain translation, this method suffers from catastrophic forgetting which leads to poor performance in the previous domains. Regularization-based methods (Dakwale and Monz, 2017; Thompson et al., 2019; Barone et al., 2017; Khayrallah et al., 2018) instead introduce an additional loss to the original objective so that the translation model can trade off between general-domain and in-domain. This kind of methods usually has all the parameters shared by general-domain and in-domain, with the assumption that the optimal parameter spaces for all the domains will overlap with each other, and retaining these overlapped parameters can balance over all the domains. This assumption is feasible when the domains are similar, but when the divergence of the domains is large, it is not reasonable anymore. In contrast, the methods with domain-specific networks (Dakwale and Monz, 2017; Wang et al., 2019; Bapna and Firat, 2019; Gu et al., 2019) can be often (but not always) immune to domain divergence as it can capture domain-specific features. But unfortunately, as the number of domains increases, the parameters of this kind of methods will surge. Besides, the structure of these networks needs to be carefully designed and tuned, which prevents them from being used in many cases.

Given the above, we propose a method of domain adaptation that can not only deal with large domain divergence during domain transferring but also keep a stable model size even with multiple domains. Inspired by the analysis work on NMT (Bau

*Corresponding author: Yang Feng.
Reproducible code: https://github.com/ictnlp/PTE-NMT.

et al., 2019; Voita et al., 2019; Gu and Feng, 2020), we find that only some important parameters in a well-trained NMT model play an important role when generating the translation and unimportant parameters can be erased without affecting the translation quality too much. According to these findings, we can preserve important parameters for general-domain translation, while tuning unimportant parameters for in-domain translation. To achieve this, we first train a model on the general domain and then shrink the model with neuron pruning or weight pruning methods, only retaining the important neurons/parameters. To ensure the model can still perform well on general-domain data, we adjust the model on in-domain data with knowledge distillation where the original whole model is used as the teacher and the pruned model as the student. Finally, we expand the model to the original size and fine-tune the added parameters on the in-domain data. Experimental results on different languages and domains show that our method can avoid catastrophic forgetting on general-domain data and achieve significant improvements over strong baselines on multiple in-domain data sets.

Our contributions can be summarized as follows:

- We prove that the parameters that are unimportant for general-domain data can be utilized to improve in-domain translation quality.
- Our model can keep superior performance over baselines even when continually transferring to multiple domains.
- Our model can fit in the continual learning scenario where the data for the previous domains cannot be got anymore which is the common situation in practice.

2 Background

2.1 The Transformer

In our work, we apply our method in the framework of TRANSFORMER (Vaswani et al., 2017) which will be briefly introduced here. However, we note that our method can also be combined with other NMT architectures. We denote the input sequence of symbols as $\mathbf{x} = (x_1, \ldots, x_J)$, the ground-truth sequence as $\mathbf{y}^* = (y_1^*, \ldots, y_{K*}^*)$ and the translation as $\mathbf{y} = (y_1, \ldots, y_K)$.

The Encoder & Decoder The encoder is composed of N identical layers. Each layer has two sublayers. The first is a multi-head self-attention sublayer and the second is a fully connected feed-forward network. Both of the sublayers are followed by a residual connection operation and a layer normalization operation. The input sequence $\mathbf{x}$ will be first converted to a sequence of vectors $\mathbf{E}_x = [E_x[x_1]; \ldots; E_x[x_J]]$ where $E_x[x_j]$ is the sum of word embedding and position embedding of the source word x_j. Then, this sequence of vectors will be fed into the encoder and the output of the N-th layer will be taken as source hidden states. and we denote it as $\mathbf{H}$. The decoder is also composed of N identical layers. In addition to the same kind of two sublayers in each encoder layer, the cross-attention sublayer is inserted between them, which performs multi-head attention over the output of the encoder. The final output of the N-th layer gives the target hidden states $\mathbf{S} = [\mathbf{s}_1; \ldots; \mathbf{s}_{K*}]$, where $\mathbf{s}_k$ is the hidden states of y_k.

The Objective We can get the predicted probability of the k-th target word over the target vocabulary by performing a linear transformation and a softmax operation to the target hidden states:

$$p(y_k|\mathbf{y}_{<k}, \mathbf{x}) \propto \exp(\mathbf{W}_o \mathbf{s}_k + \mathbf{b}_o), \qquad (1)$$

where $\mathbf{W}_o \in \mathbb{R}^{d_{model} \times |\mathrm{V}_t|}$ and $|\mathrm{V}_t|$ are the size of target vocabulary. The model is optimized by minimizing a cross-entropy loss of the ground-truth sequence with teacher forcing training:

$$\mathcal{L}(\theta) = -\frac{1}{K} \sum_{k=1}^{K} \log p(y_k^*|\mathbf{y}_{<k}, \mathbf{x}; \theta), \qquad (2)$$

where K is the length of the target sentence and θ denotes the model parameters.

2.2 Knowledge Distillation

Knowledge Distillation (KD) method (Hinton et al., 2015) is for distilling knowledge from a teacher network to a student network. Normally, the teacher network is considered to be with higher capability. A smaller student network can be trained to perform comparably or even better by mimicking the output distribution of the teacher network on the same data. This is usually done by minimizing the cross entropy between the two distributions:

$$\begin{aligned}\mathcal{L}_{\mathrm{KD}}(\theta, \theta_T) = -\frac{1}{K} \sum_{k=1}^{K} q(\mathbf{y}_k|\mathbf{y}_{<k}, \mathbf{x}; \theta_T) \\ \times \log p(\mathbf{y}_k|\mathbf{y}_{<k}, \mathbf{x}; \theta),\end{aligned} \qquad (3)$$

where q denotes the output distribution of the teacher network and θ and θ_T denote the parameters of the student and teacher network, respectively.

The parameters of the teacher network usually keep fixed during the KD process.

3 Method

The main idea of our method is that different neurons or parameters have different importance to the translation model and hence different roles in domain adaptation. Based on this, we distinguish them into important and unimportant ones and make important neurons or parameters compromise between domains while unimportant ones focus on in-domain. Specifically, our method involves the following steps shown in Figure 1. First, we train a model on the general domain and then evaluate the importance of different neurons or parameters. Then we erase the unimportant neurons or parameters and only keep the ones that are related to the general domain so that our method will not be subjected to domain divergence. Next, we further adjust our model under the framework of knowledge distillation (Hinton et al., 2015) on the in-domain with the unpruned model as the teacher and the pruned model as the student. In this way, the pruned model can regain some of its lost performance because of pruning. Finally, we expand the pruned model to the original size and fine-tune the added parameters for the in-domain.

3.1 Model Pruning

Model pruning aims to find a good subset of neurons and parameters of the general-domain model while maintaining the original performance as much as possible. Therefore, under the premise of retaining most of the model's capability, we want to remove those unimportant neurons or parameters to reduce the size of the whole model first. To achieve this, we adopt two pruning schemes. The first is neuron pruning, where we evaluate the importance of neurons directly and then prune unimportant neurons and relevant parameters. The second is weight pruning, where we evaluate and prune each parameter directly.

Neuron Pruning To evaluate the importance of each neuron, we adopt a criterion based on the Taylor expansion (Molchanov et al., 2017), where we directly approximate the change in loss when removing a particular neuron. Let h_i be the output produced from neuron i and H represents the set of other neurons. Assuming the independence of each neuron in the model, the change of loss when removing a certain neuron can be represented as:

$$|\Delta\mathcal{L}(h_i)| = |\mathcal{L}(H, h_i = 0) - \mathcal{L}(H, h_i)|, \quad (4)$$

where $\mathcal{L}(H, h_i = 0)$ is the loss value if the neuron i is pruned and $\mathcal{L}(H, h_i)$ is the loss if it is not pruned. For the function $\mathcal{L}(H, h_i)$, its Taylor expansion at point $h_i = a$ is:

$$\mathcal{L}(H, h_i) = \sum_{n=0}^{N} \frac{\mathcal{L}^n(H, a)}{n!}(h_i - a)^n + R_N(h_i), \quad (5)$$

where $\mathcal{L}^n(H, a)$ is the n-th derivative of $\mathcal{L}(H, h_i)$ evaluated at point a and $R_N(h_i)$ is N-th remainder. Then, approximating $\mathcal{L}(H, h_i = 0)$ with a first-order Taylor polynomial where h_i equals zero:

$$\mathcal{L}(H, h_i = 0) = \mathcal{L}(H, h_i) - \frac{\partial\mathcal{L}(H, h_i)}{\partial h_i} h_i - R_1(h_i). \quad (6)$$

The remainder R_1 can be represented in the form of Lagrange:

$$R_1(h_i) = \frac{\partial^2\mathcal{L}(H, h_i)}{\partial^2 \delta h_i} h_i^2, \quad (7)$$

where $\delta \in (0, 1)$. Considering the use of ReLU activation function in the model, the first derivative of loss function tends to be constant, so the second order term tends to be zero in the end of training. Thus, we can ignore the remainder and get the importance evaluation function as follows:

$$\Theta_{\mathrm{TE}}(h_i) = |\Delta\mathcal{L}(h_i)| = \left|\frac{\partial\mathcal{L}(H, h_i)}{\partial h_i} h_i\right|. \quad (8)$$

In practice, we need to accumulate the product of the activation and the gradient of the objective function w.r.t to the activation, which is easily computed during back-propagation. Finally, the evaluation function is shown as:

$$\Theta_{\mathrm{TE}}(h_i^l) = \frac{1}{T}\sum_t \left|\frac{\delta\mathcal{L}(H, h_i^l)}{\delta h_i^l} h_i^l\right|, \quad (9)$$

where h_i^l is the activation value of the i-th neuron of l-th layer and T is the number of the training examples. The criterion is computed on the general-domain data and averaged over T. Finally, we prune a certain percentage of neurons and relevant parameters in each target layer based on this criterion.

Weight Pruning We adopt the magnitude-based weight pruning scheme (See et al., 2016), where

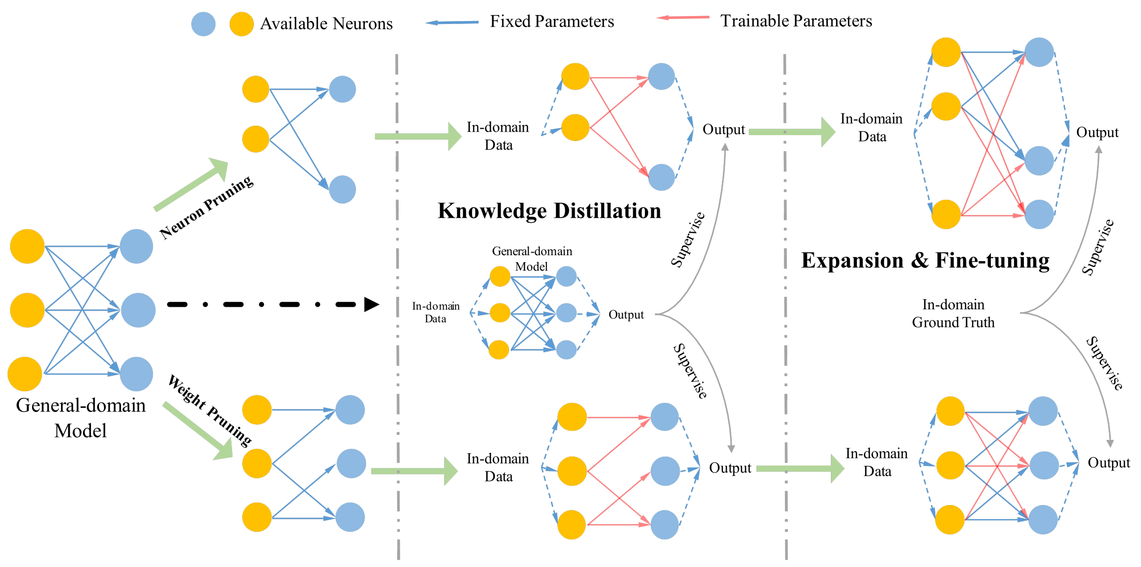


Figure 1: The whole training process of the proposed method.

the absolute value of each parameter in the target matrix is treated as the importance:

$$\Theta_{\mathrm{AV}}(\mathrm{w}_{mn}) = |\mathrm{w}_{mn}|, \mathrm{w}_{mn} \in \mathbf{W}, \tag{10}$$

where w_{mn} denotes the m-th row and n-th column parameter of the weight matrix $\mathbf{W}$. The weight matrix $\mathbf{W}$ represents different parts of the model, e.g., embedding layer, attention layer, output layer, etc. Finally, a certain percentage of parameters in each target parameter matrix are pruned.

3.2 Knowledge Distillation

Though only limited degradation will be brought in performance after removing the unimportant neurons or parameters, we want to further reduce this loss. To achieve this, we minimize the difference in the output distribution of the unpruned and pruned model. In this work, the general-domain model (parameters denoted as θ_G^*) acts as the teacher model and the pruned model (parameters denoted as θ_G) acts as the student model. So, the objective in this training phase is:

$$\mathcal{L}_{\mathrm{KD}}(\theta_G, \theta_G^*) = -\frac{1}{K}\sum_{k=1}^{K} q(\mathbf{y}_k|\mathbf{y}_{<k}, \mathbf{x}; \theta_G^*) \times \log p(\mathbf{y}_k|\mathbf{y}_{<k}, \mathbf{x}; \theta_G). \tag{11}$$

Considering that the general-domain data is not always available in some scenarios when adapting the model to new domains, e.g., continual learning, we adopt the word-level knowledge distillation method using the **in-domain data**. Because the teacher model is trained on general-domain, it can still transfer the general-domain knowledge to the student model even with the in-domain data. We can fine-tune the pruned model on general-domain if the data is available which can simplify the training procedure. We have also tried the sentence-level knowledge distillation method, but the results are much worse. The parameters of the teacher model keep fixed during this training phase and the parameters of the pruned model are updated with this KD loss. After convergence, the parameters of the pruned model (θ_G) will be solely responsible for the general-domain and will also participate in the translation of in-domain data. These parameters will be kept fixed during the following training phase, so our model won't suffer catastrophic forgetting on the general-domain during the fine-tuning process.

3.3 Model Expansion

After getting the well-trained pruned model, we add new parameters (denoted as θ_I) to it, which expands the model to its original size. Then we fine-tune these newly added parameters with in-domain data, which is supervised by the ground truth sequences. As we have indicated above, the parameters of the pruned model (denoted as θ_G), which are responsible for generating the general-domain translation, keep fixed during this training phase. The objective function is:

$$\mathcal{L}(\theta_G, \theta_I) = -\frac{1}{K}\sum_{k=1}^{K} \log p(y_k^*|\mathbf{y}_{<k}, \mathbf{x}; \theta_G, \theta_I). \tag{12}$$

After convergence, the parameters of the pruned model (θ_G) and new parameters (θ_I) are combined together for generating the in-domain translation.

4 Experiments

4.1 Data Preparation

Chinese→English. For this task, the general-domain data is from WMT 2017 Zh-En translation task that contains 23.97M sentence pairs. The data is mainly related to the **News** domain. The newsdev2017 and newstest2017 are chosen as the development and test set, respectively. We choose the parallel sentences with the domain label **Thesis** from the UM-Corpus (Tian et al., 2014) as our in-domain data. This portion covers 15 journal topics in the research area. We filter out the duplicate sentences and then choose 75K, 1K, and 1K sentences randomly as our training, development, and test data, respectively. We tokenize and truecase the English sentences with Moses scripts.[1] For the Chinese data, we perform word segmentation by using Stanford Segmenter.[2]

English→French. For this task, the general-domain data is from the UN corpus of the WMT 2014 En-Fr translation task that contains 12.78M sentence pairs, which are mainly related to the **News** domain. We choose newstest2013 and newstest2014 as our development and test set, respectively. The in-domain data with 53K sentence pairs are from WMT 2019 biomedical translation task, and it is mainly related to the **Biomedical** domain. We choose 1K and 1K sentences randomly from the corpora as our development and test data, respectively. We tokenize and truecase the corpora.

English→German. For this task, general-domain data is from the WMT16 En-De translation task which is mainly **News** texts. It contains about 4.5M sentence pairs. We choose the newstest2013 for validation and newstest2014 for test. For the in-domain data, we use the parallel training data from the IWSLT 2015 which is mainly from the **Spoken** domain. It contains about 194K sentences. We choose the 2014test for validation and the 2015test for test. We tokenize and truecase the corpora.

Besides, integrating operations of 32K, 32K, and 30K are performed to learn BPE (Sennrich et al., 2016) on the general-domain data and then applied to both the general-domain and in-domain data. Then we filter out the sentences which are longer than 128 sub-words. For the Zh-En translation task, 44K size of the Chinese dictionary and 33K size of the English dictionary are built based on the general-domain data. For the En-Fr and En-De tasks, 32K size of the dictionaries for the source and target languages are also built on the corresponding general-domain data.

[1] http://www.statmt.org/moses/

[2] https://nlp.stanford.edu/

4.2 Systems

We use the open-source toolkit called *Fairseq-py* (Ott et al., 2019) released by Facebook as our Transformer system. The contrast methods can be divided into two categories. The models of the first category are capacity-fixed while the second category are capacity-increased. The first category includes the following systems:

• **General** This baseline system is trained only with the general-domain training data.

• **In** This baseline system is trained only with the in-domain training data.

• **Fine-tuning** (Luong and Manning, 2015) This method just continues to train the general-domain model with the in-domain data.

• **SeqKD** (Kim and Rush, 2016) The in-domain source sentences are first translated by the general-domain model. Then the model is further trained with the combined pseudo and real data.

• **Multi-objective Learning (MOL)** (Dakwale and Monz, 2017) This method is based on the Fine-tuning method. Besides minimizing the loss between the ground truth words and the output distribution of the network, this method also minimizes the cross-entropy between the output distribution of the general-domain model and the network. The final objective is:

$$\mathcal{L}_{\text{MOL}}(\theta) = \mathcal{L}(\theta) + \alpha \mathcal{L}_{\text{KD}}(\theta) \qquad (13)$$

where α is the hyper-parameter which controls the contribution of the two parts. The bigger the value, the less degradation on the general-domain.

• **Elastic Weight Consolidation (EWC)** (Thompson et al., 2019) This method models the importance of the parameters with Fisher information matrix and puts more constrains on the important parameters to let them stay close to the original values during the fine-tuning process. The training objective is:

$$\mathcal{L}_{\text{EWC}}(\theta) = \mathcal{L}(\theta) + \alpha \sum_i F_i(\theta_i - \theta_i^G)^2 \qquad (14)$$

where i represents the i-th parameter and F_i is the modeled importance for the i-th parameter.

ID	System		Chinese-English				English-French				English-German			
			#Para.	Gen.	In.	Avg.	#Para.	Gen.	In.	Avg.	#Para.	Gen.	In.	Avg.
0	General		100.5M	23.26	9.97	16.62	93.3M	**33.05**	25.25	29.15	94.4M	26.22	29.53	27.88
1	In			0.87	3.64	2.26$^{-14.36}$		3.88	7.87	5.88$^{-23.27}$		12.63	24.77	18.70$^{-9.18}$
2	Fine-tuning			13.78	**17.05**	15.42$^{-1.20}$		14.03	**33.24**	23.64$^{-5.51}$		23.37	32.63	28.00$^{+0.12}$
3	SeqKD			14.02	13.45	13.74$^{-2.88}$		20.37	27.93	24.15$^{-5.00}$		24.50	30.20	27.35$^{-0.53}$
4	MOL			18.44	13.86	16.15$^{-0.47}$		26.58	28.05	27.32$^{-1.83}$		25.42	32.09	28.76$^{+0.88}$
5	EWC			17.45	15.72	16.59$^{-0.03}$		25.29	32.24	28.77$^{-0.38}$		25.62	32.17	28.90$^{+1.02}$
6	Full Bias		100.6M	23.26	11.55	17.41$^{+0.79}$	93.4M	**33.05**	26.47	9.76$^{+0.61}$	94.5M	26.22	30.00	28.11$^{+0.23}$
7	Adapter		101.3M	23.26	15.82	19.54$^{+2.92}$	94.1M	**33.05**	29.61	31.33$^{+2.18}$	95.3M	26.22	31.54	28.88$^{+1.00}$
8	MLL		117.4M	23.30	16.04	19.67$^{+3.05}$	109.7M	32.70	30.78	31.74$^{+2.59}$	111.2M	26.08	32.07	29.08$^{+1.20}$
9	PTE	0+**NP**	81.1M	18.35	9.70	14.03$^{-2.45}$	76.3M	29.27	24.86	27.07$^{-2.60}$	77.2M	24.99	26.23	25.61$^{-2.27}$
10		9+KD	81.1M	22.62	9.99	16.31$^{-0.17}$	76.3M	32.77	25.83	29.30$^{-0.37}$	77.2M	26.04	27.71	26.88$^{-1.00}$
11		10+FT	100.5M	22.62	15.94	19.28$^{+2.81}$	93.3M	32.77	30.69	31.73$^{+2.07}$	94.4M	26.04	32.57*	29.31$^{+1.43}$
12		0+**WP**	70.4M	20.74	9.54	15.14$^{-1.48}$	65.3M	29.65	25.03	27.34$^{-1.81}$	66.1M	25.02	26.66	25.84$^{-2.04}$
13		12+KD	70.4M	23.50	9.77	16.64$^{+0.02}$	65.3M	32.64	25.98	29.31$^{+0.16}$	66.1M	26.38	26.74	26.56$^{-1.32}$
14		13+FT	100.5M	**23.50**	**16.98****	**20.24**$^{+3.62}$	93.3M	32.64	**33.16****	**32.90**$^{+3.75}$	94.4M	**26.38**	**33.02****	**29.70**$^{+1.82}$

Table 1: BLEU scores on three translation tasks. '#Para.' denotes the number of parameters of the whole model, 'Gen.' and 'In.' denote the BLEU on general-domain and in-domain, and 'Avg.' denotes the average BLEU of the two test sets. 'NP', 'WP', 'KD', and 'FT' represent neuron pruning, weight pruning, knowledge distillation, and fine-tuning, respectively. The numbers on the right of 'PTE' denote that this training phase is based on the previous corresponding models. After knowledge distillation, the parameters in the pruned model (system 10, 13) are fixed, so the general-domain BLEU is unchanged after fine-tuning (system 11, 14). * and ** mean the improvements over the MLL method is statistically significant ($\rho < 0.05$ and $\rho < 0.01$, respectively). (Collins et al., 2005)

The second category indcludes the following three systems:

• **Full Bias** (Michel and Neubig, 2018) This method adds domain-specific bias term to the output softmax layer and only updates the term as other parts of the general-domain model keep fixed.

• **Adapter** (Bapna and Firat, 2019) This methods injects domain-specific adapter modules into each layer of the general-domain model. Each adapter contains a normalization layer and two linear projection layers. The adapter size is set to 64.

• **Multiple-output Layer Learning (MLL)** (Dakwale and Monz, 2017) The method modifies the general-domain model by adding domain-specific output layer for the in-domain and learning these domain specific parameters with respective learning objective. The training objective is:

$$\mathcal{L}_{\text{MLL}}(\theta_S, \theta_G, \theta_I) = \mathcal{L}(\theta_S, \theta_I) + \alpha \mathcal{L}_{\text{KD}}(\theta_S, \theta_G) \tag{15}$$

where θ_S is the domain-shared parameters, θ_G and θ_I denote the domain specific parameters for the general-domain and in-domain, respectively.

• **Our Method - Pruning Then Expanding (PTE)** Our model is trained just as the Method section describes. For the neuron pruning scheme, we prune the last 10% unimportant neurons; for the weight pruning scheme, we prune the last 30% unimportant parameters. To better show the ability of our method, we report the general- and in-domain performance after each training phase.

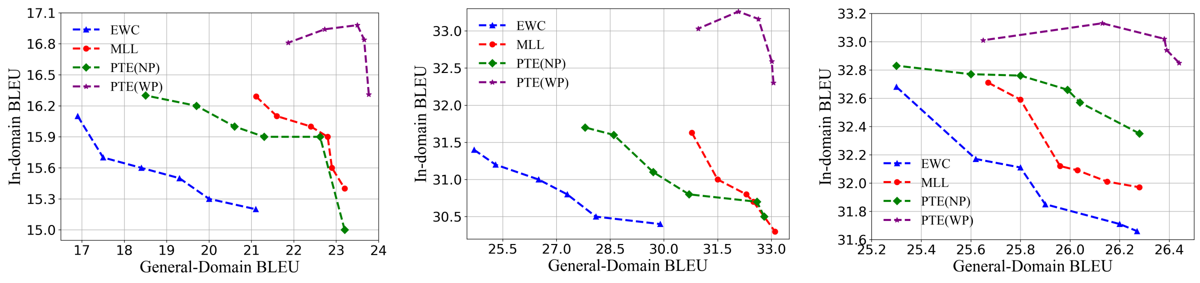
Implementation Details All the systems are implemented as the base model configuration in Vaswani et al. (2017) strictly. We set the hyper-parameter α to 1 for MOL, EWC, and MLL and we will do more analysis on the impact of this hyper-parameter in the next section. We set the learning rate during fine-tuning process to 7.5×10^{-5} for all the systems after having tried different values from 1.5×10^{-6} to 1.5×10^{-3}. In both of our methods, we don't prune the layer-normalization layers in the encoder and decoder, which can make training faster and more stable. For the neuron pruning method, we also don't prune the first layer of the encoder and the last layer of the decoder. Just like the work of Dakwale and Monz (2017), the domain of the test data is known in our experiments. Besides, we use beam search with a beam size of 4 during the decoding process.

4.3 Main Results

The final translation is detokenized and then the quality is evaluated using the 4-gram case-sensitive BLEU (Papineni et al., 2002) with the *SacreBLEU* tool (Post, 2018).[3] The results are given in Table 1. In all the datasets, our weight pruning method outperforms all the baselines. Furthermore, we get the following conclusions:

First, the contrast capacity-fixed methods can't handle large domain divergence and still suffer catastrophic forgetting. They perform well in the

[3]BLEU+case.mixed+numrefs.1+smooth.exp+tok.13a +version.1.3.6



(a) Zh→En (b) En→Fr (c) En→De

Figure 2: The performance trade-off with different hyper-parameters. the x-axis is general-domain BLEU and the y-axis is in-domain BLEU. The closer the point is to the upper right corner, the better the performance.

En-De translation task, where the data distributions are similar. They can significantly improve the in-domain translation quality without excessive damage to the general-domain translation quality. However, they perform worse in the En-Fr and Zh-En translation tasks with more different data distributions. The in-domain data contains many low-frequency or out-of-vocabulary tokens of the general-domain data. In this situation, these methods either bring limited in-domain improvements or degrade the general-domain performance too much. In contrast, our method is superior to them in all tasks, especially on the more different domains. This also validates our motivation.

Second, the capacity-increased methods can better deal with domain divergence. Compared with them, our method can achieve larger improvements on in-domain since we actually allocate more parameters for in-domain than the capacity-increased methods. Besides, our methods are also more convenient to use in practice because we don't need to specialize the model architecture. The pruning ratio is the only hyper-parameter needed tuning.

Lastly, both of our methods are immune to large domain divergence. Moreover, the knowledge distillation can bring modest improvements on the general domain. Compared with the neuron pruning method, the weight pruning method is more effective since it can prune and reutilize more parameters with smaller performance degradation.

5 Analysis

5.1 Adapting to Multi-Domain

We conduct experiments under the multi-domain scenario, which lets the model adapt to several different domains. Except for the training data used in the main experiments of the Zh-En task, which are related to the **News** and **Thesis** domain, we add two datasets from other domains, namely, **Spoken** and **Education**. Both of them are chosen randomly from the UM-corpus. Each of them contains about 75K, 1K, and 1K sentence pairs in the training, development, and test set. We test our weight-pruning based method and still prune last 30% unimportant parameters. We compare our method with the basic fine-tuning system and more effective capacity-increased method. The results are shown in Table 2. It shows that our method can get significant improvements on all the domains.

System	#Para.	Gen.	T	S	E
Fine-tuning	100.5M	12.58	**16.99**	18.64	19.43
Adapter	102.9M	23.26	15.82	17.83	18.68
MLL	151.2M	22.60	16.24	18.27	18.39
PTE(WP)	100.5M	**23.78****	16.85*	**18.69**	**19.55****

Table 2: BLEU of different domains on the Zh-En task. 'T', 'S', and 'E' denote the in-domain of Thesis, Spoken, and Education, respectively. 'PTE(WP)' denotes our weight-pruning based method.

5.2 Effects of Different Hyper-parameters

For the MOL, EWC, and MLL methods, the hyper-parameter α controls the trade-off between the general- and in-domain performance. As for our method, the proportion of model parameters to be pruned has a similar effect. To better show the full general- and in-domain performance trade-off, we conduct experiments with different hyper-parameters. We compare our method with the best capacity-fixed method EWC and best capacity-increased method MLL. For the EWC and MLL method, we vary α from 0.25 to 2.5. We vary the pruning proportion from 5% to 30% for our neuron-pruning method and from 10% to 50% for our weight-pruning method. The results are shown in Figure 2. It shows that our method outperforms EWC at all the operating points significantly. Be-

ID	System		Gen.	In.	Avg.
0	General		23.26	9.97	16.62
12	PTE	0+WP	20.74	9.54	$15.14^{-1.48}$
13		12+KD	23.50	9.77	$16.64^{+0.02}$
14		13+FT	23.50	16.98	$20.24^{+3.62}$
15	Random	0+WP	12.47	5.18	$8.83^{-7.79}$
16		15+KD	14.69	5.48	$10.09^{-6.53}$
17		16+FT	14.69	16.03	$15.36^{-1.26}$
18	Selective FT	0+FT	13.74	16.58	$15.16^{-1.46}$

Table 3: Results of the ablation study. 'Random' denotes the parameters are randomly pruned. 'Selective FT' denotes only the unimportant parameters are fine-tuned. Other denotations are the same as in Table 1.

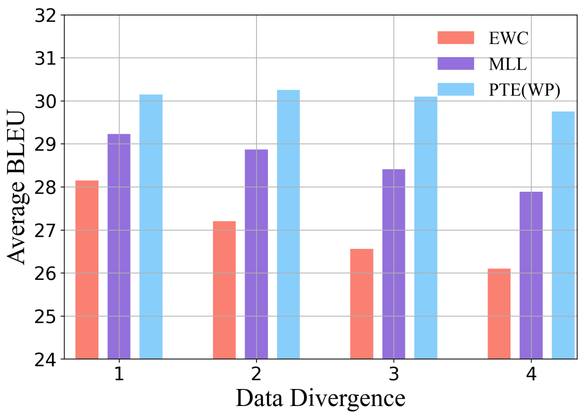


Figure 3: The average BLEU with different domain divergences on the En→Fr translation task. The x-axis represents the divergence of the data distribution and the larger the value is, the general- and in-domain data are more different.

sides, our neuron-pruning method can achieve comparable results as MLL and our weight-pruning method can surpass it with fewer parameters.

5.3 Ablation Study

To further understand the impact of each step of our method, we perform further studies by removing or replacing certain steps of our method. We first investigate the necessity of parameter importance evaluation. We train another three models following our method but with the parameters randomly pruned. The results are given in Table 3. It shows that random pruning will give excessive damage to general-domain. Besides, we also train another model that skips the model pruning and knowledge distillation steps and directly fine-tune the unimportant parameters. At last, we perform translation with the whole model on both the general- and in-domain. The results show that the change of unimportant parameters will also lead to catastrophic forgetting on general-domain, which shows the necessity of "divide and conquer".

5.4 Effects of Data Distribution Divergence

To further prove that our method is better at dealing with large domain divergence, we conduct experiments on the En-Fr translation task. Following the method in Moore and Lewis (2010), we score and rank each in-domain sentence pair by calculating the per-word cross-entropy difference between the general- and in-domain language model:

$$\texttt{Score} = (H_G(s) - H_I(s)) + (H_G(t) - H_I(t)) \quad (16)$$

where H denotes the language model which is trained with Srilm (Stolcke, 2002), s and t denote the source and target sentence. Then, we split the in-domain data into four parts with equal size and train new models with them separately. We compare our weight pruning based method with the EWC and MLL methods. The results are shown in Figure 3. It shows that we can get larger improvements as the data divergence gets larger.

6 Related Work

Domain Adaptation Recent work on DA can be divided into two categories according to the use of training data. The first category, which is also referred to as multi-domain adaptation, needs the training data from all of the domains. Chu et al. (2017) fine-tunes the model with the mix of the general-domain data and over-sampled in-domain data. Kobus et al. (2017) adds domain-specific tags to each sentence. Zhang et al. (2019b) applies curriculum learning to the DA problem. Britz et al. (2017) adds a discriminator to extract common features across domains. There are also some work (Zeng et al., 2018, 2019; Gu et al., 2019) that adds domain-specific modules to the model to preserve the domain-specific features. Currey et al. (2020) distills multiple expert models into a single student model. The work of Liang et al. (2020) has a similar motivation with ours which also fix the important parameters and prune the unimportant parameters. Compared with their method, our method doesn't need to store the general-domain training data and our method has less degradation on general-domain because we adopt the knowledge distillation method.

The second category, which is also referred to as continual learning, only needs the data from the new domain and the model in use. The biggest challenge for this kind of work is the catastrophic forgetting. Luong and Manning (2015) fine-tunes the general-domain model with the in-domain data. Freitag and Al-Onaizan (2016) ensembles the general-domain model and the fine-tuned model for generating. Saunders et al. (2019) investigates adaptive ensemble weighting for inference. Khayrallah et al. (2018) and Thompson et al. (2019) add regularization terms to let the model parameters stay close to their original values. Dakwale and Monz (2017) minimizes the cross-entropy between the output distribution of the general-domain model and the fine-tuned model. Michel and Neubig (2018) adds domain-specific softmax bias term to the output layer. Bapna and Firat (2019) injects domain-specific adapter modules into each layer of the general-domain model. Wuebker et al. (2018) only saves the domain-specific offset based on the general-domain model. Wang et al. (2020b) achieves efficient lifelong learning by establishing complementary learning systems. Sato et al. (2020) adapts the vocabulary of a pre-trained NMT model to the target domain.

Overall, our work is related to the second type of approach, which is more flexible and convenient in practice. The work of Thompson et al. (2019) and Dakwale and Monz (2017) are most related to our work. Compared with Thompson et al. (2019), our work is better at dealing with large domain divergence, since we add domain-specific parts to the model. In contrast to Dakwale and Monz (2017), our model divides each layer of the model into domain-shared and domain-specific parts, which increases the depth of the in-domain model, intuitively. Besides, our method doesn't need to add parameters, but it can be easily extended when necessary.

Model Pruning Model pruning usually aims to reduce the model size or improve the inference efficiency. See et al. (2016) examines three magnitude-based pruning schemes. Zhu and Gupta (2018) demonstrates that large-sparse models outperform comparably-sized small-dense models. Wang et al. (2020a) improves the utilization efficiency of parameters by introducing a rejuvenation approach. Lan et al. (2020) presents two parameter reduction techniques to lower memory consumption and increase the training speed of BERT.

7 Conclusion

In this work, we propose a domain adaptation method based on the importance of neurons and parameters of the NMT model. We make the important ones compromise between domains while unimportant ones focus on in-domain. Based on this, our method consists of several steps, namely, model pruning, knowledge distillation, model expansion, and fine-tuning. The experimental results on different languages and domains prove that our method can achieve significant improvements with model capacity fixed. Further experiments prove that our method can also improve the overall performance under the multi-domain scenario.

Acknowledgements

We thank all the anonymous reviewers for their insightful and valuable comments. This work was supported by National Key R&D Program of China (NO. 2017YFE0192900).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR*.

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 2019, Hong Kong, China, November 3-7, 2019*, pages 1538–1548.

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1489–1494.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James R. Glass. 2019. Identifying and controlling important neurons in neural machine translation. In *7th International Conference on Learning Representations, ICLR*.

Denny Britz, Quoc V. Le, and Reid Pryzant. 2017. Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017*, pages 118–126.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP 2014*, pages 1724–1734.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of simple domain adaptation methods for neural machine translation. *arXiv preprint arXiv:1701.03214*.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *ACL 2005*, pages 531–540.

Anna Currey, Prashant Mathur, and Georgiana Dinu. 2020. Distilling multiple domains for neural machine translation. In *EMNLP 2020*, pages 4500–4511.

Praveen Dakwale and Christof Monz. 2017. Fine-tuning for neural machine translation with limited degradation across in-and out-of-domain data. *Proceedings of the XVI Machine Translation Summit*, page 117.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation. *CoRR*, abs/1612.06897.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, pages 1243–1252.

Shuhao Gu and Yang Feng. 2020. Investigating catastrophic forgetting during continual training for neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4315–4326.

Shuhao Gu, Yang Feng, and Qun Liu. 2019. Improving domain adaptation translation with domain invariant and specific information. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 3081–3091.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *NIPS 2014 Deep Learning Workshop*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

Huda Khayrallah, Brian Thompson, Kevin Duh, and Philipp Koehn. 2018. Regularized training objective for continued training for domain adaptation in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation, NMT@ACL 2018, Melbourne, Australia, July 20, 2018*, pages 36–44.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1317–1327.

Catherine Kobus, Josep Maria 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, Varna, Bulgaria, September 2 - 8, 2017*, pages 372–378.

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 *International Conference on Learning Representations*.

Jianze Liang, Chengqi Zhao, Mingxuan Wang, Xipeng Qiu, and Lei Li. 2020. Finding sparse structure for domain specific neural machine translation. *AAAI 2021*.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 312–318.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2017. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, C] onference Track Proceedings*.

Robert C. Moore and William D. Lewis. 2010. Intelligent selection of language model training data. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, Short Papers*, pages 220–224.

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 NAACL-HLT 2019: Demonstrations*.

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 of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Shoetsu Sato, Jin Sakuma, Naoki Yoshinaga, Masashi Toyoda, and Masaru Kitsuregawa. 2020. Vocabulary adaptation for domain adaptation in neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 4269–4279.

Danielle Saunders, Felix Stahlberg, Adrià de Gispert, and Bill Byrne. 2019. Domain adaptive inference for neural machine translation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 222–228.

Abigail See, Minh-Thang Luong, and Christopher D. Manning. 2016. Compression of neural machine translation models via pruning. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 291–301.

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, ACL*.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *7th International Conference on Spoken Language Processing, ICSLP2002 - INTERSPEECH 2002, Denver, Colorado, USA, September 16-20, 2002*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Brian Thompson, Jeremy Gwinnup, Huda Khayrallah, Kevin Duh, and Philipp Koehn. 2019. Overcoming catastrophic forgetting during domain adaptation of neural machine translation. In *NAACL-HLT 2019*, pages 2062–2068.

Liang Tian, Derek F. Wong, Lidia S. Chao, Paulo Quaresma, Francisco Oliveira, and Lu Yi. 2014. Um-corpus: A large english-chinese parallel corpus for statistical machine translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014*, pages 1837–1842.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 5998–6008.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL*, pages 5797–5808.

Yong Wang, Longyue Wang, Victor O. K. Li, and Zhaopeng Tu. 2020a. On the sparsity of neural machine translation models. In *EMNLP 2020*, pages 1060–1066.

Yong Wang, Longyue Wang, Shuming Shi, Victor O. K. Li, and Zhaopeng Tu. 2019. Go from the general to the particular: Multi-domain translation with domain transformation networks. *AAAI20*.

Zirui Wang, Sanket Vaibhav Mehta, Barnabás Póczos, and Jaime G. Carbonell. 2020b. Efficient meta lifelong-learning with limited memory. In *EMNLP 2020*, pages 535–548.

Joern Wuebker, Patrick Simianer, and John DeNero. 2018. Compact personalized models for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 881–886.

Jiali Zeng, Yang Liu, Jinsong Su, Yubin Ge, Yaojie Lu, Yongjing Yin, and Jiebo Luo. 2019. Iterative dual domain 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 2019, Hong Kong, China, November 3-7, 2019*, pages 845–855.

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, Brussels, Belgium, October 31 - November 4, 2018*, pages 447–457.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019a. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4334–4343.

Xuan Zhang, Pamela Shapiro, Gaurav Kumar, Paul McNamee, Marine Carpuat, and Kevin Duh. 2019b. Curriculum learning for domain adaptation in neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 1903–1915.

Michael Zhu and Suyog Gupta. 2018. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *ICLR 2018*.