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

Ahmet Üstün*
University of Groningen
a.ustun@rug.nl

Asa Cooper Stickland*
University of Edinburgh
a.cooper.stickland@ed.ac.uk

Abstract

Parameter-efficient fine-tuning methods (PEFTs) offer the promise of adapting large pre-trained models while only tuning a small number of parameters. They have been shown to be competitive with full model fine-tuning for many downstream tasks. However, prior work indicates that PEFTs may not work as well for machine translation (MT), and there is no comprehensive study showing when PEFTs work for MT. We conduct a comprehensive empirical study of PEFTs for MT, considering (1) various parameter budgets, (2) a diverse set of language-pairs, and (3) different pre-trained models. We find that 'adapters', in which small feed-forward networks are added after every layer, are indeed on par with full model fine-tuning when the parameter budget corresponds to 10% of total model parameters. Nevertheless, as the number of tuned parameters decreases, the performance of PEFTs decreases. The magnitude of this decrease depends on the language pair, with PEFTs particularly struggling for distantly related language-pairs. We find that using PEFTs with a larger pre-trained model outperforms full fine-tuning with a smaller model, and for smaller training data sizes, PEFTs outperform full fine-tuning for the same pre-trained model.[1]

1 Introduction

There has been enormous progress on scaling up neural machine translation (NMT) in the recent years, resulting in 'massively multilingual' models that are capable of translating across many languages (Bapna et al., 2022). Most successful applications rely on sequence-to-sequence pre-training that (1) leverages web-scale monolingual data with a masking objective to build a multilingual backbone (parent) model (Liu et al., 2020; Song et al., 2019), or (2) directly targets a many-to-many NMT system by mining parallel corpora (Fan et al., 2020).

Standard practice is to fine-tune every parameter of a particular a pre-trained model to specialize it to a language pair (or domain) of interest (Zoph et al., 2016; Neubig and Hu, 2018). However, if we require specialization to many language pairs or domains, the storage and time costs of full fine-tuning may become prohibitive. Moreover, as models grow ever larger, more efficient methods become attractive.

As an alternative to full model fine-tuning, several parameter-efficient fine-tuning methods (**PEFTs**) have been proposed. Such methods only fine-tune a small number of parameters, reducing storage cost, and avoid calculating the gradients for every model parameter, reducing training time and memory cost. Examples include adapters (Houlsby et al., 2019; Bapna and Firat, 2019) and prefix-tuning (Li and Liang, 2021), which introduce a few extra parameters to fine-tune, keeping the pre-trained model fixed. Others like BitFit (Zaken et al., 2021) tune only the bias vectors of the backbone model and similarly Gheini et al. (2021) update only cross-attention layers.

PEFTs can produce results that are competitive with full fine-tuning. For instance, adapters can match full fine-tuning performance on the GLUE benchmark using only 2-4% additional parameters (Houlsby et al., 2019). However their potential for MT has not been fully explored. Prior studies indicate that PEFTs designed for classification tasks can fail for MT (Stickland et al., 2021a), and it is not known how source and target language characteristics affect PEFTs' performance.

In this work, we provide a comprehensive analysis of PEFTs for MT. For our analysis, we consider: (1) different pre-trained models with varying size

* Both authors contributed the paper equally and the order is determined by coin flip.

[1]Our code and scripts for reproducing the experiments are available at https://github.com/ahmetustun/fairseq

from 484 million to 1.2 billion total parameters, (2) several PEFTs, and (3) typographically and geographically diverse languages. Moreover, we vary the number of tuned parameters, resulting in different parameter 'budgets' for each fine-tuning experiment, ranging from 0.03% to 10% of the total number of model parameters. Our main research questions are:

RQ1: For a given parameter budget, which PEFT works best?

RQ2: How does language similarity affect the performance of PEFTs for different parameter 'budgets'?

RQ3: How does (i) the pre-training objective, and (ii) the size of the parent model affect the performance of PEFTs?

RQ4: Do PEFTs work better than fine-tuning for small dataset sizes?

Key Findings **1)** We found methods which introduce new parameters to a pre-trained model, namely adapters and prefix tuning, give us the best performance (§ 5.1). Of these, adapters retain good performance as the number of new parameters is increased, while prefix-tuning falls behind. **2)** We found a large variation across language pairs in term of PEFTs' performance. Specifically, the distance between the source and target languages is negatively correlated with performance, especially for methods tuning the smallest number of parameters and methods tuning a subset of existing parameters (like bias terms or cross attention) (§ 5.2). **3)** We observe that increasing model size, but keeping the same number of fine-tuned parameters, substantially increases MT performance (§ 5.3). Finally, **4)** we observe that adapters perform better than full fine-tuning for small datasets, with the advantage for adapters increasing as dataset size gets smaller (§ 5.4).

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. It 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, hidden dimension of 1024 and feedforward dimension of 4096. To study the effect of model size, we also use the medium size M2M-100 with 1.2B parameters, which has 24 encoder and 24 decoder layers, and feedforward dimension of 8192. Both models have a multilingual vocabulary of 128K unique tokens that are distributed across 100 languages with temperature sampling.

3 Parameter Efficient Fine-tuning 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 fine-tuning methods (PEFTs) 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:

$$A_\ell(\mathbf{h}^\ell) = W_\mathrm{u}^T \cdot f(W_\mathrm{d}^T \mathrm{LN}(\mathbf{h}^\ell) + \mathbf{b}_\mathrm{d}^\ell) + \mathbf{b}_\mathrm{u}^\ell, \quad (1)$$

where an adapter module A_ℓ at layer ℓ consists of a layer-normalization LN of the input $h^\ell \in \mathcal{R}^d$, followed by a down-projection $W_\mathrm{d} \in \mathcal{R}^{d \times b}$ with bottleneck dimension b, a non-linear function $f(\cdot)$ and an up projection $W_\mathrm{u} \in \mathcal{R}^{b \times d}$. Finally, a residual connection with input h^ℓ is added to the output of the adapter: $\mathbf{h}^\ell \rightarrow A_\ell(\mathbf{h}^\ell) + \mathbf{h}^\ell$. 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:') which the transformer can attend to, 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

$$E_p = \mathrm{Concat}(V^0, E), \quad (2)$$

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

BitFit (Zaken et al., 2021) Bias term fine-tuning was introduced in the context of fine-tuning BERT for classification tasks, and consists of training only the bias terms and the task-specific classification layer. To use this method for MT we 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 that BitFit fine-tunes bias parameters in layer-norm (LN) modules (Ba et al., 2016), since the layer-norm contains the following affine transformation:

$$\mathrm{LN}_{\mathrm{aff}}^\ell(\mathbf{z}^\ell) = \gamma \odot \mathbf{z}^\ell + \beta \quad (3)$$

	mBART		M2M-100	
	it→en	tr→en	it→en	tr→en
Full FT	38.2	31.7	36.6	30.1
X-attention	34.8	27.0	36.1	29.2
Adapter (b=1024)	38.0	30.6	36.3	30.0
Prefix (p=13)	29.7	20.3	33.0	26.7
BitFit (LN-bias)	29.3	19.9	32.4	26.2
BitFit (LN-weights)	30.5	21.1	32.6	26.4
Adapter (b=5)	29.9	21.0	33.1	26.9
Prefix (p=5)	28.4	19.1	32.4	26.3
Adapter (b=1)	27.8	15.3	32.5	26.5

Table 1: For a given parameter budget, which method works best (RQ1)? BLEU scores for it→en and tr→en when different fine-tuning methods used for mBART and M2M-100. Each block consists of methods that update approximately the same number of parameters. chrF scores for these experiments are shown in Appendix C

where $\mathbf{z}^\ell$ is the normalized input after a residual connection. $\gamma, \beta \in \mathcal{R}^d$ are learnable weights and the bias parameters of the LN module. For the standard transformer model, the LN module is always followed by a matrix multiplication plus a bias term i.e. $W_m^\ell \cdot \mathrm{LN}_{\mathrm{aff}}^\ell(\mathbf{z}^\ell) + b_m^\ell = W_m^\ell \cdot \gamma \odot \mathbf{z}^\ell + W_m^\ell \cdot \beta + b_m^\ell$. Notice the same space of functions is available by *only* updating the b_m^ℓ term in $W_m^\ell \cdot \beta + b_m^\ell$. We simply switch to updating γ instead of β, i.e. unfreezing the LN weight and freezing the bias, in order to increase expressiveness (confirmed empirically in § 5.1). We use this version of BitFit throughout this work unless stated otherwise.

X-attention Tuning (Gheini et al., 2021) refers to fine-tuning only cross-attention (X-attention) and corresponding layer-norm parameters located in each decoder layer of a transformer model. This method is based on the importance of cross-attention for MT.

4 Experiments

Datasets We conduct experiments with a selection of 12 typologically and geographically diverse languages, paired with English. In our experiments, we fine-tune the pre-trained model on only one language pair and translation direction at a time (e.g. Italian → English). The parallel data for all languages is from TED talks in order to factor out the impact of the domain differences (except Finnish and Estonian which we only use for a separate control experiment). To pick these languages, we consider variation in language families

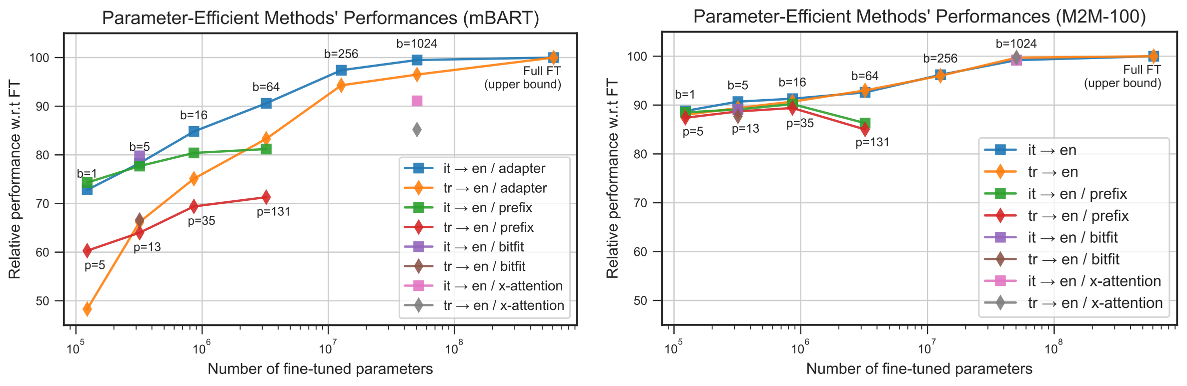

Figure 1: For increasing parameter budget, does prefix-tuning or adapters work best (RQ1)? We show relative MT performance over full fine-tuning vs. number of fine-tuned parameters for mBART and M2M-100. b and p refer to adapter bottleneck dimension and prefix length respectively. Due to the large effective sequence length, we limit prefix-tuning experiments.

and scripts. More details of the datasets are given in Appendix A.

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 also be used without fine-tuning, since it is initially pre-trained for MT (see § 2). We conduct experiments with both the base and the medium size M2M-100, to measure the impact of parent model size.

For all fine-tuning methods, we fine-tuned models with a maximum learning rate of 1e-4 with 2500 warm-up steps 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[2] (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.[3] On the recommendation of Marie et al. (2021) we report chrF (Popović, 2015) in Appendix C for each fine-tuning method.

Parameter Budget Selection In order to fairly compare different methods, we selected a series of parameter 'budgets', and adjusted the settings of each method such that they update the same number of parameters. To determine the parameter budgets, we used the number of trainable parameters for the cross-attention update and BitFit since these numbers are constant (Unlike adapters and prefix-tuning, where we have an adjustable bottleneck dimension). Additionally, when comparing adapters and prefix-tuning, we start from the parameter size of the smallest adapter where the bottleneck dimension is 1.[4]

5 Results and Discussion

In this section, we first compare the performance of various PEFTs on two language directions for different parameter budgets § 5.1. We then select a subset of these methods to test on ten language directions, in order to evaluate the effect of language similarity on the performance of PEFTs § 5.2. We use these results to explore the effect of parent model pre-training § 5.3 and parent model size § 5.3. We noticed that on the language directions with the smallest dataset size, adapter methods outperformed full fine-tuning, and therefore conducted control experiments showing that as dataset size decreases, adapters outperform full fine-tuning by a larger margin.

[2]Sacrebleu signature (BLEU): nrefs:1|case:mixed|eff:no|tok:13a|smooth:exp|version:2.0.0

[3]BLEU scores for each direction are given in Appendix C

[4]Each block corresponds to parameter budgets of approximately 50m, 320k, and 120k trainable parameters, representing {X-attention, Adapter-1024}, {BitFit, Adapter-5, Prefix-13}, and {Adapter-1, Prefix-5} respectively.

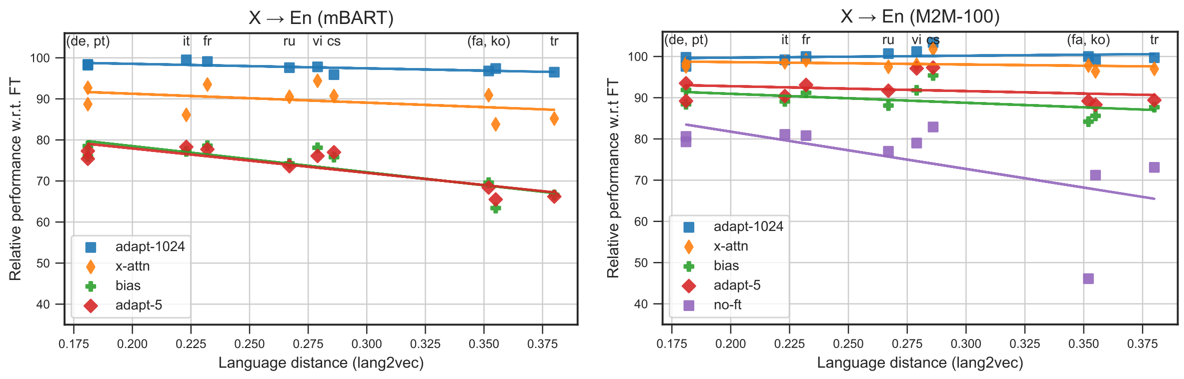

Figure 2: How does language similarity affect relative performance w.r.t. full fine-tuning (%) for PEFTs (RQ2)? Trend lines show the correlation between performance of PEFTs and language distance.

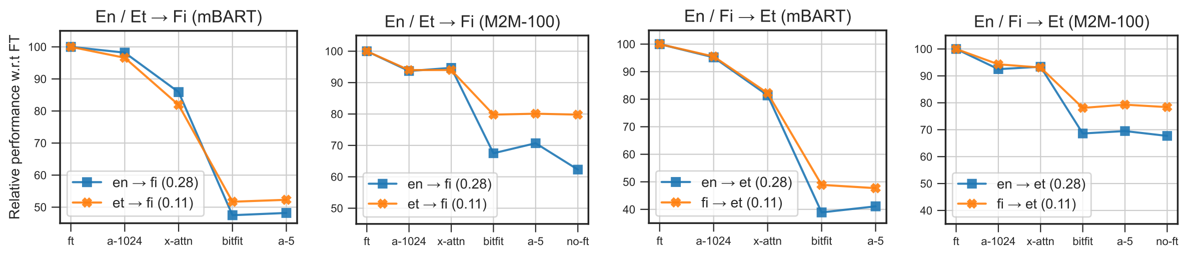

Figure 3: Decrease in relative performance (%) over full fine-tuning as the number of updated parameters decreases for translating into Finnish and Estonian with different source language (en, et, fi). `Lang2vec` distances for en↔fi, en↔et, and fi↔et are 0.28, 0.28 and 0.11 respectively.

5.1 RQ1: Comparing fine-tuning methods

Table 1 shows the **performance** of PEFTs in terms of BLEU score for it→en and tr→en. In the table, each block (separated with a dashed line) consists of PEFTs with approximately the same number of updated parameters. Adapters outperform other methods at almost all parameter budgets for both mBART and M2M-100, except the smallest budget where only 120k parameters are updated. In this block, prefix-tuning (prefix-5) performs better than adapters for mBART. However, when the portion of fine-tuned parameters increases, as shown in Figure 1, prefix-tuning quickly falls behind adapters, confirming previous findings (He et al., 2021). Furthermore, in terms of **training speed/memory cost**, prefix-tuning slows down the training relative to adapters, and imposes a significant memory cost due to a large effective sequence length; see also Appendix B.[5]

As for the methods that fine-tune existing parameters, BitFit is better than adapters for mBART and worse for M2M, whereas X-attention performs worse than adapters in all cases. We confirm that our method of tuning layer norm weights rather than biases improves performance, outperforming adapters for mBART on the language pairs in Table 1. Averaging across 10 language pairs, adapters still outperform BitFit for both parent models, see Figure 4.

	mBART		M2M-100	
	x→en	en→x	x→en	en→x
Adapter (b=1024)	*-0.43*	*0.11*	*0.23*	*-0.07*
X-attention	-0.69	*-0.21*	*-0.28*	*-0.07*
Adapter (b=5)	-0.85	*-0.53*	*-0.26*	*-0.22*
BitFit	-0.84	-0.64	*-0.47*	*-0.33*
No FT	-	-	-0.60	-0.72

Table 2: Pearson correlation coefficients between relative performance w.r.t. fine-tuning and language distance. Negative correlation means that relative performance tends to decrease as the distance between source and target language increases. Numbers in italics are not statistically significant (p=0.05).

5.2 RQ2: Impact of language relatedness

In order to evaluate how language similarity between translation pairs affects the performance of different PEFTs, we extend our experiments to 10

[5]Prefix-13 causes a 30% slow-down in training speed relative to adapter-5.

languages paired with English (x→en, en→x), representing a diverse set of linguistic typology. Figure 2 shows performance w.r.t. full fine-tuning, for mBART and M2M respectively.[6]

We found that similarity between source and target languages impacts the performance of PEFTs, with distantly related languages (e.g. English and Korean) leading to lower performance for the methods with a small number of updated parameters such as BitFit and adapter-5. And so when translating between distantly related languages, we need to tune more parameters to match full fine-tuning and get the most out of the parent model.

More concretely, relative performance w.r.t. full FT is negatively correlated with language distance measured by `lang2vec`.[7] These correlations are stronger for mBART than M2M. Methods which tune existing parameters (X-attention and BitFit) and M2M with no fine-tuning show higher correlation than adapters with similar parameter budgets; see Table 2. One explanation could be that adding parameters, and therefore increasing model capacity with adapters is beneficial for overcoming the difficulty of translating distant languages.

To investigate whether our findings extend beyond English-centric settings, we designed another set of experiments. We picked 3 languages from MultiParaCrawl, 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. Figure 3 shows results 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. Thus, when the source and target languages are typologically similar, PEFTs make better use of the parent model.

[6]Due to space limitations, Figure 2 shows results for x→en. en→x results are given in Appendix C

[7]lang2vec a python package based on the URIEL typology database (Littell et al., 2017). For language distance, we compute the cosine distance between typological features vectors of languages that consists of syntactic, phonological and inventory features.

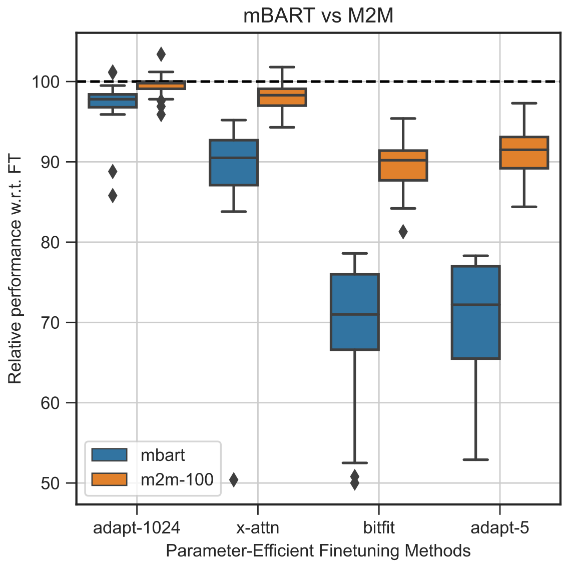

Figure 4: How does the type of parent model (different pre-training objective) affect performance (RQ3)? Statistics of relative performance w.r.t. full fine-tuning (%) for all languages (x↔e) when the model is initialized with mBART or M2M-100. The dashed line refers to full fine-tuning performance.

5.3 RQ3: Impact of parent model

Pre-training Objective Figure 4 shows the overall performances for PETFs aggregated over all languages (x↔en) when the model is initialized with mBART or M2M-100. In general, PEFTs for M2M-100 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 (monolingual[8]) denoising objective. Thus, more parameters are required at fine-tuning time to 'learn' the MT task for mBART. Finally, we note mBART results have a higher variance than M2M-100 (see Fig. 4), due to the higher negative correlation with language distance.

Model Size We investigate how parent model size affects the performance of fine-tuning methods, comparing M2M-100's base model (484M) to its medium model (1.2B). Table 3 shows the average performance of full fine-tuning and small-size adapters corresponding to approximately 300K new parameters[9]. No fine-tuning (no FT) results

[8]Although mBART-50 pre-trained on 50 languages, the pre-training objective does not use any cross-lingual signal.

[9]Both adapter-5 in the base model and adapter-2 in the medium model correspond to roughly the same number of

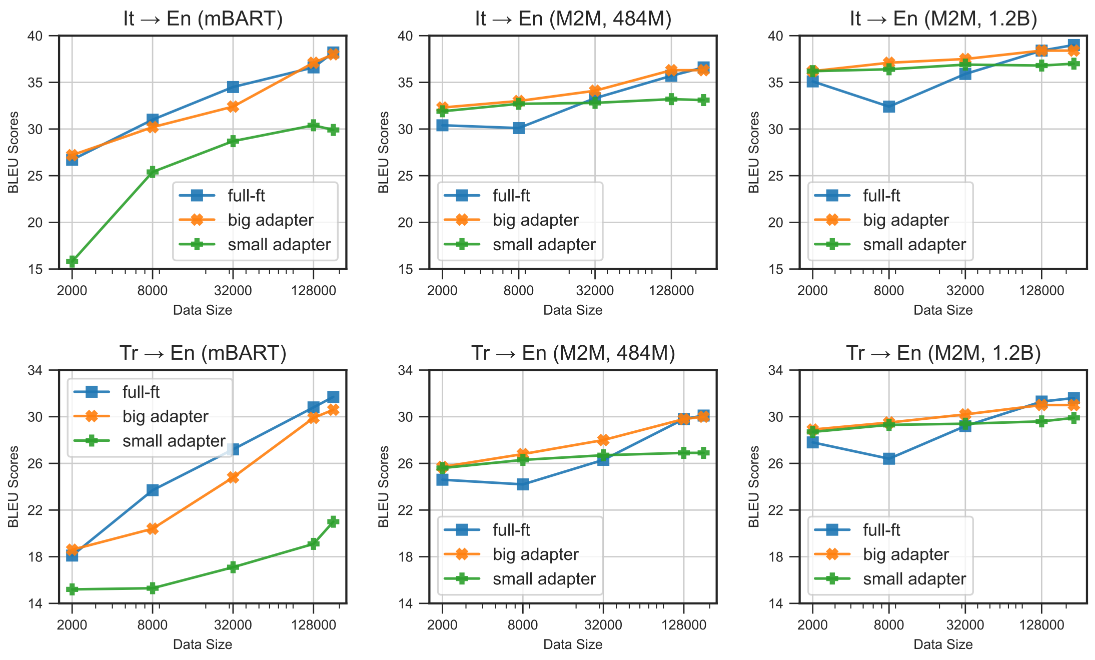

Figure 5: How does the amount of parallel data affect fine-tuning performance (RQ4)? BLEU scores for various subsets of the full training data for Italian to English and Turkish to English, with base (484m parameters) and medium (1.2 billion parameters) M2M and mBART parent models. Big and small adapters have ≈50m and ≈300k parameters respectively for all models.

	Model		
	Base (418m)	Med. (1.2b)	Δ BLEU
en →x			
No FT	21.9	24.3	2.4
Small adapter	24.8	27.4	2.6
Full FT	27.4	28.4	1.0
x →en			
No FT	26.1	28.5	2.4
Small adapter	31.7	35.3	3.6
Full FT	34.4	36.6	2.2

Table 3: How does parent model size affect performance (RQ3)? Average BLEU score across 10 languages for the base (484m parameters) and medium (1.2 billion parameters) M2M parent models, when tuning all parameters ('full FT'), when tuning small adapters, and when tuning no parameters ('no FT'). We also show the increase in BLEU when moving from the base to the medium model. See Appendix C for individual results.

are also shown, representing *lower* bounds.

Predictably, the medium model outperforms the base model across all fine-tuning methods. The magnitude of this improvement is larger when translating into English (*x*→*en*) vs. *x*→*en*, and the increase for small adapters is larger than for other methods. When translating into English, small adapters with the medium model outperform *full fine-tuning* of the base model for most languages despite tuning only 0.03% of its parent model parameters. For *en*→*x*, small adapters are still competitive with full fine-tuning of the base model with almost the same average performance. But for distantly related languages to English (Farsi, Korean and Turkish), adapters' (1.2B) performance falls behind full fine-tuning of the base model.

When it is used without *any* parameter updates ('no FT'), the medium model (while outperforming the base model for no FT) is not competitive with small size adapters for the base model, in either direction (*x*↔*en*). Furthermore, relative performance w.r.t. full fine-tuning is still negatively correlated with language distance (see Appendix Table 5). Therefore, even at large scales, parameter efficient fine-tuning is useful, taking MT performance to the *upper* bound of a smaller model.

5.4 RQ4: Impact of fine-tuning dataset size

We noticed that for the datasets with the smallest amount of training data (Vietnamese and Czech), PEFTs outperformed full fine-tuning (see Appendix C). We therefore designed a control experiment to test for the effect of the training data size on PEFT' performance, taking a random subset of

trainable parameters (0.07% of 484M and 0.03% of 1.2B total parameters).

sizes 2000, 8000, 32000 and 128000 training examples for Italian to English and Turkish to English. We then evaluated full fine-tuning, large adapters (≈50m parameters) and small adapters (≈300k parameters) on each dataset; see Figure 5.

For all models, at the smallest dataset size, large adapters outperformed full fine-tuning, and for M2M full fine-tuning only catches up at 128k examples. For mBART, small adapters lag far behind, indicating they do not provide enough capacity to 'learn' the MT task. For M2M however, small adapters are on a par with larger ones for small dataset sizes, but fall behind as dataset size increases. Again, we believe this is because more capacity is needed to get the most out of larger datasets.

Chen et al. (2022) explore the effect of fine-tuning dataset size for RoBERTa fine-tuned on English NLU tasks, finding PEFTs outperform full fine-tuning for dataset size <1000. Interestingly, for mBART, similarly small dataset sizes are required for outperforming full fine-tuning. However, for M2M, we see adapters outperforming up until dataset sizes of ≈ 128k. Perhaps the 'gap' between RoBERTa's masked language model pre-training objective and the fine-tuning objective is similar to the gap between mBART's pre-training objective and MT, whereas since M2M is pre-trained for MT, leaving the base model unchanged is viable up to larger fine-tuning dataset sizes. We leave further exploration of this to future work. Finally, we observe that full fine-tuning always converges in fewer iterations than the adapter methods, in a result similar to that of Chen et al. (2022).

6 Related Work

PEFTs have been widely used for fine-tuning Transformer models to new tasks, domains or languages. Adapters (Houlsby et al., 2019) have been used in 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. 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.

Some of these methods have been compared in a controlled setting for English classification tasks (Chen et al., 2022) or only a single language pair (English and Romanian) for MT (He et al., 2021). Chen et al. (2022) test PEFTs for various English classification tasks and observe that on the tasks with the smallest dataset sizes, PEFTs outperform fine-tuning, but they do not conduct a control experiment varying dataset size and parent model for a single task as we do. 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 PEFTs for MT, and we aim to fill this gap.

7 Conclusion

Do PEFTs work for MT? We found that the answer depends on multiple factors: the particular method, the backbone model, the number of tuned parameters and the fine-tuning language pair. Adapter layers usually have the highest performance out of all parameter-efficient fine-tuning methods (§ 5.1), although for the smallest parameter budgets we consider, prefix tuning outperforms adapters for mBART. For large parameter budgets (≈50m parameters) adapters 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 PEFTs only **outperform** full FT for smaller dataset sizes (§ 5.4), less than around ≈2k examples for mBART and ≈128k for M2M. Future work could explore in detail how the difference between pre-training objective and fine-tuning task affects this phenomenon.

Using PEFT with a larger model (M2M-100 medium size) can outperform full FT of a smaller model (M2M-100 base size). However when translating in the en$\rightarrow$x direction where x is distantly related to English e.g. Korean, full FT is superior (§ 5.3). More generally, distantly related language pairs require more parameters to be tuned to get close to full FT, for all methods (§ 5.2). 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

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. *CoRR*, abs/1907.05019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization.

Ankur Bapna, Isaac Caswell, Julia Kreutzer, Orhan Firat, Daan van Esch, Aditya Siddhant, Mengmeng Niu, Pallavi Baljekar, Xavier Garcia, Wolfgang Macherey, et al. 2022. Building machine translation systems for the next thousand languages. *arXiv preprint arXiv:2205.03983*.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1538–1548, Hong Kong, China. Association for Computational Linguistics.

Alexandre Berard, Dain Lee, Stephane Clinchant, Kweonwoo Jung, and Vassilina Nikoulina. 2021. Efficient inference for multilingual neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8563–8583, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Guanzheng Chen, Fangyu Liu, Zaiqiao Meng, and Shangsong Liang. 2022. Revisiting parameter-efficient tuning: Are we really there yet?

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–391, Vancouver, Canada. Association for Computational Linguistics.

Chenhui Chu and Rui Wang. 2018. A survey of domain adaptation for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1304–1319, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. Beyond english-centric multilingual machine translation. *arXiv preprint*.

Mozhdeh Gheini, Xiang Ren, and Jonathan May. 2021. Cross-attention is all you need: Adapting pretrained Transformers for machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1765, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mitchell A Gordon, Kevin Duh, and Jared Kaplan. 2021. Data and parameter scaling laws for neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5915–5922, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Thamme Gowda and Jonathan May. 2020. Finding the optimal vocabulary size for neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3955–3964, Online. Association for Computational Linguistics.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4884–4896, Online. Association for Computational Linguistics.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *CoRR*, abs/2110.04366.

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. In *International Conference on Machine Learning*, pages 2790–2799.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 565–576, Online. Association for Computational Linguistics.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah Smith. 2021. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *International Conference on Learning Representations*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Proceedings of ICLR*.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2018. The IIT Bombay English-Hindi parallel corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Patrick Littell, David R. Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. URIEL and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 8–14.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Benjamin Marie, Atsushi Fujita, and Raphael Rubino. 2021. Scientific credibility of machine translation research: A meta-evaluation of 769 papers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7297–7306, Online. Association for Computational Linguistics.

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, Copenhagen, Denmark. Association for Computational Linguistics.

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.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Jerin Philip, Alexandre Berard, Matthias Gallé, and Laurent Besacier. 2020. Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods* *in Natural Language Processing (EMNLP)*, pages 4465–4470, Online. Association for Computational Linguistics.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

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.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 529–535, New Orleans, Louisiana. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

Danielle Saunders. 2021. Domain adaptation and multi-domain adaptation for neural machine translation: A survey. *CoRR*, abs/2104.06951.

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

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pages 5926–5936.

Asa Cooper Stickland, Alexandre Bérard, and Vassilina Nikoulina. 2021a. Multilingual domain adaptation for NMT: decoupling language and domain information with adapters. *Sixth Conference on Machine Translation (WMT2021)*.

Asa Cooper Stickland, Xian Li, and Marjan Ghazvininejad. 2021b. Recipes for adapting pre-trained monolingual and multilingual models to machine translation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3440–3453, Online. Association for Computational Linguistics.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, pages 5986–5995.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning. *arXiv preprint arXiv:2008.00401*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Ahmet Üstün, Alexandre Berard, Laurent Besacier, and Matthias Gallé. 2021. Multilingual unsupervised neural machine translation with denoising adapters. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6650–6662, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language adaptation for truly Universal Dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2302–2315, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2021. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

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, Austin, Texas. Association for Computational Linguistics.

A Reproducibility Report

Datasets All datasets that are used in our experimetns are publicly available. We used TED talks (Qi et al., 2018) for (cs, fr, ko, ru, pt, tr, fa)$\leftrightarrow$en, IWSLT15 and IWSTL17 (Cettolo et al., 2012) for vi$\leftrightarrow$en and (it, de)$\leftrightarrow$en respectively, IITB (Kunchukuttan et al., 2018) for hi$\leftrightarrow$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. For full fine-tuning (and not other methods) with the 1.2 billion size M2M model we use the Adafactor optimizer (Shazeer and Stern, 2018) in order to save memory (and use learning rate 5e-5), and otherwise use the Adam optimizer (Kingma and Ba, 2014). 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 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 6 shows chrF scores[10] for the experiments comparing different PEFTs on it$\rightarrow$en and tr$\rightarrow$en (Table 1). These results confirms that the trends discussed in Section 4 are the same regardless of metric used for MT quality.

In Tables 7, 8 and 9, 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. Note that for M2M, for the two smallest dataset sizes (cs and vi) we see adapter-1024 (and adapter-2 for the medium size M2M) outperforming full fine-tuning, similarly to § 5.4.

In Table 8 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

[10] Sacrebleu signature (chrF2++):
nrefs:1|case:mixed|eff:yes|nc:6|nw:2|space:no|version:2.0.0

Language	Language family	Dataset source	Train size (k)	Dev size (k)	Test size (k)
Czech **(cs)**	Slavic	TED	103	3.5	3.8
French **(fr)**	Romance	TED	192	4.3	4.9
Korean **(ko)**	Korean	TED	205	4.4	5.6
Russian **(ru)**	Slavic	TED	208	4.8	5.5
Italian **(it)**	Romance	TED	231	0.9	1.6
Portuguese **(pt)**	Romance	TED†	184	4	4.9
Turkish **(tr)**	Turkic	TED	182	4	5
Vietnamese **(vi)**	Austri-Asiatic	TED†	133	1.6	1.3
German **(de)**	Germanic	TED†	206	0.9	1.6
Farsi **(fa)**	Iranian	TED	150	3.9	4.5
Finnish* **(fi)**	Finnic	mParacrawl	200	3	3
Estonian* **(et)**	Finnic	mParacrawl	200	3	3

Table 4: Details of dataset that is used in our experiments. We gather language pairs (x↔en) from TED (Qi et al., 2018) and IWSLT† (Cettolo et al., 2012) that are both compiled from TED talks. '*' indicates a set of separate controlled experiments where we randomly sampled 200k parallel sentences from MultiParacrawl (mParaCrawl) for corresponding language pairs.

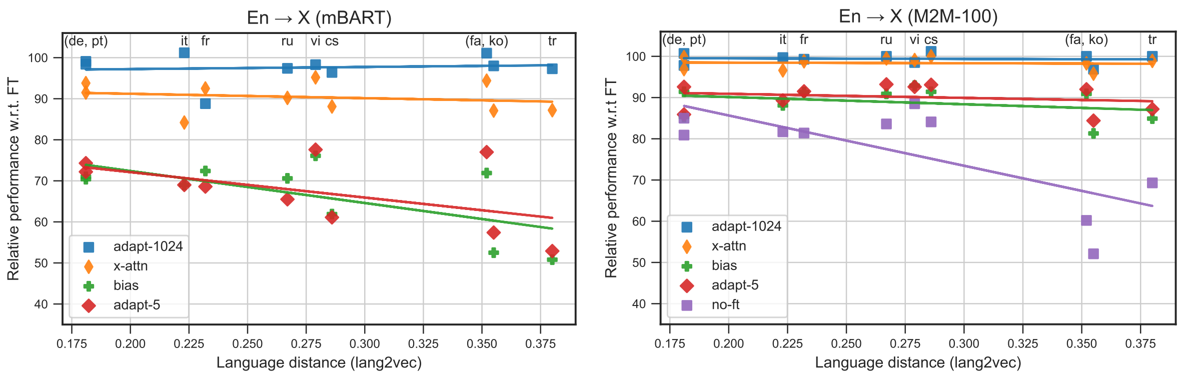

Figure 6: How does language similarity affect relative performance w.r.t. full fine-tuning (%) , for various methods with M2M as the parent model (RQ4).

	en→x	*x→en*
Adapter (b=2)	-0.75	*-0.39*
No fine-tuning	-0.75	*-0.55*

Table 5: Correlation coefficients between language distance and relative performance for the 1.2 billion size M2M model; see also Table 2. Numbers in italics are not statistically significant (p=0.05).

models. For the *x→en* direction almost all of our methods based on pre-trained models outperformed the 'from scratch' baseline, however in the *en→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.

	mBART		M2M-100	
	it→en	tr→en	it→en	tr→en
Full FT	59.4	53.3	58.2	52.6
X-attention	56.6	48.9	57.7	51.6
Adapter (b=1024)	59.2	52.3	57.8	52.2
Prefix (p=13)	52.4	42.8	55.3	49.7
BitFit (LN-bias)	51.8	41.7	55.0	49.3
BitFit (LN-weights)	52.7	42.8	55.1	49.5
Adapter (b=5)	52.4	42.8	55.5	49.8
Prefix (p=5)	51.4	41.4	54.9	49.5
Adapter (b=1)	50.5	36.5	55.0	49.5

Table 6: chrF 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.

M2M-100	No. Params	fa 150k	it 230k	de 208k	ru 208k	ko 205k	fr 192k	pt 184k	tr 182k	vi 133k	cs 103k
en→x											
Full FT	484m	17.6	32.8	32.0	22.0	9.6	41.3	42.1	17.9	33.9	24.5
Full FT (1.2b)	1.2b	17	33.7	33.6	23.8	9.9	42.8	43.1	18.8	35.2	26.3
Adapter (b=1024)	50m	17.6	32.7	31.3	22.0	9.3	41.0	42.4	17.9	33.4	24.8
X-attention	50m	17.3	31.7	31.0	21.9	9.2	40.8	42.0	17.7	33.6	24.5
BitFit	335k	16.0	28.9	27.3	20.0	7.8	37.7	38.5	15.2	31.5	22.4
Adapter (b=5)	320k	16.2	29.3	27.5	20.5	8.1	37.8	39.0	15.6	31.4	22.8
Adapter (b=2; 1.2B)	344k	14.6	32.5	32.1	23.1	8.9	42.2	43.1	16.7	34.6	26.4
No FT (1.2B)	0	9.7	29.6	29.9	21.1	5.5	37.6	39.6	13.2	32.9	24.0
No FT (484M)	0	10.6	26.8	25.9	18.4	5.0	33.6	35.8	12.4	30.0	20.6
x→en											
Full FT	484m	32.3	36.6	37.2	27.8	22.2	43.2	47.9	30.1	34.3	32.8
Full FT (1.2b)	1.2b	36.1	39	39.3	29.9	23.9	45.1	49.8	31.6	35.7	35.3
Adapter (b=1024)	50m	32.3	36.3	36.3	28.0	22	43.2	47.8	30	34.7	33.9
X-attention	50m	31.6	36.1	36.3	27.1	21.4	42.8	47.1	29.2	33.6	33.4
BitFit	335k	27.2	32.6	32.9	24.5	19.0	39.4	44	26.4	31.5	31.3
Adapter (b=5)	320k	28.8	33.1	33.2	25.5	19.6	40.2	44.8	26.9	33.3	31.9
Adapter (b=2; 1.2B)	344k	31.5	37.3	37.7	28.9	22.2	44.0	48.7	29.9	37.5	35.6
No FT (1.2B)	0	14.9	32.5	32.1	24.1	17.6	37.5	42.0	24.2	29.9	30.1
No FT (484m)	0	14.9	29.7	29.5	21.4	15.8	34.9	38.6	22.0	27.1	27.2

Table 7: x↔en results in terms of BLEU for M2M-100 experiments.

mBART	No. Params	fa 150k	it 230k	de 208k	ru 208k	ko 205k	fr 192k	pt 184k	tr 182k	vi 133k	cs 103k
en→x											
Full FT	610m	17.8	32.9	33.1	23.5	10.1	42.7	43.5	18.7	35.2	25.2
Adapter (b=1024)	50m	18.0	33.3	32.8	22.9	9.9	37.9	42.8	18.2	34.6	24.3
X-attention	50m	16.8	27.7	30.3	21.2	8.8	39.5	40.8	16.3	33.5	22.2
BitFit	335k	12.8	22.7	23.3	16.6	5.3	30.9	30.9	9.5	26.8	15.6
Adapter (b=5)	320k	13.7	22.7	23.9	15.4	5.8	29.3	32.3	9.9	27.3	15.4
From Scratch	40m	25.0	23.9	22.9	15.3	5.5	32.5	35.4	11.0	26.2	17.0
x→en											
Full FT	610m	33.9	38.2	34.1	29.6	23.5	44.8	49.4	31.7	36.0	34.3
Adapter (b=1024)	50m	32.8	38.0	33.5	28.9	22.9	44.4	48.6	30.6	35.2	32.9
X-attention	50m	30.8	32.9	31.6	26.8	19.7	41.9	43.8	27.0	34.0	31.1
BitFit	335k	23.6	29.5	25.9	22.0	14.9	35.2	38.8	21.1	28.1	26.0
Adapter (b=5)	320k	23.2	29.9	25.7	21.8	15.4	34.8	38.2	21.0	27.4	26.4
From Scratch	40m	20.9	27.3	26.3	19.2	11.6	34.3	39.4	19.1	21.9	23.8

Table 8: x↔en results in terms BLEU for mBART experiments.

	M2M-100						**mBART**					
	en <	> fi	en <	> et	fi <	> et	en <	> fi	en <	> et	fi <	> et
Full FT	43.9	37.9	40.4	33.4	33.6	33.4	45.4	39.8	42.3	35.5	34.8	35.4
Adapter (b=1024)	42.7	35.5	39.6	30.9	31.6	31.5	45.3	39.1	41.9	33.8	33.6	33.8
X-attention	42.9	35.9	39.5	31.2	31.6	31.1	40.6	34.2	36.1	28.9	28.5	29.1
BitFit	35.4	25.6	33.9	22.9	26.8	26.1	28.9	18.9	25.0	13.8	18.0	17.3
Adapter (b=5)	36.1	26.8	34.3	23.2	26.9	26.5	28.9	19.2	24.3	14.6	18.2	16.9
Adapter (b=2; 1.2B)	41.9	32.0	39.6	28.8	31.8	31.4	-	-	-	-	-	-
No FT (1.2B)	40.3	28.6	38.1	27.3	31.3	31.0	-	-	-	-	-	-
No FT (484M)	34.1	23.6	32.9	22.6	26.8	26.2	-	-	-	-	-	-

Table 9: (en, et, 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. 3).