Interspeech 2024
1-5 September 2024, Kos, Greece

AdaRA: Adaptive Rank Allocation of Residual Adapters for Speech Foundation Model

Zhouyuan Huo, Dongseong Hwang, Gan Song, Khe Chai Sim, Weiran Wang
Google LLC, USA
zhhuo@google.com

Abstract

A recent paradigm shift in artificial intelligence has witnessed the emergence of foundation models. These foundation models, possessing billions of parameters and trained on extensive datasets, are anticipated to demonstrate superior generalization across diverse downstream tasks. Residual Adapters represent a broadly employed methodology for efficient adaptation, achieved by updating a limited set of additive parameters while maintaining a fixed bottleneck dimension. However, when the parameter budget is constrained, allocating additive parameters uniformly across layers proves sub-optimal. In this paper, we propose a novel adaptive efficient adaptation method that automatically determines the optimal number of bottleneck dimensions for Residual Adapters at different layers. Experimental results confirm that the proposed method effectively learns an optimal additive parameter allocation, surpassing the performance of comparable methods in speech recognition domain adaptation.

Index Terms: Speech Recognition, Speech Foundation Model, Parameter Efficient Fine-Tuning

1. Introduction

Foundation models [1] (also known as large language models, LLMs) are trained on massive unsupervised datasets and can be fine-tuned for various tasks. They have achieved impressive results in natural language processing (NLP) tasks [2, 3, 4, 5]. The speech community is exploring self-supervised pre-training of foundation models on massive unlabeled speech data, showing promising improvements in speech recognition [6, 7]. Two main categories of self-supervised learning algorithms are used: first, reconstruction-based methods: These directly predict the input feature, like Auto-regressive Predictive Coding [8] and Multi-Predictive Coding [9]. Second, BERT-style methods: They bridge the gap between continuous speech signal and discrete text tokens, including Wav2vec2.0 [10], HuBERT [11] and BEST-RQ [12]. After pre-training, the foundation model is fine-tuned on supervised data for downstream tasks. For example, in speech recognition, the encoder is initialized from a pre-trained foundation model and fine-tuned on the supervised data of the target domain. This fine-tuned model can then recognize speech in that domain.

Researchers have explored ways to reduce the number of trainable parameters during fine-tuning, enabling efficient adaptation to numerous tasks. BitFit [13] method focuses on updating only the bias terms of the model, ignoring the larger weight matrices, leading to sparse fine-tuning. Hierarchical Feature Fusion (HFF) [14, 15] proposes a compute- and parameter-efficient method by treating the foundation model as a frozen feature extractor and fuses features from multiple intermediate layers of the foundation model. Adapter [16], which are small, trainable neural networks inserted between layers of a frozen pre-trained model, allows targeted adaptation without modifying the base network. Other low-rank matrix approximation of adapters [17, 18] further reduce the parameter count by finding compact representations of the adapter weight matrices with minimal accuracy compromise. These parameter-efficient methods achieve reasonable performance on various tasks while significantly reducing the number of parameters that need to be trained. This is crucial for efficient adaptation to many tasks with limited data.

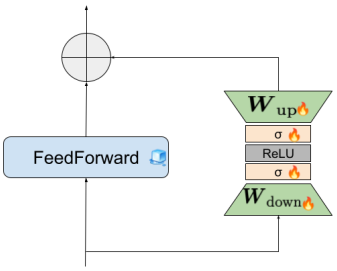

Figure 1: *Adaptive rank allocations (AdaRA) of parallel Residual Adapters. Removing σ makes it a standard Residual Adapter.*

In Residua Adapter, the bottleneck dimension is a predetermined hyper-parameter allocating an equal amount of memory at each layer. Houlsby et al. [16] established a single bottleneck dimension value across all layers. Chen et al. [19] explored design patterns within parameter-efficient methodologies, concluding that uniform allocation of trainable parameters per layer is suboptimal. However, their study was limited to five group patterns (e.g., increasing and decreasing), with bottleneck dimensions requiring manual configuration prior to training. Furthermore, multiple studies [20, 10, 21] have demonstrated that different layers in a pre-trained speech foundation model encode distinct levels of information. Consequently, the ideal additional parameter count for adaptation would vary by layer. This highlights the critical need for a novel method capable of automatically determining optimal bottleneck dimensions for parameter-efficient fine-tuning.

In this paper, we propose a novel adaptive efficient adaptation method that can automatically learn the required number of bottleneck dimensions for Adapters at different layers. We demonstrate the benefit of the proposed method on adapt-

10.21437/Interspeech.2024-45

ing pre-trained foundation models to the target domain. After visualizing the sparsity process of the residual adapters and comparing the proposed method with other parameter-efficient fine-tuning methods, the results show that our method performs better than other methods with a similar amount of additional parameters.

2. Methods

In this section, we initially review the Residual Adapter and subsequently propose a novel adaptive rank allocation method for parameter-efficient domain adaptation. Additionally, we provide a detailed description of a proximal operator employed to solve the new loss function within the algorithm.

2.1. Residual Adapter

Residual Adapter represents a widely used adaptation methodology in various fields, prized for their straightforward implementation and adaptability [22]. In general, Residual Adapter computes an additive adaptation vector by subjecting their input to a sequence of typically two or more transformations. Within the standard setting [16], Residual Adapters are positioned following each Transformer block. They receive input features generated by the previous backbone block. The first layer within the adapter serves to project the input feature down to a vector of reduced dimension while introducing non-linearity. Subsequently, the second layer projects this low-dimensional activation back to the original input dimension. The adapter concludes by adding this resultant adaptation vector to its original input.

Concretely, the first layer of the adapter takes in the input $h_i \in \mathbf{R}^d$ extracted from the i^{th} backbone block and projects it to a lower dimensional hidden activation $u_i \in \mathbf{R}^{bottleneck}$ as following:

$$u_i = ReLU(W_{down} h_i) \tag{1}$$

where $W_{down} \in \mathbf{R}^{bottleneck \times d}$ is the trainable parameter. The second layer then calculates a final adapted feature representation $\hat{h}_i \in \mathbf{R}^d$ from the low dimensional activation u_i by first applying an up-projection transformation and then adding the resulting high dimensional vector to the original input h_i. This computation can be written as follows:

$$\hat{h}_i = W_{up} u_i + h_i \tag{2}$$

where the weight matrix $W_{up} \in \mathbf{R}^{d \times bottleneck}$ is trainable. The adapted feature representation $\hat{h}_i$ is used in the forward computation of the backbone model instead of h_i. We omit the biasing term in the above linear layer for simplicity.

The weight matrices $W_{down} \in \mathbf{R}^{bottleneck \times d}$ and $W_{up} \in \mathbf{R}^{d \times bottleneck}$ compose the trainable parameters for the Residual Adapter. Given T backbone blocks in the foundation model, the total number of trainable parameters is $(2 \times bottleneck \times d) \times T$ and is determined by the bottleneck dimension. The Residual Adapter usually sets the bottleneck dimension size to a value smaller than the input dimension ($bottleneck \ll d$) for parameter efficiency, creating a bottleneck at its hidden layer.

Residual Adapter provides flexibility in the positioning of the adapter module within the backbone model, allowing for customization based on the specific task and architecture. In the context of speech Conformer models, the optimal configuration for WER performance is achieved by placing adapter module in parallel with each Feed-Forward Network (FFN) within the individual Conformer blocks [23]. Consistent with this finding, our experiments employ a similar setup, positioning adapters in parallel with the final FFN of each block.

Algorithm 1 Adaptive Rank Allocation of Residual Adapter

Initialize: $W_{up,l} := \mathbf{0}^{d \times d}$, $\sigma_l := \mathbf{1}^d$, $W_{down,l} \sim \mathcal{N}(0,1)$ at all layers $l \in \{1, 2, ..., L\}$;
Stage I: Update $\{W_{up,l}, W_{down,l}, \sigma_l\}_{l=1:L}$ by minimizing Equation (4) and get the expected parameter efficiency through Equation (5);
Stage II: Fix $\{\sigma_l\}_{l=1:L}$ at all Residual Adapters and update $\{W_{up,l}, W_{down,l}\}_{l=1:L}$ only.

2.2. Adaptive Rank Allocation Method

When the total amount of the additional parameters in the Residual Adapter is fixed, it is obviously sub-optimal that all adapters share the same number of bottleneck dimension. In this paper, we propose a novel adaptive rank allocation method (AdaRA) that can learn the bottleneck dimension for each adapter automatically. To achieve this, we change the dimension of trainable weights W_{down} and W_{up} to $\mathbf{R}^{d \times d}$ in the training stage and introduce an additional trainable vector $\sigma \in \mathbf{R}^d$ and its corresponding diagonal matrix $\text{diag}(\sigma) \in \mathbf{R}^{d \times d}$ at each adapter. All the non-diagonal elements in $\text{diag}(\sigma)$ are $0s$. Thus, the Equation (1) and (2) can be re-written as following:

$$\begin{array}{rcl} u_i & = & ReLU(\hat{W}_{down} h_i) \\ \hat{h}_i & = & \hat{W}_{up} u_i + h_i \\ \hat{W}_{down} & = & \text{diag}(\sigma) \times W_{down} \\ \hat{W}_{up} & = & W_{up} \times \text{diag}(\sigma) \end{array} \tag{3}$$

When the vector σ is sparse, it means that the corresponding rows in W_{down} and columns in W_{up} are zeroed out. Through learning a set of $\{\sigma_1, \sigma_2, ..., \sigma_L\}$ with different sparsities, we can learn L different bottleneck dimensions $\{\sigma_l d\}_{l=1}^L$ adaptatively. To one extreme, we can remove the adapter at layer l when $\sigma_l = 0$. To enforce sparsity in σ, we add the ℓ_1 regularization of σ to the original loss function f:

$$\min_{\{W_{up,l}, W_{down,l}, \sigma_l\}_{l=1:L}} f(\{W_{up,l}, W_{down,l}\}_{l=1:L}) + \lambda \sum_{l=1}^{L} |\sigma_l| \tag{4}$$

To solve the new loss (4), we propose to use proximal gradient method [24] which applies a proximal operator on σ after gradient descent updates. The ℓ_1-norm proximal operator is as following:

$$\sigma = \text{sign}(\sigma) \cdot \max\{|\sigma| - \lambda, 0\} \tag{5}$$

The proposed Adaptive Rank Allocation method (AdaRA) can be described in two stages as in Algorithm 1. To avoid Residual Adapters affecting the base model in the beginning, we initialize W_{up} to be a matrix of 0. σ is initialized to be a vector of 1 to make sure that all parameters of Residual Adapters share the same significance. After stage I, the bottleneck dimension of the adapter at layer l is determined by the number of non-zero elements in σ_l.

3. Experimental Setup

In the paper, we study the parameter efficiency of the proposed method on the domain adaptation task of the speech foundation model.

Table 1: *Comparing parameter-efficient methods on the domain adaptation of speech recognition task. ↓ denotes the smaller the better. The (+x% ↓) represents the relative regression compared to Fine-Tune all parameters.*

Fine-Tuning Method	**# Trainable Encoder Parameters ↓**	**VS WER (%) ↓**	**VS Contact WER (%) ↓**
Fine-Tune All	1.82 B	4.4	15.7
BitFit	1.4 M	5.8 (+31.8%)	18.1 (+15.3%)
Residual Adapter (bottleneck=128)	12.7 M	5.0 (+13.6%)	16.7 (+6.4%)
Residual Adapter (Magnitude Pruned)	12.7 M	5.1 (+15.9%)	16.9 (+7.6%)
LoRA (rank=16)	20.5 M	4.9 (+11.4%)	16.6 (+5.7%)
AdaRA Residual Adapter	12.7 M	**4.6 (+4.5%)**	**16.1 (+2.5%)**
AdaRA Residual Adapter	6.3 M	4.9 (+11.4%)	16.5 (+5.1%)

3.1. Foundation Model And Task

Our adaptation employs a Universal Speech Model (USM) as its foundation [25]. This model utilizes a Connectionist Temporal Classification (CTC) loss [26] and a ConformerXL encoder architecture [6]. The encoder contains 32 conformer blocks and a total parameter count of approximately 2 billion. To pretrain the encoder, we selected BEST-RQ [12], a self-supervised learning method that demonstrates superior performance. The decoder comprises a fully-connected layer and a Softmax layer with $4,096$ output units representing the word-piece units. In the speech recognition domain adaptation task, we apply parameter-efficient methods in the encoder and fine-tune the trainable parameters in the encoder and the CTC decoder.

3.2. Training Data

The USM model leverages extensive pre-training on YouTube data spanning various sub-domains [27] through the BEST-RQ method [12]. It is then adapted for American English short-form speech using approximately $490k$ hours of voice search (VS) data. This adaptation dataset is primarily transcribed with the guidance of a teacher model [28].

We evaluate word error rate (WER) of compared methods on two American English Voice Search test sets: **VS** and **VS Contact**. The inclusion of contact names in the latter enables us to evaluate the adaptation methods' capacity to generalize and accurately recognize infrequent words.

4. Results

In this section we present our experimental study on adapting pre-trained foundation models to the target Voice Search domain. The goal of this study is to show how σ achieves parameter-efficiency adaptively in the Residual Adapters and demonstrate better performance than other compared parameter-efficient fine-tuning methods.

4.1. Comparison of Different Parameter-Efficient Methods

To validate the proposed Adaptive Rank Allocation (AdaRA) method of Residual Adapter, we compare it to several related parameter-efficient fine-tuning algorithms. Specifically, we compare with three representative and strong parameter-efficient methods: BitFit [29], Residual Adapter with fixed bottleneck dimension [16] and LoRA [18]. BitFit [29] updates the bias term of the foundation model only. For Residual Adapters, the bottleneck dimension is set to be 128 across all layers. We apply LoRA to every 2D matrix in the Conformer FeedForward Network modules with rank=16. For each method, we update the trainable encoder parameters and the CTC decoder with 6.3M parameters for $200k$ steps. For AdaRA method, we set $\lambda = 3e^{-5}$ and run the stage I in Algorithm 1 after required sparsities are achieved, 12.7M and 6.3M respecitvely in the paper. After that, we update W_{up} and W_{down}, while fixing σ. Experimental results in Table 1 show that the proposed AdaRA Residual Adapter method achieves the best performance when there is the same amount of trainable encoder parameters (12.7M), which reduces the WER regression of parameter-efficient fine-tuning method to within 5% on both **VS** and **VS Contact** test-sets. AdaRA Resdiual Adapter also achieve comparable performance as other parameter-efficient methods when there is only half of the number of trainable encoder parameters (6.3M).

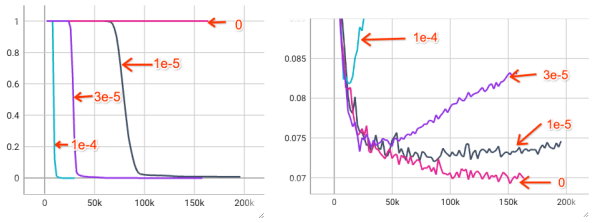

Figure 2: *Left: Effect on non-zero rates of σ at layer 0 when $\lambda \in \{0, 1e^{-5}, 3e^{-5}, 1e^{-4}\}$. x axis denotes the training steps and y axis denotes the non-zero rate of σ. Right: **VS** WER when $\lambda \in \{0, 1e^{-5}, 3e^{-5}, 1e^{-4}\}$. x axis denotes the training steps and y axis denotes **VS** WER.*

4.2. Effect of ℓ_1-norm Hyper-Parameter λ

Hyper-parameter λ is to control strength of ℓ_1-norm regularization on the normal loss function. In this subsection, we visualize how the value of λ affect the sparsity of σ in the left of Figure 2 and how WER changes when we vary the value of λ in the right of Figure 2. In the left of Figure 2, we can observe that the non-zero rates of σ at the first layer converges faster towards 0 when we increase the value of λ. When $\lambda = 0$, the non-zero rates are always 1, which means that all parameters in the Residual Adapter are updated and cannot be pruned. In the right of Figure 2, we can see that the WER goes down in the beginning

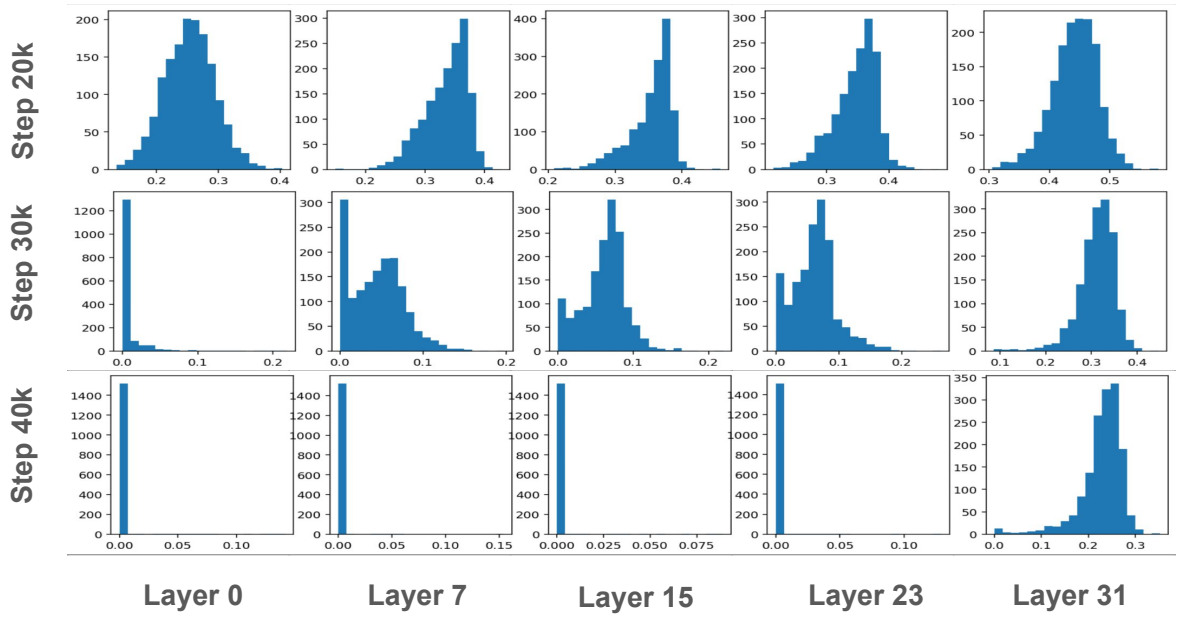

Figure 3: *Histogram of σ from layers $\{0, 7, 15, 23, 31\}$ at steps $\{20k, 30k, 40k\}$. In each subfigure, x axis denotes the values of the parameter, and the y axis denotes the frequency.*

and goes up after a point when $\lambda \neq 0$. It is because that after a sparsity level, further squeezing of the amount of the trainable parameters starts to affect the capacity of the model and degrades the performance. It is obvious that the change point of $\lambda = 1e^{-4}$ appears more earlier than others because it enforces a stronger sparsity regularization.

4.3. Behavior of σ During Training

Figure 3 visualizes the progress of learning a sparse σ at 5 layers from the bottom to the top. At step $20k$, the elements in σ present normal distributions at all layers and most of the values are non-zeros. At step $30k$, it is obvious that nearly all the elements in σ at layer 0 become 0. At layer 7, 15, and 23, we can also see that most of the values are moving towards 0 and corresponding σ are getting sparser. At step $40k$, σ at all layers become very sparse except layer 31. The non-zero rates of σ at all layers from 0 to 31 are presented in Figure 4. There is a significant difference in the sparsity of σ between top 4 layers and all other layers. It is easy to see that the non-zero rates of σ from layer 0 to layer 27 are almost 0. Therefore, given desired sparsity, an optimal allocations of parameters at different layers are learned through our algorithm.

4.4. Compare with Magnitude Filtering

We also compare the proposed AdaRA method with filtering by the magnitude of the parameters in σ after $100k$ steps. After filtering, both methods are fine-tuned until $200k$ steps. Results in Table 1 show that our proposed method (4.6%) outperforms the magnitude pruned method (5.1%) significantly when the number of trainable encoder parameters is 12.7M. And it is even worse than AdaRA method when we have only half of the number of trainable encoder parameters 6.3M. Figure 4 shows

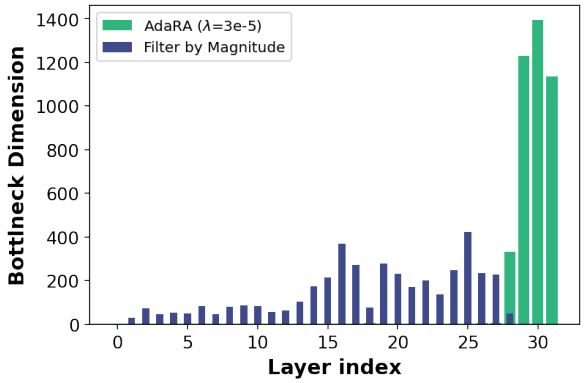

Figure 4: *Comparison of non-zero rates of σ at all layers to get residual adapter with additional 12.7M parameters.*

the parameter allocations of these two methods when the total amount of parameters is 12.7M. It is obvious that the proposed AdaRA method can learn a better allocation.

5. Conclusion

In this paper, we propose a novel adaptive rank allocation of Residual Adapters for speech foundation model, which can learn the bottleneck dimension across all layers automatically. After visualizing the sparsity process of the residual adapters and comparing with other parameter-efficient fine-tuning method, it is obvious that our method outperforms other methods given a similar amount of additional parameters.

6. References

[1] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[3] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8748–8763.

[4] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer." *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.

[5] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[6] Y. Zhang, D. S. Park, W. Han, J. Qin, A. Gulati, J. Shor, A. Jansen, Y. Xu, Y. Huang, S. Wang *et al.*, "Bigssl: Exploring the frontier of large-scale semi-supervised learning for automatic speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, 2022.

[7] D. Hwang, A. Misra, Z. Huo, N. Siddhartha, S. Garg, D. Qiu, K. C. Sim, T. Strohman, F. Beaufays, and Y. He, "Large-scale asr domain adaptation using self-and semi-supervised learning," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6627–6631.

[8] Y.-A. Chung and J. Glass, "Generative pre-training for speech with autoregressive predictive coding," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3497–3501.

[9] W. Wang, Q. Tang, and K. Livescu, "Unsupervised pre-training of bidirectional speech encoders via masked reconstruction," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6889–6893.

[10] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in neural information processing systems*, vol. 33, pp. 12 449–12 460, 2020.

[11] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[12] C.-C. Chiu, J. Qin, Y. Zhang, J. Yu, and Y. Wu, "Self-supervised learning with random-projection quantizer for speech recognition," in *International Conference on Machine Learning*. PMLR, 2022, pp. 3915–3924.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[14] Z. Huo, K. C. Sim, B. Li, D. Hwang, T. N. Sainath, and T. Strohman, "Resource-efficient transfer learning from speech foundation model using hierarchical feature fusion," *arXiv preprint arXiv:2211.02712*, 2022.

[15] Z. Huo, K. C. Sim, D. Hwang, T. Munkhdalai, T. Sainath, and P. Moreno, "Re-investigating the efficient transfer learning of speech foundation model using feature fusion methods."

[16] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2790–2799.

[17] R. Karimi Mahabadi, J. Henderson, and S. Ruder, "Compacter: Efficient low-rank hypercomplex adapter layers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 1022–1035, 2021.

[18] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[19] J. Chen, A. Zhang, X. Shi, M. Li, A. Smola, and D. Yang, "Parameter-efficient fine-tuning design spaces," *arXiv preprint arXiv:2301.01821*, 2023.

[20] A. Pasad, J.-C. Chou, and K. Livescu, "Layer-wise analysis of a self-supervised speech representation model," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2021, pp. 914–921.

[21] A. Arunkumar, V. N. Sukhadia, and S. Umesh, "Investigation of ensemble features of self-supervised pretrained models for automatic speech recognition," *arXiv preprint arXiv:2206.05518*, 2022.

[22] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, "Learning multiple visual domains with residual adapters," *Advances in neural information processing systems*, vol. 30, 2017.

[23] Q. Li, B. Li, D. Hwang, T. N. Sainath, and P. M. Mengibar, "Modular domain adaptation for conformer-based streaming asr," *arXiv preprint arXiv:2305.13408*, 2023.

[24] N. Parikh, S. Boyd *et al.*, "Proximal algorithms," *Foundations and trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[25] Y. Zhang *et al.*, "Google USM: Scaling automatic speech recognition beyond 100 languages," *arXiv preprint arXiv:2303.01037*, 2023.

[26] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[27] A. Narayanan, A. Misra, K. C. Sim, G. Pundak, A. Tripathi, M. Elfeky, P. Haghani, T. Strohman, and M. Bacchiani, "Toward domain-invariant speech recognition via large scale training," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 441–447.

[28] D. Hwang, K. C. Sim, Z. Huo, and T. Strohman, "Pseudo label is better than human label," *arXiv preprint arXiv:2203.12668*, 2022.

[29] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," *arXiv preprint arXiv:2106.10199*, 2021.