Lightweight Adapter Tuning for Multilingual Speech Translation

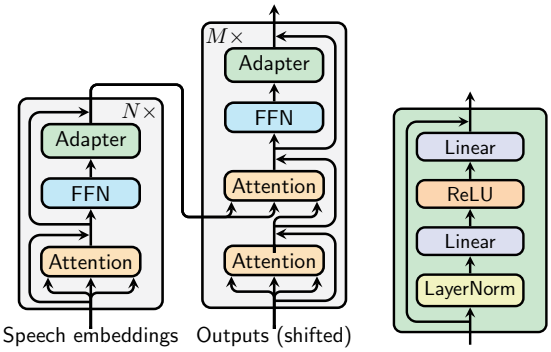
Hang Le[1] **Juan Pino**[2] **Changhan Wang**[2]
Jiatao Gu[2] **Didier Schwab**[1] **Laurent Besacier**[1,3]
[1]Univ. Grenoble Alpes, CNRS, LIG [2]Facebook AI [3]Naver Labs Europe
{hang.le, didier.schwab, laurent.besacier}@univ-grenoble-alpes.fr
{juancarabina, changhan, jgu}@fb.com

Abstract

Adapter modules were recently introduced as an efficient alternative to fine-tuning in NLP. Adapter tuning consists in freezing pre-trained parameters of a model and injecting lightweight modules between layers, resulting in the addition of only a small number of task-specific trainable parameters. While adapter tuning was investigated for multilingual neural machine translation, this paper proposes a comprehensive analysis of adapters for multilingual speech translation (ST). Starting from different pre-trained models (a multilingual ST trained on parallel data or a multilingual BART (mBART) trained on non-parallel multilingual data), we show that adapters can be used to: (a) efficiently specialize ST to specific language pairs with a low extra cost in terms of parameters, and (b) transfer from an automatic speech recognition (ASR) task and an mBART pre-trained model to a multilingual ST task. Experiments show that adapter tuning offer competitive results to full fine-tuning, while being much more parameter-efficient.

1 Introduction

The question of *versatility* versus *specialization* is often raised in the design of any multilingual translation system: is it possible to have a single model that can translate from any source language to any target one, or does it have to be multiple models each of which is in charge of one language pair? The former is referred to as a *multilingual* model, while the latter are *bilingual* ones. These two paradigms have their own strengths and limitations. From a practical point of view, a multilingual model seems to be highly desirable due to its simplicity in *training* and *deployment*, in terms of both time and space complexities. However, in terms of *accuracy*, a multilingual model could be outperformed by its bilingual counterparts, especially on high-resource language pairs.



(a) Transformer with adapters. (b) An adapter cell.

Figure 1: (a) Transformer with adapters at its FFN sub-layers. For simplicity, layer normalization (Ba et al., 2016) is omitted. During fine-tuning, only the adapters are trained. (b) A typical adapter architecture.

In practice, a certain trade-off between the aforementioned factors (and thus more generally between versatility and specialization) has often to be made, and depending on the application, one can be favored more than the other. One way to move along the spectrum between multilingual and bilingual models is to use adapter tuning which consists in freezing pre-trained parameters of a multilingual model and injecting lightweight modules between layers resulting in the addition of a small number of language-specific trainable parameters. While adapter tuning was investigated for multilingual neural machine translation (NMT) (Bapna and Firat, 2019), to our knowledge, this paper proposes the first comprehensive analysis of adapters for multilingual speech translation.

Our contributions are the following: (1) we show that both versatility and specialization can be achieved by tuning language-specific adapter modules on top of a multilingual system. Bilingual models with higher accuracy than the original multilingual model are obtained, yet keeping a low maintenance complexity; (2) starting from a different initialization point, we show that adapters can also be

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Short Papers), pages 817–824
August 1–6, 2021.

used as a glue to connect off-the-shelf systems (an automatic speech recognition (ASR) model and a multilingual denoising auto-encoder mBART (Liu et al., 2020; Tang et al., 2020)) to perform the multilingual ST task. Extensive experiments on the MuST-C dataset (Di Gangi et al., 2019) show that adapter-based fine-tuning can achieve very competitive results to full fine-tuning—while being much more parameter-efficient—in both standard and low-resource settings. Our code based on FAIRSEQ S2T (Wang et al., 2020) is publicly available.[1]

2 Related Work

Adapter layers (or *adapters* for short) were first proposed in computer vision (Rebuffi et al., 2017), then explored for text classification tasks in NLP (Houlsby et al., 2019). Adapters are generally inserted between the layers of a pre-trained network and finetuned on the adaptation corpus. Bapna and Firat (2019) studied adapters in the context of NMT and evaluated them on two tasks: domain adaptation and massively multilingual NMT. Philip et al. (2020) later introduced monolingual adapters for zero-shot NMT. Other research groups made contributions on the use of adapters in NLP (Pfeiffer et al., 2020b, 2021) and a framework built on top of HuggingFace Transformers library (Wolf et al., 2020) was also released to facilitate the downloading, sharing, and adapting state-of-the-art pre-trained models with adapter modules (Pfeiffer et al., 2020a). Also very relevant to our paper is the work of Stickland et al. (2021) where adapters are used to adapt pre-trained BART (Lewis et al., 2020) and mBART25 (multilingual BART pre-trained on 25 languages) (Liu et al., 2020) to machine translation.

As far as speech processing is concerned, adapters were mostly used in ASR (Kannan et al., 2019; Lee et al., 2020; Winata et al., 2020; Zhu et al., 2020). Recently, they have also been explored for ST as well but in a limited scope. Escolano et al. (2020) addressed a very specific setting (zero-shot ST), while Li et al. (2020) used only a single adapter after a Transformer encoder.

3 Adapters for Speech Translation

In this section, we describe the integration of adapters into a given backbone model for speech translation. As the Transformer (Vaswani et al., 2017) has become increasingly common in speech processing,[2] it will be used as our backbone. Our method, however, can be easily applied to any other architectures, e.g., dual-decoder Transformer (Le et al., 2020).

Adapter modules can be introduced into a Transformer in a *serial* or *parallel* fashion. Consider a layer represented by a function f that produces an output $\mathbf{y}$ from an input $\mathbf{x}$, i.e., $\mathbf{y} = f(\mathbf{x})$. This can be an entire encoder or decoder layer, or just one of their sub-layers (e.g., the self-attention or the final feed-forward network (FFN) component). Suppose that our adapter layer is represented by a function g. The new "adapted" output is then given by:

$$\mathbf{y}_{\text{serial}} = g(f(\mathbf{x})), \quad \mathbf{y}_{\text{parallel}} = f(\mathbf{x}) + g(\mathbf{x}).$$

Intuitively, a serial adapter modifies the output directly, while a parallel one performs the operations in parallel before merging its output to the layer. In Figure 1a, we show an example of serial adapters being integrated to the Transformer, or more precisely to its FFN sub-layers. A common adapter module (Bapna and Firat, 2019) is presented in Figure 1b. Here g is a small FFN with a residual connection. The first linear layer is typically a down projection to a bottleneck dimension, and the second one projects the output back to the initial dimension. Bottleneck allows us to limit the number of parameters. Other adapter architectures also exist, e.g., Stickland and Murray (2019) explored parallel adapters consisting of a multi-head attention (MHA) layer in a multi-task setup.

For multilingual ST, we adopt the following general recipe for adapter-based fine-tuning. Starting from a pre-trained backbone, an adapter is added for each language pair and then finetuned on the corresponding bilingual data (while the rest of the backbone is frozen). The pre-trained backbone plays a crucial role in this recipe. We explore two common scenarios to obtain this pre-trained model, namely *refinement* and *transfer learning*. We present them in details, together with extensive experimental results, in Section 5 and 6. In the next section, we present our experimental setup.

4 Experimental Setup

4.1 Dataset

MuST-C We evaluate our recipes on MuST-C (Di Gangi et al., 2019), a large-scale one-to-many

[1] https://github.com/formiel/fairseq/tree/master/examples/speech_to_text/docs/adapters.md

[2] For speech applications (Inaguma et al., 2020; Wang et al., 2020), the embedding layer of the encoder is often a small convolutional neural network (Fukushima and Miyake, 1982; LeCun et al., 1989).

	Dict	D	Adapter d	Adapter ENC	Adapter DEC	Finetune ENC	Finetune DEC	# params (M) trainable/total	de	es	fr	it	nl	pt	ro	ru	avg
								Training data (hours)	408	504	492	465	442	385	432	489	
1	mono	256	-	-	-	-	-	8×31.1/8×31.1	22.16	30.42	27.92	22.92	24.10	27.19	21.51	14.36	23.82
2	multi		-	-	-	-	-	32.1/32.1	22.37	30.40	27.49	22.79	24.42	27.32	20.78	14.54	23.76
3	multi		64	-	✓	-	-	8×0.2/33.7	22.32	30.50	27.55	22.91	24.51	27.36	21.09	14.74	23.87
4	multi		64	✓	✓	-	-	8×0.6/36.9	22.75	31.07	28.03	23.04	24.75	28.06	21.20	14.75	24.21
5	multi		128	-	✓	-	-	8×0.4/35.3	22.45	30.85	27.71	23.06	24.57	27.52	20.93	14.57	23.96
6	multi		128	✓	✓	-	-	8×1.2/41.7	22.84*	31.25*	28.29*	23.27*	24.98*	28.16*	21.36*	14.71	24.36
7	multi		-	-	-	-	✓	8×14.6/8×32.1	23.49	31.29	28.40	23.63	25.51	28.71	21.73	15.22	24.75
8	multi		-	-	-	✓	✓	8×32.1/8×32.1	23.13*	31.39*	28.67*	23.80*	25.52*	29.03*	22.25*	15.44*	24.90
9	mono	512	-	-	-	-	-	8×74.3/8×74.3	21.93	30.46	27.90	22.64	23.98	25.98	20.50	14.01	23.42
10	multi		-	-	-	-	-	76.3/76.3	23.98	32.47	29.24	24.97	26.20	29.81	22.74	15.30	25.59
11	multi		64	-	✓	-	-	8×0.4/79.5	24.24	32.52	29.47	24.74	26.13	29.72	22.53	15.25	25.57
12	multi		64	✓	✓	-	-	8×1.2/85.9	24.13	32.80	29.55	24.90	26.04	30.25	22.73	15.31	25.72
13	multi		128	-	✓	-	-	8×0.8/82.7	24.34	32.86	29.51	24.73	26.15	30.01	22.58	15.07	25.66
14	multi		128	✓	✓	-	-	8×2.4/95.5	24.30	32.61	29.72*	25.07	26.29	30.46*	22.99	15.47	25.86
15	multi		256	-	✓	-	-	8×1.6/89.1	24.38	32.78	29.69	24.72	26.25	29.93	22.63	15.40	25.72
16	multi		256	✓	✓	-	-	8×4.8/114.7	24.61	32.94	29.67	**25.12**	26.16	**30.53**	22.66	15.31	25.88
17	multi		-	-	-	-	✓	8×35.5/8×36.3	**24.67**	**33.12**	**30.11**	25.05	**26.33**	29.85	**23.04**	**15.61**	**25.97**
18	multi		-	-	-	✓	✓	8×76.3/8×76.3	24.54*	32.95*	29.96*	25.01	26.31	30.04	22.66	15.54*	25.88

Table 1: BLEU on MuST-C dev set for **refinement**. In the **Dict** column, `mono` and `multi` mean, respectively, monolingual and multilingual dictionary. D is the Transformer hidden dimension. In the **Adapter** group, d is the adapter bottleneck dimension, ENC and DEC mean adding adapters to encoder and decoder, respectively; and idem for the **Finetune** group. Rows 1–2 and rows 9–10 represent our bilingual and multilingual baselines for each D. Values lower than the multilingual baselines are colored in blue. The highest values in each group of D are underlined, while the highest values of each column are in **bold** face. Furthermore, we select the top configurations (6, 8, 14, 18) and perform statistical significance test using bootstrap re-sampling (Koehn, 2004). Results passing the test (compared to the corresponding multilingual baselines, with p-value < 0.05) are marked with a star.

ST dataset from English to eight target languages including Dutch (`nl`), French (`fr`), German (`de`), Italian (`it`), Portuguese (`pt`), Romanian (`ro`), Russian (`ru`), and Spanish (`es`). Each direction includes a triplet of speech, transcription, and translation. Sizes range from 385 hours (`pt`) to 504 hours (`es`).

MuST-C-Imbalanced We built a low-resource version of MuST-C, called MuST-C-Imbalanced, in which we randomly keep only $X\%$ of the original training data, where $X = 100$ for `es`, `fr`; $X = 50$ for `ru`, `it`; $X = 20$ for `nl`, `ro`; and $X = 10$ for `de`, `pt` (same order of the languages in the original MuST-C if we sort them in decreasing amount of data). The amount of speech data ranges from 41 hours (`de`) to 504 hours (`es`) in this version, better reflecting real-world data imbalance scenarios.

4.2 Implementation details

Our implementation is based on the FAIRSEQ S2T toolkit (Wang et al., 2020). We experiment with two architectures: a small Transformer model with dimension $D = 256$ and a medium one where $D = 512$. All experiments use the same encoder with 12 layers. The decoder has 6 layers, except for the transfer learning scenario where we used the mBART decoder for initialization. We used 8k and 10k unigram vocabulary (Kudo and Richardson, 2018) for bilingual and multilingual models, respectively. The speech features are 80-dimensional log mel filter-bank. Utterances having more than 3000 frames are removed for GPU efficiency. We used SpecAugment (Park et al., 2019) with LibriSpeech basic (LB) policy for data augmentation.

We used the Adam optimizer (Kingma and Ba, 2015) with learning rate linearly increased for the first 10K steps to a value $\eta_{\max}$, then decreased proportionally to the inverse square root of the step counter. For all adapter experiments, $\eta_{\max}$ is set to $2\text{e}{-3}$. For the others, however, we perform a grid search over three values $\{2\text{e}{-3}, 2\text{e}{-4}, 2\text{e}{-5}\}$ and select the best one on the dev set, as they are more sensitive to the learning rate.

5 Refinement

In this section, a fully trained multilingual ST backbone is further refined on each language pair to boost the performance and close potential gaps with bilingual models. We compare adapter tuning with other fine-tuning approaches as well as the bilingual and multilingual baselines (the latter being the starting point for all fine-tuning approaches) (Bapna and Firat, 2019). Starting from

	D	Adapter d	Adapter ENC	Adapter DEC	Finetune ENC	Finetune DEC	# params (M) trainable/total	de	es	fr	it	nl	pt	ro	ru	avg
							Training data (hours)	41	504	492	232	89	38	86	245	
1	256	-	-	-	-	-	32.1/32.1	15.99	30.51	28.17	21.80	20.27	22.47	17.38	13.18	21.22
2	256	128	✓	✓	-	-	8×1.2/41.7	17.02	30.71	28.42	22.37	21.01	23.74	18.55	13.52	21.92
3	256	-	-	-	✓	✓	8×32.1/8×32.1	16.93	30.86	28.34	22.42	20.86	23.44	18.49	13.63	21.87
4	512	-	-	-	-	-	76.3/76.3	17.05	31.92	29.06	22.91	21.64	24.15	19.18	14.09	22.50
5	512	256	✓	✓	-	-	8×4.8/114.7	17.46	**31.94**	29.09	**23.11**	21.76	**24.96**	**19.50**	14.10	**22.74**
6	512	-	-	-	✓	✓	8×76.3/8×76.3	**17.49**	31.67	**29.27**	22.97	**21.80**	24.80	19.43	**14.17**	22.70

Table 2: BLEU on MuST-C dev set for **refinement** in the low-resource scenario where the models were trained on MuST-C-Imbalanced dataset. We refer to Table 1 for other notation.

	Method	# params (M) trainable/total	de	es	fr	it	nl	pt	ro	ru	avg
Ours	Baseline	76.3/76.3	24.18	28.28	**34.98**	24.62	**28.80**	**31.13**	23.22	15.88	26.39
Ours	Best adapting	8 × 4.8/76.3	**24.63**	**28.73**	34.75	**24.96**	**28.80**	30.96	23.70	**16.36**	**26.61**
Ours	Best fine-tuning	8 ×35.5/8 × 76.3	24.50	28.67	34.89	24.82	28.38	30.73	**23.78**	16.23	26.50
Li et al.	LNA-D	53.5/76.3	24.16	28.30	34.52	24.46	28.35	30.51	23.29	15.84	26.18
Li et al.	LNA-E	48.1/76.3	24.34	28.25	34.42	24.24	28.46	30.53	23.32	15.89	26.18
Li et al.	LNA-E,D	25.3/76.3	24.27	28.40	34.61	24.44	28.25	30.53	23.27	15.92	26.21

Table 3: BLEU on MuST-C test set. Our method compares favorably with (Li et al., 2020).

these backbones, we either add language-specific adapters and train them only, or we finetune the backbone on each language pair, either fully or partially. All these trainings are performed on MuST-C. The results are shown in Table 1. There are two main blocks corresponding to two architectures: $D = 256$ (small) and $D = 512$ (medium). Rows 1 and 9 provide the bilingual baselines, while rows 2 and 10 serve as the multilingual baselines for each block. In addition, we compare adapter-tuning with full fine-tuning and multilingual-training (baseline) on MuST-C-Imbalanced. Table 2 displays the results for this set of experiments.

Bilingual vs. Multilingual For the small architecture ($D = 256$), the bilingual models slightly outperform their multilingual counterpart (rows 1, 2). Looking further into the performance of each language pair, the multilingual model is able to improve the results for 4 out of 8 pairs (de, nl, pt, ru), mainly those in the lower-resource direction, but the joint multilingual training slightly hurts the performance of higher-resource pairs such as es, fr, it, and ro. Finally, we observe that the medium model ($D = 512$) performs better in the multilingual setting than the bilingual one (rows 9, 10).

Adapter tuning vs. Fine-tuning Both recipes yield improvements over the multilingual baseline and recover the lost performance of higher-resource directions compared to the bilingual baseline for the small model ($D = 256$). For the medium one ($D = 512$), one adapter tuning (row 14) can slightly improve the scores in all directions and even approach the results of the best fine-tuning experiment (row 17) while maintaining much lower model sizes (95.5M vs. 8× 36.3M parameters).

Low-resource scenario The obtained results on small models show that adapter-tuning achieved the best performance, producing clear improvements over the baseline, especially for the low-resource languages: +1.1 BLEU on average on nl, ro, de, pt; +0.3 BLEU on average on es, fr, ru, it; which is competitive to full fine-tuning (+0.9 and +0.4 BLEU, respectively) while being more parameter-efficient as well as simpler for training and deployment (one model with adapters versus eight separate models). For larger models, however, the improvement is smaller: +0.4 BLEU on average on the lower-resource pairs and +0.1 on the higher-resource ones; while those of full fine-tuning are +0.4 and roughly no improvement, respectively.

Results on test set We select the best-performing fine-tuning recipes on the dev set (rows 16 and 17 in Table 1) for evaluation on the test set. For reference, we also include the multilingual baseline (row 10). Moreover, to go beyond conventional fine-tuning approaches, we also compare our recipes with a contemporary work in which only several components of the network are finetuned (Li et al., 2020). For a fair comparison, we did not use large pre-trained components such as wav2vec (Baevski et al., 2020) or mBART (Tang et al., 2020) but instead considered the same pre-trained compo-

	Adapter			Finetune	# params (M)								
	d	ENC	DEC	xattn	trainable/total	de	es	fr	it	nl	pt	ro	ru	avg
1	-	-	-	-	8×31.1/8×31.1	22.16	30.42	27.92	22.92	24.10	27.19	21.51	14.36	23.82
2	-	-	-	✓	38 / 486	18.41	25.42	23.46	18.44	20.87	20.55	17.19	11.79	19.52
3	512	-	✓	-	101 / 587	0.94	0.65	0.93	0.76	0.95	0.89	0.52	0.93	0.82
4	512	-	✓	✓	139 / 587	21.98	29.47	27.05	22.89	24.06	26.34	21.0	14.35	23.39
5	512	✓	✓	-	152 / 638	11.04	18.62	16.10	12.37	13.18	14.29	10.62	6.95	12.90
6	512	✓	✓	✓	190 / 638	22.62	30.85	28.23	23.09	24.43	26.56	22.13	14.92	24.10

Table 4: BLEU on MuST-C dev set for **transfer learning** from pre-trained ASR and mBART models. We compare the results with the bilingual baselines (trained from scratch), shown in row 1 (which is identical to row 1 in Table 1). The column "Finetune xattn" means updating the cross-attention parameters. We refer to Table 1 for other notation.

nents used in our previous experiments. Following (Li et al., 2020), we considered six variants: fine-tuning LayerNorm + Attention in the encoder (LNA-E), or the decoder (LNA-D), or both (LNA-E,D); each with or without the length adapter. We found that adding the length adapter did not help in our experiments. Table 3 shows that our approach compares favorably with (Li et al., 2020) in terms of both performance and parameter-efficiency.

Other comments For small models, the encoder adapters boost the performance (0.3–0.4 BLEU on average) in all directions (rows 3 and 4, 5 and 6, Table 1), indicating that language-specific adapters can tweak the encoder representations to make them better suited for the decoder. In larger models, however, the impact of the encoder adapters is varied depending on languages and bottleneck dimensions. We also notice that increasing the bottleneck dimension slightly improves performance while remaining parameter-efficient. Fine-tuning remains the best option to optimize the models in most cases but leads to much larger model sizes. The adapter-tuning approach is competitive to fine-tuning while being much more parameter-efficient.

6 Transfer Learning

In this section, we show that adapters can be used to combine available pre-trained models to perform a multilingual ST task. In particular, we initialize the encoder using a pre-trained ASR encoder (on MuST-C)[3] provided by Wang et al. (2020) and the decoder using mBART50, a multilingual denoising auto-encoder pre-trained on 50 languages (Tang et al., 2020). We tune language independent cross-attention and language-specific adapters on top of these backbone models (using MuST-C as well). The results presented in Table 4 highlight that fine-tuning cross-attention is crucial to transfer to multilingual ST (rows 3 and 5 show poor results without doing so). Adding adapters to the backbone decoder (row 4) or to both encoder and decoder (row 6) further boosts performance, demonstrating the ability of adapters to connect off-the-shelf models in a modular fashion. The best-performing model in this recipe (row 6) also outperforms bilingual systems (row 1) despite having fewer trainable parameters (190M vs. 248M). It is also important to mention that while we experiment on 8 target languages of MuST-C corpus, the multilingual ST model of row 2 should be practically able to decode into 50 different target languages. Investigating such a zero-shot ST scenario is left for future work.

[3]Pre-training on ASR data and then transferring to ST is not new but rather standard. See, e.g., Bansal et al. (2019).

7 Conclusion

We have presented a study of adapters for multilingual ST and shown that language-specific adapters can enable a fully trained multilingual ST model to be further specialized in each language pair. With these adapter modules, one can efficiently obtain a single multilingual ST system that outperforms the original multilingual model as well as multiple bilingual systems while maintaining a low storage cost and simplicity in deployment. In addition, adapter modules can also be used to connect available pre-trained models such as an ASR model and a multilingual denoising auto-encoder to perform the multilingual speech-to-text translation task.

Acknowledgments

This work was supported by a Facebook AI SRA grant, and was granted access to the HPC resources of IDRIS under the allocation 2020-AD011011695 made by GENCI. It was also done as part of the Multidisciplinary Institute in Artificial Intelligence MIAI@Grenoble-Alpes (ANR-19-P3IA-0003). We thank the anonymous reviewers for their insightful questions and feedback.

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *NeurIPS*.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2019. Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. In *NAACL-HLT (1)*, pages 58–68. Association for Computational Linguistics.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *EMNLP/IJCNLP (1)*, pages 1538–1548. Association for Computational Linguistics.

Mattia A Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. Must-c: a multilingual speech translation corpus. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2012–2017. Association for Computational Linguistics.

Carlos Escolano, Marta R. Costa-jussà, José A. R. Fonollosa, and Carlos Segura. 2020. Enabling zero-shot multilingual spoken language translation with language-specific encoders and decoders. *CoRR*, abs/2011.01097.

Kunihiko Fukushima and Sei Miyake. 1982. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer.

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 *ICML*.

Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Enrique Yalta Soplin, Tomoki Hayashi, and Shinji Watanabe. 2020. Espnet-st: All-in-one speech translation toolkit. *arXiv preprint arXiv:2004.10234*.

Anjuli Kannan, Arindrima Datta, Tara N. Sainath, Eugene Weinstein, Bhuvana Ramabhadran, Yonghui Wu, Ankur Bapna, Zhifeng Chen, and Seungji Lee. 2019. Large-scale multilingual speech recognition with a streaming end-to-end model. In *INTERSPEECH*, pages 2130–2134. ISCA.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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.

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, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pages 66–71. Association for Computational Linguistics.

Hang Le, Juan Pino, Changhan Wang, Jiatao Gu, Didier Schwab, and Laurent Besacier. 2020. Dual-decoder transformer for joint automatic speech recognition and multilingual speech translation. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020*. Association for Computational Linguistics.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Taewoo Lee, Min-Joong Lee, Tae Gyoon Kang, Seokyeoung Jung, Minseok Kwon, Yeona Hong, Jungin Lee, Kyoung-Gu Woo, Ho-Gyeong Kim, Jiseung Jeong, et al. 2020. Adaptable multi-domain language model for transformer asr. *arXiv preprint arXiv:2008.06208*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Xian Li, Changhan Wang, Yun Tang, Chau Tran, Yuqing Tang, Juan Pino, Alexei Baevski, Alexis Conneau, and Michael Auli. 2020. Multilingual speech translation with efficient finetuning of pretrained models. *arXiv e-prints*, pages arXiv–2010.

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. *Trans. Assoc. Comput. Linguistics*, 8:726–742.

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 2613–2617. ISCA.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. In *EACL*, pages 487–503. Association for Computational Linguistics.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulic, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterhub: A framework for adapting transformers. In *EMNLP (Demos)*, pages 46–54. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulic, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: an adapter-based framework for multi-task cross-lingual transfer. In *EMNLP (1)*, pages 7654–7673. Association for Computational Linguistics.

Jerin Philip, Alexandre Bérard, Laurent Besacier, and Matthias Gallé. 2020. Language adapters for zero-shot neural machine translation. In *EMNLP 2020*.

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

Asa Cooper Stickland, Xian Li, and Marjan Ghazvininejad. 2021. Recipes for adapting pre-trained monolingual and multilingual models to machine translation. In *EACL*, pages 3440–3453. 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 *ICML*.

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. *CoRR*, abs/2008.00401.

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.

Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino. 2020. Fairseq S2T: fast speech-to-text modeling with fairseq. In *AACL/IJCNLP (System Demonstrations)*, pages 33–39. Association for Computational Linguistics.

Genta Indra Winata, Guangsen Wang, Caiming Xiong, and Steven Hoi. 2020. Adapt-and-adjust: Overcoming the long-tail problem of multilingual speech recognition. *arXiv preprint arXiv:2012.01687*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP (Demos)*, pages 38–45. Association for Computational Linguistics.

Yun Zhu, Parisa Haghani, Anshuman Tripathi, Bhuvana Ramabhadran, Brian Farris, Hainan Xu, Han Lu, Hasim Sak, Isabel Leal, Neeraj Gaur, et al. 2020. Multilingual speech recognition with self-attention structured parameterization. *Proc. Interspeech 2020*, pages 4741–4745.

A Parallel Adapters

In this section, we present our preliminary experiments in which we explore different positions of the parallel adapters: in parallel with either Transformer layers or their sub-layers. We perform experiments where the adapters are added to the decoder. The results are shown in Table 5.

	Adapter			# params (M)	
	d	h	type	trainable/total	en-de
1	-	-	-	32.1/32.1	22.37
2	128	-	`ser`	0.4/32.5	22.45
3	128	4	`par-TL`	0.8/32.9	21.67
4	128	4	`par-SA`	0.8/32.9	19.55
5	128	4	`par-XA`	0.8/32.9	19.22

Table 5: BLEU on dev set for **parallel vs. serial adapters**. In the "Adapter" block, d is the adapter's dimension, h is the number of heads, `ser` stands for serial adapters, and `par` stands for parallel ones. The suffixes denote the position of the parallel adapters: in parallel with the Transformer layer (`TL`), or with self-attention sub-layer (`SA`), or with cross-attention sub-layer (`XA`).

Among the parallel variants, the one that performs operations in parallel with a full layer produces the best result. However, its performance still could not surpass the serial adapter (row 2) as well as the starting point (row 1).

B Specializing

In addition to the refinement recipe where language-specific adapters tailor the frozen multilingual ST model to translate in the corresponding direction, we also propose a recipe to facilitate the specialization in individual language pairs: by replacing the multilingual vocabulary by the monolingual ones corresponding to each target language. This recipe allows us to transfer from multilingual models to monolingual ones. A practical benefit is that one can easily leverage pre-trained multilingual models for new languages.

	Dict	D	Adapter d	Adapter ENC	Adapter DEC	Finetune ENC	Finetune DEC	# params (M) trainable/total	de	es	fr	it	nl	pt	ro	ru	avg
1	mono	256	-	-	-	-	-	8×31.1/8×31.1	22.16	30.42	27.92	22.92	24.10	27.19	21.51	14.36	23.82
2	multi		-	-	-	-	-	32.1/32.1	22.37	30.40	27.49	22.79	24.42	27.32	20.78	14.54	23.76
3	mono		64	-	✓	-	-	8×4.3/8×31.3	23.28	30.95	28.31	23.25	24.76	27.84	21.55	14.60	24.32
4	mono		64	✓	✓	-	-	8×4.7/8×31.7	23.53	31.16	28.83	23.29	24.43	28.18	21.38	14.66	24.44
5	mono		128	-	✓	-	-	8×4.5/8×31.5	23.33	31.05	28.67	23.43	24.83	28.10	21.44	14.58	24.43
6	mono		128	✓	✓	-	-	8×5.3/8×32.3	22.09	30.09	27.63	22.53	24.24	27.09	20.36	14.19	23.53
7	mono		-	-	-	-	✓	8×13.6/8×31.1	24.03	31.79	29.64	24.16	25.55	28.92	22.11	15.00	25.15
8	mono		-	-	-	✓	✓	8×31.1/8×31.1	23.89	31.72	29.23	23.65	25.14	28.23	21.83	14.80	24.81
9	mono	512	-	-	-	-	-	8×74.3/8×74.3	21.93	30.46	27.90	22.64	23.98	25.98	20.5	14.01	23.42
10	multi		-	-	-	-	-	76.3/76.3	23.98	32.47	29.24	24.97	26.20	29.81	22.74	15.30	25.59
11	mono		64	-	✓	-	-	8×8.6/8×74.7	23.85	31.79	29.63	24.26	25.77	28.97	22.18	15.02	25.18
12	mono		64	✓	✓	-	-	8×9.4/8×75.5	23.74	31.62	29.44	24.02	25.56	29.23	22.25	15.39	25.16
13	mono		128	-	✓	-	-	8×9.0/8×75.1	23.91	32.05	29.47	24.08	25.86	29.28	22.30	15.28	25.28
14	mono		128	✓	✓	-	-	8×10.6/8×76.7	23.98	32.28	29.40	24.46	25.46	29.28	21.90	15.15	25.24
15	mono		256	-	✓	-	-	8×9.8/8×75.9	23.91	32.12	29.45	24.17	25.67	29.01	22.31	15.37	25.25
16	mono		256	✓	✓	-	-	8×13/8×79.1	24.39	32.33	29.46	24.07	25.72	29.84	22.07	15.25	25.39
17	mono		-	-	-	-	✓	8×33.4/8×74.3	24.95	32.85	30.33	25.02	26.08	29.97	23.01	15.69	25.99
18	mono		-	-	-	✓	✓	8×74.3/8×74.3	24.77	32.35	30.14	24.79	25.79	29.85	22.71	15.77	25.77

Table 6: BLEU on MuST-C dev set for **specialization**. We refer to Table 1 for all notation.

Table 6 displays the results of the specializing recipe. Starting from a trained multilingual ST model, one can obtain an improvement of 1.3–1.4 BLEU on average (row 8 vs. row 1 and 2) compared to the bilingual and multilingual baselines trained from scratch for the small architecture where $D = 256$. However, for a larger network ($D = 512$), the gain is more modest (0.4 BLEU on average).