Simple, Scalable Adaptation for Neural Machine Translation

Ankur Bapna **Orhan Firat**
Google AI
{ankurbpn,orhanf}@google.com

Abstract

Fine-tuning pre-trained Neural Machine Translation (NMT) models is the dominant approach for adapting to new languages and domains. However, fine-tuning requires adapting and maintaining a separate model for each target task. We propose a simple yet efficient approach for adaptation in NMT. Our proposed approach consists of injecting tiny task specific adapter layers into a pre-trained model. These lightweight adapters, with just a small fraction of the original model size, adapt the model to multiple individual tasks simultaneously.

We evaluate our approach on two tasks: (i) Domain Adaptation and (ii) Massively Multilingual NMT. Experiments on domain adaptation demonstrate that our proposed approach is on par with full fine-tuning on various domains, dataset sizes and model capacities. On a massively multilingual dataset of 103 languages, our adaptation approach bridges the gap between individual bilingual models and one massively multilingual model for most language pairs, paving the way towards universal machine translation.

1 Introduction

Recent developments in deep learning have led to significantly improved quality on Neural Machine Translation (NMT) (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2014; Vaswani et al., 2017). While NMT performance on sentence level translation for high resource languages seems to be dramatically improved (Wu et al., 2016; Hassan et al., 2018), performance on out-of-domain data or low resource languages, still remains pretty poor (Duh et al., 2013; Koehn and Knowles, 2017; Farajian et al., 2017; Dong et al., 2015; Zoph et al., 2016). This has generated significant interest in adaptation approaches, that can leverage the huge amounts of parallel data available for high resource languages, to improve translation performance on low resource tasks. In this work we focus on two adaptation tasks: (i) Domain Adaptation, to improve the performance of in-domain translation by leveraging out-of-domain datasets (Luong and Manning, 2015; Freitag and Al-Onaizan, 2016), and (ii) Multilingual NMT, to improve the translation quality on low resource languages via co-training with similar languages (Dong et al., 2015; Firat et al., 2016; Johnson et al., 2016; Zoph et al., 2016; Neubig and Hu, 2018; Aharoni et al., 2019; Arivazhagan et al., 2019).

While several approaches have been explored in literature (Chu and Wang, 2018), full fine-tuning of model parameters remains the dominant approach for adapting to new domains and languages (Luong and Manning, 2015; Neubig and Hu, 2018). However, fine-tuning requires training and maintaining a separate model for every language, for every domain. As the number of languages and domains grow, training, maintaining and serving a separate model for every task becomes infeasible. This is further exacerbated by the increasing model capacity of state-of-the-art NMT models (Shazeer et al., 2018; Bapna et al., 2018; Huang et al., 2018); full fine-tuning is just too parameter inefficient. In addition to the growing number of models, fine-tuning requires very careful hyper-parameter tuning (eg. learning rate, regularization knobs etc.) during adaptation, and is prone to rapid over-fitting (Sennrich et al., 2016; Miceli Barone et al., 2017). This sensitivity to hyper-parameters and over-fitting to the adaptation corpus become worse in the setting of high capacity models.

These weaknesses beckon the need for parameter-efficient, scalable and hyper-parameter insensitive approaches for adaptation. The ideal adaptation approach should also offer the flexi-

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 1538–1548, Hong Kong, China, November 3–7, 2019.

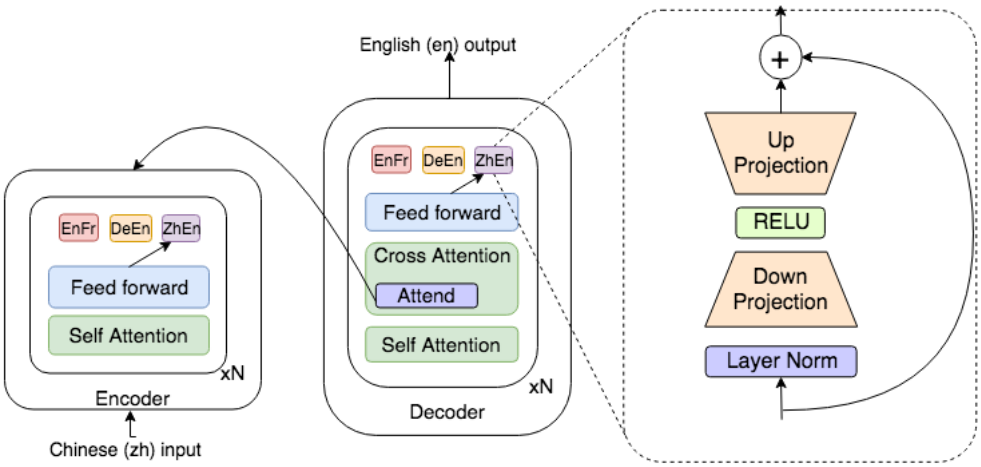

Figure 1: Diagrams depicting (i) Left: the proposed layout of a Transformer enhanced with language specific adapters (ii) Right: the architecture of each residual adapter layer. While the figure depicts use of adapters for multilingual NMT, the same formulation can be used for domain adaptation.

bility to adapt to tasks of varying complexity and adaptation corpus sizes, within a single model.

In this work we propose using light-weight adapter layers, which are transplanted between the layers of a pre-trained network and fine-tuned on the adaptation corpus. Adapting only the light-weight layers enables our approach to be parameter efficient, and eases the scalability of the approach to large models. The capacity of these adapters can be adjusted to match the requirements of the target task, making them suitable for a variety of adaptation tasks. By separating the parameters of the original network and each adaptation task, our approach circumvents catastrophic interference (McCloskey and Cohen, 1989) with the original model parameters, and allows us to simultaneously adapt a single model to multiple domains and languages, while retaining the quality on the source languages and domains.

We make three major contributions in this work: (i) we propose a formulation of adapter layers for NMT adaptation that enables us to tune their capacity according to the target task complexity and corpus size, (ii) we evaluate our approach on domain adaptation, and demonstrate that light-weight adapters match the performance of full fine-tuning based adaptation at a fraction of the per-domain parameter cost, and (iii) we use adapters to train a massively multilingual model on 103 languages, and demonstrate that it is possible to train a single model that significantly improves transfer performance on low resource languages, without huge regression on high resource language pairs.

By demonstrating the effectiveness of adapters on domain adaptation and massively multilingual translation, we make progress towards a flexible universal translation model for all languages and domains.

2 Related Work

Several approaches have been proposed in recent literature that try to address the shortcomings of full fine-tuning when applied to domain adaptation (Chu and Wang, 2018). Michel and Neubig (2018) proposed a space efficient approach to adaptation that introduces domain-specific biases to the output vocabulary, enabling extreme personalization in settings where small amounts of data are available for a lot of different domains. Thompson et al. (2018) fine-tune selected components of the base model architecture, in order to determine how much fine-tuning each component contributes to the final adaptation performance. Wuebker et al. (2018) propose introducing sparse offsets from the base model parameters for every domain, reducing the memory complexity of loading and unloading domain specific parameters in real world settings. Bapna and Firat (2019) train the base model to utilize neighboring samples from the training set, enabling the model to adapt to new domains without the need for additional parameter updates. Learning Hidden Unit Contribution (LHUC) (Vilar, 2018) is perhaps closest to our work in spirit.

They introduce domain specific gates that control the contribution of hidden units feeding into the next layer. However, they introduce a limited amount of per-domain capacity which doesn't scale well when a lot of domain specific data is available.

Residual Adapters were first introduced for adapting vision models in Rebuffi et al. (2017), but their formulation used a single projection layer, without any tunable hyper-parameters that could be used to adjust capacity based on the target domain. Houlsby et al. (2019) utilized a new formulation of adapters to adapt BERT (Devlin et al., 2018) to multiple tasks simultaneously. Our formulation of adapters is motivated by theirs, but differs in a few respects. Houlsby et al. (2019) introduce adapters after every sub-layer (self-attention, feed-forward) within a transformer layer, and re-train existing layer normalization parameters for every new domain. We simplify this formulation by leaving the parameters frozen, and introducing new layer normalization parameters for every task, essentially mimic-ing the structure of the transformer feed-forward layer.

3 Approach

Our approach consists of two phases: (i) Training a generic base model, and (ii) adapting it to new tasks with added small network modules. We first take a standard NMT model which is trained on a large source corpus. Following convergence,[1] all model parameters are frozen, preserving the information learned during this pre-training phase. Next, per-task light-weight adapter layers (see Fig. 1 right pane) are introduced after every layer in the encoder and the decoder (see Fig. 1 left pane). We fine-tune the parameters of these task-specific adapters on the adaptation corpus. This procedure can be followed for every additional task, allowing us to train a single model for all tasks simultaneously, with a small set of task-specific adapters.

Adapter Modules Our design principles for adapter modules are simplicity and flexibility. We propose a simple single hidden-layer feed-forward network formulation for adapters, with a non-linear activation function between the two projection layers. The inner dimension of these two projections is the only knob to tune. This allows us to adjust the capacity of the adapter module easily, depending on the complexity of the target task or domain. Additionally, we normalize the input of the adapters, in order to make the module plug-able into any part of the base network, irrespective of the variations in the activation patterns/distributions. This parametrized normalization layer allows the module to learn the activation pattern of the layer it's injected into. Finally, to allow the adapter module to represent a no-op if necessary, we wrap it with a residual connection.

Formulation Let z_i be the output of the i-th layer, of dimension d. We first apply layer-normalization (Ba et al., 2016) to the inputs of the adapter corresponding to task T.

$$\tilde{z}_i^T = LN_T(z_i). \tag{1}$$

This is followed by a projection layer of dimension b. The dimension of the projection can be tuned based on the task complexity and the size of the adaptation corpus. This now allows us to make it a bottleneck layer for compression, or over-parametrize it with a dimension larger than the input dimension.[2]

$$h_i^T = relu(W_{bd}^T \tilde{z}_i^T). \tag{2}$$

Lastly, the inner representation is projection back to the input dimension d, and combined with a residual connection (He et al., 2015):

$$x_i^T = W_{db}^T h_i^T + z_i. \tag{3}$$

Our proposed formulation for adapters, and their incorporation into Transformers (Vaswani et al., 2017) is illustrated in Figure 1. This self-contained adapter module can be injected between any two layers of the network, without disrupting the original operation.

4 Domain Adaptation

We first compare the adaptation performance of the light-weight residual adapters against full fine-tuning and LHUC (Vilar, 2018) on a large scale English-French domain adaptation task.

[1] We leave the methodology for defining convergence task specific, e.g. early stopping on validation set accuracy or number of training steps.

[2] Following the wiring choices of Transformer feed-forward network (Vaswani et al., 2017).

4.1 Dataset

We use the WMT En-Fr training set (36M pairs) as our out-of-domain (source) training corpus. NMT models trained on WMT are then adapted to the IWSLT'15 En-Fr corpus, consisting of 237k sentence pairs. We also evaluate adaptation performance on the JRC-Acquis dataset [3], which is an extremely narrow domain dataset containing 797k sentence pairs in the training split. For IWSLT, we use the test corpora from 2012-14 for validation, and the test corpus from 2015 as the test set. For JRC-Acquis the test and validation set contain 6574 and 5121 sentence pairs respectively. We also evaluate the translation performance of the non-adapted base model on newstest-2014.

Dataset	Base	FT	LHUC	Adap.
WMT'14	42.80	-	-	-
IWSLT'15	41.33	**44.59**	43.33	**44.63**
JRC	54.60	**64.13**	57.10	63.48

Table 1: Domain adaptation performance with different adaptation strategies. Base refers to the baseline NMT model trained on the WMT'14 En-Fr training corpus. FT refers to the fine-tuning upper bound, adapting all the model parameters by incrementally training on in-domain training data. LHUC adds additional task-specific gating parameters to the pre-trained model, which are trained on the in-domain data, as described in Vilar (2018). Adap. is the proposed adaptation approach, adding domain specific adapter layers trained on the in-domain training corpus.

4.2 Using Adapters for Domain Adaptation

When using adapters for domain adaptation, we follow the following two step approach:

- Pre-training: Pre-train the NMT model on a large open-domain corpus. Freeze all the parameters of this pre-trained model.
- Adaptation: Inject a set of domain-specific adapter layers for every target domain. These adapters are then fine-tuned to maximize performance on the corresponding domains. This step can be applied any time a new domain is added to the model.

As discussed in Section 5.2, we follow a slightly different approach when using adapters for multilingual NMT.

[3] http://opus.nlpl.eu/JRC-Acquis.php

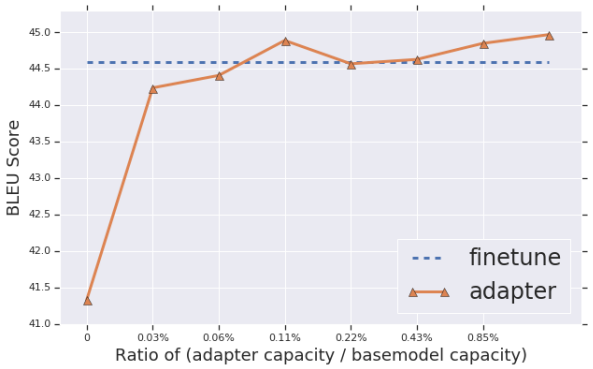

Figure 2: IWSLT Adaptation performance vs adapter capacity in terms of percentage of the base model's capacity. The range of model capacity plotted here corresponds to adapter bottleneck dimensions of 4, 8...256 respectively.

4.3 Models and Hyper-parameters

We use a larger version of Transformer Big containing 375M parameters as our base model. Our model is identical to Vaswani et al. (2017), having 6 encoder and decoder layers (12 in total), except that we use hidden layers of size 8192 instead of 4096, and a learning rate schedule of (3.0, 40K)[4], following Chen et al. (2018).

For full fine-tuning, we continue training on the in-domain corpus without resetting the optimizer accumulators or the learning rate schedule. This allows us to fine-tune gradually, avoiding rapidly over-fitting to the target domain. We also conducted fine-tuning experiments with SGD, but report results with the approach described above, based on better performance on the validation sets. When adapting with LHUC or lightweight adapters, we train using the same learning rate schedule and optimizer used during pre-training, but restart from the 0-th step, resetting the optimizer accumulators. BLEU scores are computed on the checkpoint with the best validation performance, on tokenized, true-cased output and references using *multi-bleu.perl* from Moses. All our experiments were performed using the open source Tensorflow Lingvo (Shen et al., 2019) framework.

4.4 Results and Analysis

The results of our adaptation experiments are documented in Table 1. On both, IWSLT and JRC-

[4] (3.0, 40K) schedule is the shorthand for a learning rate of 3.0, with 40K warm-up steps for the schedule, which is decayed with the inverse square root of the number of training steps after warm-up.

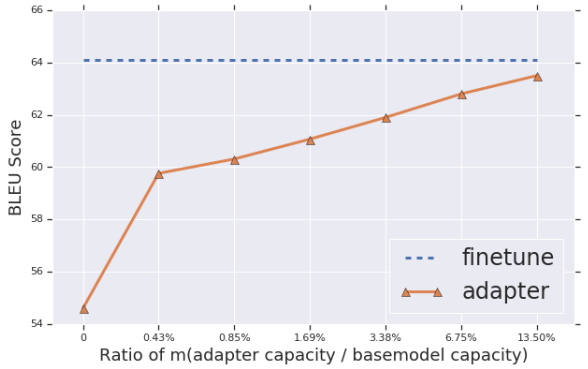

Figure 3: JRC-Acquis Adaptation performance vs adapter capacity in terms of percentage of the base model's capacity. The range of model capacity plotted here corresponds to adapter bottleneck dimensions of 64, 128...2048 respectively.

Acquis, full model fine-tuning (Full-FT columns in Table 1) on in-domain data significantly improves translation performance compared to the base, non-adapted Transformer Big by a huge margin, 3 BLEU points for IWSLT and 9 BLEU points for JRC. LHUC also improves performance over the base model, but lags behind a fully fine-tuned model for both domains and model capacities.

On IWSLT, adapters match the performance of the fine-tuning upper bound within error margins, while adding less than 0.11% of the original model parameters. On JRC-Acquis adapters recover around 90% of fine-tuning improvements without updating any existing parameters, while adding around 13.5% additional parameters.

To demonstrate the flexibility of our approach, we quantify the trade-off between adapter capacity and adaptation performance on both IWSLT and JRC-Acquis. In Figures 2 and 3, we plot the adaptation performance on IWSLT and JRC-Acquis respectively, while varying adapter capacity. On IWSLT, we notice that residual adapters reach within 0.5 BLEU of the full fine-tuning upper bound with just 0.03% of the model capacity, corresponding to a hidden dimension of size 4. By increasing capacity further we were able to improve over the full fine-tuning baseline by around 0.5 BLEU. On the other hand, on JRC-Acquis, adapter capacity had to be increased up to 13.5% of the total model capacity, corresponding to a hidden dimension of size 2048, before we were within 0.5 BLEU of the full fine-tuning performance. This highlights a key strength of the approach: by varying adapter capacity it is possible to adapt the same model to domains of varying complexity and amounts of data.

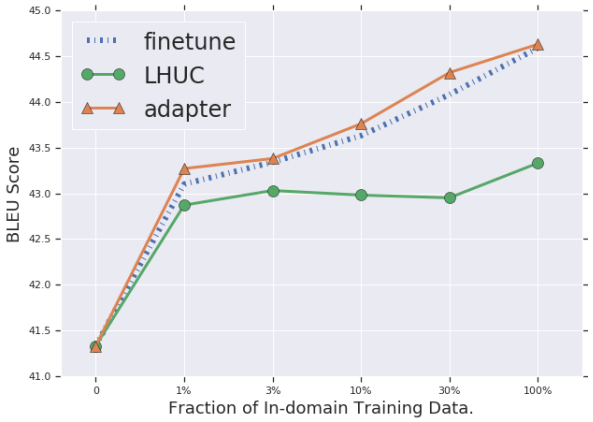

Figure 4: IWSLT Adaptation performance vs fraction of the in-domain training corpus used for adaptation. The blue solid line plots the performance of fine-tuning. The red dotted line tracks the performance of LHUC, while the yellow dashed line tracks the performance of the proposed residual adapter based approach with varying amounts of training data.

To evaluate the effectiveness of adapters when adapting with small in-domain corpora, we further compare the performance of adapters with fine-tuning on varying amounts of training data. In Figure 6 we plot the adaptation performance on IWSLT, when using different fractions of the training corpus for adaptation. While LHUC is competitive with full fine-tuning and light-weight adapters for extremely small fractions, the lack of capacity limits the applicability of the approach when larger quantities of adaptation data are available. On the other hand, by tuning the capacity of adapters to match the requirements for the adaptation corpus size, we are able to match and outperform fine-tuning on almost all evaluated datapoints.

In order to monitor the learning behavior of light-weight adapters, we compare the validation BLEU scores during the course of the fine-tuning process. Figure 5 illustrates the comparison of the two approaches, full fine-tuning and light-weight adapters. We notice that for a reasonably small adapter size, adapter performance gradually converges to its peak and stays steady, with almost no over-fitting for a long enough period, easing final model selection. On the other hand, with full fine-tuning, optimal model selection becomes challenging due to rapid over-fitting on the adaptation corpus. This, in fact, can be remedied by care-

fully tuning the learning rate (and/or batch size) during adaptation, but is not trivial and needs to be done individually for every different domain, model and corpus size, favoring the simplicity of our proposed approach.

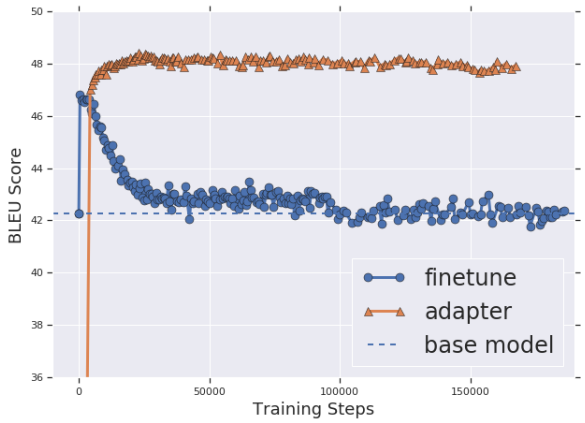

Figure 5: IWSLT dev performance vs number of in-domain adaptation steps when adapting with fine-tuning vs adapters.

5 Massively Multilingual Machine Translation

To stress test our adapters based approach, we apply this to a massively multilingual translation task on a real world dataset (Arivazhagan et al., 2019). Most previous literature in multilingual NMT focuses on improving the performance of low resource languages (Zoph et al., 2016; Firat et al., 2016; Neubig and Hu, 2018), often ignoring the source language performance of the adapted model. However, the goal of our work is to enable training a single model for all language pairs, in order to get benefits of transfer on low resource language pairs, without losing performance in the high resource setting.

5.1 Dataset

To highlight the flexibility of an adapters based approach, we study multilingual NMT on a massive scale, using a corpus generated by crawling and extracting parallel sentences from the web. Our corpus contains parallel documents for 102 languages, to and from English, containing a total of 25 billion sentence pairs (Arivazhagan et al., 2019).[5] The number of parallel sentences per language in our corpus ranges from around 10s of thousands to almost 2 billion. Figure 6 illustrates the data distribution across languages for all 102 languages studied in this paper.

5.2 Using Adapters for multilingual NMT

Our approach to using adapters for multilingual NMT diverges from domain adaptation, owing to the differences in the two tasks.

In the domain adaptation setting, while the input and output distributions of the adaptation domain might differ from that of the base, the set of inputs and outputs is pretty much the same. In mathematical terms, both the base and adaptation domain distributions, D_S and D_T respectively, are defined on the same support set $\{X, Y\}$.

On the other hand, in multilingual NMT, the support sets of different language pairs have very little overlap. In this setting, adapting a model to a new language pair without learning the embeddings and softmax parameters (which correspond to the input and output support sets) would be an extremely difficult task. Following the approach used for domain adaptation in Section 4.2 might not be possible here. We modify our approach to expand the input-output distribution of our initial pre-trained model to all language pairs we are interested in supporting, i.e. we can't add any new language pairs to the model during adaptation, but we use the adaptation stage to improve performance on languages learnt during pre-training.

For multilingual NMT, we follow the following two step approach:

- Global training: Train a fully shared model on all language pairs, with the goal of maximizing transfer to low resource languages.
- Refinement: Fine-tuning language pair specific adapters for all high resource languages, to recover lost performance during step 1. This step can only be applied for language pairs learned during global training.

5.3 Models and Hyper-parameters

We first train dedicated bilingual models on all language pairs to ground our multilingual analyses. We perform all our experiments with variants of the Transformer architecture (Vaswani et al., 2017). For most bilingual experiments, we use a larger version of Transformer Big containing 375M parameters (Chen et al., 2018), and a shared source-target sentence-piece model (SPM) (Kudo

[5] Limited to approximately this amount for experimentation.

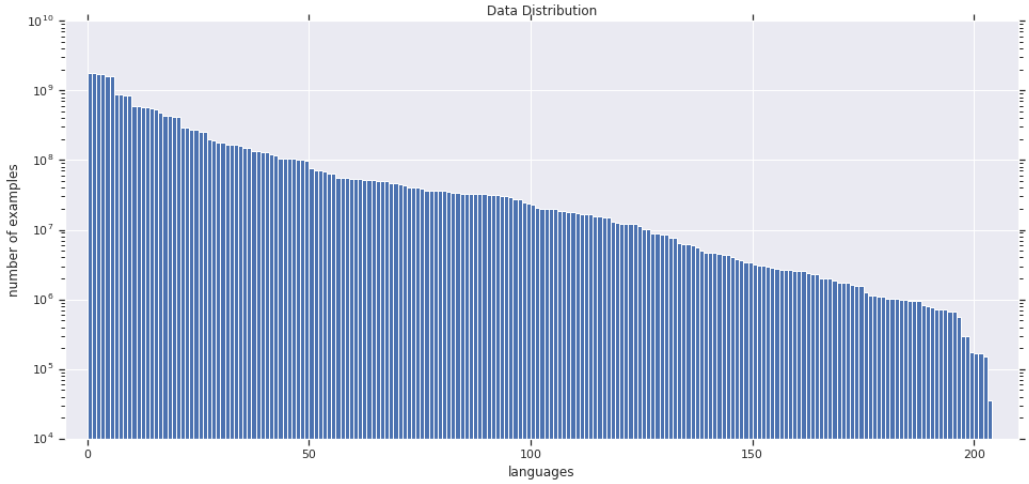

Figure 6: Per language pair data distribution of the dataset used for our multilingual experiments (for 102 languages, 204 language pairs to and from English). The y-axis depicts the number of training examples available per language pair on a logarithmic scale. Dataset sizes range from the order of 10^4 for the lowest resource language pairs to the order of 10^9 for the largest.

and Richardson, 2018) vocabulary with 32k tokens. We tune different values of dropout (Srivastava et al., 2014), depending on the dataset size for each language pair. For most medium and low resource languages we also experiment with Transformer Base. All our models are trained with Adafactor (Shazeer and Stern, 2018) with momentum factorization, a learning rate schedule of (3.0, 40K), and a per-parameter norm clipping threshold of 1.0. For Transformer Base models, we use a learning rate schedule of (2.0, 8K). BLEU scores are computed on the checkpoint with the best validation performance, on true-cased output and references.[6]

We now describe our approach for training the multilingual models. Due to the large imbalance in our training dataset (Figure 6), we first design a sampling strategy to simultaneously train a single model on all 204 language pairs. Sampling directly from the data distribution results in good performance on high resource languages, but low resource languages get starved. Sampling equally from all language pairs results in huge boost in low resource translation performance, but high resource languages perform significantly worse than their bilingual baselines.

To balance between high and low resource language pairs, we use a temperature based sampling strategy (Arivazhagan et al., 2019). For a given language pair, l_{12}, let D_{l12} be the size of the available parallel corpus. Then if we sample from the union of the datasets, the probability of the sample being from language pair l_{12} is $p_{l_{12}} = \frac{D_{l_{12}}}{\Sigma_{l_{12}} D_{l_{12}}}$. We set the probability of our sampled distribution to be proportional to $p_{l_{12}}^{\frac{1}{T}}$, where T is the sampling temperature. Now, $T = 1$ corresponds to true data distribution and $T = 100$ corresponds to an (almost) equal number of samples for each language. We use $T = 5$ for our multilingual model.

We train a single Transformer Big simultaneously on all 204 language pairs (102 languages to and from English), with the same hyper-parameter settings as the bilingual model. However, we use a shared SPM vocab with 64K tokens, generated using the same sampling distribution ($T = 5$) used during training. We additionally use character coverage of 0.999995 to ensure our vocab contains most of the alphabets for all 103 languages. Please refer (Arivazhagan et al., 2019) for additional training details for the base multilingual model.

Following global pre-training on all language pairs, we inject and fine-tune language pair specific adapters. The fine-tuning stage is performed separately for each language pair to reduce the device memory needed for the training process. The fine-tuned adapters can then be combined together into a single model after this stage. For fine-tuning, we use the same hyper-parameter used during global pre-training, but reset our optimizer

[6]We used an in-house implementation of mteval-v13a.pl from Moses to evaluate BLEU scores for our multilingual experiments.

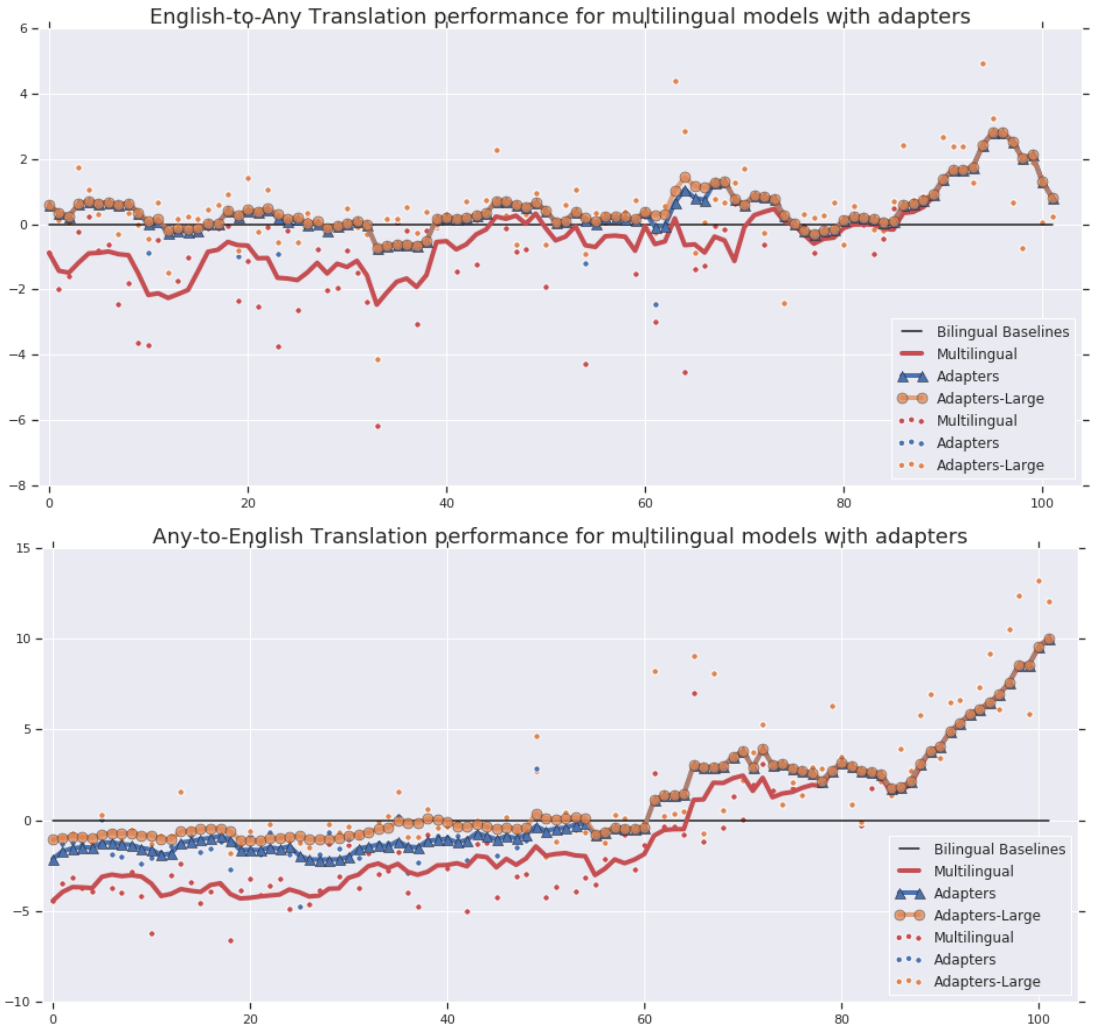

Figure 7: Trendlines depicting translation performance improvement in multilingual models with residual adapters. From left to right, languages are arranged in decreasing order of available training data. y-axis depicts the BLEU score relative to the bilingual baseline trained on the corresponding language pair. The plots correspond to the following models: (1.) Red: Multilingual model trained with sampling temperature, $T = 5$ (2.) Blue: Multilingual model + Small adapters, $b = 2048$ (3.) Pink: Multilingual model + Large adapters, $b = 4096$.
Note: For adapter experiments, we choose the best performance between $b = 0$, $b = 2048$ and $b = 4096$.

accumulators and restart from step 0. For our experiments we use the same bottle-neck dimension, $b = 2048$, for all language pairs. This was meant to reduce the number of experiments given the large number of language pairs in our setup. For language pairs that were worse than their bilingual models after adding adapters with $b = 2048$, we re-run fine-tuning with larger adapters, with $b = 4096$. In an ideal setting, the bottle-neck could be larger for the highest resource languages and $b = 0$ (no adapters) for the smallest languages.

5.4 Results and Analysis

We plot the translation quality on different language pairs in Figure 7. As we can see, the multilingual model significantly out-performs the bilingual baselines in the extremely low resource setting. These gains are even more amplified when translating into English, agreeing with previous work in multilingual NMT (Neubig and Hu, 2018; Aharoni et al., 2019). However, owing to the huge training corpus, we observe significant performance deterioration in the high resource languages. We attribute this deterioration to two factors: (i) Languages compete for capacity given the limited model size, and (ii) The model converges much before it trains on significant portions of the high resource datasets.

As is apparent from Figure 7, performance on high and medium resource languages improves

by huge margins after the second stage of training (adapter based refinement). Fine-tuning with adapters allows the model to see larger portions of the training data for high resource languages, and converges faster than training a model from scratch since it only updates a very small fraction of the model parameters (for most language pairs, the second stage converges within 20-50k steps, depending on the corpus size). For high resource languages, especially when translating into English, we observe further performance improvements when increasing adapter size. This again highlights the flexibility of adapters, it is possible to adjust the adapter capacity to match the complexity and resource size of the target task.

While adapters help us bridge most of the gap between bilingual and multilingual models, we still observe a minor regression for high resource languages translating into English, compared to the bilingual baselines. Although it might be possible to reduce this gap further by increasing the adapter size beyond $b = 4096$, there might be more efficient ways to approach this problem, including more expressive network architectures for adapters, joint fine-tuning of adapters and global model parameters, etc. However, we leave these studies to future work.

6 Conclusion

In this work, we proposed *light-weight adapters*, a simple yet efficient way for adapting large scale neural machine translation models. Our proposed approach, by injecting small task-specific adapter layers between the frozen base model layers during adaptation, enables the final network to adapt to multiple target tasks simultaneously, without forgetting the original parameters.

We evaluate light-weight adapters on two different adaptation tasks, domain adaptation and multilingual NMT. Experiments support the flexibility and scalability of *light-weight adapters*, (i) yielding comparable or better results when compared with the standard full fine-tuning or bilingual baselines, (ii) without the need for any hyper-parameter tuning across varying adaptation dataset sizes and model capacities.

With a large set of globally shared parameters and small interspersed task-specific layers, adapters allow us to train and adapt a single model for a huge number of languages and domains. We hope that this work would motivate further research into massively multitask and universal translation models.

Acknowledgments

We would like to thank the Google Translate and Lingvo development teams for their foundational contributions to this project. We would also like to thank Neil Houlsby, Anjuli Kannan and Yonghui Wu for helpful discussions early on in the project, and Wolfgang Macherey, Markus Freitag, Ciprian Chelba, Naveen Arivazhagan, Zhifeng Chen and the anonymous EMNLP reviewers for their insightful comments.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. *CoRR*, abs/1903.00089.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ankur Bapna, Mia Xu Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. 2018. Training deeper neural machine translation models with transparent attention. *arXiv preprint arXiv:1808.07561*.

Ankur Bapna and Orhan Firat. 2019. Non-parametric adaptation for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1921–1931.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. *arXiv preprint arXiv:1804.09849*.

Chenhui Chu and Rui Wang. 2018. A survey of domain adaptation for neural machine translation. *arXiv preprint arXiv:1806.00258*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1723–1732.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 678–683.

M Amin Farajian, Marco Turchi, Matteo Negri, and Marcello Federico. 2017. Multi-domain neural machine translation through unsupervised adaptation. In *Proceedings of the Second Conference on Machine Translation*, pages 127–137.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073*.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp.

Yanping Huang, Yonglong Cheng, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, and Zhifeng Chen. 2018. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *arXiv preprint arXiv:1811.06965*.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

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, Seattle, Washington, USA. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

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*, pages 1489–1494. Association for Computational Linguistics.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 875–880. Association for Computational Linguistics.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, pages 506–516.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, Ryan Sepassi, and Blake Hechtman. 2018. Mesh-tensorflow: Deep learning for supercomputers.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *arXiv preprint arXiv:1804.04235*.

Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, Mia X. Chen, Ye Jia, Anjuli Kannan, Tara Sainath, Yuan Cao, Chung-Cheng Chiu, Yanzhang He, Jan Chorowski, Smit Hinsu, Stella Laurenzo, James Qin, Orhan Firat, Wolfgang Macherey, Suyog Gupta, Ankur Bapna, Shuyuan Zhang, Ruoming Pang, Ron J. Weiss, Rohit Prabhavalkar, Qiao Liang, Benoit Jacob, Bowen Liang, HyoukJoong Lee, Ciprian Chelba, Sbastien Jean, Bo Li, Melvin Johnson, Rohan Anil, Rajat Tibrewal, Xiaobing Liu, Akiko Eriguchi, Navdeep Jaitly, Naveen Ari, Colin Cherry, Parisa Haghani, Otavio Good, Youlong Cheng, Raziel Alvarez, Isaac Caswell, Wei-Ning Hsu, Zongheng Yang, Kuan-Chieh Wang, Ekaterina Gonina, Katrin Tomanek, Ben Vanik, Zelin Wu, Llion Jones, Mike Schuster, Yanping Huang, Dehao Chen, Kazuki Irie, George Foster, John Richardson, Klaus Macherey, Antoine Bruguier, Heiga Zen, Colin Raffel, Shankar Kumar, Kanishka Rao, David Rybach, Matthew Murray, Vijayaditya Peddinti, Maxim Krikun, Michiel A. U. Bacchiani, Thomas B. Jablin, Rob Suderman, Ian Williams, Benjamin Lee, Deepti Bhatia, Justin Carlson, Semih Yavuz, Yu Zhang, Ian McGraw, Max Galkin, Qi Ge, Golan Pundak, Chad Whipkey, Todd Wang, Uri Alon, Dmitry Lepikhin, Ye Tian, Sara Sabour, William Chan, Shubham Toshniwal, Baohua Liao, Michael Nirschl, and Pat Rondon. 2019. Lingvo: a modular and scalable framework for sequence-to-sequence modeling. *arXiv preprint arXiv:1902.08295*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

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, Huda Khayrallah, Antonios Anastasopoulos, Arya D. McCarthy, Kevin Duh, Rebecca Marvin, Paul McNamee, Jeremy Gwinnup, Tim Anderson, and Philipp Koehn. 2018. Freezing subnetworks to analyze domain adaptation in neural machine translation.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

David Vilar. 2018. Learning hidden unit contribution for adapting neural machine translation models. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 500–505.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Joern Wuebker, Patrick Simianer, and John DeNero. 2018. Compact personalized models for neural machine translation. *arXiv preprint arXiv:1811.01990*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575.