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].

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

[1] We will share our code and scripts to reproduce all experiments in the paper.

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:

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

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 a up projection $W_\mathrm{u} \in \mathcal{R}^{b \times d}$. Finally, a residual connection with the input h^ℓ is added to the output of the adapter: $\mathbf{h}^\ell \rightarrow \mathrm{A}_\ell(\mathbf{h}^\ell) + \mathbf{h}^\ell$. We write 'adapter-$b$' to mean adapters with bottleneck dimension b throughout this work.

Language	Language family	Dataset source	Parallel data (K)
Czech **(cs)**	Slavic	TED	103
French **(fr)**	Romance	TED	192
Korean **(ko)**	Korean	TED	205
Russian **(ru)**	Slavic	TED	208
Italian **(it)**	Romance	IWSLT17	231
Portuguese **(pt)**	Romance	TED	184
Turkish **(tr)**	Turkic	TED	182
Vietnamese **(vi)**	Austri-Asiatic	IWSLT15	133
German **(de)**	Germanic	IWSLT17	206
Farsi **(fa)**	Iranian	TED	150
Hindi **(hi)**	Indic	IITB	1600
Finnish* **(fi)**	Finnic	mParacrawl	200
Estonian* **(et)**	Finnic	mParacrawl	200

Table 1: Languages that are used in the experiments. We gather language pairs (x↔en) from TED (Qi et al., 2018), IWSLT (Cettolo et al., 2012), MultiParacrawl (mParaCrawl) and IITB (Kunchukuttan et al., 2018). '*' indicates that we randomly sampled 200k parallel sentences from the original datasets for corresponding language pairs.

Fine-tuning Method	# Trainable Parameters	Parameter Ratio (%)
mBART		
Full FT	610m	100
Adapter (b=1024)	50m	8.2
X-attention	50m	8.2
BitFit	335k	0.05
Adapter (b=5)	320k	0.05
Prefix (p=13)	320k	0.05
Adapter (b=1)	123k	0.02
Prefix (p=5)	123k	0.02
M2M-100 (base, 484M parameters)		
Full FT	484m	100
Adapter (b=1024)	50m	10.3
X-attention	50m	10.3
BitFit	335k	0.07
Adapter (b=5)	320k	0.07
Prefix (p=13)	320k	0.07
Adapter-1	123k	0.03
Prefix (p=5)	123k	0.03
No FT	0	0
M2M-100 (medium, 1.2B parameters)		
Adapter (b=2)	344k	0.03*
No FT	0	0

Table 2: Fine-tuning methods used in our experiments. '*' indicates that Adapter-2's parameter ratio is calculated w.r.t M2M-100 (medium, 1.2B), but its parameter count matches Adapter (b=5) (base) and BitFit (base).

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

$$E_p = \text{Concat}(V^0, E), \tag{2}$$

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 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 following affine transformation:

$$\text{LN}^\ell_{\text{aff}}(\mathbf{z}^\ell) = \gamma \odot \mathbf{z}^\ell + \beta \tag{3}$$

where $\mathbf{z}^\ell$ is the normalized input after a residual connection. $\gamma, \beta \in \mathcal{R}^d$ 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. $W^\ell_m \cdot \text{LN}^\ell_{\text{aff}}(\mathbf{z}^\ell) + b^\ell_m = W^\ell_m \cdot \gamma \odot \mathbf{z}^\ell + W^\ell_m \cdot \beta + b^\ell_m$ Notice the same space of functions is available by *only* updating the b^ℓ_m term in $W^\ell_m \cdot \beta + b^\ell_m$. We simply switch to updating γ instead of β, 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

	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 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

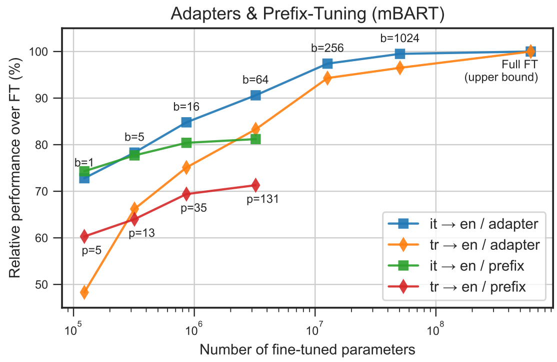

Figure 1: Relative MT performance over full fine-tuning vs. number of fine-tuned parameters for mBART. b and p refer to adapter bottleneck dimension and prefix length respectively. Due to the large effective sequence length, we limit prefix-tuning experiments.

method is based off the importance of cross-attention for MT.

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[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.

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[4]). 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

[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] Due to computational constraints, we did not perform experiments on all combinations of method and language pair.

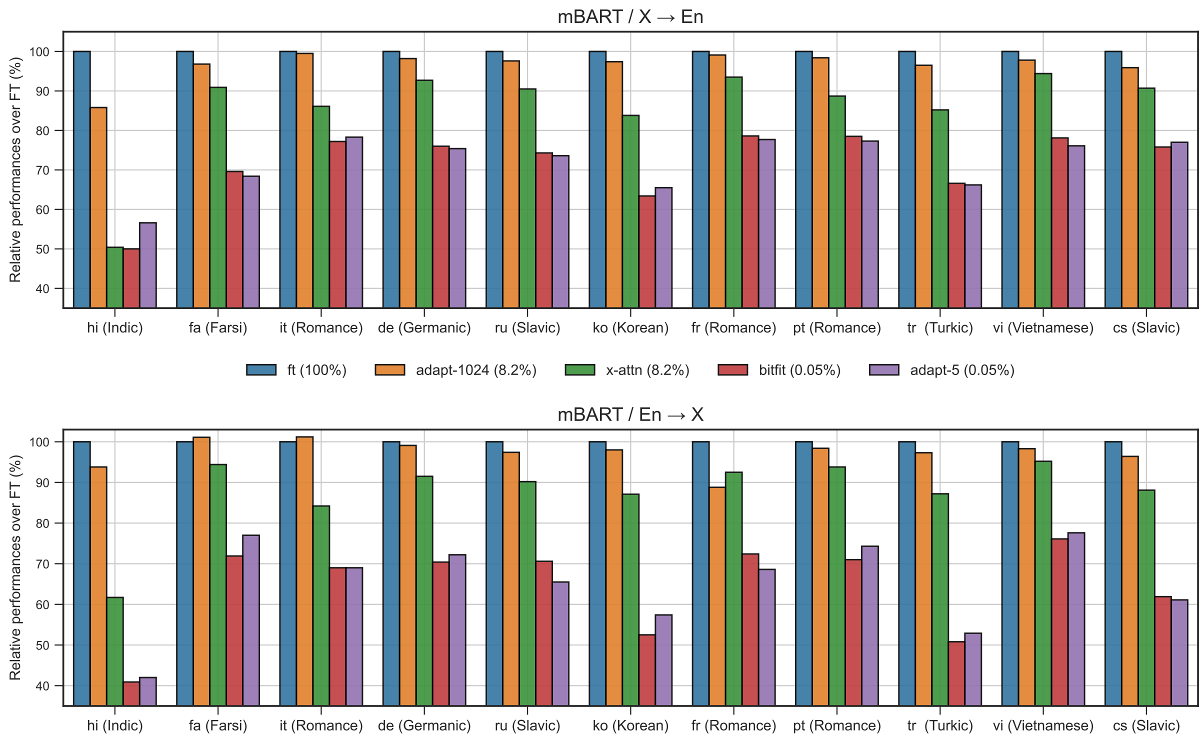

Figure 2: Relative performances over the full fine-tuning (%) for $x \leftrightarrow en$ when mBART is fine-tuned by using different methods. We show in brackets the percentage of total mBART parameters that are fine-tuned for each method.

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 ($x \leftrightarrow en$). For other languages this decreases to ≈85%, and for Hindi (hi) to 50.4% and 61.7% in $x \rightarrow en$ and $en \rightarrow x$ respectively.

For smaller parameter budgets (BitFit and adapter-5; 0.05% of mBART parameters), we see better performance when translating *into* English ($x \rightarrow en$). 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 $en \rightarrow x$ (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 ($x \leftrightarrow en$). 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] English is the largest portion (55M tokens, 300GB) of the data that is used for mBART pre-training (Liu et al., 2020)

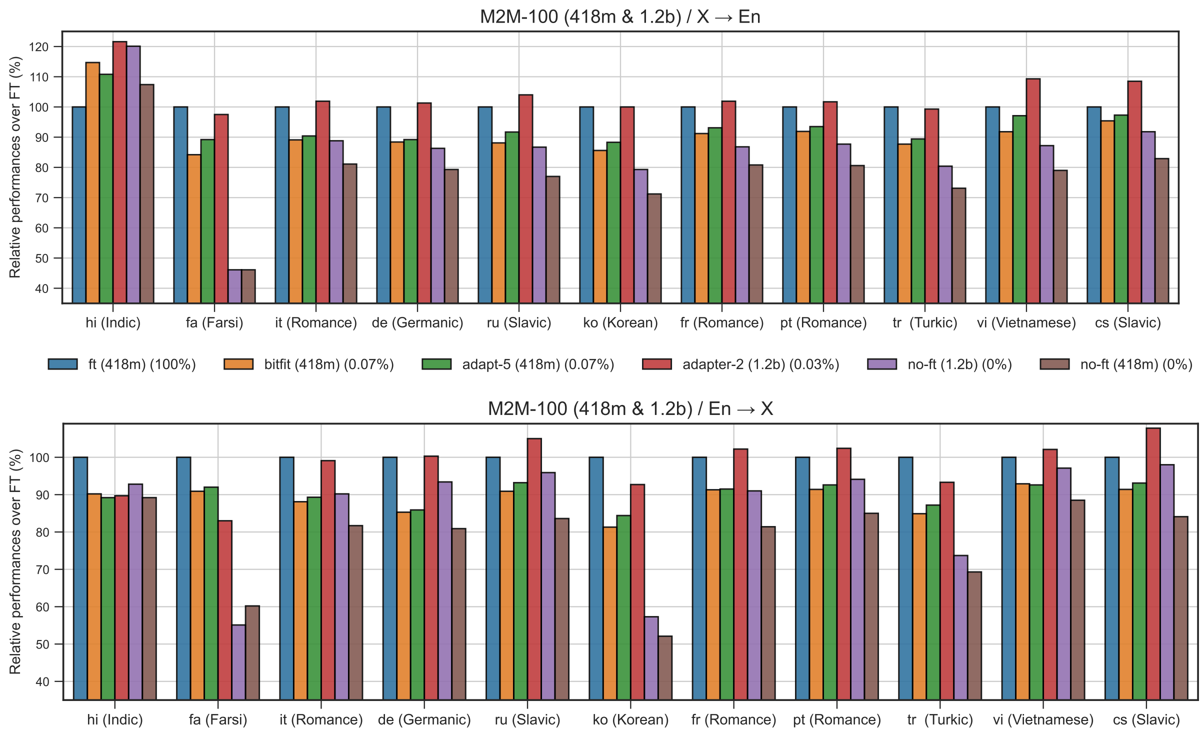

Figure 3: Relative performance w.r.t. full fine-tuning (%) for *x*↔*en* when M2M-100 is fine-tuned with different methods. Note that 'adapt-1024' and 'x-attn' are not shown as they perform similarly to full FT (§ Appendix C)

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 (monolingual[6]) denoising objective. Perhaps more parameters are required at fine-tuning time to 'learn' the MT task for mBART. Tun-

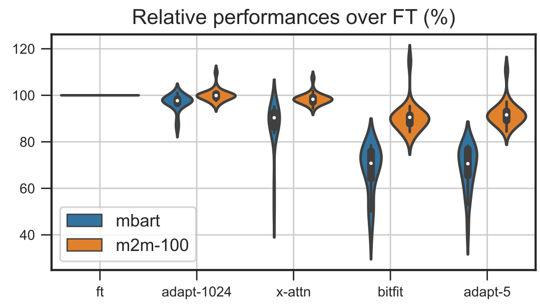

Figure 4: Relative performance w.r.t. full fine-tuning (%) for all languages (*x*↔*en*) when the model is initialized with mBART or M2M-100.

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

ing fewer parameters is better at revealing innate model capability than full FT or larger adapters, with the differences between parent models pretty minor for larger parameter budgets. Finally, we note mBART results have a higher variance than M2M-100 (see Fig. 4).

5.2 Impact of parent model size

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 ($x \rightarrow en$), 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 $x \rightarrow en$, 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 ($x \leftrightarrow en$). 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.

5.3 Impact of language relatedness

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 pretrained **model capability**, we find methods that don't add any additional parameters (x-attn 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 $x \rightarrow en$, BitFit (0.84) > adapter-5 (0.77) > x-attn (0.73) > adapter-1024 (0.66), where the number in brackets is the Pearson product-moment correlation coefficient. We have the same ranking for $en \rightarrow x$. This shows that both a small parameter budget and not adding additional parameters i.e. adapters seems to be important for

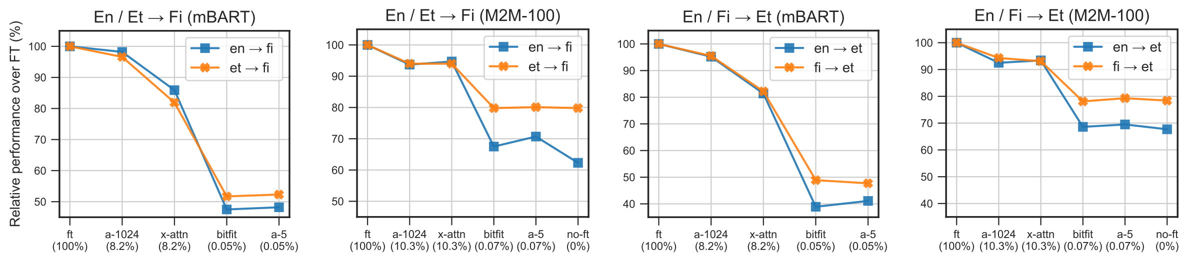

Figure 5: Decrease in relative performance (%) over full fine-tuning as the number of updated parameters decreases for translating into Finnish and Estonian with different source language (en, et, fi).

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

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 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.

Damián Blasi, Antonios Anastasopoulos, and Graham Neubig. 2021. Systematic inequalities in language technology performance across the world's languages. *arXiv preprint arXiv:2110.06733*.

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.

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*.

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.

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.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems*.

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.

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*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

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.

Language	Dev Size (k)	Test Size (k)	Train Size (k)
Czech **(cs)**	3.5	3.8	103
French **(fr)**	4.3	4.9	192
Korean **(ko)**	4.4	5.6	205
Russian **(ru)**	4.8	5.5	208
Italian **(it)**	0.9	1.6	231
Portuguese **(pt)**	4	4.9	184
Turkish **(tr)**	4	5	182
Vietnamese **(vi)**	1.6	1.3	133
German **(de)**	0.9	1.6	206
Farsi **(fa)**	3.9	4.5	150
Hindi **(hi)**	0.5	2.5	1600
Finnish* **(fi)**	3	3	200
Estonian* **(et)**	3	3	200

Table 4: Train, dev and test splits for the languages that are used in the experiments, in thousands of parallel sentences. '*' indicates that we randomly sampled parallel sentences from the original datasets for corresponding language pairs.

	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 5: 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.

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)↔en, IWSLT15 and IWSTL17 (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 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[7] for the experiments comparing different parameter-efficient methods on it→en and tr→en (Table 3). These results confirms 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 sentencepiece (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→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.

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

M2M-100	No. Params	hi 1.6m	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	19.4	17.6	32.8	32.0	22.0	9.6	41.3	42.1	17.9	33.9	24.5
Adapter (b=1024)	50m	18.6	17.6	32.7	31.3	22.0	9.3	41.0	42.4	17.9	33.4	24.8
X-attention	50m	18.3	17.3	31.7	31.0	21.9	9.2	40.8	42.0	17.7	33.6	24.5
BitFit	335k	17.5	16.0	28.9	27.3	20.0	7.8	37.7	38.5	15.2	31.5	22.4
Adapter (b=5)	320k	17.3	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	17.4	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	18.0	9.7	29.6	29.9	21.1	5.5	37.6	39.6	13.2	32.9	24.0
No FT (484M)	0	17.3	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	20.4	32.3	36.6	37.2	27.8	22.2	43.2	47.9	30.1	34.3	32.8
Adapter (b=1024)	50m	22.4	32.3	36.3	36.3	28.0	22	43.2	47.8	30	34.7	33.9
X-attention	50m	21.9	31.6	36.1	36.3	27.1	21.4	42.8	47.1	29.2	33.6	33.4
BitFit	335k	23.4	27.2	32.6	32.9	24.5	19.0	39.4	44	26.4	31.5	31.3
Adapter (b=5)	320k	22.6	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	24.8	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	24.5	14.9	32.5	32.1	24.1	17.6	37.5	42.0	24.2	29.9	30.1
No FT (484m)	0	21.9	14.9	29.7	29.5	21.4	15.8	34.9	38.6	22.0	27.1	27.2

Table 6: x↔en results in terms of BLEU for M2M-100 experiments.

mBART	No. Params	hi 1.6m	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	19.3	17.8	32.9	33.1	23.5	10.1	42.7	43.5	18.7	35.2	25.2
Adapter (b=1024)	50m	18.1	18.0	33.3	32.8	22.9	9.9	37.9	42.8	18.2	34.6	24.3
X-attention	50m	11.9	16.8	27.7	30.3	21.2	8.8	39.5	40.8	16.3	33.5	22.2
BitFit	335k	7.9	12.8	22.7	23.3	16.6	5.3	30.9	30.9	9.5	26.8	15.6
Adapter (b=5)	320k	8.1	13.7	22.7	23.9	15.4	5.8	29.3	32.3	9.9	27.3	15.4
From Scratch	40m	5.3	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	22.6	33.9	38.2	34.1	29.6	23.5	44.8	49.4	31.7	36.0	34.3
Adapter (b=1024)	50m	19.4	32.8	38.0	33.5	28.9	22.9	44.4	48.6	30.6	35.2	32.9
X-attention	50m	11.4	30.8	32.9	31.6	26.8	19.7	41.9	43.8	27.0	34.0	31.1
BitFit	335k	11.3	23.6	29.5	25.9	22.0	14.9	35.2	38.8	21.1	28.1	26.0
Adapter (b=5)	320k	12.8	23.2	29.9	25.7	21.8	15.4	34.8	38.2	21.0	27.4	26.4
From Scratch	40m	5.0	20.9	27.3	26.3	19.2	11.6	34.3	39.4	19.1	21.9	23.8

Table 7: 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 8: (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. 5).