Recipes for Adapting Pre-trained Monolingual and Multilingual Models to Machine Translation

Asa Cooper Stickland[♣] **Xian Li**[♠] **Marjan Ghazvininejad**[♠]

[♣] University of Edinburgh, [♠] Facebook AI
a.cooper.stickland@ed.ac.uk, {xianl,ghazvini}@fb.com

Abstract

There has been recent success in pre-training on monolingual data and fine-tuning on Machine Translation (MT), but it remains unclear how to best leverage a pre-trained model for a given MT task. This paper investigates the benefits and drawbacks of freezing parameters, and adding new ones, when fine-tuning a pre-trained model on MT. We focus on 1) Fine-tuning a model trained only on English monolingual data, BART. 2) Fine-tuning a model trained on monolingual data from 25 languages, mBART. For BART we get the best performance by freezing most of the model parameters, and adding extra positional embeddings. For mBART we match or outperform the performance of naive fine-tuning for most language pairs with the encoder, and most of the decoder, frozen. The encoder-decoder attention parameters are most important to fine-tune. When constraining ourselves to an out-of-domain training set for Vietnamese to English we see the largest improvements over the fine-tuning baseline.

1 Introduction

Machine Translation (MT) has recently seen significant advances, with improvements in modeling, especially since the advent of neural models (Sutskever et al., 2014; Bahdanau et al., 2015), and the availability of large parallel corpora for training such systems (Smith et al., 2013; Kocmi and Bojar, 2017; Tiedemann, 2012). However, often standard neural systems do not perform well on *low-resource* language pairs (Koehn and Knowles, 2017), especially when the language pairs are only distantly related. Since these languages are spoken by a large fraction of the world's population, reducing the gap in performance between high and low-resource MT could have a large impact.

An explosion of interest in large-scale pre-training in Natural Language Processing has led to increased performance on smaller datasets, by simple *fine-tuning* of large pre-trained models on downstream tasks. The typical approach is to train a large model on text from the web (for example English Wikipedia), with a common objective predicting masked out tokens using the unmasked context. For Natural Language Generation (for example summarization of text), performance can be improved by pre-training a sequence-to-sequence model (Song et al., 2019; Lewis et al., 2019).

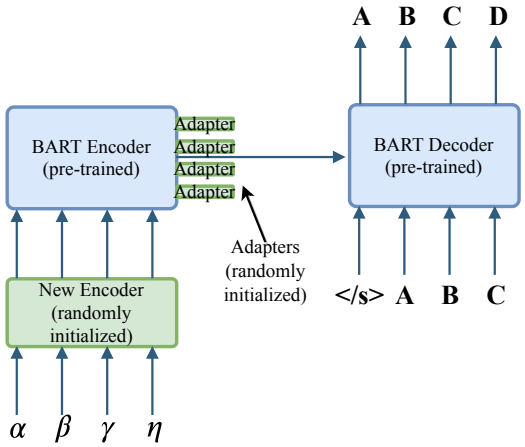

Figure 1: Schematic diagram showing the components of our system for adapting BART to MT. We learn a new encoder that takes as input the source language, with a potentially different vocabulary to the original BART system. We freeze most BART parameters (frozen model components are shown in blue).

However previous work has shown that on NLP tasks such as Natural Language Inference, the relative performance of fine-tuning vs. keeping the pre-trained model frozen depends on the similarity of the pre-training and downstream tasks (Peters et al., 2019). We observe empirically that simple fine-tuning of a monolingual model for MT can result in worse performance than training from scratch (e.g. Table 1). For MT the more common mono-

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pages 3440–3453
April 19 - 23, 2021.

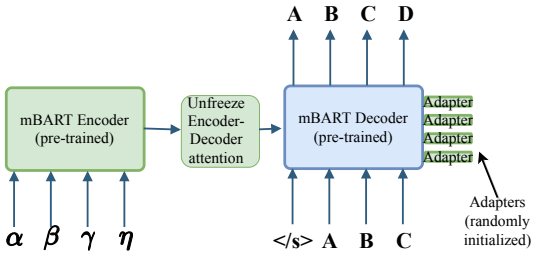

Figure 2: Schematic diagram showing one method of adapting mBART to MT, unfreezing the encoder and encoder-decoder attention, and adding adapters in the decoder. Model components colored blue are not updated during fine-tuning.

lingual (usually only English) pre-training (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019; Yang et al., 2019b; Liu et al., 2019) may be inadequate since the input or output domain for the downstream task will be a non-English language.

Multilingual pre-training offers a solution, by modifying the pre-training objective to include many languages. Using a multilingual pre-trained model for MT gives good performance, especially on lower-resource language directions (Liu et al., 2020). However it is challenging to balance the training data so that higher-resource languages do not overwhelm lower-resource ones (Arivazhagan et al., 2019; Conneau et al., 2019). For a particular language it may be hard to source monolingual data, or it may be simply not included in training.

We also consider multilingual MT (training on many language pairs and sharing all or most model parameters) as a downstream task. Sharing 'knowledge' across language directions can improve performance on low-resource language pairs by transfer from other pairs included in training. Previous work observed problems of performance degradation, often on high-resource languages, due to interference and constrained capacity (Johnson et al., 2017; Tan et al., 2019). And when initialising from a pre-trained model, we want to avoid 'catastrophic forgetting', where by fine-tuning on a particular language pair we lose the knowledge about another language pair that is stored in the model weights.

Previous work has explored how to improve on simple fine-tuning, by freezing pre-trained model parameters (Peters et al., 2019; Houlsby et al., 2019) and using lightweight 'adapter modules' (Houlsby et al., 2019; Stickland and Murray, 2019) which are inserted between the layers of the pre-trained network. We aim to explore and improve on these approaches for both bilingual and multilingual MT (in contrast to previous work largely focusing on text classification). We explore freezing different subsections of the pre-trained model.We expect freezing to be particularly useful when the parallel data is of low quality, in which case naive fine-tuning may, for example, over-specify the pre-trained model to a particular domain.

Our main contributions are:

- A novel fine-tuning approach, similiar to Lewis et al. (2019) but with adapter modules in the encoder of the pre-trained sequence-to-sequence model and combining both learnable, and fixed sinusoidal, positional embeddings in the input module (see sections 3.1 and 3.2) that feeds into the pre-trained encoder.

- Extensive experiments with fine-tuning a multilingual pre-trained model for MT, showing the benefits and drawbacks of freezing various parameters. We find we should freeze the decoder but unfreeze the encoder-decoder attention when fine-tuning on $Xx \rightarrow En$ data, and in the other direction we should freeze the encoder but unfreeze the entire decoder (section 5.3). We find monolingual models benefit more from freezing parameters than multilingual models (section 5.2).

- Results on fine-tuning a multilingual pre-trained model for multilingual MT showing that freezing parameters improves performance on some, mostly distantly related, language directions (section 5.5).

2 Background and Related Work

BART and mBART We briefly describe the pre-trained models we focus on in this work. In order to perform machine translation with the minimum of modifications to the pre-trained model, we prefer models that can perform conditional sequence generation. We concentrate on the BART (Bidirectional and Auto-Regressive Transformer) model (Lewis et al., 2019) and the multilingual BART (mBART; Liu et al., 2020) model. BART and mBART are sequence-to-sequence models with the standard transformer-based neural machine translation architecture, i.e. an encoder and autoregressive decoder. The pre-training task they are trained on is reconstructing a document from

a noisy version of that document (so called 'denoising autoencoder'). Examples of noise added to the training data include randomly shuffling the order of the original sentences, randomly changing the start position of the document, and using a masking scheme where arbitrary length spans of text are replaced with a single mask token. BART and mBART are trained entirely on monolingual data from the web, with English data for BART and data from 25 different languages for mBART.

BART and mBART have almost identical architectures, with 12 encoder layers and 12 decoder layers with model dimension of 1024 and 16 attention heads. BART has a vocabulary of approximately 40k and $\sim$ 406M parameters, whereas mBART has a larger vocabulary of size 250k and $\sim$ 610M parameters.

Pre-trained Models for MT There has been much recent progress in pre-training for NLP applications (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019; Yang et al., 2019b; Liu et al., 2019), with the most relevant for our work focusing on text generation (Radford et al., 2019; Song et al., 2019; Dong et al., 2019; Raffel et al., 2019; Lewis et al., 2019) Specifically for MT, Ramachandran et al. (2017) proposed pre-training the encoder-decoder modules as two separate language models, and Yang et al. (2019a); Zhu et al. (2020) explored approaches incorporating BERT model weights into the usual seq-to-seq architecture.

Multilingual MT *Multilingual translation* (Firat et al., 2016; Viégas et al., 2016; Aharoni et al., 2019; Arivazhagan et al., 2019) aims to jointly train one translation model that translates multiple language directions, and shares representations to improve the translation performance on low-resource languages (Gu et al., 2018). Our freezing approach is similar in spirit to Sachan and Neubig (2018) who investigate which parameters are most useful to share for multilingual MT with transformer models. We start from a multilingual pre-trained model, and decide between sharing or freezing parameters.

Transfer Learning for MT *Transfer learning* hopes to leverage a related task to perform well on a target task, for example by initialising the model weights from those resulting from training on a related task. For MT various approaches have been explored, with a common method training on high-resource language(s) and fine-tuning on a low-resource language (Neubig and Hu, 2018).

Closely related to our work is that of Bapna and Firat (2019), who introduce freezing and adapters (extra parameters inserted within the transformer) for domain adaption in MT. They take an MT model trained on a large parallel corpus, and fine-tune in a different domain (e.g. legal text). We differ in that we start from a pre-trained model that has not been trained on parallel text, and study adapting it to MT. Approaches based on freezing various model components have also been proposed (Thompson et al., 2018; Zoph et al., 2016), but have focused on RNN models pre-trained with parallel data, not transformer models pre-trained on monolingual data.

3 Methods

Because BART has been trained on only English input, we need to use different techniques when fine-tuning BART and mBART for MT, with a schematic overview shown in Figure 1 and Figure 2. BART and mBART are standard sequence-to-sequence models, where an *encoder* consumes a sequence of source-side tokens, and a *decoder* acts as a conditional language model, generating target tokens given a source sequence. Intuitively, we want the encoder and decoder to be performing roughly the same tasks during fine-tuning as they were during pre-training. For BART this means the input to the encoder should be similar to (embedding vectors of) noisy English text. Therefore when training on say, Vietnamese to English, we first transform the Vietnamese source sentence into a representation useful for BART. We introduce new parameters (the 'Input Module') that consume the source sentence and produce hidden vectors we can feed into the BART encoder. We describe the Input Module architecture in section 3.1.

mBART can be fine-tuned without modification since during pre-training it saw the languages it will be fine-tuned on. To increase flexibility when freezing parts of the network, we optionally add extra parameters to both BART and mBART, described in section 3.3.

3.1 Input Module Architecture

We refer to the network that takes in the source language text and outputs hidden vectors useful for BART as an 'Input Module' or $\mathrm{IM}(\cdot)$. To improve performance on low-resource MT, we use smaller token embedding vectors on the source side of size $d_{\mathrm{s}} = 512$, whereas BART uses hidden vectors of

size $d_{\text{BART}} = 1024$. The full network is as follows, with $\{\mathbf{e}_t\}_{t=0}^{l}$ token embeddings for a source sentence with l tokens,

$$\text{BART}(\text{IM}(\{\mathbf{e}_t\}_{t=0}^{l})), \tag{1}$$

where $\text{BART}(\cdot)$ is the full BART encoder-decoder model. Where we would normally input token embeddings to the BART model we use the outputs of the Input Module. The t-th element of $\text{IM}(\{\mathbf{e}_t\}_{t=0}^{l})$ as follows:

$$\alpha\text{LN}(\mathbf{W}\text{Transformer}(\{\mathbf{e}_t\}_{t=0}^{l})_t) \tag{2}$$

and where $\text{LN}(\cdot)$ is layer-norm, $\mathbf{W}$ is a matrix projecting up from d_{s} to d_{BART}, and $\text{Transformer}(\cdot)$ is the application of a series of Transformer layers. α is a scalar, in our case equal to $\sqrt{d_{\text{BART}}}$, which is required to insure the input to BART is on the same scale as the embedding vectors BART was trained on. If we remove $\text{LN}(\cdot)$, $\mathbf{W}$ and α, and set $d_{\text{s}} = d_{\text{BART}}$, we recover the method introduced by Lewis et al. (2019) for fine-tuning BART on MT.

3.2 Extra Positional Embeddings

We found empirically that the details of positional embedding vectors are important for good performance (see Table 1), perhaps because of the need for the BART model to deal with different word order to that it was trained on. Transformer models normally have either learnable positional embedding vectors, or fixed sinusoidal positional embedding (Vaswani et al., 2017) vectors $\mathbf{p}^t$, with $\mathbf{p}^t_i = \sin(t/10000^{i/(d_s/2-1)})$, if $0 \leq i < d_s/2$, and $\mathbf{p}^t_i = \cos(t/10000^{(i-(d_s/2-1))/(d_s/2-1)})$ if $d_s/2 \leq i < d_s$, where t indexes position and i indexes dimension.

Note that positional embedding are typically only added to the token embeddings. We use learnable positional embeddings at the embedding layer. But to get extra positional information, we optionally add fixed sinusoidal positional embedding to the input of each transformer layer in $\text{IM}(\cdot)$, i.e. the input to layer i, $\mathbf{h}^i_t = \mathbf{o}^{i-1}_t + \mathbf{p}^t$, with $\mathbf{o}^{i-1}_t$ the previous layer output. This means the network has access to both *learned positional embeddings* (only at the embedding layer), and *fixed sinusoidal* ones at the input to each layer.

3.3 Within-Network Adapter Architecture

When freezing parts of a pre-trained model (either BART or mBART in our case), we may want to add flexibility by modifying the pre-trained model architecture. One approach is to use 'adapters', introduced by Houlsby et al. (2019); Stickland and Murray (2019) which are newly-initialised neural network layers that can be 'slotted in' to the layers of the pre-trained model.

We only considered simple adapter architectures, essentially feed-forward networks, with one hidden layer, and a residual connection to the output. The dimension of the hidden layer can be much smaller than the model dimension to reduce computational cost and parameter count. We use one adapter per transformer layer, inserting them at the end of the layer (Stickland and Murray, 2019; Bapna and Firat, 2019). We use the following architectures, with $\mathbf{h}$ the hidden state of a particular token after the usual transformer layer, and $\mathbf{h}_{\text{out}}$ the hidden state of the token after the adapter layer:

$$\begin{aligned} \mathbf{z} &= \text{gelu}(\mathbf{W}_d\mathbf{h}) \\ \mathbf{h}_{\text{out}} &= \tanh(\mathbf{W}_u\mathbf{z}) + \mathbf{h} \end{aligned} \tag{3}$$

The $\tanh$ non-linearity helped with stability in early experiments, probably because it prevents the adapter output exploding by constraining it between -1 and 1.

We also considered a version of the adapter based on the 'gated linear unit' (GLU; Dauphin et al., 2016) architecture:

$$\begin{aligned} \mathbf{z} &= 2\sigma(\mathbf{W}_g\mathbf{h}) \odot \text{gelu}(\mathbf{W}_d\mathbf{h}) \\ \mathbf{h}_{\text{out}} &= \tanh(\mathbf{W}_u\mathbf{z}) + \mathbf{h}. \end{aligned} \tag{4}$$

We found the network was sensitive to changes in the magnitude of the hidden states the adapter produced, and therefore multiply the sigmoid gate by 2 so that it approximately leaves the magnitude of the hidden states unchanged.

3.4 Freezing Details

BART We freeze all parameters of BART except the weights and biases of the layer-norm modules (following Houlsby et al. (2019)), and additionally unfreeze the self-attention module of the first layer in the BART encoder, which is a small fraction of total BART parameters ($24 \cdot 2d_{\text{BART}}$ from layer-norm parameters and $4d^2_{\text{BART}}$ from the self-attention module). We freeze BART token embeddings (used in the softmax layer).

mBART In most of our experiments we unfreeze layer-norm parameters, positional and token embeddings, and either the entire encoder or decoder

Languages	Vi-En	Tuned Params (m)
(1): BART + InputModule (unfreeze all)	9.5	374
(2): BART (frozen) + InputModule	27.9	26
(3): (2) + unfreeze layer-norm	28.4	26
(3) + sinusoidal positional embeddings	18.3	26
(1) + extra positional embeddings	22.0	26
(4): (3) + extra positional embeddings	29.0	26
(5): (3) + encoder adapters	28.9	29
(3) + decoder adapters	28.3	29
(6): (5) + extra positional embeddings	30.0	29
(7): (6) + GLU adapters	30.5	29

Table 1: Ablation study for various choices in the frozen BART method, with validation set BLEU score. We organise model settings by a number in brackets, (n), and define a new model configuration in bold as **(n)**:. We use '+' to indicate the addition of new model settings on top of the previous ones. Method **(2)** is similar to the method introduced by Lewis et al. (2019). '+ sinusoidal positional embeddings' refers to adding sinusoidal positional embeddings to token embeddings, while '+ extra positional embeddings' refers to adding them within each transformer layer (see section 3.2). 'Tuned Params (m)' refers to the number of tunable parameters for each method in millions. Test set results are listed in Table 3 (as 'Frozen BART').

Languages	It-En	Si-En
(1): BART + InputModule + LN	34.1	5.1
(2): (1) + encoder adapters	35.0	7.3
(1) + decoder adapters	35.5	6.8
(3): (2) + extra pos. embeddings	36.3	8.7
(4): (3) + GLU adapters	35.7	9.2

Table 2: Further Ablation study for key settings of the frozen BART method, with validation set BLEU score. Test set results are listed in Table 3 (as 'Frozen BART').

module (or the encoder and subsections of the decoder). We unfreeze the self-attention module of the first layer in the mBART encoder and decoder.

4 Experimental Settings

We use the fairseq (Ott et al., 2019) library for all experiments. The final models are selected based on validation likelihood, except for multilingual fine-tuning where we evaluate the models after 10000 training steps. We use beam-search with beam size 5 for decoding, and evaluate all BLEU scores using SacreBLEU (Post, 2018) [1]. We use ISO 693-2 language codes in this work for convenience, and use the same parallel data as Liu et al. (2020), both listed in listed in Table 11 of the Appendix.

We fine-tune frozen BART and an Input Module on bilingual parallel text, feeding the source language into the Input Module. For mBART we feed the source language into the encoder, and use the same hyper-parameters as Liu et al. (2020). When using adapters we use 0.1 dropout in the adapter bottleneck layer ($\mathbf{z}$ in section 3.3), and a hidden dimension of either 128, or $\lfloor 2/3 \cdot 128 \rceil$ when using a gated linear unit adapter. We use the Adam (Kingma and Ba, 2015) optimizer. Hyper-parameters are listed in Appendix B, and we use the same hyper-parameter search space for frozen and non-frozen models.

4.1 Multilingual MT

We train with a very large effective batch size, training on 32 GPUs with a per-GPU batch size of 4096 tokens, meaning our total batch size is $N \cdot 32 \cdot 4096$ tokens, where N is the number of language pairs. We evaluate our model after 10000 training steps (amounting to $N \cdot 10000$ forwards-backwards passes through the model).

4.2 Vocabulary

BART uses the GPT-2 tokenizer, which uses the BPE (Sennrich et al., 2016) approach (on the level of bytes, not characters). BART could technically take any Unicode string as input, however the BPE is learned on English text. When fine-tuning BART on machine translation we therefore learn a

[1] SacreBLEU signature: `BLEU+case.lc+lang.[src-lang]-[tgt-lang]+numrefs.1+smooth.exp+tok.13a+version.1.3.6`

Languages **Size**	**Vi-En**† 110k	**Vi-En** 133k	**It-En** 250k	**My-En** 259k	**Ne-En** 564k	**Si-En** 647k	**Cs-En** 11M	**Es-En** 15M	**Pars (m)**
(1): Freeze decoder	12.1	30.0	36.5	27.4	11.0	13.6	26.6	34.1	407
Freeze encoder	12.0	29.7	36.6	25.2	8.8	12.3	25.6	33.8	457
(2): (1) + adapters	12.2	30.0	36.7	27.7	10.8	14.2	27.4	34.4	410
(2) + ft enc-attn	12.3	30.6	37.0	29.0	11.4	14.9	27.0	35.1	461
(2) + ft self-attn	11.7	30.4	36.1	28.3	10.6	14.3	27.4	34.7	461
(2) + ft last 3 lyrs	12.1	30.6	36.6	28.1	11.5	14.7	27.6	34.9	461
Test (random init)	8.1	23.6	31.7	23.3	7.6	7.2	22.0	29.0	N/A
Test (frozen BART)	-	35.2	38.5	21.0	0.5	7.8	-	-	29
Test (ft all)	14.1	**36.7**	**39.8**	**27.6**	14.1	**14.0**	29.2	**34.5**	610
Test (ft enc-attn)	**14.9**	**36.4**	39.4	**27.9**	**14.6**	**14.1**	**29.8**	**34.4**	461

Table 3: Validation BLEU score (unless stated otherwise) obtained by freezing various parts of the mBART and of adding adapters for *Xx* → *En*. 'ft' refers to fine-tuning, i.e. unfreezing. Vi-En† refers to a new parallel, 'out-of-domain' dataset constructed similarly to the Flores (Guzmán et al., 2019) train sets (see section 5.2). 'Test (frozen BART)' indicates results from English-only BART with the best performing method from Table 2 or Table 1. 'Test (random init)' refers to training models (of various sizes) from scratch on the bitext for that language pair. 'Pars (m)' refers to the number of tunable parameters for each method in millions (note token embeddings are tuned in every method and account for 256m parameters). Bold indicates the best test set score and all scores whose difference from the best is not statistically significant (with p-value less than 0.05). (Statistical significance is computed via bootstrapping (Koehn, 2004).)

	Vi-En	**It-En**	**My-En**	**Ne-En**	**Si-En**
Freeze decoder (don't ft layer-norm)	26.6	35.1	26.6	10.3	13.1
Freeze encoder (don't ft layer-norm)	29.4	36.1	24.1	8.7	12.1

Table 4: Ablation study on improvement from fine-tuning layer-norm. Compare to the 'Freeze decoder' and 'Freeze encoder' methods in the first two rows of Table 3.

new subword vocabulary (using the sentencepiece (Kudo and Richardson, 2018) library) on the source data from the fine-tuning dataset, and use a smaller vocabulary size of 5000, which empirically performs better for low-resource MT (Guzmán et al., 2019; Sennrich and Zhang, 2019). We don't change the mBART tokenizer or vocabulary.

5 Results and Discussion

5.1 Frozen BART

Table 1 shows the effects of various choices we made in fine-tuning BART for MT. *Freezing* is important: we see an 18.4 BLEU point improvement from fine-tuning a frozen BART model compared to fine-tuning an unfrozen BART (both with an Input Module; see section 3.1).

Adding extra flexibility with within-network adapters helps performance, especially when added to the BART *encoder*. It is important to use *learned positional embeddings* at the embedding layer in the Input Module, with an 10.1 BLEU score drop if we use fixed positional embeddings (at the embedding layer). We see consistent gains in Table 1 and Table 2 by adding additional, fixed sinusoidal positional embeddings to the input of every transformer layer of the Input Module (see section 3.2), even when using an unfrozen BART. The BART encoder 'expects' English input, and it may be the Input Module with extra fixed embeddings can better account for the different word order in the input language. In the next section we compare to mBART and baselines.

5.2 Frozen mBART

In Table 3 and Table 5 we list results from freezing various parts of mBART. We get better performance than fine-tuning ('ft all' in Table 3) with our freeze decoder + fine-tune encoder-decoder attention method ('ft enc-attn' in Table 3) on Ne-En and Cs-En for *Xx* → *En*, and mostly similar results to

Languages	En-Vi	En-It	En-My	En-Ne	En-Si	En-Cs	En-Es	Pars (m)
Freeze decoder	29.7	32.2	35.0	5.8	2.1	17.7	35.4	407
(1): Freeze encoder	30.1	31.5	36.0	5.3	3.7	16.5	35.0	457
(2): (1) + encoder adapters	30.3	32.3	36.9	5.4	4.2	16.6	35.3	461
Test (ft all)	**35.4**	34.0	**36.9**	**7.4**	**3.3**	**18.0**	**34.0**	610
Test (freeze enc. + adapters)	35.0	**34.3**	35.9	6.9	**3.3**	16.7	32.5	461

Table 5: Validation BLEU score (unless stated otherwise) obtained by freezing various parts of the mBART and of adding adapters for for *En* → *Xx*. 'Pars (m)' refers to the number of tunable parameters for each method in millions.

the baseline otherwise.

We believe a benefit to freezing, when fine-tuning on training data from a different domain to test data, will be avoiding specialising the pre-trained model to the fine-tuning train data domain. To test this we constructed a new Vi-En parallel dataset (Vi-En† in Table 3) using the some of the same sources as the Flores (Guzmán et al., 2019) training data (the Si-En and Ne-En training sets used in this work), specifically GNOME/KDE/Ubuntu domain from the OPUS repository[2] and Bible translations from the bible-corpus[3], and use the same test and validation sets as the IWSLT15 Vi-En dataset. By constraining ourselves to this out-of-domain training set we see the largest gains out of the language pairs we considered over the fine-tuning baseline (0.9 BLEU).

We also consider the effect of the size of the fine-tuning dataset. If we constrain the training data to a random subset of 200k training examples from Ro-En (Table 6), the 'ft enc-attn' method outperforms simple fine-tuning. This effect generalises to an mBART variant that was pre-trained on only Ro and En monolingual data (using the same data as Liu et al. (2020)). Further results on Ro-En data are available in the Appendix, Table 10, and show similar trends to Table 3, with fine-tuning encoder-decoder attention the most important.

Table 3 shows the relative performance of frozen BART, frozen mBART and baselines. Fine-tuning mBART gave consistently better results than frozen BART especially for distantly related languages. For Si, Ne and My the performance of frozen BART is roughly on par with a randomly initialised model (or much worse in the case of Ne-En). The parallel data for these languages is often lower quality, and the BART system has to learn about the non-English language from noisy or out-of-domain text (e.g. text from the Ubuntu manual for the En-Ne pair). For Vi and It, we have high quality parallel data, and the frozen BART method is only approximately 1.5 BLEU points behind the best mBART results. We note mBART was trained on more English data than BART, and with different noising function hyper-parameters.

5.3 What Should be Unfrozen?

Layer-Norm We find large benefits to simply fine-tuning the weights and biases of the pre-trained layer-norm weights (recall that after normalisation, the layer-norm module multiplies each hidden dimension by a weight and adds a bias); this was observed in the setting of BERT by Houlsby et al. (2019). This gains e.g. 0.5 BLEU for frozen BART (see Table 1) and an average of 0.8 BLEU across five languages for mBART (see Table 4 compared to Table 3). Since these weights and biases are only $2d$ parameters per layer-norm, where d is the model dimension. This is parameter-efficient, with adding more parameters with 'Adapters' on top of unfrozen layer-norm providing a smaller improvement.

Encoder vs Decoder For the *Xx* → *En* direction (Table 3) we can see that freezing the decoder always performs better than freezing the encoder (except for It-En where they perform roughly the same.) For the *En* → *Xx* direction (Table 5) we see slightly weaker evidence for the opposite trend, with the decoder more useful to fine-tune; but for the high resource languages Es and Cs freezing the decoder works better. There is more English data in mBART pre-training than data in other languages, which may account for better results with a frozen encoder (when English is the source language) or decoder (when English is the target language). Adding flexibility with adapters in the frozen layers

[2] http://opus.nlpl.eu/

[3] https://github.com/christos-c/bible-corpus/

Model	mBART		En-Ro mBART	
Languages (Size)	**Ro-En (608k)**	**Ro-En (200k)**	**Ro-En (608k)**	**Ro-En (200k)**
Test (ft all)	37.8	36.4	38.5	37.7
Test (ft enc-attn)	37.8	36.8	38.1	37.9

Table 6: Validation set BLEU (unless stated otherwise) comparing freezing various parts of mBART and En-Ro mBART (pre-trained only on En and Ro data), fine-tuned on *Ro* $\rightarrow$ *En* parallel data. 'ft' referes to fine-tuning, i.e. unfreezing. 'Ro-En (200k)' refers to a random subset of the Ro-En training data of size 200k.

Src. Lang.	**Ru**	**Fr**	**De**	**Zh**	**Es**	**Cs**	**Lv**	**Fi**	**Lt**	**Et**	**Hi**	**Si**
Size	32M	29M	28M	25M	15M	11M	4.5M	2.7M	2.1M	1.9M	788k	647k
Finetune all	**33.6**	**39.0**	**33.1**	**20.2**	33.7	**29.9**	**21.1**	**29.0**	**22.8**	**28.6**	**25.4**	**16.9**
Ft enc-attn	**33.4**	38.2	32.6	**20.2**	**34.0**	29.7	**20.8**	**29.1**	**22.7**	**28.3**	25.1	**16.7**
Src. Lang.	**Ro**	**Ne**	**My**	**Ar**	**It**	**Nl**	**Ko**	**Ja**	**Tr**	**Vi**	**Kk**	**Gu**
Size	612k	563k	259k	251k	251k	237k	230k	223k	207k	133k	91k	12k
Finetune all	**37.8**	**20.7**	**31.0**	**37.0**	**39.6**	**43.3**	**25.0**	**18.7**	**24.0**	**37.4**	14.6	**18.7**
Ft enc-attn	**37.9**	**20.8**	30.5	**36.9**	**39.3**	**43.0**	24.2	**18.8**	23.7	**37.5**	**15.0**	**18.3**

Table 7: Test set BLEU score on many-to-one (*Xx* $\rightarrow$ *En*) multilingual MT with a simple round-robin training schedule. 'Ft enc-attn' refers to fine-tuning the encoder, and fine-tuning the encoder-decoder attention module in every decoder layer, leaving the other decoder sub-modules frozen. The 'Ft enc-attn' model setting uses adapter modules in the decoder to increase flexibility after freezing parameters. Bold indicates the best score and all scores whose difference from the best is not statistically significant (with p-value less than 0.05). For clarity we underline language pairs where the 'Ft enc-attn' method matches or outperforms naive fine-tuning.

improves performance in all languages and directions, except for Ne$\rightarrow$En.

We explore more fine-grained unfreezing for the *Xx* $\rightarrow$ *En* direction (Table 3). We fine-tuned three equally sized subsets of the decoder: the encoder-decoder attention layers (approx. $12 \cdot 4d^2_{\text{BART}}$ parameters), the self-attention layers in the decoder (approx. $12 \cdot 4d^2_{\text{BART}}$ parameters), or the entire last three layers of the decoder (approx. $3 \cdot 16d^2_{\text{BART}}$ parameters). We observe that fine-tuning the encoder-decoder attention performed well (note the last three layers include three encoder-decoder attention layers), with fine-tuning self-attention the least useful. We hypothesize that the pre-training task of mBART (reconstructing noisy monolingual sentences) does not help with teaching the encoder-decoder attention to align source and target text of different languages.

5.4 Memory Cost

Freezing parameters means we no longer need to allocate memory to storing their gradients. We will obtain additional memory savings when using an optimizer that stores various other quantities (i.e. the Adam optimizer stores running averages of the first and second moments of gradients.). The memory savings allow for roughly 45-75% larger batches for the methods we consider in this work (see Table 8 for our mBART methods), but for larger pre-trained models the proportion of GPU memory freed up by freezing will increase. At inference time we no longer require gradients and we have the same memory cost.

	Tokens per GPU
Finetune all	2304
(1): Freeze decoder	4096
Freeze encoder	3584
(2): (1) + decoder adapters	4096
(2) + ft enc-attn	3328

Table 8: Maximum number of tokens that would fit on one NVIDIA Volta GPU when fine-tuning mBART on the En-Vi training set. We evaluated batch sizes in increments of 256 tokens.

5.5 Multilingual Fine-tuning of mBART

We explore freezing parts of the mBART model when fine-tuning on a challenging multilingual MT

task. Table 7 lists results from a naive fine-tuning baseline, and results from freezing most of the decoder but unfreezing the encoder-decoder attention (when freezing we use GLU adapters in the decoder, see section 3.3). Freezing parameters hurts performance on some language pairs, and since freezing removes flexibility from the model and we have to adapt to 25 different directions this is perhaps not surprising. The language pairs where we match or improve on the baseline are Zh, Es, Fi, Ne, Ja, Vi and Kk. These are mostly (five out of seven) non-European languages, and distantly related to En. However since most of these results are not statistically significant further study is needed to verify this. Note we see a clear benefit over bilingual fine-tuning for some language pairs (e.g. compare our best Ne result from Table 3, 14.6 BLEU vs. 20.8 BLEU for multilingual fine-tuning). We leave to future work a more thorough investigation of the multilingual MT setting.

6 Conclusion

We recommend: For a language with high quality parallel data but without a pre-trained model trained on monolingual data from that language, using a frozen (English-only) BART model with additional parameters at the source side (the 'input module') improves performance over a randomly initialised baseline. For this approach it is important to freeze the pre-trained model. We also give the model both learned positional embeddings at the embedding layer, and fixed sinusoidal positional embeddings at each layer of the input module.

For a multilingual pre-trained model, we found performance improvements on some (mostly distantly related) languages for multilingual many-to-one fine-tuning. For bilingual $En \rightarrow Xx$ fine-tuning we did not see any improvement, although the performance drops are small, and by freezing parameters we need less memory at training time compared to fine-tuning. For $Xx \rightarrow En$ bilingual fine-tuning it is important to unfreeze the encoder-decoder attention, and keep the rest of the decoder frozen. This can improve on simple fine-tuning, especially for distantly-related language pairs or those with out-of-domain training data.

We recommend fine-tuning layer-norm parameters as a parameter-efficient complement to adapter layers. For our mBART experiments we found it was necessary to fine-tune the token embeddings, which correspond to a large number of parameters, and future work could remove this cost by working out a subset of the vocabulary to fine-tune, or another method.

Acknowledgments

We'd like to thank James Cross, Mike Lewis, Naman Goyal, Jiatao Gu, Iain Murray, Yuqing Tang and Luke Zettlemoyer for useful discussion. We also thank our colleagues at FAIR and FAIAR for valuable feedback.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *CoRR*, abs/1612.08083.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of

deep bidirectional transformers for language understanding. In *North American Association for Computational Linguistics (NAACL)*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *NAACL*.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. 2018. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354, New Orleans, Louisiana. Association for Computational Linguistics.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The FLORES evaluation datasets for low-resource machine translation: Nepali–English and Sinhala–English. 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 6097–6110, Hong Kong, China. Association for Computational Linguistics.

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 *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799, Long Beach, California, USA. PMLR.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Tom Kocmi and Ondřej Bojar. 2017. Curriculum learning and minibatch bucketing in neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 379–386, Varna, Bulgaria. INCOMA Ltd.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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, Brussels, Belgium. Association for Computational Linguistics.

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 *North American Association for Computational Linguistics (NAACL): System Demonstrations*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *North American Association for Computational Linguistics (NAACL)*.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. *CoRR*, abs/1903.05987.

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.

Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Prajit Ramachandran, Peter J Liu, and Quoc Le. 2017. Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 383–391.

Devendra Sachan and Graham Neubig. 2018. Parameter sharing methods for multilingual self-attentional translation models. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 261–271, Belgium, Brussels. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Association for Computational Linguistics (ACL)*, pages 1715–1725.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221, Florence, Italy. Association for Computational Linguistics.

Jason R. Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. 2013. Dirt cheap web-scale parallel text from the common crawl. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1374–1383, Sofia, Bulgaria. Association for Computational Linguistics.

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 (ICML)*.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995, Long Beach, California, USA. PMLR.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Xu Tan, Jiale Chen, Di He, Yingce Xia, Tao Qin, and Tie-Yan Liu. 2019. Multilingual neural machine translation with language clustering. 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 963–973, Hong Kong, China. Association for Computational Linguistics.

Brian Thompson, Huda Khayrallah, Antonios Anastasopoulos, Arya D. McCarthy, Kevin Duh, Rebecca Marvin, Paul McNamee, Jeremy Gwinnup, Tim Anderson, and Philipp Koehn. 2018. Freezing subnetworks to analyze domain adaptation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 124–132, Belgium, Brussels. Association for Computational Linguistics.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 2214–2218, Istanbul, Turkey. European Languages Resources Association (ELRA).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Fernanda Viégas, Greg Corrado, Jeffrey Dean, Macduff Hughes, Martin Wattenberg, Maxim Krikun, Melvin Johnson, Mike Schuster, Nikhil Thorat, Quoc V Le, et al. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation.

Jiacheng Yang, Mingxuan Wang, Hao Zhou, Chengqi Zhao, Yong Yu, Weinan Zhang, and Lei Li. 2019a. Towards making the most of bert in neural machine translation. *arXiv preprint arXiv:1908.05672*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019b. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2020. Incorporating bert into neural machine translation. *arXiv preprint arXiv:2002.06823*.

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 Additional Ablation Study

In Table 9 we reproduce Table 4 of the main paper with more context to study the effect of unfreezing layer-norm parameters when fine-tuning mBART. Across all language pairs we see improvements from fine-tuning layer norm parameters over not fine-tuning them, and additional, smaller, improvements from adding adapters, indicating both forms of adding flexibility are useful. In Table 10 we present additional results on the Ro-En pre-trained model (see section 3.2 of the main body).

B Fine-tuning Hyper-parameters

For all experiments with bilingual datasets we use a batch size of 2048×16 tokens, i.e. 2048 tokens per GPU and 16 GPUs (we investigate larger batch sizes for frozen models only to test GPU memory usage, and do not evaluate models trained with larger batch sizes). Ranking of hyper-parameters was done by validation set BLEU score.

Frozen BART We train with 0.3 dropout for the frozen BART parameters, and 0.2 dropout for the Input Module parameters, 0.1 label smoothing, 0.2 dropout for the self-attention scores in the Input Module, 5000 warm-up steps, and $7\mathrm{e}{-4}$ maximum learning rate. We performed a grid search over learning rates in $\{7\mathrm{e}{-4}, 5\mathrm{e}{-4}, 3\mathrm{e}{-4}\}$, dropout for Input Module parameters in $\{0.2, 0.1\}$, and dropout for self-attention scores in $\{0.2, 0.1\}$. We train for a maximum of 50K training updates for all low and medium resource pairs and 100K for high resource pairs (which takes roughly 8 hours and 16 hours respectively).

Frozen mBART We train with 0.3 dropout, 0.2 label smoothing, 2500 warm-up steps, and $3\mathrm{e}{-5}$ maximum learning rate. We did not search over hyper-parameters, simply re-using those of Liu et al. (2020). Despite the adapter parameters being randomly initialised, the small learning rate did not affect performance (we performed a small sweep of larger learning rates and found only marginal gains, and so kept the same settings for simplicity). We use a maximum of 40K training updates for all low and medium resource pairs and 100K for high resource pairs (Es and Cs in our case), this takes roughly 12 hours and 30 hours respectively.

Multi-lingual MT We train with 0.3 dropout, 0.1 dropout for self-attention scores, 4000 warm-up steps, and $1\mathrm{e}{-4}$ maximum learning rate.

Out-of-domain Vi-En Baseline To train a randomly initialised baseline for the out-of-domain Vi-En data (Vi-En† in Table 3 of the main body) we used the same model architecture and training settings as those of Guzmán et al. (2019) use for training MT systems on similar data (but with Si or Ne source language). Specifically a seq2seq transformer with 5 encoder and decoder layers, hidden dimension 512. shared embeddings between the input and softmax layers, and strong regularisation (e.g. 0.4 dropout on hidden states, 0.2 dropout on attention scores, 0.2 label smoothing). We learn a BPE vocabulary (joint across source and target data) of size 5000 on the training data. For full details of hyper-parameters we refer the reader to Guzmán et al. (2019) and the associated GitHub repository[4].

C Pre-training Languages

We reproduce in Table 11 the details from Liu et al. (2020) of the size of each pre-training language corpus for mBART.

[4]https://github.com/facebookresearch/flores

	Vi-En	It-En	My-En	Ne-En	Si-En
Freeze decoder	26.6	35.1	26.6	10.3	13.1
Freeze encoder	29.4	36.1	24.1	8.7	12.1
(1): Freeze decoder + ft layer norm	30.0	36.5	27.4	11.0	13.6
Freeze encoder + ft layer norm	29.7	36.6	25.2	8.8	12.3
(1) + decoder adapters	30.0	36.7	27.2	10.8	14.2

Table 9: Validation BLEU score (unless stated otherwise) obtained by fine-tuning layer-norm parameters and of adding adapters for mBART, for *Xx* $\rightarrow$ *En*. 'ft' refers to fine-tuning, i.e. unfreezing. Note we are simply reproducing rows from Table 3 and Table 4 of the main paper for ease of comparison.

	mBART		En-Ro mBART	
	Ro-En (608k)	Ro-En (200k)	Ro-En (608k)	Ro-En (200k)
(1): Freeze decoder	38.8	37.9	40.4	39.9
Freeze encoder	39.1	38.3	40.0	39.2
(2): (1) + decoder adapters	39.3	38.0	40.6	40.0
(1) + ft enc-attn	39.8	39.0	40.5	40.5
(1) + ft self-attn	39.6	38.3	40.4	40.1
(1) + ft last 3 lyrs	39.6	38.6	40.5	40.3
Test (ft enc-dec)	37.8	36.8	38.1	37.9
Test (ft all)	37.8	36.4	38.5	37.7

Table 10: Validation set BLEU (unless stated otherwise) comparing freezing various parts of mBART and En-Ro mBART (pre-trained only on En and Ro data rather than 25 languages), fine-tuned on *Ro* $\rightarrow$ *En* parallel data. 'ft' refers to fine-tuning, i.e. unfreezing. 'Ro-En (200k)' refers to a random subset of the Ro-En training data of size 200k.

Code	Language	Tokens(M)	Size(GB)	Parallel data source
En	English	55608	300.8	
Ru	Russian	23408	278.0	WMT19
Vi	Vietnamese	24757	137.3	IWSLT15
Ja	Japanese	530 (*)	69.3	IWSLT17
De	German	10297	66.6	WMT19
Ro	Romanian	10354	61.4	WMT16
Fr	French	9780	56.8	WMT19
Fi	Finnish	6730	54.3	WMT17
Ko	Korean	5644	54.2	IWSLT17
Es	Spanish	9374	53.3	WMT19
Zh	Chinese (Sim)	259 (*)	46.9	WMT19
It	Italian	4983	30.2	IWSLT17
Nl	Dutch	5025	29.3	IWSLT17
Ar	Arabic	2869	28.0	IWSLT17
Tr	Turkish	2736	20.9	IWSLT17
Hi	Hindi	1715	20.2	ITTB
Cs	Czech	2498	16.3	WMT19
Lt	Lithuanian	1835	13.7	WMT19
Lv	Latvian	1198	8.8	WMT17
Kk	Kazakh	476	6.4	WMT19
Et	Estonian	843	6.1	WMT18
Ne	Nepali	237	3.8	FLoRes
Si	Sinhala	243	3.6	FLoRes
Gu	Gujarati	140	1.9	WMT19
My	Burmese	56	1.6	WAT19

Table 11: **Languages and Statistics of the CC25 Corpus.** A list of the 25 languages used in mBART pretraining ranked with monolingual corpus size. (*) The Chinese and Japanese corpora are not segmented, so the token counts here are sentence counts.