Liu *et al.*
EURASIP Journal on Audio, Speech, and Music Processing (2024) 2024:29
https://doi.org/10.1186/s13636-024-00349-3

EURASIP Journal on Audio, Speech, and Music Processing

EMPIRICAL RESEARCH **Open Access**

Exploration of Whisper fine-tuning strategies for low-resource ASR

Yunpeng Liu[1*], Xukui Yang[1] and Dan Qu[1]

Abstract

Limited data availability remains a significant challenge for Whisper's low-resource speech recognition performance, falling short of practical application requirements. While previous studies have successfully reduced the recognition error rates of target language speech through fine-tuning, a comprehensive exploration and analysis of Whisper's fine-tuning capabilities and the advantages and disadvantages of various fine-tuning strategies are still lacking. This paper aims to fill this gap by conducting comprehensive experimental exploration for Whisper's low-resource speech recognition performance using five fine-tuning strategies with limited supervised data from seven low-resource languages. The results and analysis demonstrate that all fine-tuning strategies explored in this paper significantly enhance Whisper's performance. However, different strategies vary in their suitability and practical effectiveness, highlighting the need for careful selection based on specific use cases and resources available.

Keywords Whisper, Fine-tune, Low-resource, ASR

1 Introduction

Automatic speech recognition (ASR) [1–3] refers to the use of machines to convert human speech into corresponding text. After more than half a century of rapid development, ASR technology has been widely applied in various fields of human social production and life, including economy, military, and culture [4–7]. Currently, ASR systems developed for international languages such as English have achieved human-level recognition capabilities, with robust, fast, and accurate performance. However, among the more than 7000 languages globally, the majority do not have sufficient training resources like the major languages such as English. According to statistics, about 40% of languages are facing extinction, with a user base of fewer than 1000 users [8–10]. It is the scarcity of the transcribed data for these low-resource language communities that prevents large neural networks from being substantially trained, leading to poor performance and a lack of real-world applications. Therefore, ASR for low-resource languages has gradually become a research focus.

With the technological wave of large language models such as GPT (Generative Pre-Training Transformer) [11, 12] and BERT (Bidirectional Encoder Representation from Transformers) [13], large models in speech domain have also ushered in a development opportunity. Recently, multilingual and multi-task large speech models have gradually become a popular paradigm for solving the problem of low-resource ASR. Among them, OpenAI's weakly supervised speech processing model Whisper can achieve multiple tasks such as ASR, speech translation, language identification, and speech activity detection for 100 languages simultaneously [14]. It has excellent evaluation results on many open-source datasets. Therefore, Whisper becomes the research object of this paper.

Despite its excellent capabilities, Whisper still falls short of meeting practical application requirements in low-resource ASR tasks, leaving considerable room for improvement. Currently, some studies use a small

*Correspondence:
Yunpeng Liu
lypspeech@163.com
[1] School of Information Systems Engineering, University of information Engineering, Science Avenue, Zhengzhou 450000, Henan, China

amount of supervised data to fine-tune Whisper and thereby improve its performance for target languages. Sicard et al. fine-tuned Whisper to reduce the recognition error rate of Swiss German dialects [15]. Liu et al. proposed a parameter-efficient fine-tuning method that can quickly adapt to child ASR tasks [16]. Xie et al. fine-tuned Whisper on a homemade dataset and effectively improved performance on mixed-language ASR tasks [17]. Waheed et al. established an ASR and dialect recognition system for various Arabic dialects by fine-tuning Whisper and some other models [18].

Although these studies have made effective explorations in theory and practice, there are still three issues that need further in-depth research regarding Whisper's low-resource ASR task. *First, to what extent fine-tuning can improve performance? Second, which part of the model has critical importance in fine-tuning? Third, what are the advantages and disadvantages of parameter-efficient fine-tuning methods?* These issues involve different fine-tuning strategies for Whisper, but they share commonalities and can be comprehensively compared and analyzed.

In view of this, this paper deeply explores various fine-tuning strategies for Whisper in low-resource ASR and conducts detailed analysis and discussion. Specifically, this paper uses a small amount of supervised data from seven low-resource languages and adopts three fine-tuning methods (vanilla fine-tuning, fine-tuning with specific parameters, and fine-tuning with additional modules) to experimentally analyze above issues. The results and analysis show that all fine-tuning strategies explored in this paper can significantly improve the performance of Whisper. The vanilla fine-tuning can greatly enhance the performance of target languages, specific parameter fine-tuning can further improve performance based on the vanilla fine-tuning, and additional module fine-tuning can effectively prevent catastrophic forgetting with negligible performance loss and achieve parameter-efficient fine-tuning.

The main contributions of this paper are as follows: (1) exploring the ability range that Whisper's fine-tuning strategies can achieve in low-resource ASR; (2) further exploring the inherent mechanism of Whisper's speech encoding capabilities by observing and analyzing the specific manifestations of different submodules in the model; (3) comparing and analyzing the advantages and disadvantages of different fine-tuning strategies subjectively and objectively. The related conclusions can be directly applied to future research and engineering practices.

Table 1 Architecture details of the Whisper model family

Model	Layers	Width	Heads	Parameters
Tiny	4	384	6	39M
Base	6	512	8	74M
Small	12	768	12	244M
Medium	24	1024	16	769M
Large	32	1280	20	1150M

2 Material and method

2.1 Whisper

Whisper, developed by OpenAI, is an advanced speech processing model capable of performing various tasks such as ASR, speech translation, language recognition, and speech activity detection in 100 languages.

In terms of data construction, the development team formed a weakly supervised dataset of 680,000 h of speech data through extensive collection, automation screening, and processing. They also conducted multi-task standardized annotation on the transcribed text. When it comes to model architecture, the team chose a multi-layer stacked Transformer with encoder-decoder structure as the basic network structure. Depending on the number of layers, the dimension of feature representation (width), and the number of attention heads, the model was divided into five versions: Tiny, Small, Base, Medium, and Large. Table 1 summarizes the specifics of each version. Additionally, on the basis of the large version, Large-v2 had 2.5 times more training iterations, while Large-v3 used data collection and processing as well as pseudo-labeling with Large-v2 to increase the training data to 5 million hours. Both the large-v2 and large-v3 models outperform the large model, and the large-v3 model demonstrates even stronger capabilities than the large-v2. In terms of model training, Whisper uses multi-task training to update and optimize model parameters, including recognition, English translation, speech activation detection, and language identification.

Whisper boasts superior multi-language ASR and translation capabilities. In some languages, its performance is even comparable to or better than that of humans. Currently, Whisper is becoming increasingly popular due to its advanced features and extensive applications.

2.2 Fine-tuning

Fine-tuning involves adjusting the model parameters to fit the hypothesis space of target task using a much smaller amount of data compared to the pretraining. It is typically used when the source and target domains are similar. There are numerous studies on using fine-tuning techniques to improve model performance. Well-known self-supervised speech models such as Wav2vec series [19, 20], Hubert [21], WavLM [22], and MMS (Massively Multilingual Speech) [23] require fine-tuning on domain-specific data to adapt to downstream tasks. Jain et al. explored different pretraining and fine-tuning methods for the Wav2vec 2.0 model on the ASR task for child speech [24]. Zhang et al. analyzed various combinations of pretraining and fine-tuning on 15 low-resource languages in the OpenASR21 challenge [25]. Pasad et al. used various metrics, including canonical correlation analysis, mutual information, word recognition, and word similarity, to study and analyze the characteristics of Wav2vec 2.0's layer representations and guide model improvements for better fine-tuning strategies [26].

However, there are three main challenges with fine-tuning. Firstly, due to the large parameter size of the models, which can be in the hundreds of millions or even billions, updating all parameters during fine-tuning can be computationally and time-consuming. Secondly, fitting a small number of data with large amounts of parameters may lead to overfitting, resulting in poor generalization performance. Finally, large models tend to have general capabilities for multiple tasks or languages, but after fine-tuning, they may only perform well on the target task or language, losing their general abilities, which is known as catastrophic forgetting. Both overfitting and catastrophic forgetting can degrade the generalization ability of a trained model, but the former is specific to the training and test data of a single language, while the latter concerns multiple languages.

To address these issues, numerous research efforts have been made to explore improved fine-tuning strategies. Rosin et al. used a partial parameter freezing strategy when adapting an ASR system to German and explored the impact of different freezing configurations on system performance [27]. Pasad et al. reinitialized the last 1–3 layers of the Wav2vec 2.0 model, achieving even better results than the pretrained model [26]. Kannan et al. introduced a bottleneck adapter into model, facilitating full adaptation to specific languages [28]. Yu et al. introduced LoRA (Low-Rank Adaptation) into ASR systems with decreased training times by factors between 5.4 and 3.6 while using only 0.08% of the parameters of the pretrained model [29].

3 Experiments and analysis

3.1 Data and baseline

We use seven languages in the Fleurs dataset [30] for our experiments: Afrikaans (Af), Belarusian (Be), Icelandic (Is), Kazakh (Kk), Marathi (Mr), Nepali (Ne), and Swahili (Sw). We have listed the areas affiliated with all experimental languages in Table 2. We believe that there is a relationship between areas and the basic phonological structures of languages, which is useful for our in-depth analysis of the interactions between multiple languages. Therefore, adopting languages from multiple areas can ensure the generalization of our experimental conclusions. Fleurs is a multi-parallel speech dataset where each language contains about 12 h of speech supervision. It can be used for various speech tasks such as ASR, speech translation, machine translation, and retrieval. The training set contains less than 10 h of supervision, and the speakers in the training set are different from those in the development and test sets. The data are sourced from Hugging Face[1], and we conducted training, validation, and testing using the train, validation, and test set of each language, respectively. Table 2 presents the detailed statistics of the data applied in this paper.

Table 2 The detailed statistics of the data applied in this paper. *h* means hour. Num_rows is the number of speech rows, and the numbers in "( )" represent (train: validation: test)

Language	Area	Duration (*h*)	Num_rows
Afrikaans	Sub-Saharan Africa	5.18	(1032:198:264)
Belarusian	Eastern Europe	15.20	(2433:408:927)
Icelandic	Western Europe	3.15	(926:36:46)
Kazakh	Central-Asia	17.16	(3200:369:856)
Marathi	South-Asia	17.37	(3269:443:1015)
Nepali	South-Asia	14.51	(3332:305:726)
Swahili	Sub-Saharan Africa	16.20	(3070:211:487)

Table 3 shows the performance of Whisper on seven languages [14], measured using the word error rate (WER) as the evaluation metric. Due to changes in the training data for Large-v3, many low-resource languages have more data, deviating from the low-resource trend. The focus of this paper is on comparing and analyzing fine-tuning strategies for Whisper, rather than on the application of lightweight models. Therefore, we selected Large-v2 as the baseline for our experiments and analysis, which we refer to as the pretrained model (PT). We believe that the relevant conclusions of this paper can be directly transferred to smaller models.

3.2 Vanilla fine-tuning

To explore the extent to which vanilla fine-tuning can improve the low-resource ASR performance of Whisper, we first fine-tuned the model using the seven language datasets mentioned above. We used the example[2]

[1] https://huggingface.co/datasets/google/fleurs

[2] https://huggingface.co/blog/fine-tune-whisper

Table 3 WERs (%) of 7 languages tested by OpenAI. The number after each language is the duration of training data in Whisper pretraining, and h means hour. These 7 languages have less than 20 h of data, which satisfies the low-resource feature

Model	Af(4.1 h)	Be(2.4 h)	Is(16 h)	Kk(12 h)	Mr(0.6 h)	Ne(0.6 h)	Sw(5.4 h)	Average
Tiny	91.2	94.0	113.3	165.2	100.3	101.8	100.9	109.53
Base	81.5	91.3	95.5	109.2	100.3	102.4	92.5	96.10
Small	61.1	75.1	72.6	70.3	60.2	69.5	73.7	68.93
Medium	44.9	60.4	49.9	48.8	63.2	54.4	52.8	53.49
Large	42.6	56.6	43.0	43.8	43.7	52.2	47.9	47.11
Large-v2	36.7	45.4	38.2	37.7	38.3	47.1	39.3	40.39
Large-v3	32.4	42.5	30.4	32.4	34.1	40.2	34.1	35.18

Table 4 WERs (%) of 7 languages tested before (PT) and after (FT) fine-tuning

Model	Af	Be	Is	Kk	Mr	Ne	Sw	Average
PT	36.7	45.4	38.2	37.7	38.3	47.1	39.3	40.39
FT	19.33	12.98	25.56	13.53	15.17	19.57	15.62	17.39

provided by Hugging Face as a reference for fine-tuning. Specifically, we used AdamW [31] for optimization and training with a total of 35 epochs. After warming up 10% of the total steps, the learning rate reached 0.00001 and then decayed linearly. This setting ensures sufficient training. During training, checkpoints were saved every 200 steps, and then, the performance of all checkpoints was evaluated using the test set. Finally, the checkpoint with the lowest WER was selected as the fine-tuned model. We used a single 48GB NVIDIA A40 GPU, and the batch size for all languages was 8.

Table 4 compares the test results before (PT) and after (FT) fine-tuning. It is evident that fine-tuning Whisper-large-v2 with insufficient 20 h of data can significantly reduce the error rate on that language. Specifically, compared with PT, FT reduced the average WER by 56.94%, relatively. This result indicates that Whisper's multi-language and multi-task pretraining not only enables it to have preliminary capabilities for ASR in many languages, but also updates the model parameters to a suitable initial high-dimensional space, enabling rapid adaptation to target tasks. Vanilla fine-tuning can further migrate the model parameters towards a target task-friendly direction even with limited data.

3.3 Fine-tuning with specific parameters

There are some related research summaries that multi-layer stacked encoder can gradually extract deep information of speech from bottom to top [25, 32]. The bottom-level representation often stores the most basic information of speech, such as pronunciation units, lip and tooth movements, while the high-level representation tends to express advanced features of speech content. Due to the similarity of human lip and tooth structures, the pronunciation methods are similar, and the differences between languages are not significant. Therefore, updating the bottom-level parameters of the encoder is redundant during model fine-tuning. Reducing unnecessary parameter updates during model fine-tuning has two benefits: first, the number of parameters to be updated is reduced, which can alleviate the consumption of computational power and time; second, it can effectively prevent overfitting when training large models with very limited data.

To further analyze the speech encoding capabilities of Whisper and determine the redundant areas of encoder parameter updates or the core areas that affect task performance in the model encoder, we used canonical correlation analysis (CKA) to visualize the representations of each layer of the encoder before and after fine-tuning. Specifically, we randomly sampled 100 utterances from each language dataset and used the CKA method in [26] to calculate the similarity of the representations of all layers of the encoder before and after fine-tuning.

Figure 1 visually demonstrates the similarity. It can be observed that for all languages, the change pattern of the representations of each layer of the encoder before and after fine-tuning is quite similar. Taking Kk (yellow line) as an example, the similarity of the representations of the first 15 layers of the encoder remains almost unchanged, with a value above 0.95. Starting from the 16th layer, the similarity gradually decreases, with smaller similarity at higher layers, and the downward trend becoming more evident. We believe that this pattern is consistent with

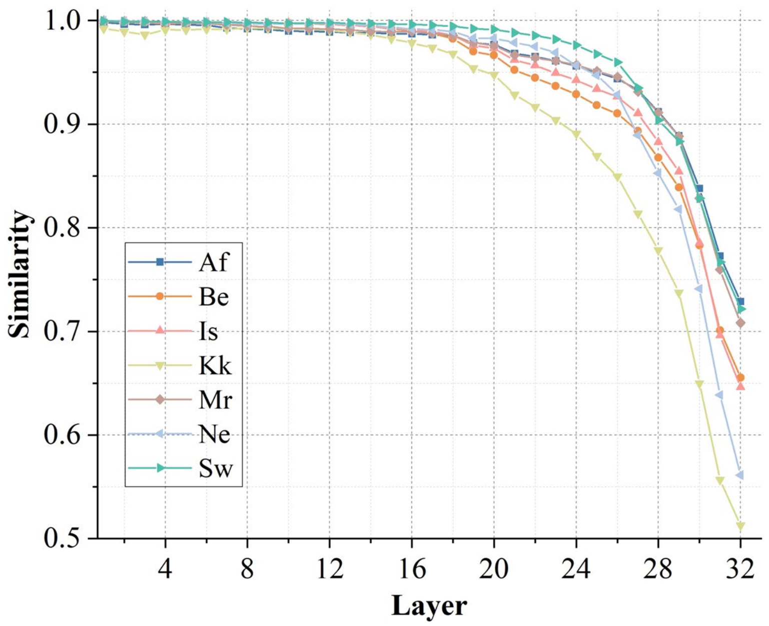

Fig. 1 Similarities of encoder layer-wise representations by CKA in 7 languages before and after fine-tuning

the conclusions of existing researches [25], which states that the bottom layers of the model capture shared features among languages, while the higher layers extract language-specific features. The intermediate layers serve as a smooth transition from shared features to language-specific features. It should be noted that Kk exhibits uniqueness in Fig. 1, where the similarity of its initial layers first decreases and then increases, and the curve for Kk does not fall within the set formed by the curves of the other six languages. We believe that the unique performance of Kk is related to its geographical origin. Among all seven languages, only Kk originates from Central Asia and belongs to the Turkic language family. Differences in language characteristics lead to variations in model performance.

To further confirm the validity of the above analysis, we made changes to the fine-tuning method of Whisper and analyzed it through objective results. During model parameter updates, we froze the bottom layers of the encoder, with the number of frozen layers being n. The other layers of the model were still updated during training, with n=10, 11,. . ., 31. All layers are updated when n = 32, which is fine-tuning. We do not have a comparison of n=1, 2, ..., 9, because operations like freezing lose their original meaning when the number of freezing layers is too small.

Figure 2 shows the results for each language under different numbers of frozen layers. It can be observed that first, as the number of frozen layers increases, the performance of the model on all languages consistently tends to worsen, indicating that the conclusion that the bottom and top layers of the encoder encode different information is correct; second, the best results for each language appear when $n \leq 20$, indicating that updating the bottom-level parameters during ordinary fine-tuning is redundant.

Table 5 shows the performance of fine-tuning with frozen encoder bottom layers (FFT) and vanilla fine-tuning, showing only the best results of FFT. By comparison, we found that for all languages, compared to FT, FFT achieved consistent WER reduction by 6.27%, relatively. Combined with the above results, it can be found that freezing the parameters of the lower layers of the encoder not only effectively reduces the number of parameters to be updated during fine-tuning, computational complexity, and time consumption, but also slightly improves performance.

In addition, according to the conclusions in [26], the possible reason for the large changes in the top layers of the encoder before and after fine-tuning is that the pre-training did not provide good initial parameters for the top layers, i.e., the initial state deviated from the target task. Reinitializing the top layers may make the initial parameters more suitable for the target task. To further discuss and analyze the speech encoding capabilities of Whisper, we implemented top-layer reinitialization fine-tuning. We reinitialized the top m layers while freezing the bottom (32 - m) layers, where m = 1, 2, . . ., 5. In Whisper, each layer of the encoder consists of two sublayers: a linear layer and a normalization layer. For the linear layer,

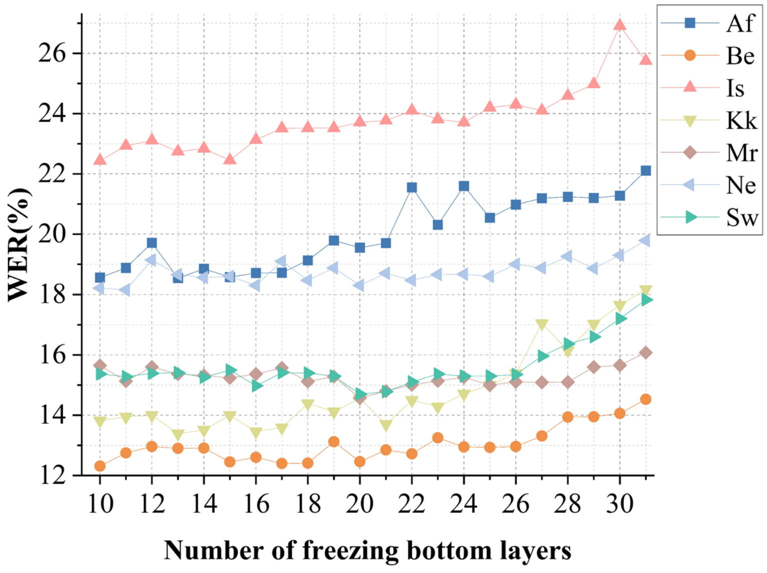

Fig. 2 WERs (%) for different number of freezing bottom layers per language

Table 5 WERs (%) of 7 languages tested by FT, bottom layers freezing fine-tuning (FFT) and top-layer reinitialization fine-tuning (RIFT). The number before "/" is the best WER (%) of all results per language. The number after "/" is the number of layer for freezing or reinitialization

Model	Af	Be	Is	Kk	Mr	Ne	Sw	Average
FT	19.33	12.98	25.56	13.53	15.17	19.57	15.62	17.39
FFT	18.54/13	12.31/10	22.44/15	13.39/13	14.57/20	18.15/11	14.70/20	16.30
RIFT	22.63/1	15.93/1	30.71/1	18.75/1	17.56/1	20.32/1	19.85/1	20.82/1

we initialize the weight parameters with Xavier Normalization [33] and initialize the bias parameters to 0. For the normalization layer, we initialize the weight parameters to 1 and the bias parameters to 0.

Figure 3 shows the results of reinitialization fine-tuning (RIFT) for each language, and Table 5 presents the best results across all reinitialization layers. It can be observed that, for any language, the results continuously worsen as the number of reinitialized layers increases. When m = 5, the ASR capability is completely lost, and the effect of only reinitializing the last layer of the encoder is the best. However, compared with RIFT and FT, we find that reinitialization does not improve the performance. This indicates that the reinitialization operation is ineffective, suggesting that Whisper's multi-language multi-task training has helped its ASR capability find a reasonable initialization parameter space, and randomly changing the model parameters will worsen its performance.

3.4 Fine-tuning with additional modules

The above fine-tuning methods have two main drawbacks: First, as shown in Table 1, the Large-v2 model has 1.15 B parameters. Even if the bottom layers of the encoder are frozen, the remaining number of parameters still exceeds 1 B (19.87 M per layer in the encoder). Therefore, both vanilla fine-tuning and fine-tuning with specific parameters require a large number of parameter updates, consuming a significant amount of computational power and time. Second, while adjusting and updating model parameters to optimize performance on the target task, it also necessarily leads to a decrease in the generalizability of the model on the original multi-task, which is known as catastrophic forgetting.

One effective way to address the above issues is to insert additional lightweight modules into the model [34]. During fine-tuning on the target language, only the parameters of the additional module are updated, while the original parameters of the model are frozen. This allows the updated parameters of the additional module to effectively transfer to the target language while preserving the original functionality of the model. Additionally, because the additional module generally has lightweight properties, compared with large models based on transformer as the basic network structure,

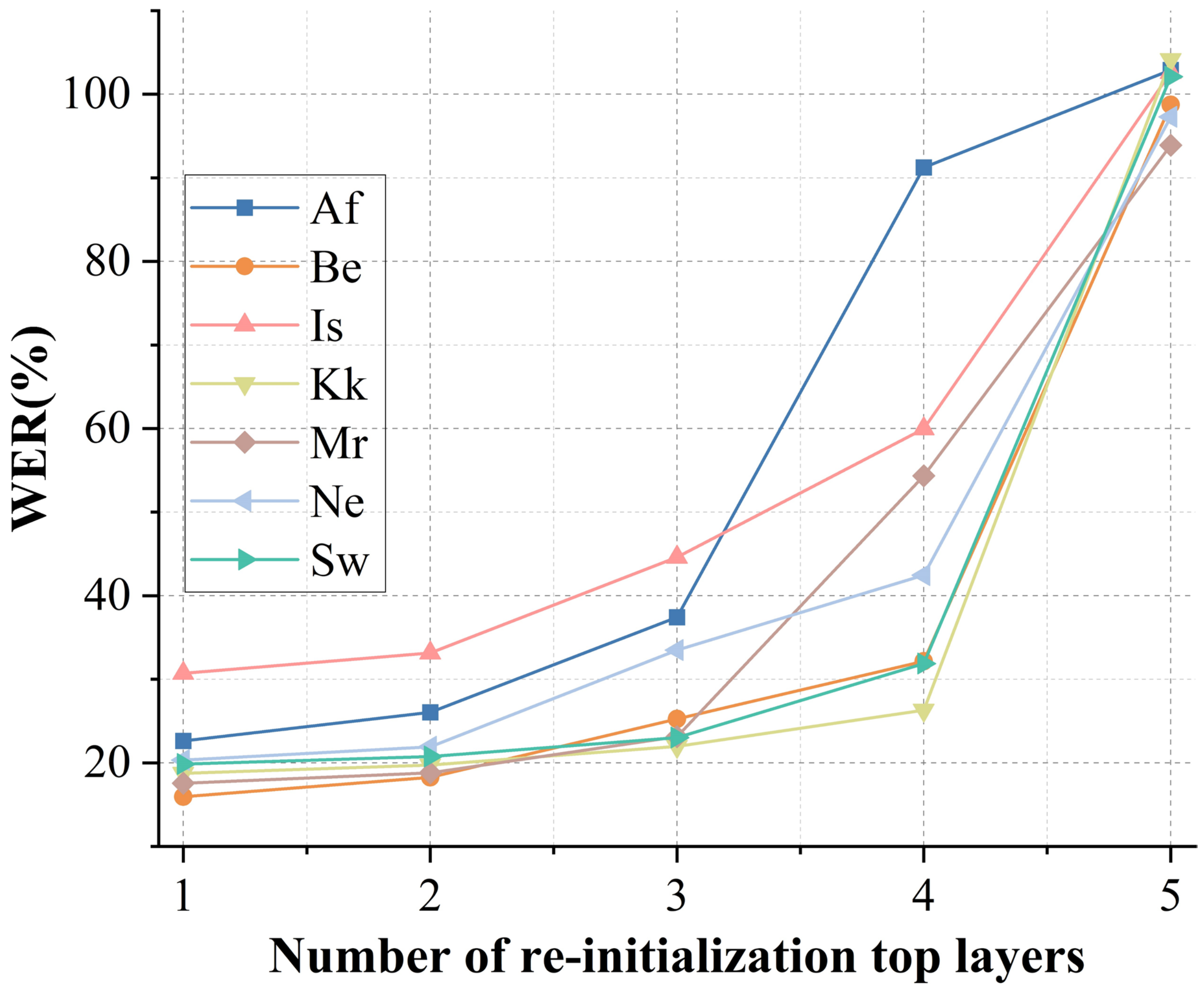

Fig. 3 WERs (%) for different number of re-initialization top layers per language

the increase in parameter quantity is very small, typically around 1% of the pre-trained model. This lightweight characteristic greatly reduces the number of parameters that can be updated during model fine-tuning, enabling efficient fine-tuning with parameters. *In order to explore the advantages and disadvantages of efficient parameter tuning methods*, in this paper, we conducted experiments and analysis on two classes of additional module: the bottleneck adapter (BA) [34] and LoRA [35].

For BA, we inserted it into every layer of the encoder and decoder to be the last module. Figure 4 shows the insertion method of BA and its internal structure. The input to the BA module first undergoes layer normalization (LN), then down-samples to a specific smaller dimension d (MLP), passes through a GLEU (Gaussian Error Linear Units) activation, and then up-samples back to the original dimension (MLP). The input and output maintain residual connections. During fine-tuning with additional BA modules, we used a learning rate of 0.0003, and otherwise followed the same procedures as regular fine-tuning. We first explored the impact of different values of d on the results (we assumed that language is independent of the value of d, so we only used Icelandic for experimental analysis), with $d = 32, 64, 128, 256$. Figure 5 shows the impact of BA's intermediate dimension, with $d = 64$ achieving the best results.

Then, with $d = 64$, we inserted BA after each layer of the encoder and decoder. Table 6 shows additional BA fine-tuning (BAFT) with PT and FT. Compared with FT, the average WER of BAFT increased by 1.63%. However, during the fine-tuning process, the number of parameters that were updated was approximately 0.8% of the original model, and the fine-tuning time was reduced by 5.2 times. This indicates that after inserting BA, although the performance of Whisper slightly decreased, the costs of computational resources and time during training were

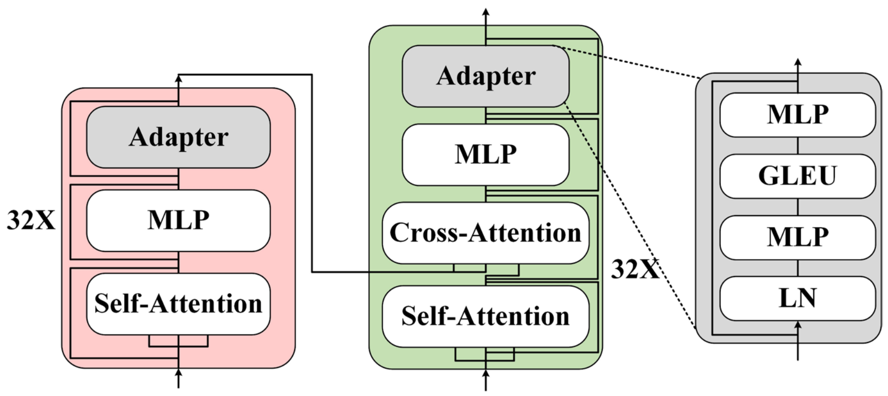

Fig. 4 Schematic diagram of inserting an adapter into Whisper

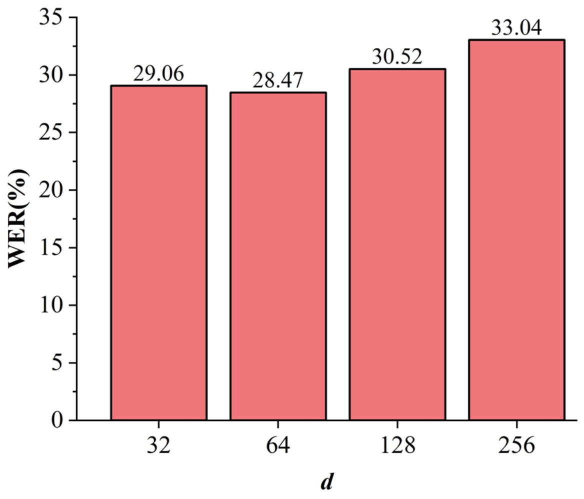

Fig. 5 WER (%) of different value of d for Icelandic

significantly reduced. Additionally, comparing BAFT and PT, it can be found that the average WER decreased by 52.91%, relatively. This suggests that fine-tuning with BA is still an effective fine-tuning strategy. It should be noted that the insertion of BA increases the model's parameter count, leading to longer inference times. Nevertheless, because of BA's lightweight nature, this increase in inference time is insignificant, and the inclusion of BA both minimizes the need for parameter updates and mitigates catastrophic forgetting, highlighting its significant practical benefits.

For fine-tuning with LoRA, we applied it to the *query* and *value* positions within the attention modules of each layer of the encoder and decoder, following the example[3] in PEFT [36]. The rank was set to 32, and alpha was set to 64. We used a single 80G NVIDIA A100 GPU with a batch size of 4, and the remaining hyperparameters were consistent with vanilla fine-tuning.

As shown in Table 6, the results of fine-tuning with LoRA (LoRAFT) are slightly inferior to those of BAFT. However, the number of parameters that need to be updated for LoRAFT is even smaller, only 0.67% of the original model. Compared with PT, the relative WER of LoRAFT can be reduced by 47.01%, indicating that this strategy is still effective. In addition, we have found that BAFT reduces training time by 5.2 times, and LoRAFT reduces training time by 5.6 times. Therefore, as concluded in [29], the overall performance of the model is significantly enhanced by introducing lightweight additional modules.

The above results and analysis demonstrate that both fine-tuning with BA and fine-tuning with LoRA can improve model performance while significantly reducing the amount and cost of parameter updates. However, when compared with vanilla fine-tuning, they perform slightly worse. Therefore, it is reasonable to choose an appropriate fine-tuning strategy based on practical conditions.

Finally, keeping BA inserted in all layers of the decoder, but we only insert it into the top p layers of the encoder, with $p = 1, \ldots, 10$. Figure 6 shows the results of BAFT inserted in the top layers of the encoder (BAFT-T). It can be observed that as the number of layers increases, the performance gradually improves. Table 6 provides the best results for each language and the corresponding number of top layers. It can be seen that the optimal number of top layers for inserting BA in the encoder differs among different target languages. However, for most languages (Af, Be, Kk, Ne), the best results are achieved when $p = 9$. There are also some languages that achieve the best results when $p = 8$ (Sw) or $p = 10$ (Mr), which are close to $p = 9$. Only Icelandic performs best when $p = 6$. We believe that the uniqueness of Icelandic's performance

[3] https://github.com/huggingface/peft/tree/main/examples/int8_training

Table 6 WERs (%) of 7 languages tested by PT, FT, BAFT, LoRAFT, and BAFT-T. The two columns, "Paras" and "Time," represent the number of trainable parameters and the average training time, respectively

Model	Paras	Time	Af	Be	Is	Kk	Mr	Ne	Sw	Average
PT	1.15B	/	36.7	45.4	38.2	37.7	38.3	47.1	39.3	40.39
FT	1.15B	14.6*h*	19.33	12.98	25.56	13.53	15.17	19.57	15.62	17.39
BAFT	9.2M	2.8*h*	21.13	13.81	28.47	15.68	16.54	20.81	16.69	19.02
LoRAFT	7.7M	2.6*h*	23.86	16.47	29.10	18.62	19.26	23.51	18.97	21.40
BAFT-T	/	/	21.94/9	14.04/9	26.14/6	15.59/9	16.58/10	19.77/9	16.52/8	18.65

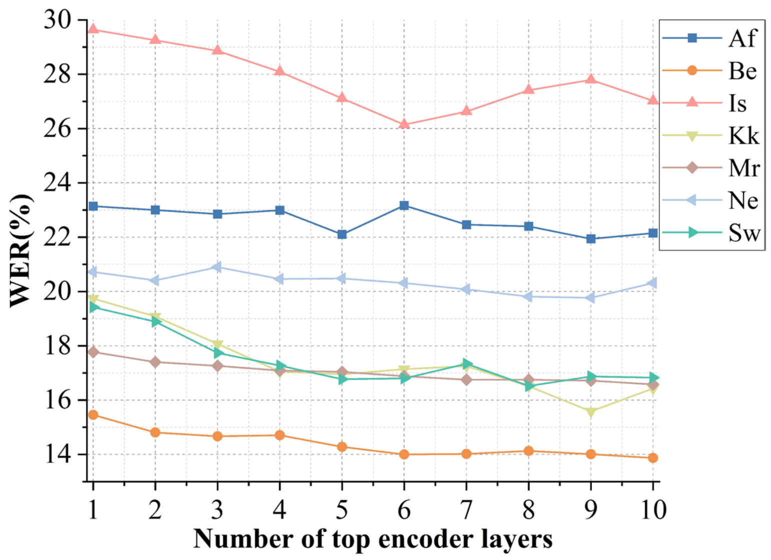

Fig. 6 WERs (%) of BAFT for only top encoder layers

is related to the duration of its training data. Specifically, among the seven languages, only Icelandic's training data is less than 3 h. The small amount of data leads to effective training of BA when $p = 6$, but the fit between data and model is not good when $p > 6$. Additionally, comparing BAFT-T with BAFT, it can be seen that inserting BA in the top layer of the encoder has a similar effect to inserting it in all layers, but the former can further reduce the number of parameters that need to be updated. This can be considered a more effective approach.

4 Conclusions

Multilingual and multitask large speech models are becoming a popular paradigm for solving low-resource speech recognition problems. By using a small amount of data for fine-tuning, the model's performance on the target task can be effectively improved. This paper starts from the current research's limitations, exploring the scope of performance improvement that fine-tuning can bring to Whisper through experiments and analysis of five fine-tuning strategies. We found that all fine-tuning strategies mentioned in this paper can effectively improve the performance of Whisper on the target language ASR. Among them, vanilla fine-tuning can greatly improve the ASR performance in the target language, fine-tuning with freezing the bottom layers has the strongest ability, while re-initializing the top layers is ineffective. Adding bottleneck adapters and LoRA fine-tuning can significantly reduce computational and time costs, while sacrificing only a small amount of speech recognition ability. The above conclusions can be applied in domain applications and engineering practices.

Comparing all the experimental results and analysis in this paper: Fine-tuning with additional adapter has the dual benefits of avoiding catastrophic forgetting and reducing training time, thus being the focus of our next research. Moreover, the similarity among languages can help further improve the model's performance in the target language, making multi-lingual data fine-tuning another promising avenue for exploration.

Acknowledgements
Not applicable.

Authors' contributions
YL was the main contributor to the implementation of the experiment and the writing of the manuscript, XY analyzed the experimental results, and DQ carefully proofread the manuscript. All authors read and approved the final manuscript.

Funding
This work was supported by the National Natural Science Foundation of China (No.62171470), Natural Science Foundation of Henan Province of China (No.232300421240), and Henan Zhongyuan Science and Technology Innovation Leading Talent Project (No.234200510019).

Availability of data and materials
The data generated and analyzed during the current study are not publicly available [due to privacy restrictions] but are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 2 January 2024 Accepted: 26 April 2024
Published online: 01 June 2024

References

1. A. Graves, A.R. Mohamed, G. Hinton, in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. Speech Recognition with Deep Recurrent Neural Networks (2013), pp. 6645–6649. https://doi.org/10.1109/ICASSP.2013.6638947
2. W. Chan, N. Jaitly, Q. Le, O. Vinyals, in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition (2016), pp. 4960–4964. https://doi.org/10.1109/ICASSP.2016.7472621
3. Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, S. Kumar, in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Transformer Transducer: A Streamable Speech Recognition Model with Transformer Encoders and RNN-T Loss (2020), pp. 7829–7833. https://doi.org/10.1109/ICASSP40776.2020.9053896
4. P. Karmakar, S.W. Teng, G. Lu, Thank you for attention: a survey on attention-based artificial neural networks for automatic speech recognition. CoRR **abs/2102.07259** (2021). https://arxiv.org/abs/2102.07259. Accessed 5 Apr 2023
5. Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. Laurent, Y. Bengio, A.C. Courville, Towards end-to-end speech recognition with deep convolutional neural networks. CoRR **abs/1701.02720** (2017). http://arxiv.org/abs/1701.02720. Accessed 1 Oct 2022
6. S. Alharbi, M. Alrazgan, A. Alrashed, T. Alnomasi, R. Almojel, R. Alharbi, S. Alharbi, S. Alturki, F. Alshehri, M. Almojil, Automatic speech recognition: Systematic literature review. IEEE Access **9**, 131858–131876 (2021). https://doi.org/10.1109/ACCESS.2021.3112535
7. J. Li. Recent advances in end-to-end automatic speech recognition (2022). http://arxiv.org/abs/2111.01690
8. L. Miao, J. Wu, P. Behre, S. Chang, S. Parthasarathy, in *2022 Ninth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. Multilingual transformer language model for speech recognition in low-resource languages (2022), pp. 1–5. https://doi.org/10.1109/SNAMS58071.2022.10062774
9. D.N. Krishna, Multilingual speech recognition for low-resource indian languages using multi-task conformer. CoRR **abs/2109.03969** (2021). https://arxiv.org/abs/2109.03969. Accessed 8 Dec 2023
10. H. Yadav, S. Sitaram. A survey of multilingual models for automatic speech recognition (2022). https://arxiv.org/abs/2202.12576. Accessed 8 Dec 2023
11. R. Alec, N. Karthik, S. Tim, S. Ilya, Improving language understanding by generative pre-training (2018). https://www.mikecaptain.com/resources/pdf/GPT-1.pdf. Accessed 8 Dec 2023
12. A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners (2019). https://api.semanticscholar.org/CorpusID:160025533. Accessed 8 Dec 2023
13. J. Devlin, M.W. Chang, K. Lee, K. Toutanova, in *North American Chapter of the Association for Computational Linguistics*. Bert: Pre-training of deep bidirectional transformers for language understanding (2019). https://api.semanticscholar.org/CorpusID:52967399. Accessed 8 Dec 2023
14. A. Radford, J.W. Kim, T. Xu, G. Brockman, C. Mcleavey, I. Sutskever, in *Proceedings of the 40th International Conference on Machine Learning*, *Proceedings of Machine Learning Research*. Robust speech recognition via large-scale weak supervision, vol. 202 (2023), pp. 28492–28518. https://proceedings.mlr.press/v202/radford23a.html. Accessed 8 Dec 2023
15. C. Sicard, K. Pyszkowski, V. Gillioz, Spaiche: Extending state-of-the-art ASR models to Swiss German dialects. ArXiv **abs/2304.11075** (2023). https://api.semanticscholar.org/CorpusID:258291445. Accessed 8 Dec 2023
16. W.M. Liu, Y. Qin, Z. Peng, T. Lee, Sparsely shared LoRA on whisper for child speech recognition. ArXiv **abs/2309.11756** (2023). https://api.semanticscholar.org/CorpusID:262084086. Accessed 8 Dec 2023
17. P. Xie, X. Liu, Z. Chen, K. Chen, Y. Wang. Whisper-mce: Whisper model finetuned for better performance with mixed languages (2023). https://arxiv.org/abs/2310.17953v1. Accessed 8 Dec 2023
18. W. Abdul, T. Bashar, S. Peter, E. AbdelRahim, A.M. Muhammad, in *ARABICNLP*. VoxArabica: A robust dialect-aware Arabic speech recognition system (2023). https://api.semanticscholar.org/CorpusID:264172944. Accessed 8 Dec 2023
19. S. Schneider, A. Baevski, R. Collobert, M. Auli. wav2vec: Unsupervised pre-training for speech recognition (2019). https://arxiv.org/abs/1904.05862. Accessed 8 Dec 2023
20. A. Baevski, H. Zhou, A. Mohamed, M. Auli, wav2vec 2.0: A framework for self-supervised learning of speech representations. CoRR **abs/2006.11477** (2020). https://arxiv.org/abs/2006.11477. Accessed 8 Dec 2023
21. W. Hsu, B. Bolte, Y.H. Tsai, K. Lakhotia, R. Salakhutdinov, A. Mohamed, Hubert: Self-supervised speech representation learning by masked prediction of hidden units. CoRR **abs/2106.07447** (2021). https://arxiv.org/abs/2106.07447. Accessed 8 Dec 2023
22. S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao, J. Wu, L. Zhou, S. Ren, Y. Qian, Y. Qian, J. Wu, M. Zeng, F. Wei, WavLM: Large-scale self-supervised pre-training for full stack speech processing. CoRR **abs/2110.13900** (2021). https://arxiv.org/abs/2110.13900. Accessed 8 Dec 2023
23. V. Pratap, A. Tjandra, B. Shi, P. Tomasello, A. Babu, S. Kundu, A. Elkahky, Z. Ni, A. Vyas, M. Fazel-Zarandi, A. Baevski, Y. Adi, X. Zhang, W.N. Hsu, A. Conneau, M. Auli. Scaling speech technology to 1,000+ languages (2023). http://arxiv.org/abs/2305.13516. Accessed 8 Dec 2023
24. R. Jain, A. Barcovschi, M.Y. Yiwere, D. Bigioi, P. Corcoran, H. Cucu, A wav2vec2-based experimental study on self-supervised learning methods to improve child speech recognition. IEEE Access **11**, 46938–46948 (2023). https://doi.org/10.1109/ACCESS.2023.3275106
25. J. Zhao, W.Q. Zhang, Improving automatic speech recognition performance for low-resource languages with self-supervised models. IEEE J. Sel. Top. Signal Process. **16**(6), 1227–1241 (2022). https://doi.org/10.1109/JSTSP.2022.3184480
26. A. Pasad, J. Chou, K. Livescu, Layer-wise analysis of a self-supervised speech representation model. CoRR **abs/2107.04734** (2021). https://arxiv.org/abs/2107.04734. Accessed 8 Dec 2023
27. T. Pekarek-Rosin, S. Wermter, Replay to remember: Continual layer-specific fine-tuning for german speech recognition. ArXiv **abs/2307.07280** (2023). https://api.semanticscholar.org/CorpusID:259924527. Accessed 8 Dec 2023
28. A. Kannan, A. Datta, T.N. Sainath, E. Weinstein, B. Ramabhadran, Y. Wu, A. Bapna, Z. Chen, S. Lee, Large-scale multilingual speech recognition with a streaming end-to-end model. ArXiv **abs/1909.05330** (2019). https://api.semanticscholar.org/CorpusID:202565562. Accessed 8 Dec 2023
29. Y. Yu, C.H.H. Yang, J. Kolehmainen, P.G. Shivakumar, Y. Gu, S. Ryu, R. Ren, Q. Luo, A. Gourav, I.F. Chen, Y.C. Liu, T. Dinh, A. Gandhe, D. Filimonov, S. Ghosh, A. Stolcke, A. Rastrow, I. Bulyko, Low-rank adaptation of large language model rescoring for parameter-efficient speech recognition.

ArXiv **abs/2309.15223** (2023). https://api.semanticscholar.org/CorpusID:263152679. Accessed 8 Dec 2023

30. A. Conneau, et al., in Proceeding of 2022 IEEE Spoken Language Technology Workshop (SLT), FLEURS: FEW-Shot Learning Evaluation of Universal Representations of Speech (Doha, Qatar, 2022), pp. 798–805. https://doi.org/10.1109/SLT54892.2023.10023141
31. I. Loshchilov, F. Hutter, Fixing weight decay regularization in adam. CoRR **abs/1711.05101** (2017). http://arxiv.org/abs/1711.05101. Accessed 26 Jan 2023
32. K. Peterson, A. Tong, Y. Yu, in *Proc. Interspeech 2022*. OpenASR21: The Second Open Challenge for Automatic Speech Recognition of Low-Resource Languages (2022), pp. 4895–4899. https://doi.org/10.21437/Interspeech.2022-10972
33. X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks. J. Mach. Learn. Res. Proc. Track **9**, 249–256 (2010)
34. N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, S. Gelly, Parameter-efficient transfer learning for NLP. CoRR **abs/1902.00751** (2019). http://arxiv.org/abs/1902.00751. Accessed 26 Jan 2023
35. E.J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, W. Chen, LoRA: Low-rank adaptation of large language models. CoRR **abs/2106.09685** (2021). https://arxiv.org/abs/2106.09685. Accessed 26 Jan 2023
36. S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, B. Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. (2022). https://github.com/huggingface/peft. Accessed 26 Jan 2023