PARAMETER-EFFICIENT TUNING WITH ADAPTIVE BOTTLENECKS FOR AUTOMATIC SPEECH RECOGNITION

Geoffroy Vanderreydt[1], *Amrutha Prasad*[2,3], *Driss Khalil*[2], *Srikanth Madikeri*[2], *Kris Demuynck*[1], *Petr Motlicek*[2,3]

[1]IDLab, Ghent University - imec, Ghent, Belgium
[2]Idiap Research Institute, Martigny, Switzerland
[3]Brno University of Technology, Speech@FIT, Brno, Czech Republic

ABSTRACT

Transfer learning from large multilingual pretrained models, like XLSR, has become the new paradigm for Automatic Speech Recognition (ASR). Considering their ever-increasing size, fine-tuning all the weights has become impractical when the computing budget is limited. Adapters are lightweight trainable modules inserted between layers while the pretrained part is kept frozen. They form a parameter-efficient fine-tuning method, but they still require a large bottleneck size to match standard fine-tuning performance. In this paper, we propose ABSADAPTER, a method to further reduce the parameter budget for equal task performance. Specifically, ABSADAPTER uses an Adaptive Bottleneck Scheduler to redistribute the adapter's weights to the layers that need adaptation the most. By training only 8% of the XLSR model, ABSADAPTER achieves close to standard fine-tuning performance on a domain-shifted Air-Traffic Communication (ATC) ASR task.

Index Terms— ASR, XLSR, Adapters, ATC

1. INTRODUCTION

Finetuning large pretrained models to a downstream task has become the new paradigm of Natural Language Processing (NLP) and Automatic Speech Recognition (ASR). The performance on the downstream task scales well with increasing model capacity [1], hence new state-of-the-art (SOTA) models keep increasing in model size. Standard fine-tuning becomes then impractical when the compute budget is limited. Additionally, fine-tuning all the model's weights is time consuming and memory inefficient as it results in a copy of the model for each downstream task. Recent works showed that standard fine-tuning tends to overfit in low-resource situation [2] and it is prone to catastrophic forgetting [3]. Lastly, these large SOTA models are massively over-parameterized and the same performance can be obtained with a subset of the parameters trained [1, 2].

Parameter-efficient tuning (PET) consists in freezing the pretrained parameters and train lightweight modules injected in the model. This results in the addition of only a small number of task-specific trainable parameters. A large variety of PET methods have been proposed in the literature; we refer to [1] for a unified overview. Many of these methods have been applied to Natural Language Processing (NLP) [4] and more recently to speech processing [2]. Among them, adapters [5] present an attractive trade-off between parameter efficiency and performance. Adapters are lightweight trainable modules inserted between layers while the pretrained part is kept frozen. They have been successfully integrated in ASR systems using the wav2vec2.0 model [6] and the conformer model [7, 3, 8, 9]. Adapters have been used to train multilingual ASR models [10, 11], where each adapter serves one language. They have been applied to the XLSR model [12] for language adaptive training [13]. In [14], they were used to adapt the ASR model to accents unseen during training. The authors in [3] show that adapters enable to overcome catastrophic forgetting. In [15], adapters are exploited for cross-lingual low-resource ASR.

In this paper, we investigate adapters in the context of domain-shifted adaptation of the XLSR model for Air Traffic Communication (ATC) ASR. ATC speech is noisy, accented and often pronounced at high speech rate. Hence it constitutes a significant domain shift compared to the pretraining data of XLSR. ATC can contain very specific jargon so we adopt a hybrid ASR approach to allow contextual boosting [16]. To the best of our knowledge, we are the first to study adapters in a hybrid ASR framework that employs large pre-trained models.

Previous works have shown that lower layers (close to the input signal) extract generic speech features, while higher layers encode more phonemic information [7, 17]. Hence, for transfer learning from self-supervised speech models to downstream ASR, bottom layers require less adaptation than top layers. For NLP, dropping the adapters in the bottom layers has shown to be a simple and effective method to further reduce the parameter budget with minimal performance loss [5, 18, 19]. Recently, this has been successfully transferred to ASR [7, 9]. Other works [17] suggest that the adapter weights

979-8-3503-0689-7/23/$31.00

can be pruned for equal performance, supporting the fact that there is room for further parameter budget reduction. Based on these findings, we propose ABSADAPTER, a simple, yet effective approach to redistribute the adapter's weights to the layers that need adaptation the most. ABSADAPTER relies on an adaptive bottleneck scheduler (ABS), based on the adaptation importance of each transformer layer. In this work, we employ a simple linear scheduler that we hope can be a baseline for future work. A recent work in NLP, named AdaLoRA [20], builds upon the low-rank adaptation method LoRA [21], and proposes to adaptively allocate budget among weight matrices according to their importance score. This method assigns a higher rank to the critical matrix increments. Since LoRA can be seen as a special variant of the adapter (scaled and in parallel insertion) [1], our proposed method can be viewed as a simplified version of this recent work, without the need to add additional computation such as singular value decomposition.
The contributions of this paper are:

- First time integration of adapters in a hybrid ASR framework that employs the large pretrained XLSR model. Are the design elements for adapters transferrable to this architecture?
- Insights on the adaptation to domain shifted data: which layers need more adaptation?
- We propose ABSADAPTER: a parameter-efficient tuning method based on adaptive bottlenecks.

2. ADAPTER IN XLSR-TDNNF-LFMMI

Following [22], we fine-tune the wav2vec2.0 models with the E2E-LFMMI criterion [23]. In this architecture, the output of wav2vec2.0 is passed to a multi-layer factorized time-delay network (TDNNF). In [22], it was shown that finetuning wav2vec2.0 with CTC or LF-MMI results in similar performance. In our case, the LF-MMI based hybrid model allows us to boost words directly in the decoding graph [16]. This is interesting for the ATC domain because each airport can have a specific set of words that need to be properly detected. [24] gives baseline results for XLSR-TDNNF-LFMMI on similar ATC data.
Figure 1 depicts how we insert adapters in the XLSR-TDNNF-LFMMI architecture. We follow the standard approach to insert the adapter in the transformer layers, as proposed in [15]: one adapter after the attention block (ATTN) and another after the feed-forward network (FFN). We did not observe any improvement by inserting the adapters in parallel, as suggested in prior work [9]. The adapter consists of a down-projection, a *Gelu* non-linearity [25], an up-projection, layer normalization and a skip-connection that adds the input to the output. In the remaining of the paper, we refer to the intermediate projection dimension as the adapter's bottleneck size. The Parameter Efficient Tuning (PET) method refers to training the adapters while keeping the rest of the XLSR model frozen.

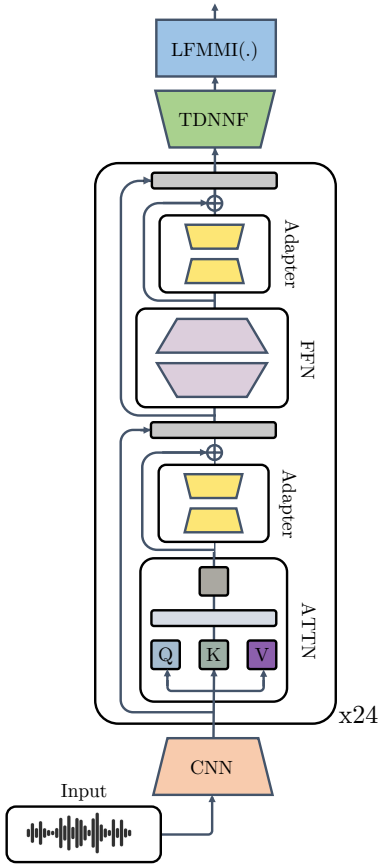

Fig. 1. Adapters in the XLSR-TDNNF-LFMMI framework. The adapters are inserted sequentially after the attention block and the feed-forward network of the transformer layer.

3. EXPERIMENTS

3.1. Datasets

We train and evaluate the ASR models on Air Traffic Communication (ATC) speech data. ATC ASR is challenging: the speech data is generally noisy, accented and has a high speaking rate. Also the vocabulary is very specific to the domain. Hence, the fine-tuning data can be considered out-of-distribution compared to the pretraining data in XLSR [12].
We follow the experimental setup in prior work [26]. We group three public datasets to form a training set with manually annotated data. The language is English with various accents. The test set, referred to as *NATS* [24], contains aircraft approach communication between the controller and the pilot from an airport in London. The datasets are summarized in Table 1.

Table 1. Air traffic control communications datasets.

Database	Accents	Hrs
Training datasets		
ATCOSIM [27]	de, fr, de-CH	10
UWB-ATCC [28]	cs	13
LDC-ATCC [29]	en-US	26
Total (after cleaning)		41
Evaluation datasets		
NATS [24]	en-GB	2

3.2. Experimental setup

The input to the XLSR is raw speech sampled at 16kHz. The output is passed into a three layered factorized time-delay network (TDNNF). We fine-tune the XLSR together with TDNNF layers using flat start LFMMI (E2E-LFMMI) for 5 epochs. We use a batch size of 8. The E2E-LFMMI is trained with biphone units. We do 3-fold data augmentation using speed perturbation with 0.9 and 1.1 speed rates.

We use a tri-stage learning schedule; the learning rate follows a linear increase to $3e{-}5$ over 10% of the updates, remains constant for 40% of the updates, and then linearly decreases for the remaining 50% of the updates. For the TDNNF model parameters, the learning rate is set to 20 times the current learning rate used for XLSR model updates. Additionally, we employ the natural gradient update method for training with the E2E-LFMMI objective, as described in [30].

All models are trained using PyTorch [31]. For fine-tuning with E2E-LFMMI we rely on the Espresso toolkit [32], which implements the LFMMI loss using PyChain [33]. The PyTorch implementation from [34] is used for the natural gradient update. For decoding we use a WFST decoder from Kaldi [35] with beam width 15.

3.3. Adapters vs finetuning

In this section, we show the effectiveness of adapters in the XLSR-TDNNF-LFMMI framework and compare them to standard fine-tuning (FT) with respect to parameter efficiency and Word Error Rate (WER). Inspired by other works [36], we explore their variants where only the feed-forward network in the transformer layer (Figure 1) is fine-tuned or adapted.

The XLSR model (wav2vec2.0 Large) accounts for 316M parameters and the subsequent TDNNF layers for 12M parameters. During standard finetuning, all the model weights are trained, as described in Section 3.2. The adapter-based PET consists in training the adapters and TDNNF layers while keeping the wav2vec2.0 model frozen (the CNN-based feature extractor and the transformer layers). Since the TDNNF layers are trained in any case, we will refer to the percentage of trained parameters with respect to the wav2vec2.0 model only. Figure 2 shows the parameter budget saving using Adapters (bottleneck size 256): by training only 8% of the wav2vec2.0 model, we achieve 10.3 WER, which is an 8% relative WER increase (ΔWER) compared to standard fine-tuning (FT).

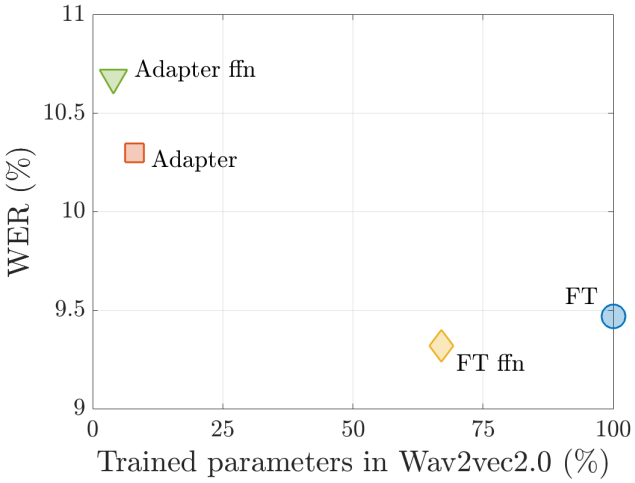

Fig. 2. Trade-off between trained parameters (percentage of wav2vec2.0 model) and WER. *FT* refers to standard fine-tuning, *FT ffn* to fine-tuning the transformer's feedforward network only, *Adapter* to the baseline PET method described in Section 2 and *Adapter ffn* to a model where adapters are inserted solely after the feedforward network, as proposed in [36].

Previous work [36] showed that it is sufficient to insert the adapter after the feedforward network (FFN) module only for transfer learning to a downstream task. We refer to this model as *Adapter ffn* and we compare it to fine-tuning the FFN only, referred to as *FT ffn*. As illustrated in Figure 2, *FT ffn* achieves similar performance to full fine-tuning *FT ffn* (slight improvement). However, because of the high intermediate dimension, the FFN still accounts for a large amount of the total number of parameters (67%). On the other hand, the adapter variant (*Adapter ffn*) halves the number of trained parameters compared to the baseline *Adapter*, resulting in only 4% of the wav2vec2.0 model being trained. If we set a maximum relative WER increase ΔWER compared to standard fine-tuning at 10%, the *Adapter ffn* method is unsatisfacotry (18% ΔWER). In the next sections, we investigate design elements to improve the trade-off between parameter budget and performance.

3.4. Bottleneck size

In this section, we investigate how the adapter's bottleneck size affects the ASR performance. Figure 3 shows the WER on NATS when we vary the adapter's bottleneck size from 32 to 512 by doubling it at every step. Doubling the bottleneck

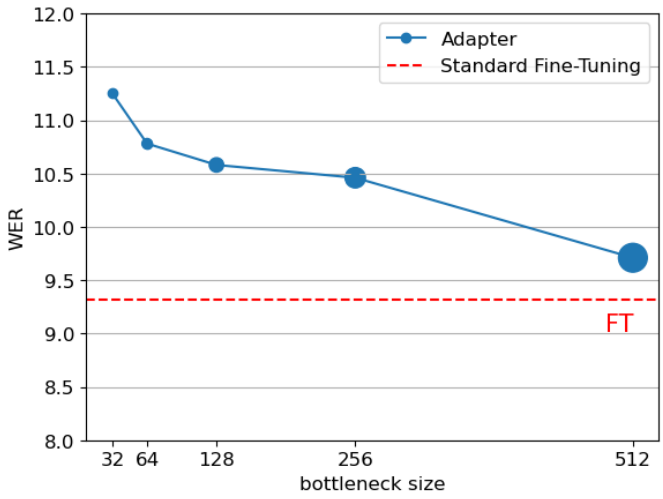

Fig. 3. The WER decreases with increasing adapter's bottleneck size. The marker size indicates the parameter budget. Using a bottleneck size of 512 results in close to full fine-tuning performance while training 16% of the model's weights.

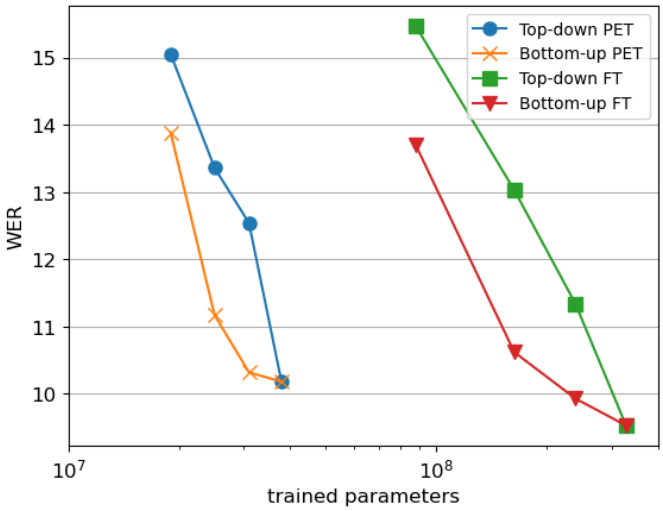

Fig. 4. PET: Parameter-efficient tuning using adapters. FT: standard Fine-tuning. Top-down refers to tuning the top layers first and sequentially tune additional lower layers. Bottom-up refers to tuning the bottom layers first and sequentially tune additional higher layers. the XLSR model has 24 layers and we use 6 layers steps, thus 4 data points per curve.

size doubles the trained parameters in wav2vec2.0. Overall, The ASR model improves with increasing bottleneck size and achieves close to standard fine-tuning performance with bottleneck size 512. Those observations are in accordance with findings in prior works [9, 17].

3.5. Domain adaptation vs task adaptation

Prior works in NLP suggest that top layers (closer to the encoder output) are more important to finetune [19] and consequently adapters in the bottom layers (closer to the input signal) can be dropped without significant loss of performance [5, 18]. This has been confirmed for wav2vec2.0+CTC based and conformer based ASR [7, 9]. We investigate if this observation conveys to domain-shifted transfer learning.

We fine-tune the bottom quarter (first 6 layers), the bottom half (first 12 layers) and the bottom three quarters (first 18 layers) of the XLSR layers, while keeping the remaining layers frozen. Similarly, we fine-tune the top quarter, the top half and the top three quarters of the XLSR layers. The resulting WERs are depicted in Figure 4 (right). We conduct similar experiments with adapter-based parameter-efficient tuning by increasing the number of adapters inserted in a top-down and bottom-up fashion (Figure 4 left).The results are consistent with both tuning methods; with this setup, it is more important to adapt the bottom layers. Dropping the top 6 layers, reduces the trained parameters in XLSR from 8% to 6%, for only a small performance loss (less than 2% relative ΔWER). One may think that due to the large speech domain shift in this setup between unsupervised (pretraining) data [12] and supervised (fine-tuning) data (ATC), the bottom layers - which are commonly associated with the extraction of generic speech features - require significant adaptation.

In order to acquire better insight, we compare the WER evolution during training of the model, in which on one side only the 6 top layers are fine-tuned and on the other side only the 6 bottom layers are finetuned.The WER curves - as function of hours of speech data - are illustrated in Figure 5. We conduct this experiments both on the domain-shifted ATC setup and on in-domain Swahili speech data from the Babel dataset [37], that has been used for self-supervised learning (SSL) during the pretraining stage of XLSR [12]. We observe that for both tasks, the model adapts faster to the new task (ASR as opposed to SSL) when we fine-tune the top layers only. In the in-domain case (Babel Swahili), eventually both fine-tuning methods converge. However, in the domain-shifted case (ATC), fine-tuning the bottom layers prevails over fine-tuning the top layers. This example confirms that the bottom layers are the important layers to adapt in domain-shifted transfer learning setup. More broadly, it supports the theory that domain adaptation is done by the lower layers and task adaptation is done by the higher layers.

3.6. Adaptive bottlenecks

In the previous sections, we concluded that increasing the adapter's bottleneck size improves the downstream ASR performance, with the downside of increasing the total parameter budget. We also concluded that dropping a few adapters can reduce the parameter budget with a minimal loss in perfor-

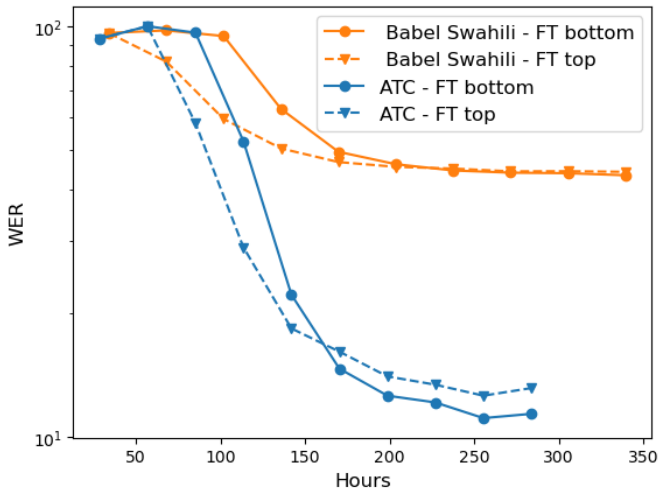

Fig. 5. Tuning (standard FT and adapter-based PET) the 6 bottom layers versus the 6 top layers. The models are evaluated on NATS. Tuning the top layers learns faster the downstream task but tuning the bottom layers is better on the long run.

model	bn	WER	ΔWER	% params
FT	-	9.3	0%	100%
Adapter	256	10.5	+13%	8%
Adapter	512	9.8	+5%	16%
ABSADAPTER	512-32	10.0	+7%	8%

Table 2. Trade-off between parameter saving and performance loss for the baseline adapters and the proposed ABSADAPTER. *bn* refers to the bottleneck size.

mance on the ASR task. In our ATC ASR setup, these layers corresponded to the top layers. Based on these takeaways, we propose ABSADAPTER: a parameter-efficient tuning method based on adaptive bottlenecks. By allowing different bottleneck sizes depending on the layer depth, we redistribute the adapters' weights where adaptation is needed the most. In our ATC setup, this corresponds to attributing a high bottleneck size to the bottom layers - for better domain adaptation - and a low bottleneck size to the top layers. The goal is to reduce the parameter budget for equal performance, or equivalently, increase the performance for equal parameter budget, compared to a constant bottleneck method. As described in Section 1, we propose a simple yet effective approach to schedule the bottleneck size: a linear decrease with the layer depth. We set a maximum bottleneck size for the first layer, a minimum bottleneck size for the last layer, and decrease the size linearly along the layers. This method is simple, and does not require any additional computation as in [20].

Table 2 compares the relative WER increase (ΔWER) and the amount of trained parameters (%params) for the proposed method and the two best performing adapter-based methods discussed in this work. We set the maximum and minimum bottleneck sizes to 512 and 32, respectively. ABSADAPTER performs better than the adapter using 256 bottleneck size, with a similar amount of trained parameters: 8% of the XLSR model. It slightly underperforms the Adapter with 512 bottleneck size while having half its number of parameters. These preliminary results show the effectiveness of the method and drives us to further investigate on broader setups.

4. CONCLUSIONS AND FUTURE WORK

In this work, we studied lightweight trainable modules - called adapters - in a hybrid ASR framework that employs large pre-trained models. We investigated design elements such as the bottleneck size choice and the adapter's position to find the best trade-off between parameter budget and ASR performance. Our study on adapters applied to Air Traffic Communication gave insights on domain-shifted transfer learning from self-supervised speech models to downstream ASR. We proposed a simple method to adapt the bottleneck size based on the layer adaptation importance. Results on ATC show that the method improves ASR performance while keeping the same parameter budget. In future work, we would like to apply the proposed ABSADAPTER to a larger benchmark that involves more in-domain data. We suspect the benefits of the method are limited in the current setup, as the large pretrained XLSR needs to do both speech domain adaptation in the lower layers and task adapatation in the higher layers. Another investigation direction would be to explore more complex schedulers than the linear one, potentially adressing the domain-shifted transfer learning better.

5. ACKNOWLEDGEMENTS

This work was supported by CRiTERIA, an EU project, funded under the Horizon 2020 programme, grant agreement no. 101021866.

6. REFERENCES

[1] Jonas Pfeiffer, Sebastian Ruder, Ivan Vulić, and Edoardo Maria Ponti, "Modular deep learning," *arXiv preprint arXiv:2302.11529*, 2023.

[2] Zih-Ching Chen, Chin-Lun Fu, Chih-Ying Liu, Shang-Wen Daniel Li, and Hung-yi Lee, "Exploring efficient-tuning methods in self-supervised speech models," in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023, pp. 1120–1127.

[3] Steven Vander Eeckt and Hugo Van Hamme, "Using adapters to overcome catastrophic forgetting in end-to-end automatic speech recognition," in *ICASSP*

2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2023, pp. 1–5.

[4] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig, "Towards a unified view of parameter-efficient transfer learning," *arXiv preprint arXiv:2110.04366*, 2021.

[5] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2790–2799.

[6] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in neural information processing systems*, vol. 33, pp. 12449–12460, 2020.

[7] Bethan Thomas, Samuel Kessler, and Salah Karout, "Efficient adapter transfer of self-supervised speech models for automatic speech recognition," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7102–7106.

[8] Somshubra Majumdar, Shantanu Acharya, Vitaly Lavrukhin, and Boris Ginsburg, "Damage control during domain adaptation for transducer based automatic speech recognition," *arXiv preprint arXiv:2210.03255*, 2022.

[9] Nanxin Chen, Izhak Shafran, Yu Zhang, Chung-Cheng Chiu, Hagen Soltau, James Qin, and Yonghui Wu, "Efficient adapters for giant speech models," *arXiv preprint arXiv:2306.08131*, 2023.

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

[11] Anjuli Kannan, Arindrima Datta, Tara N Sainath, Eugene Weinstein, Bhuvana Ramabhadran, Yonghui Wu, Ankur Bapna, Zhifeng Chen, and Seungji Lee, "Large-scale multilingual speech recognition with a streaming end-to-end model," *arXiv preprint arXiv:1909.05330*, 2019.

[12] Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli, "Unsupervised cross-lingual representation learning for speech recognition," *arXiv preprint arXiv:2006.13979*, 2020.

[13] Yizhou Lu, Mingkun Huang, Xinghua Qu, Pengfei Wei, and Zejun Ma, "Language adaptive cross-lingual speech representation learning with sparse sharing sub-networks," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6882–6886.

[14] Xun Gong, Yizhou Lu, Zhikai Zhou, and Yanmin Qian, "Layer-wise fast adaptation for end-to-end multi-accent speech recognition," *arXiv preprint arXiv:2204.09883*, 2022.

[15] Wenxin Hou, Han Zhu, Yidong Wang, Jindong Wang, Tao Qin, Renjun Xu, and Takahiro Shinozaki, "Exploiting adapters for cross-lingual low-resource speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 317–329, 2021.

[16] Iuliia Nigmatulina, Juan Zuluaga-Gomez, Amrutha Prasad, Seyyed Saeed Sarfjoo, and Petr Motlicek, "A two-step approach to leverage contextual data: speech recognition in air-traffic communications," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6282–6286.

[17] Shwai He, Liang Ding, Daize Dong, Miao Zhang, and Dacheng Tao, "Sparseadapter: An easy approach for improving the parameter-efficiency of adapters," *arXiv preprint arXiv:2210.04284*, 2022.

[18] Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych, "Adapterdrop: On the efficiency of adapters in transformers," *arXiv preprint arXiv:2010.11918*, 2020.

[19] Jeremy Howard and Sebastian Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.

[20] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao, "Adaptive budget allocation for parameter-efficient fine-tuning," *arXiv preprint arXiv:2303.10512*, 2023.

[21] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[22] Apoorv Vyas, Srikanth Madikeri, and Hervé Bourlard, "Comparing ctc and lfmmi for out-of-domain adaptation of wav2vec 2.0 acoustic model," *arXiv preprint arXiv:2104.02558*, 2021.

[23] Hossein Hadian, Hossein Sameti, Daniel Povey, and Sanjeev Khudanpur, "End-to-end speech recognition using lattice-free mmi.," in *Interspeech*, 2018, pp. 12–16.

[24] Juan Zuluaga-Gomez, Amrutha Prasad, Iuliia Nigmatulina, Saeed Sarfjoo, Petr Motlicek, Matthias Kleinert, Hartmut Helmke, Oliver Ohneiser, and Qingran Zhan, "How does pre-trained wav2vec2.0 perform on domain shifted asr? an extensive benchmark on air traffic control communications," *IEEE Spoken Language Technology Workshop (SLT), Doha, Qatar*, 2023.

[25] Dan Hendrycks and Kevin Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[26] Juan Zuluaga-Gomez, Amrutha Prasad, Iuliia Nigmatulina, Petr Motlicek, and Matthias Kleinert, "A virtual simulation-pilot agent for training of air traffic controllers," *Aerospace*, vol. 10, no. 5, pp. 490, 2023.

[27] Konrad Hofbauer, Stefan Petrik, and Horst Hering, "The ATCOSIM corpus of non-prompted clean air traffic control speech," in *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, 2008, European Language Resources Association (ELRA).

[28] Luboš Šmídl, Jan Švec, Daniel Tihelka, Jindřich Matoušek, Jan Romportl, and Pavel Ircing, "Air traffic control communication (ATCC) speech corpora and their use for ASR and TTS development," *Language Resources and Evaluation*, vol. 53, no. 3, pp. 449–464, 2019.

[29] John Godfrey, "The Air Traffic Control Corpus (ATC0) - LDC94S14A," 1994.

[30] Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur, "Parallel training of dnns with natural gradient and parameter averaging," *arXiv preprint arXiv:1410.7455*, 2014.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[32] Yiming Wang, Tongfei Chen, Hainan Xu, Shuoyang Ding, Hang Lv, Yiwen Shao, Nanyun Peng, Lei Xie, Shinji Watanabe, and Sanjeev Khudanpur, "Espresso: A fast end-to-end neural speech recognition toolkit," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 136–143.

[33] Yiwen Shao, Yiming Wang, Daniel Povey, and Sanjeev Khudanpur, "Pychain: A fully parallelized pytorch implementation of lf-mmi for end-to-end asr," *arXiv preprint arXiv:2005.09824*, 2020.

[34] Srikanth Madikeri, Sibo Tong, Juan Zuluaga-Gomez, Apoorv Vyas, Petr Motlicek, and Hervé Bourlard, "Pkwrap: a pytorch package for lf-mmi training of acoustic models," *arXiv preprint arXiv:2010.03466*, 2020.

[35] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011, number CONF.

[36] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych, "Adapterfusion: Non-destructive task composition for transfer learning," *arXiv preprint arXiv:2005.00247*, 2020.

[37] Mark JF Gales, Kate M Knill, Anton Ragni, and Shakti P Rath, "Speech recognition and keyword spotting for low-resource languages: Babel project research at cued," in *Fourth International workshop on spoken language technologies for under-resourced languages (SLTU-2014)*. International Speech Communication Association (ISCA), 2014, pp. 16–23.