When does Parameter-Efficient Transfer Learning Work for Machine Translation?

Anonymous ACL submission

Abstract

We study parameter-efficient transfer learning methods that adapt a pre-trained model by fine-tuning a small number of parameters, for machine translation. We conduct experiments across a diverse set of languages, comparing different fine-tuning methods in terms of (1) parameter budget, (2) language-pair, and (3) different pre-trained models. We show that methods such as adapters and prefix-tuning that add parameters to a pre-trained model perform best. However, methods which fine-tune a subset of existing parameters, e.g. BitFit and cross-attention tuning, are better correlated with pre-trained model capability. Furthermore, we found a large performance variation across language pairs, with parameter-efficient methods particularly struggling for distantly related language-pairs. Finally, we show that increasing model size, but tuning only 0.03% of total parameters, can outperform tuning 100% of the parameters of a smaller model.  

1 Introduction

There has been recent progress on scaling up neural machine translation models, improving performance. Such scale allows for ‘massively multilingual’ models, i.e. a single model that can translate between any pair of languages. Driving this trend is the availability of web-scale data in many languages, used to train sequence-to-sequence Transformer models. One approach leverages monolingual data with a denoising auto-encoder or masked language modelling objective (Liu et al., 2020; Xue et al., 2021). Another approach directly targets many-to-many multilingual machine translation (MT) by mining parallel corpora (Fan et al., 2020).

However models that are trained on monolingual data need to be fine-tuned for MT. As for multilingual MT systems that are trained on parallel data, they may need specialisation to a language pair (or domain) of interest (Neubig and Hu, 2018). Therefore in order to get the most out of available pre-trained models we may need to adapt them to a particular setting, simply fine-tuning all the parameters (Zoph et al., 2016) of the pre-trained model to learn MT and/or specialize to a language-pair.

Nonetheless, there are many reasons to fine-tune less than 100% of the pre-trained model’s parameters: (1) To avoid the large memory cost at training time associated with full fine-tuning, especially as model size increases. (2) Similarly, to prevent the storage cost of using many different large models for particular language pairs or domains. Furthermore, it enables us to probe model capability by measuring performance on different tasks or languages when only a small number of parameters are changed. We use parameter-efficient methods as shorthand for the more precise ‘parameter-efficient fine-tuning methods’.

Many such methods have been proposed in NLP, namely adapter-tuning (Houlsby et al., 2019), BitFit (Zaken et al., 2021), prefix-tuning (Li and Liang, 2021), and specifically for MT, updating only cross-attention layers (Gheini et al., 2021). These methods show promising results for many NLP tasks, e.g. recent work shows that for some classification tasks the performance of full fine-tuning can be matched by only training 20k parameters for a model (T5) with 11 billion parameters (Lester et al., 2021). However, their potential for MT across different language-pairs, parameter budgets, and based on different pre-trained (parent) models has not been covered yet. Previous work has found parameter-efficient methods designed for classification can fail for MT (Stickland et al., 2021a), and it is well known that NLP performance is unequal across the world’s languages (Blasi et al., 2021).

In this work, we provide a comprehensive analysis of parameter-efficient methods for MT, covering typographically and geographically diverse languages. Our main focus is on methods tuning
less than 1% of total model parameters, but also cover methods with more parameters that are able to match full fine-tuning. We experiment with models pre-trained on both monolingual and parallel data, varying from around 400m to 1 billion total parameters. Our main research questions are:

1. How do different parameter-efficient methods perform on MT for different languages/parameter budgets?
2. How does pre-trained model size effect the performance of parameter-efficient methods?
3. How do parameter-efficient methods differ in terms of performance and ability to reveal model capability?

Findings We found methods which add parameters to a pre-trained model, namely adapters and prefix tuning, give us the best performance (§ 4.1), while methods tuning a subset of existing parameters (like bias terms or cross attention) are better correlated with pre-trained model capability (§ 5.4). We found a large performance variation across language pairs, with translating between distantly related languages decreasing performance, especially for the most parameter-efficient methods (§ 5.3). Finally, we observe that increasing model size, but keeping the same number of fine-tuned parameters, substantially increases MT performance (§ 5.2).

2 Background

This section briefly describes the two multilingual pre-trained models that we focus on in this work, namely mBART and M2M-100.

**Multilingual Denoising Pre-training** Multilingual BART, mBART (Liu et al., 2020), is a sequence-to-sequence transformer model (Vaswani et al., 2017) that consists of an encoder and an autoregressive decoder. It is pre-trained with a denoising objective, reconstructing a document from a noisy version. mBART uses span masking and sentence permutation to noise the original document. Its architecture consists of 12 encoder and 12 decoder layers, with hidden dimension of 1024 and 16 attention heads. mBART is trained entirely on monolingual data that includes multiple languages and it has a large multilingual vocabulary of 250k tokens. In our experiments, we use mBART-50 (Tang et al., 2020) which was pre-trained on 50 languages.

**Many-to-Many Multilingual MT** The M2M-100 model (Fan et al., 2020) is a many-to-many multilingual translation system that is pre-trained on a large-scale parallel dataset for 100 languages and 100×99 translation directions. This dataset is automatically constructed with a novel data mining method based on language similarities and back-translation. The model is trained in a many-to-many fashion, balancing languages using sinkhorn temperature sampling. In our experiments, we use the base size M2M-100 with 484M parameters that consists of 12 encoder and 12 decoder layers, and feedforward dimension of 4096. To study the effect of the model size, we also use the medium size M2M-100 with 1.2B parameters. Both models have a multilingual vocabulary of 128K unique tokens that are distributed across 100 languages with temperature sampling.

3 Parameter-efficient Methods

All of our experiments fall under the umbrella of specialising a pre-trained sequence-to-sequence transformer model for MT of a particular language pair, with source language x and target language y. If the pre-training task was MT, and x and y were included, then a lower bound will be simply applying the pre-trained model without any changes. Conversely an upper bound is fine-tuning 100% of the pre-trained model parameters (‘full fine-tuning’). In between full fine-tuning and directly using the pre-trained model, we consider the following parameter-efficient-methods in this work:

**Adapter-tuning (Houlsby et al., 2019)** ‘Adapter layers’ are lightweight, learnable units inserted between transformer layers. They typically take the form of a feedforward network inserted as the final operation in a transformer layer. Formally, we follow the architecture introduced by Bapna and Firat (2019) for MT:

\[
A_{\ell}(h^\ell) = W_u^T \cdot f(W_d^T \text{LN}(h^\ell) + b_d^\ell) + b_u^\ell, \tag{1}
\]

where an adapter module \(A_\ell\) at layer \(\ell\) consists of a layer-normalization LN of the input \(h^\ell \in \mathcal{R}^d\), followed by a down-projection \(W_u \in \mathcal{R}^{d \times b}\) with bottleneck dimension \(b\), a non-linear function \(f(\cdot)\) and a up projection \(W_d \in \mathcal{R}^{b \times d}\). Finally, a residual connection with the input \(h^\ell\) is added to the output of the adapter: \(h^\ell \rightarrow A_\ell(h^\ell) + h^\ell\). We write ‘adapter-b’ to mean adapters with bottleneck dimension \(b\) throughout this work.
Prefix-tuning (Li and Liang, 2021) prepends a sequence of continuous task-specific vectors (‘prefixes’) to the model input, in analogy to natural language prompts (e.g. ‘translate this sentence:’). The transformer can attend to the prefix as if it were a sequence of ‘virtual tokens’, but the prefix consists entirely of free parameters. For each transformer layer, the prefix is replaced with a new set of vectors, increasing the expressiveness of the method. Concretely, we replace token embeddings by

\[ E_p = \text{Concat}(V^0, E), \]

with \( E \in \mathbb{R}^{L \times d} \) the original token embeddings packed into a matrix, \( V^0 \in \mathbb{R}^{p \times d} \) the prefix vectors, and \( L \) the original sequence length, \( p \) the prefix length and \( d \) model dimension. Before transformer layer \( \ell \) we additionally set the first \( p \) hidden states to a new prefix vector, i.e.

\[ H^\ell_t[p:p] = V^\ell \quad \text{with} \quad H \in \mathbb{R}^{(L+p) \times d} \]

the hidden states and \( V^\ell \in \mathbb{R}^{p \times d} \).

BitFit (Zaken et al., 2021) Bias term fine-tuning was introduced in the context of fine-tuning BERT for classification tasks, and consists of freezing most of the transformer-encoder parameters, and training only the bias terms and the task-specific classification layer. To use this method for MT we simply additionally fine-tune all decoder bias terms, and do not need the classification head.

We introduce a simple improvement to BitFit, based on replacing redundant parameters with ones that increase the expressiveness of the method. Note BitFit fine-tunes bias parameters in layer-norm (LN) modules (Ba et al., 2016), since layer-norm contains the affine transformation:

\[ \text{LN}_{\text{aff}}^\ell(z) = \gamma \odot z^\ell + \beta \]

where \( z^\ell \) is the normalized input after a residual connection. \( \gamma, \beta \in \mathbb{R}^d \) are learnable weight and the bias parameters of the layer-norm module. For the standard transformer model we consider in this work, the LN module is always followed by a matrix multiplication plus a bias term i.e. \( W^\ell_m \cdot \text{LN}_{\text{aff}}^\ell(z^\ell) + b^\ell_m = W^\ell_m \cdot \gamma \odot z^\ell + W^\ell_m \cdot \beta + b^\ell_m \).

Notice the same space of functions is available by only updating the \( b^\ell_m \) term in \( W^\ell_m \cdot \beta + b^\ell_m \). We simply switch to updating \( \gamma \) instead of \( \beta \), i.e. unfreezing the LN weight parameters and freezing the bias term in order to increase expressiveness and downstream performance (confirmed empirically in § 4.1). We use this version of BitFit throughout this work unless stated otherwise.

X-attention Tuning (Gheini et al., 2021) refers fine-tuning only cross-attention (X-attention) and corresponding layer-norm parameters located in each decoder layer of a transformer model. This
Adapters & Prefix-Tuning (mBART)

Table 3: BLEU scores for it→en and tr→en when different fine-tuning methods used for mBART and M2M-100. Each block represents same ratio of updated parameters, respectively 100%, 8.2/10.3%, 0.05/0.07%, and 0.02/0.03% for mBART/M2M-100. chrF scores for these experiments are shown in Appendix C.

<table>
<thead>
<tr>
<th>Method</th>
<th>it→en</th>
<th>tr→en</th>
<th>it→en</th>
<th>tr→en</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full FT</td>
<td>38.2</td>
<td>31.7</td>
<td>36.6</td>
<td>30.1</td>
</tr>
<tr>
<td>X-attention</td>
<td>34.8</td>
<td>27.0</td>
<td>36.1</td>
<td>29.2</td>
</tr>
<tr>
<td>Adapter (b=1024)</td>
<td>38.0</td>
<td>30.6</td>
<td>36.3</td>
<td>30.0</td>
</tr>
<tr>
<td>Prefix (p=13)</td>
<td>29.7</td>
<td>20.3</td>
<td>33.0</td>
<td>26.7</td>
</tr>
<tr>
<td>BitFit (LN-bias)</td>
<td>29.3</td>
<td>19.9</td>
<td>32.4</td>
<td>26.2</td>
</tr>
<tr>
<td>BitFit (LN-weights)</td>
<td>30.5</td>
<td>21.1</td>
<td>32.6</td>
<td>26.4</td>
</tr>
<tr>
<td>Adapter (b=5)</td>
<td>29.9</td>
<td>21.0</td>
<td>33.1</td>
<td>26.9</td>
</tr>
<tr>
<td>Prefix (p=5)</td>
<td>28.4</td>
<td>19.1</td>
<td>32.4</td>
<td>26.3</td>
</tr>
<tr>
<td>Adapter (b=1)</td>
<td>27.8</td>
<td>15.3</td>
<td>32.5</td>
<td>26.5</td>
</tr>
</tbody>
</table>

Figure 1: Relative MT performance over full fine-tuning vs. number of fine-tuned parameters for mBART. b and p refer to adapter bottleneck dimension and prefix length respectively. Due to the large effective sequence length, we limit prefix-tuning experiments.

4 Experiments & Results

Datasets We selected 13 typologically and geographically diverse languages for our experiments. Language families and dataset sources are shown in Table 1. For each language x, we paired it with English (en), and fine-tuned the pre-trained models separately. To pick these languages, we consider variation in language families and scripts.

Experimental Settings We used mBART-50 (Liu et al., 2020; Tang et al., 2020) and M2M-100 (Fan et al., 2020) as our multilingual pre-trained models, and all the languages we experiment with are included in their pre-training data. mBART needs to learn machine translation with parallel data, but M2M-100 can be used without fine-tuning, due to their pre-training tasks (see § 2). We experimented with medium size M2M-100 (1.2B parameters), to measure the impact of parent model size.

Table 2 shows all the fine-tuning methods (§ 3) we use, with their base model, number of trainable parameters and parameter ratio over full fine-tuning. As prefix-tuning is computationally expensive for large prefix lengths and generally does not perform as well as adapter-tuning for the same parameter budget, we do not include it in the experiments on every language pair (see § 4.1).

For all directions (x→en) and fine-tuning methods, we fine-tuned models with 1e-4 maximum learning rate for 100K training updates. We picked the best model based on dev set perplexity. We used a maximum batch size of 1024 tokens for mBART and 600 tokens for M2M-100, with a gradient accumulation step (update-frequency) of 2 for both models. All experiments are performed with the fairseq (Ott et al., 2019) library. Additional details including dataset splits are in Appendix A.

We use BLEU scores to estimate MT quality, calculated from Sacrebleu (Post, 2018). To compare fine-tuning methods across different languages, we often report relative performance with respect to full fine-tuning (FT) for each language by calculating the ratio of each method’s BLEU score w.r.t. the full FT BLEU score. On the recommendation of Marie et al. (2021) we report chrF (Popović, 2015) in Appendix C for each fine-tuning method.

4.1 Comparing fine-tuning methods

We can compare fine-tuning methods on several dimensions. Table 3 shows performance in terms of BLEU score for it→en and tr→en (similar and dissimilar language pairs). Adapters outperform other methods at almost all parameter budgets. At the largest parameter budget, adapter-1024 outperforms X-attention. For medium budgets (adapter-5 size) prefix-tuning is in second place, but for the smallest parameter budget (adapter-1 size) we consider, prefix-tuning outperforms adapters for mBART. However, prefix-tuning quickly falls behind adapters as parameter count, i.e. prefix length or adapter size, increases (see Fig. 1), in a result

Sacrebleu signature (BLEU):
refs:1case:mixed:edl:off:13alsmooth:exp:version:2.0.0

Due to computational constraints, we did not perform experiments on all combinations of method and language pair.
similar to He et al. (2021). Tuning LN weights rather than LN biases in the BitFit method outperforms the version tuning LN biases, confirming that our version improves expressiveness.

In terms of training speed/memory cost, prefix-13 causes a 30% slow-down in training speed relative to adapter-5, and larger models impose significant costs due to a large effective sequence length; see also Appendix B. BitFit and adapters have similar training speed.

### 4.2 Comparing language pairs

**mBART**  Fig. 2 shows the performance of several parameter-efficient method as we vary language pair, when initialized from mBART. Only adapter-1024 (8.2% of mBART parameters) is consistently competitive with full FT. Updating only cross-attention blocks (x-attn; 8.2% of mBART parameters) generates +90% relative performance with respect to full FT for Farsi, German, Russian, French, Portuguese, Vietnamese, and Czech in both directions (x→en). For other languages this decreases to ≈85%, and for Hindi (hi) to 50.4% and 61.7% in x→en and en→x respectively.

For smaller parameter budgets (BitFit and adapter-5; 0.05% of mBART parameters), we see better performance when translating into English (x→en). We expect better representation quality for English given the unequal amount of data per language used in mBART pre-training\(^5\). We observe that adapter-5 consistently outperforms BitFit in en→x (see also § 4.1). Finally note Hindi, Korean, and Turkish are particularly challenging for these methods, in both directions.

**M2M-100**  Fig. 3 shows relative MT performance when initializing with M2M-100. Here, we also include results for M2M-100 with no fine-tuning (‘no FT’), as M2M-100 is pre-trained with parallel data for MT. Again, languages such as Korean and Turkish present a bigger challenge than others (≈85% vs +90% performance relative to full FT) when tuning with either zero or a small number of parameters, although the performance drop is not as large as for mBART.

Adapter-5 again achieves better results than BitFit (+1% overall performance), in both directions (x→en). M2M-100 without fine-tuning (no FT) generally performs the worst; No FT reaches 78% mean relative MT performance w.r.t full FT,\(^5\)

---

\(^5\)English is the largest portion (55M tokens, 300GB) of the data that is used for mBART pre-training (Liu et al., 2020)
whereas adapter-5 achieves 92%. And for no FT the performance difference between languages is larger as can be seen in Farsi, Korean, and Turkish.

Interestingly, the results for Hindi (hi) do not follow the same trend as mBART. For en→hi, compared to Korean or Turkish we see better relative performance for small parameter budgets. For hi→en, full fine-tuning gives the worst performance. However, updating a small number of parameters (BitFit; 0.07% of the model parameters) outperforms the base model with no fine-tuning (115% vs 107%). The corresponding en→hi dataset consists of noisily aligned parallel sentences, and for the hi→en direction we speculate that fitting larger numbers of parameters gives the model enough capacity to model these noisy sentence pairs, hurting generalization. Finally, for fa→en, M2M-100 performance is considerably lower than for other language-pairs when we do not fine-tune the model.

5 Analysis
5.1 Impact of parent model & pre-training

Fig. 4 shows the relative performances over full fine-tuning for all languages (x→en) when the model is initialized with mBART or M2M-100. Overall, parameter-efficient fine-tuning of M2M-100 consistently provides higher relative performance than mBART (Fig. 4). This difference is larger when the number of trainable parameters is small (BitFit and adapter-5). While M2M-100 is pre-trained for MT with parallel data, mBART is pre-trained with a (monolingual6) denoising objective. Perhaps more parameters are required at fine-tuning time to ‘learn’ the MT task for mBART. Tun-

6Although mBART-50 pre-trained on 50 languages, the pre-training objective does not use any cross-lingual signal.
We investigate how parent model size affects the performance of fine-tuning methods across languages, comparing M2M-100’s base model with 418M total parameters to its medium size version (1.2B parameters). Fig. 3 shows the relative performances over full fine-tuning (484M) for adapter-5 with the base model and adapter-2 with the medium model, which correspond to roughly the same number of trainable parameters (0.07% of 418M parameters or 0.03% of 1.2B). No fine-tuning (no FT) results are also shown, representing lower bounds.

When translating into English (x→en), adapter-2 with the medium model outperforms full fine-tuning of the base model for most languages despite tuning only 0.03% of its parent model parameters. Compared to adapter-5 (484M) the difference is even larger (104.3% vs 93.6% mean relative MT performance w.r.t FT). Moreover, adapter-2 (1.2B) has a lower variance in performance compared to other models. For x→en, adapter-2 is still competitive with full fine-tuning of the base model with almost the same average performance. However, the difference between adapter-2 (1.2B) and adapter-5 (484M) is lower in this direction (97.9% vs 90.1%). Furthermore, the performance variation across languages is more visible: for Hindi, Farsi, Korean and Turkish adapter-2 (1.2B) performance falls behind full fine-tuning of the base model.

When it is used without any parameter updates, the medium model shows mixed results. Although performance is considerably higher than the base model without fine-tuning, the medium model is not competitive with adapter-5 (484M), in either direction (x→en). Furthermore, there is relatively high variance in results across language, with some languages remaining challenging. Therefore, for large parent models, parameter-efficient fine-tuning (∼1%) can take MT performance to the upper bound of a smaller model, showing the usefulness of fine-tuning even at large scales.

In order to investigate the impact of language relatedness on parameter-efficient fine-tuning, we designed another set of controlled experiments. We pick 3 languages from MultiParaCrawl, namely Finnish, Estonian and English, where Finnish and Estonian are from the same language family and typologically similar. We measure translation performance into Finish from Estonian and English, for different fine-tuning methods, and similarly for translation into Estonian. Fig. 5 shows relative MT performances with respect to full fine-tuning for adapter-1024, X-attention, BitFit and adapter-5, corresponding to decreasing numbers of trainable parameters, for both mBART and M2M-100.

As shown in the first two plots, when translating into Finnish, Estonian as the source language gives an advantage over English for BitFit and adapter-5 (This advantage is higher in M2M-100 than mBART). Likewise, for translation into Estonian, as the number of trainable parameters decreases, relative MT performance drops less when Finnish is the source language compared to English, for both parent models. These results suggest that, when the source and target languages are typologically similar, parameter-efficient fine-tuning methods make better use of the parent model.

Similarly, Fig. 1 shows relative MT performance with an increasing number of trainable parameters in mBART for a similar language pair (it→en) and a dissimilar one (tr→en). At low parameter budgets tr→en performance is much lower than it→en but the gap between the two decreases as parameter budget increases.

5.4 Revealing model capability
Comparing methods on their ability to reveal pre-trained model capability, we find methods that don’t add any additional parameters (x-att and BitFit) are the most useful. These methods show the most variation across language pairs (see e.g. Fig. 4). Additionally since for M2M-100 we can measure pre-trained model capability by evaluating the performance of the model without fine-tuning (“no FT”), for M2M-100 we also calculate the correlation between relative performance of different fine-tuning methods and no FT performance for each language. We find, for x→en, BitFit (0.84) > adapter-5 (0.77) > x-att (0.73) > adapter-1024 (0.66), where the number in brackets is the Pearson product-moment correlation coefficient. We have the same ranking for en→x. This shows that both a small parameter budget and not adding additional parameters i.e. adapters seems to be important for...
revealing model capability.

6 Related Work

In NLP, parameter-efficient methods have been widely used for fine-tuning of Transformer models to new tasks, domains or languages. Among those that add additional parameters, adapters (Houlsby et al., 2019) are ‘modular’, adding separate networks to the base model. As well as simple fine-tuning, they can be used in contexts such as multi-task learning (Stickland and Murray, 2019; Pfeiffer et al., 2021; Karimi Mahabadi et al., 2021), cross-lingual transfer (Üstün et al., 2020; Pfeiffer et al., 2020) and multilingual NMT (Bapna and Firat, 2019; Philip et al., 2020; Stickland et al., 2021b; Üstün et al., 2021).

Prefix-tuning (Li and Liang, 2021) and Prompt-tuning (Lester et al., 2021; Qin and Eisner, 2021) (i.e. only using soft prompt tokens without prefix vectors in each layer), have a natural interpretation in terms of virtual tokens. They can be used as task embeddings for inter-task transferability (Vu et al., 2021). LoRA (Hu et al., 2021) injects trainable low-rank matrices into query and value projection matrices of each transformer layer. Concurrently to our work, He et al. (2021) present a unified framework that integrates the above methods. Diff-pruning (Guo et al., 2021) modifies model parameters with a sparse vector. Some methods don’t add any parameters: BitFit (Zaken et al., 2021) fine-tunes only existing bias vectors, for classification tasks, and for MT, Gheini et al. (2021) propose updating only cross-attention blocks in decoder layers of the model.

Some of these methods have been compared in a controlled setting for English classification tasks (Mahabadi et al., 2021) or only a single language pair (English and Romanian) for MT (He et al., 2021). Aspects of efficiency and scale in MT in terms of inference cost (Kasai et al., 2021; Berard et al., 2021), vocabulary size (Gowda and May, 2020) data (Gordon et al., 2021), model size (Gordon et al., 2021; Arivazhagan et al., 2019) and number of languages (Arivazhagan et al., 2019) have been explored. Other work aims to improve full FT for domain adaptation by mixing in different data (Chu et al., 2017), regularisation (Miceli Barone et al., 2017) or many other methods (Chu and Wang, 2018; Saunders, 2021). However, none of these works study parameter-efficient transfer-learning methods for MT, and we aim to fill this gap.

7 Conclusion

We recommend: when fine-tuning a pre-trained model for MT, adapter layers usually have the highest performance out of all parameter-efficient fine-tuning methods (§ 4.1). For large parameter budgets (≈50m parameters) they almost recover full fine-tuning performance, and even for lower budgets, if the pre-training task was MT, i.e. M2M-100, adapters can recover >90% of full FT performance. However methods like BitFit which only tune existing parameters are better correlated with pre-trained model capability (§ 5.4), and for the smallest parameter budgets we consider, prefix tuning outperforms adapters for mBART.

Tuning only a small fraction of a larger model’s (M2M-100 medium size) parameters can outperform full FT of a smaller model (M2M-100 base size). However when translating in the en→x direction where x is distantly related to English e.g. Korean, full FT is superior (§ 5.2). More generally, distantly related language pairs require more parameters to be tuned to get close to full FT, for all methods (§ 5.3). Although we attempted to cover a diverse set of languages, future work could explore truly low resource languages, and those not included in the pre-training data of our models, where one would expect even larger performance gaps.
References


A Reproducibility Report

Datasets All datasets that are used in our experiments are publicly available. We used TED talks (Qi et al., 2018) for (cs, fr, ko, ru, pt, tr, fa)→en, IWSLT15 and IWSLT17 (Cettolo et al., 2012) for vi→en and (it, de)→en respectively, IITB (Kunchukuttan et al., 2018) for hi→en. Finally, for (en, et, fi) experiments, we randomly sampled 200k parallel sentences for each language-pair from MultiParacrawl by using OPUS (Tiedemann, 2012). Sizes of train, dev and test splits are given in Table 4. All datasets have licenses allowing non-commercial use.

Pre-trained models and Hyper-parameters We used mBART (Liu et al., 2020) that is extended to 50 languages (Tang et al., 2020). For M2M-100 (Fan et al., 2020), we used base- and medium-size models that consist of 484M and 1.2B parameters respectively.

For all experiments we used the hyper-parameters that are reported by Liu et al. (2020) except learning rate. For the learning rate, we follow Üstün et al. (2021) and used maximum of 1e-4 with polynomial learning rate decay, based on their adapter-tuning experiments. We fine-tune models by using 0.3 dropout, 0.2 label smoothing, 2500 warm-up steps for 100K training updates with an early-stopping patience of 10 epochs. We used a maximum batch size of 1024 tokens for mBART and 600 tokens for M2M-100, with a gradient accumulation step (update-frequency) of 2 for both models. We report the result of a single random seed/training run throughout this work whenever we list BLEU scores. All parameter-efficient fine-tuning methods are implemented on top of the Fairseq framework (Ott et al., 2019). We will share our code and scripts to reproduce all experiments.

Computing Budget and Infrastructure All the experiments are conducted using Tesla V100 GPUs with mixed precision (fp16). Parameters that are fine-tuned for each model are reported in the experiments section (§4). Each individual experiment took 3-10 hours on one GPU depending on the fine-tuning method and the language-pair.

B Prefix-tuning Details

There is a relationship between memory cost and training time for prefix-tuning: including virtual tokens in a sentence will increase the effective length of that sentence, and we can either impose additional memory cost for the virtual tokens, or we can reduce the total number of ‘real’ i.e. natural language as opposed to virtual tokens in each batch. With the latter method we avoid a large memory cost, however the time taken to iterate through a given number of training examples will be longer, since the number of real tokens per batch will be decreased, increasing training time. We use the latter (decreased ‘real’ tokens) method in all experiments.

Finally we note that inference speed will decrease as we increase the number of virtual tokens, since the decoder attention mechanism needs to attend to virtual tokens, i.e. when decoding token n it will attend to n − 1 + p previous tokens for prefix length p.
C Additional Results and Metrics

Table 5 shows chrF scores\textsuperscript{7} for the experiments comparing different parameter-efficient methods on it$\rightarrow$en and tr$\rightarrow$en (Table 3). These results confirm that the trends discussed in Section 4 are the same regardless of metric used for MT quality.

In Tables 6, 7 and 8, we show BLEU scores for other experiments presented in the paper only in terms of performance relative to full FT. Additionally we show adapter-1024 and X-attention scores for M2M-100; in general adapter-1024 outperforms X-attention, and both methods come close to full FT performance or slightly outperform it.

In Table 7 we show results of a smaller (40m parameters) transformer model trained from scratch on each dataset separately, with an architecture consisting of 6 encoder and decoder layers, hidden dimension of 512 and feed-forward hidden dimension 1024. We train a unique sentence-piece (Kudo and Richardson, 2018) vocabulary for each dataset, shared between source and target language, of size approximately 16k. Training hyper-parameters were the same as our other models. For the $x$$\rightarrow$en direction almost all of our methods based on pre-trained models outperformed the ‘from scratch’ baseline, however in the en$\rightarrow$x direction for mBART the most parameter efficient methods sometimes fall short (see e.g. Turkish or French). For translating into Farsi no pre-trained model outperformed the from scratch model, even with full fine-tuning, suggesting a weakness for particularly low resource resource languages like Farsi.

Note per-dataset hyper-parameter search would likely improve performance, especially for ‘from scratch’ results, but we did not attempt this due to computational constraints.

\textsuperscript{7}Sacrebleu signature (chrF2++): nrefs:1case:mixedleff:yesnlc:6lnw:2space:nolversion:2.0.0
### Table 6: \text{en} \leftrightarrow \text{en} results in terms of BLEU for M2M-100 experiments.

<table>
<thead>
<tr>
<th>M2M-100</th>
<th>No.</th>
<th>en</th>
<th>fi</th>
<th>de</th>
<th>ru</th>
<th>ko</th>
<th>fr</th>
<th>pt</th>
<th>tr</th>
<th>vi</th>
<th>cs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1.6m</td>
<td>150k</td>
<td>230k</td>
<td>208k</td>
<td>208k</td>
<td>205k</td>
<td>192k</td>
<td>184k</td>
<td>182k</td>
<td>133k</td>
</tr>
<tr>
<td>en FT</td>
<td>484m</td>
<td>19.4</td>
<td>17.6</td>
<td>32.8</td>
<td>32.0</td>
<td>22.0</td>
<td>9.6</td>
<td>41.3</td>
<td>42.1</td>
<td>17.9</td>
<td>33.9</td>
</tr>
<tr>
<td>Adapter (b=1024)</td>
<td>50m</td>
<td>18.6</td>
<td>17.6</td>
<td>32.7</td>
<td>31.3</td>
<td>22.0</td>
<td>9.3</td>
<td>41.0</td>
<td>42.4</td>
<td>17.9</td>
<td>33.4</td>
</tr>
<tr>
<td>X-attention</td>
<td>50m</td>
<td>18.3</td>
<td>17.3</td>
<td>31.7</td>
<td>31.0</td>
<td>21.9</td>
<td>9.2</td>
<td>40.8</td>
<td>42.0</td>
<td>17.7</td>
<td>33.6</td>
</tr>
<tr>
<td>BitFit (b=5)</td>
<td>320k</td>
<td>17.3</td>
<td>16.2</td>
<td>32.5</td>
<td>32.1</td>
<td>23.1</td>
<td>8.9</td>
<td>42.2</td>
<td>43.1</td>
<td>16.7</td>
<td>34.6</td>
</tr>
<tr>
<td>No FT (1.2B)</td>
<td>344k</td>
<td>17.4</td>
<td>14.6</td>
<td>32.5</td>
<td>32.1</td>
<td>23.1</td>
<td>8.9</td>
<td>42.2</td>
<td>43.1</td>
<td>16.7</td>
<td>34.6</td>
</tr>
<tr>
<td>No FT (484M)</td>
<td>0</td>
<td>18.0</td>
<td>9.7</td>
<td>29.6</td>
<td>29.9</td>
<td>21.1</td>
<td>5.5</td>
<td>37.6</td>
<td>39.6</td>
<td>13.2</td>
<td>32.9</td>
</tr>
</tbody>
</table>

### Table 7: \text{x} \leftrightarrow \text{en} results in terms of BLEU for mBART experiments.

<table>
<thead>
<tr>
<th>mBART</th>
<th>No.</th>
<th>hi</th>
<th>fa</th>
<th>it</th>
<th>de</th>
<th>ru</th>
<th>ko</th>
<th>fr</th>
<th>pt</th>
<th>tr</th>
<th>vi</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1.6m</td>
<td>150k</td>
<td>230k</td>
<td>208k</td>
<td>208k</td>
<td>205k</td>
<td>192k</td>
<td>184k</td>
<td>182k</td>
<td>133k</td>
</tr>
<tr>
<td>en FT</td>
<td>610m</td>
<td>19.3</td>
<td>17.8</td>
<td>32.9</td>
<td>33.1</td>
<td>23.5</td>
<td>10.1</td>
<td>42.7</td>
<td>43.5</td>
<td>18.7</td>
<td>35.2</td>
</tr>
<tr>
<td>Adapter (b=1024)</td>
<td>50m</td>
<td>18.1</td>
<td>18.0</td>
<td>33.3</td>
<td>32.8</td>
<td>22.9</td>
<td>9.9</td>
<td>37.9</td>
<td>42.8</td>
<td>18.2</td>
<td>34.7</td>
</tr>
<tr>
<td>X-attention</td>
<td>50m</td>
<td>11.9</td>
<td>16.8</td>
<td>27.7</td>
<td>30.3</td>
<td>21.2</td>
<td>8.8</td>
<td>39.5</td>
<td>40.8</td>
<td>16.3</td>
<td>33.5</td>
</tr>
<tr>
<td>BitFit (b=5)</td>
<td>335k</td>
<td>7.9</td>
<td>12.8</td>
<td>22.7</td>
<td>23.3</td>
<td>16.6</td>
<td>5.3</td>
<td>30.9</td>
<td>30.9</td>
<td>9.5</td>
<td>26.8</td>
</tr>
<tr>
<td>Adapter (b=2; 1.2B)</td>
<td>344k</td>
<td>24.8</td>
<td>31.5</td>
<td>37.3</td>
<td>37.7</td>
<td>28.9</td>
<td>22.2</td>
<td>44.0</td>
<td>48.7</td>
<td>29.9</td>
<td>37.5</td>
</tr>
<tr>
<td>No FT (1.2B)</td>
<td>0</td>
<td>24.5</td>
<td>14.9</td>
<td>32.5</td>
<td>32.1</td>
<td>24.1</td>
<td>17.6</td>
<td>37.5</td>
<td>42.0</td>
<td>24.2</td>
<td>29.9</td>
</tr>
<tr>
<td>No FT (484M)</td>
<td>0</td>
<td>21.9</td>
<td>14.9</td>
<td>29.7</td>
<td>29.5</td>
<td>21.4</td>
<td>15.8</td>
<td>34.9</td>
<td>38.6</td>
<td>22.0</td>
<td>27.1</td>
</tr>
</tbody>
</table>

### Table 8: \text{en}, \text{et}, \text{fi} results in terms of BLEU for M2M-100 and mBART experiments. Note that BLEU scores are not directly comparable as the datasets are different for each language-pair. For a comparison between fine-tuning methods, we refer to relative performances over full fine-tuning (Fig. 5).

<table>
<thead>
<tr>
<th>M2M-100</th>
<th>\text{en} \leftrightarrow \text{et}</th>
<th>\text{en} \leftrightarrow \text{fi}</th>
<th>\text{et} \leftrightarrow \text{fi}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>\text{en} &lt; fi</td>
<td>\text{en} &lt; et</td>
<td>\text{fi} &lt; et</td>
</tr>
<tr>
<td>Full FT</td>
<td>43.9</td>
<td>37.9</td>
<td>40.4</td>
</tr>
<tr>
<td>Adapter (b=1024)</td>
<td>42.7</td>
<td>35.9</td>
<td>39.6</td>
</tr>
<tr>
<td>X-attention</td>
<td>42.9</td>
<td>35.9</td>
<td>39.5</td>
</tr>
<tr>
<td>BitFit (b=5)</td>
<td>35.4</td>
<td>25.6</td>
<td>33.9</td>
</tr>
<tr>
<td>Adapter (b=2; 1.2B)</td>
<td>41.9</td>
<td>32.0</td>
<td>39.6</td>
</tr>
<tr>
<td>No FT (1.2B)</td>
<td>40.3</td>
<td>28.6</td>
<td>38.1</td>
</tr>
<tr>
<td>No FT (484M)</td>
<td>34.1</td>
<td>23.6</td>
<td>32.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>mBART</th>
<th>\text{en} \leftrightarrow \text{et}</th>
<th>\text{en} \leftrightarrow \text{fi}</th>
<th>\text{et} \leftrightarrow \text{fi}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>\text{en} &lt; fi</td>
<td>\text{en} &lt; et</td>
<td>\text{fi} &lt; et</td>
</tr>
<tr>
<td>Full FT</td>
<td>45.4</td>
<td>39.8</td>
<td>42.3</td>
</tr>
<tr>
<td>Adapter (b=1024)</td>
<td>45.3</td>
<td>39.1</td>
<td>41.9</td>
</tr>
<tr>
<td>X-attention</td>
<td>40.6</td>
<td>34.2</td>
<td>36.1</td>
</tr>
<tr>
<td>BitFit (b=5)</td>
<td>28.9</td>
<td>18.9</td>
<td>25.0</td>
</tr>
<tr>
<td>Adapter (b=2; 1.2B)</td>
<td>28.9</td>
<td>19.2</td>
<td>24.3</td>
</tr>
<tr>
<td>No FT (1.2B)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>No FT (484M)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>