Interspeech 2024
1-5 September 2024, Kos, Greece

Adapter pre-training for improved speech recognition in unseen domains using low resource adapter tuning of self-supervised models

Sathvik Udupa[1], *Jesuraj Bandekar*[1], *Saurabh Kumar*[1], *Deekshitha G*[1], *Sandhya B*[1], *Abhayjeet S*[1], *Savitha Murthy*[1], *Priyanka Pai*[2], *Srinivasa Raghavan*[2], *Raoul Nanavati*[2], *Prasanta Kumar Ghosh*[1]

[1]Electrical Engineering Department, Indian Institute of Science (IISc), Bangalore-560012, India
[2]Navana Tech (NT), Mumbai-400001, India

sathvikudupa66@gmail.com, prasantag@iisc.ac.in

Abstract

Adapter tuning is an approach to fine-tune large neural network models on new tasks. These methods can be used to efficiently fine-tune large self-supervised learning (SSL) models for speech recognition tasks. In this work, we aim to perform improved low-resource adaptation of SSL features from source to target domain. Toward this, we experiment with adapter pre-training for Wav2Vec2-based models over different source and target configurations. We experiment over 3 datasets consisting 14 languages, including very low-resource languages. Further, we show the consistency of this method across different adapter dimensions and analyse the feature transformation due to the adapter pre-training process. With the proposed methods, we obtain over 10%-30% relative improvement in WER and CER with Viterbi decoding in 13 languages. Further, we obtain consistent performance gains using LM decoding on many of these languages.

Index Terms: speech recognition, adapters, low-resource

1. Introduction

Recently, advances in deep learning architectures have led to a lot of progress in Automatic Speech Recognition (ASR). The use of transformers [1] and conformers [2] over Recurrent Neural Networks (RNN) and Long Short-Term Memory networks (LSTM) has been a major milestone in sequence-to-sequence tasks including ASR. Additionally, the use of various loss functions like Connectionist Temporal Classification (CTC) [3], attention-based encoder-decoder with transformers and Recurrent Neural Network Transducers (RNN-T) [4] have shown significant improvement in Word Error rate (WER). Furthermore, a combination of these losses to train a single model in a multi-loss setup has also proven to be advantageous [5].

A prominent direction that has been adopted in the domain of ASR research is pre-training models using Self Supervised Learning (SSL) methods and then fine-tuning for ASR tasks [6, 7, 8, 9, 10]. These models are generally pre-trained using a large amount of un-transcribed data and then fine-tuned with a smaller amount of transcribed data. This has been shown to provide better performance, compared to ASR with only supervised training. Additionally, researchers have been working on different techniques to improve SSL-based pre-training and fine-tuning results [11, 12, 13, 14].

With SSL models, significant strides have been made in architectural innovations. However, the final fine-tuned ASR performance remains a function of pre-training data. Previous research [15] has demonstrated that fine-tuning using languages present during pre-training yields superior performance compared to fine-tuning with languages that were not encountered during pre-training. Pre-training large SSL models use considerable compute resources - XLSR [7] was pre-trained with 128-200 GPUs, MMS [8] was pre-trained with 48-64 A100 GPUs and HuBERT [9] was pre-trained using 32-256 GPUs. This is a major bottleneck for researchers across the world, and this has led to limited options in open-sourced pre-trained SSL models. Some of the available Wav2Vec2 [6] based models are as follows (we indicate the pre-training languages in paranthesis): Wav2Vec2 (English), XLSR (53 languages), MMS (1000+ languages), VP-100k [16] (23 European languages), IndicWav2Vec [17] (40 Indian languages) and CLSRIL-23 [18] (23 Indian languages).

While MMS [8] may seem like an ideal SSL model as it covers a very large number of languages, it still has higher concentrations of few languages, which can harm the performance of lower-resource languages. For example, a Wav2Vec2-based model trained on Indian languages [14] can outperform MMS-300M on Indian language test sets[1]. One approach towards improving fine-tuning Wav2Vec2 models can be to continue pre-training with the target labelled data, and then fine-tune the model with the same data [6]. This has been shown to improve performance in many languages [6, 19]. This, again can be a bottleneck for many researchers due to computational load. Recent models have also followed the trend of scaling up to a large number of model parameters such as MMS-1b [8]. Consequently, the process of pre-training or fine-tuning these models becomes another formidable obstacle in itself. In response to this issue, adapters have been proposed in recent years to address this problem of fine-tuning large models.

Adapters are extra parameters injected in between layers of a transformer model, which are trained during the adaptation while freezing the rest of the parameters of the model. Adapter layers consist of feed-forward layers, with down-projection of features [20]. Due to this configuration, a large model can be adapter tuned using much lesser parameters, compared to full-fine-tuning. The use of adapters has shown benefits in Machine Translation [21, 22], transfer learning [23, 24], language modelling [25, 26, 27] and computer vision tasks [28].

Adapters have also been used in speech. In speech synthesis, the use of adapters has been shown to improve speaker adaptability [29, 30]. In speech recognition, adapters have been used for domain adaptation [31], accent adaptation and speaker adaptation [32], low-resource language adaptation and multi-lingual ASR models [33, 34]. Adapters are also beneficial in avoiding catastrophic forgetting [35]; authors have used adapters to train different tasks using the same ASR model [36].

In this work, we aim to utilise adapter pre-training to improve the performance of low-resource adapter tuning. There has been a work with adapter pre-training for Wav2Vec2 based

[1]https://asr.iitm.ac.in/models

10.21437/Interspeech.2024-1587

ASR [37]. The authors propose building a continual Wav2Vec2 model by starting with a pre-trained Wav2Vec2 model and continuing pre-training with only adapters in two languages. After adapter-pre-training, the authors propose adding another set of adapters and fine-tuning the model for the ASR task with the 2nd adapters.

Our work differs from the previous work in the following ways. The previous work is motivated towards continual learning with Wav2Vec2 and faster pre-training on new languages using adapters. Our main motivation is to reduce the domain mismatch between the Wav2Vec2 model trained on source dataset $\mathcal{S}$ to the target ASR fine-tuning dataset $\mathcal{T}$. We aim to adapt pre-trained Wav2Vec2$_{\mathcal{S}}$ $\rightarrow$ Wav2Vec2$_{\mathcal{T}}$ using adapters. This process can effectively handle the two problems that were highlighted for low-resource SSL fine-tuning - 1) lack of target language data in pre-training leads to degradation in ASR performance, and 2) performing full pre-training has large computation overhead. Thus, we experiment with different pre-trained Wav2Vec2 models, and target datasets to understand the gap between $\mathcal{S}$ and $\mathcal{T}$. Unlike previous work, which has compared fine-tuning on two languages, we show the performance across three datasets and 14 languages. We obtain 10%-30% and 3%-18% relative improvement in WER and CER with Viterbi and language model (LM) decoding, respectively. Thus, the contributions of our work are as follows:

- We identify the significance of adapter pre-training based domain adaptation on source Wav2Vec2$_{\mathcal{S}}$ model to target datasets of different domains $\mathcal{T}$
- We investigate the benefits of adapter pre-training for low resource fine-tuning while transferring between similar domains ($\hat{\mathcal{T}} \rightarrow \mathcal{T}$) from pre-trained in-domain Wav2Vec2$_{\hat{\mathcal{T}}}$
- We quantify and visualise the changes in feature representation after Wav2Vec2$_{\mathcal{S}}$ $\rightarrow$ Wav2Vec2$_{\mathcal{T}}$ based domain adaptation

2. Dataset

In this work, we use 3 different datasets to test the domain adaptation capabilities of adapter-pre-training. They are summarized below -

- **RESPIN:** We use data from the *RESPIN*[2] corpus containing dialectal speech data in 9 *Indian* languages - *Bhojpuri* (bh), *Bengali* (bn), *Hindi* (hi), *Magahi* (mg), *Maithili* (mt), *Telugu* (te), *Chhattisgarhi* (ch), *Kannada* (kn) and *Marathi* (mr). Four of these languages - *Bhojpuri*, *Magahi*, *Maithili* and *Chhattisgarhi* are very low resource *Indian* languages. Further, all languages consist of many dialects. Thus, many of these languages and dialects may not be available for SSL pre-training. We create 10 hour dialect-balanced low-resource train splits on each of these languages. We also use dialect-balanced train and test splits, with the dev set ranging from 1-2 hours across languages and the test set ranging from 4-8 hours across languages. We denote this dataset as *RESPIN*.
- **Fleurs:** To test the performance on languages across the world, we use 4 different languages from *Fleurs* [38] corpus. We choose *Yong* (yo_ng) - a Southwestern *Tai* language of *Thailand*, *Afrikaans* (af_za) - a language spoken in *South Africa* and other *African* nations, *Armenian* (hy_am) - an *Indo-European* language spoken in *Armenia* and *Welsh* (cy_gb), a language spoken in *Great Britain*. *Afrikaans* has 2.8 hours of training data, while the other three languages have around 10 hours. The dev set varies from 0.5 - 1.7 hours and the test set duration varies from 0.75 to 3 hours.
- **LibriLight:** To test the performance on *English* (en) as target dataset $\mathcal{T}$, we use the 10hr split from *LibriLight* [39] corpus. We use *LibriSpeech* [40] dev_other as the dev set and report results on *LibriSpeech* test sets, which we denote as en-clean and en-other.

[2] https://respin.iisc.ac.in/

3. Proposed Methodology

In this section, we will first describe the model components, and then talk about our low-resource adaptation formulation.

3.1. Wav2Vec2

Wav2vec 2.0 [6] is an SSL model that can be trained on large amounts of non-transcribed speech data. It takes raw speech waveform $\mathcal{X}$ as input and processes it through Convolutional Neural Network (CNN) layers, which learns feature representation $\mathcal{Z}$ and down samples the features. Some frames are randomly masked and then passed on to a transformer [1] encoder to obtain features $\mathcal{C}$. Each layer of the transformer encoder consists of self-attention operation and feed-forward networks. The input features $\mathcal{Z}$ are further quantised to $\mathcal{Q}$. The network is then trained using contrastive loss using $\mathcal{C}$ and $\mathcal{Q}$. Thus, this training procedure over large amounts of speech results in rich feature representation. After pre-training, a dense layer can be added and the model can be fine-tuned with an ASR objective, such as CTC. In this work, we denote this model as Wav2Vec2.

3.2. Adapters

Adapters are small neural networks inserted after the feed-forward networks in a transformer layer [20]. We use Layer Norm, a dense layer to down-project the input features of **D** dimension to adapter features of **K** dimension, followed by a ReLU activation. This is passed to another dense layer which projects back the **K** features to output **D** dimensional features. Unlike [37], we use a single adapter per layer and do not observe the need for further changes to the transformer encoder.

3.3. Domain adaption of pre-trained Wav2Vec2 features

Our goal is to convert the pre-trained model trained on source domain $\mathcal{S}$ to target dataset domain $\mathcal{T}$. Here, the change in domain corresponds to difference in the type of data in terms of languages, dialects, recording conditions etc. We propose performing this process with a pre-trained Wav2Vec2$_{\mathcal{S}}$ model using adapter pre-training. We freeze all parameters of pre-trained Wav2Vec2$_{\mathcal{S}}$ model and add randomly initialised adapters at every layer. We then continue to pre-train this model on the low-resource target supervised dataset $\mathcal{T}$. At the end of this adaptation process, we expect the Wav2Vec2$_{\mathcal{T}}$ model to output features more representative of target data $\mathcal{T}$.

We then proceed with adapter tuning the adapted model Wav2Vec2$_{\mathcal{T}}$ on the target dataset on the ASR objective. Again, we freeze all Wav2Vec2 layers and unfreeze only the pre-trained adapter layers. We also add a dense layer at the end for the CTC objective. Thus, we fine-tune for ASR objective using adapter and dense layers only. Unlike [37], we do not add additional adapter layers, rather we continue to fine-tune the same adapters that were pre-trained. We denote our approach as pre-trained adapter tuning (PAT). We consider 3 different configurations as follows:

1. **Wav2Vec2$_{\mathcal{S}\rightarrow\mathcal{T}}$:** In this configuration, we choose *English* as source $\mathcal{S}$ and other datasets - *RESPIN* and *Fleurs* as target $\mathcal{T}$. Thus, we use Wav2Vec2$_{\mathcal{S}}$ model trained on the *LibriSpeech* dataset as the source SSL model. We then perform PAT with each language from *RESPIN* and *Fleurs* separately. We denote this adaptation process as Wav2Vec2$_{\mathcal{S}\rightarrow\mathcal{T}}$.

2. **IndicWav2Vec2$_{\hat{\mathcal{T}}\rightarrow\mathcal{T}}$:** In this configuration, we choose *Indian* languages as the target $\mathcal{T}$. We start with SSL model trained on *Indian* languages - IndicWav2Vec2$_{\hat{\mathcal{T}}}$ [17] and tune it on the *RESPIN* dataset. IndicWav2Vec2 pre-training may have data similar to *RESPIN*, but it has not used *RESPIN* samples for pre-training. Thus, we differentiate *RESPIN* and IndicWav2Vec2 as $\mathcal{T}$ and $\hat{\mathcal{T}}$ domains, respectively. We denote this adaptation process as IndicWav2Vec2$_{\hat{\mathcal{T}}\rightarrow\mathcal{T}}$.
3. **IndicWav2Vec2$_{\mathcal{S}\rightarrow\mathcal{T}}$:** In this configuration, we choose *English* languages as the target $\mathcal{T}$ and SSL pre-trained on Indian languages as source - IndicWav2Vec2$_{\mathcal{S}}$. Thus, we use PAT to tune the model to adapt it to *English*. We denote this adaptation process as IndicWav2Vec2$_{\mathcal{S}\rightarrow\mathcal{T}}$.

4. Experimental setup

We train all models using Fairseq [41]. We use the Wav2Vec2-base model with 94.4M parameter for all experiments. We perform all experiments - pre-training and fine-tuning with a **single 24GB GPU**. For full-fine-tuning (FFT), we fine-tune all 94.4M parameters, without using any adapters. We run experiments with an adapter dimension of 64. While performing adapter tuning (AT), we freeze the Wav2Vec2 model and fine-tune only 1.2M parameters from adapters. While pre-training adapters, we train with 1.4M parameters - this includes the adapters as well as the projection layer for Wav2Vec2 Contrastive loss. When we fine-tune the pre-trained adapter Wav2Vec2, we again fine-tune only on the 1.2M parameters from adapters. Additionally, we also show the performance across different adapter dimensions - 32, 64, 96 and 128 for one language.

We perform adapter pre-training for $15,000$ steps, with a gradient accumulation of 8. We perform all fine-tuning for $50,000$ steps, with a gradient accumulation of 4. We have performed fine-tuning for a large number of steps to ensure that all models have converged to their optimal performance. We use early stopping with a patience of 5 on the dev set loss. We decode all models with and without language model (LM). We use Viterbi decoding to infer without a language model. While decoding with a language model, we use KenLM [3] model trained only on train sentences. We will be releasing all codes and config files for the sake of reproducibility and open science here[4].

4.1. Feature analysis

To understand the feature transformation with the PAT approach for SSL domain adaptation, we use layer-wise analysis [42]. We compute Canonical correlation analysis (CCA) between the features from all transformer layers and one-hot phone representation. To perform this, we extract phoneme timestamps from force-alignment and extract the alignments for English (39 phones) and Hindi (53 phones). We use a CCA implementation [5] to compute the correlation on each layer from the pre-trained Wav2Vec2$_{\mathcal{S}}$ model and adapter-pre-trained Wav2Vec2$_{\mathcal{S}\rightarrow\mathcal{T}}$ model.

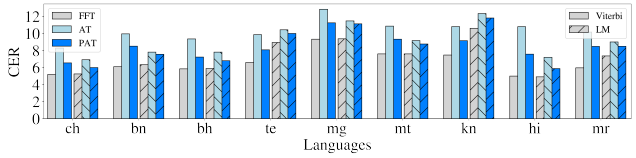
5. Results and Discussion

In this section, we begin with the results of domain mismatched fine-tuning on *RESPIN* and *Fleurs*, followed by training across similar domains and adapter dimension ablations. Finally, we present the CCA analysis on 2 languages.

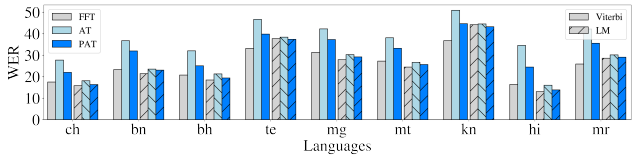
[3] https://github.com/kpu/kenlm

[4] https://github.com/bloodraven66/AdapterPretrainASR

[5] https://github.com/ankitapasad/layerwise-analysis



(a) *CER*



(b) *WER*

Figure 1: *Performance of RESPIN dataset on 9 languages across three models - FFT, AT and PAT. We show the WER and CER for two types of decoding - Viterbi and LM decoding.*

5.1. Wav2Vec2$_{\mathcal{S}\rightarrow\mathcal{T}}$ on RESPIN

Figure 1 shows the performance of full fine-tuning (FFT), adapter tuning (AT) (Section 4) and pre-trained adapter tuning (PAT) (Section 3.3) on 9 languages from *RESPIN* low-resource dataset. We have two main observations - FFT always obtains the best CER and WER, and the proposed PAT has better performance than AT. We find that using LM sometimes harms the performance, in languages such as *Telugu* (te), *Kannada* (kn) and *Marathi* (mr). We attribute this to low resource language modelling which can have out-of-vocabulary (OOV) issues. In Table 1, we present the relative improvements in CER and WER in PAT models over AT models. We observe significant performance improvements in Viterbi decoding across all languages in CER and WER. With LM, we have more than 5% relative improvement in CER for *Chhattisgarhi* (ch), *Bhojpuri* (bh), *Hindi* (hi) and *Marathi* (mr). Similarly in WER, we obtain more than 5% relative improvement in *Chhattisgarhi* (ch), *Bhojpuri* (bh) and *Hindi* (hi). We perform t-test between file level WER and CER between PAT and AT results and find that the p-value is less than 0.05 for all languages with Viterbi decoding. Additionally, with LM decoding, the statistical test for WERs for 7 languages and CERs for 5 languages have p-value less than 0.05. Thus, we can summarise that pre-training adapters can significantly lower WER and CER when there is a domain mismatch between the source pre-trained Wav2Vec2$_{\mathcal{S}}$ and target fine-tuning dataset $\mathcal{T}$.

Table 1: *Relative improvement on CER and WER using PAT over AT. We find significant improvements on domain mismatched training with 9 languages of RESPIN*

language	CER (relative %)		WER (relative %)	
	viterbi	LM	viterbi	LM
ch	21.01	13.31	20.81	9.52
bn	14.57	3.33	13.09	2.04
bh	22.86	13.06	21.78	8.83
te	18.05	4.22	14.67	2.58
mg	12.46	2.97	11.72	3.44
mt	14.18	4.37	12.79	3.79
kn	15.28	4.22	12.37	2.90
hi	29.91	18.36	29.20	13.54
mr	16.55	5.56	15.67	3.62

5.2. Wav2Vec2$_{\mathcal{S}\rightarrow\mathcal{T}}$ on Fleurs

Here, we refer to Table 2, where we show the mismatched fine-tuning results on 4 Fleurs languages. Compared to AT, we find that PAT obtains over 12% relative improvement in CER

Table 2: *Performance of AT and PAT on domain mismatched training for 4 Fleurs languages*

Languages	Viterbi				LM			
	AT		PAT		AT		PAT	
	CER	WER	CER	WER	CER	WER	CER	WER
af_za	19.34	57.16	**18.11**	**52.33**	21.65	45.68	**20.44**	**43.08**
cy_gb	14.27	48.91	**12.47**	**41.83**	17.95	34.86	**17.43**	**33.66**
hy_am	9.13	45.45	**8.03**	**39.99**	15.32	42.2	**15.16**	**41.35**
yo_ng	27.00	70.86	**24.62**	**65.41**	**27.9**	56.47	29.24	**55.62**

and WER with Viterbi decoding for *Welsh* (cy_gb) and *Armenian* (hy_am), while *Yong* (yo_ng) and *Afrikaans* (af_za) obtains around 6%-8% improvement. We find LM to degrade the performance of *Armenian* and *Yong*, while we observe minor improvements for *Welsh* and *Afrikaans*. This further validates that pre-trained adapter tuning is necessary to improve performance on adaptation with low-resource languages when there is a domain mismatch between the source model and target dataset.

5.3. IndicWav2Vec2$_{\hat{\mathcal{T}}\rightarrow\mathcal{T}}$ on RESPIN

In this case, we will consider the first three rows from Table 3. This represents adaptation from the similar domain, where we use a model pre-trained on *Indian* languages - IndicWav2Vec2$_{\hat{\mathcal{T}}}$ to continue to pre-train adapters on *Bengali* (bn) *Chhattisgarhi* (ch) and *Telugu* (te). In this case, for *Chhattisgarhi* we find around 11% relative improvement with PAT, whereas we see 3%-7% relative improvement on other languages when compared to AT. With LM decoding, we observe 1% -3% relative improvement across languages. Further, if we compare the results for these 3 languages from Table 1, we can observe that all 3 languages have better performance with IndicWav2Vec when compared to Wav2Vec2. Thus, from these observations, we can conclude that - 1) PAT across similar domains can still work, though results in minor improvement. 2) PAT with IndicWav2Vec2$_{\hat{\mathcal{T}}\rightarrow\mathcal{T}}$ has better performance than Wav2Vec2$_{\hat{\mathcal{S}}\rightarrow\mathcal{T}}$. This shows that more pre-training in the target domain will generally be beneficial. 3) Further, PAT with Wav2Vec2$_{\mathcal{S}\rightarrow\mathcal{T}}$ on *Chhattisgarhi* has better performance with LM decoding, than AT on IndicWav2Vec2$_{\hat{\mathcal{T}}}$. This suggests that performing PAT on a domain mismatched source can be beneficial over AT on a similar-domain source.

5.4. IndicWav2Vec2$_{\mathcal{S}\rightarrow\mathcal{T}}$ on Librilight

For this configuration, we will consider the last two rows from Table 3. This represents the mismatch case training where we have an IndicWav2Vec2$_{\mathcal{S}}$ pre-trained on *Indian* languages and fine-tuned on target *English* dataset of *Librilight*. Using PAT, we find 3%-7% relative improvement in CER and WER with Viterbi decoding. Whereas, we find 1%-4% relative improvement with LM decoding. With the domain mismatched training for English, we do not see as much improvement as shown in *RESPIN* or *Fleurs*. Nevertheless, we observe that. we still lower the WER and CER with PAT, when compared to AT.

Table 3: *AT and PAT on similar domain training (first 3 rows) and domain mismatch training (last 2 rows).*

languages	Viterbi				LM			
	AT		PAT		AT		PAT	
	CER	WER	CER	WER	CER	WER	CER	WER
bn	7.20	27.86	**6.87**	**26.56**	6.79	22.00	**6.72**	**21.59**
ch	6.30	21.69	**5.56**	**19.18**	5.96	16.57	**5.68**	**15.84**
te	7.36	37.11	**6.87**	**34.73**	9.65	36.89	**9.47**	**36.41**
en-clean	11.59	37.03	**10.81**	**34.25**	11.66	26.04	**11.30**	**25.01**
en-other	20.02	52.33	**19.27**	**49.64**	20.56	39.10	**20.20**	**37.82**

Table 4: *Performance of AT and PAT on domain mismatched training across different adapter dimensions for Telugu*

adapter dim	Viterbi				LM			
	AT		PAT		AT		PAT	
	CER	WER	CER	WER	CER	WER	CER	WER
32	11.46	52.26	**9.64**	**45.80**	11.18	39.7	**10.55**	**38.40**
64	9.86	46.62	**8.08**	**39.78**	10.43	38.36	**9.99**	**37.37**
96	9.24	44.07	**7.52**	**37.15**	10.17	38.12	**9.58**	**36.90**
128	8.81	42.53	**7.32**	**36.44**	10.07	38.15	**9.51**	**37.07**

5.5. Adapter dimension

In this section, we show Wav2Vec2$_{\mathcal{S}\rightarrow\mathcal{T}}$ for different adapter configurations on *Telugu* (te) language from *RESPIN*. We find that the WER and CER gradually lower with an increase in adapter dimension, though adapter dim 128 results in higher WER with LM decoding. Additionally, we find that, for all configurations, PAT observes significant relative improvements with Viterbi decoding and 2%-5% improvement with LM decoding. Thus, we can see the PAT method being generalisable across different parameter sizes of adapters.

5.6. CCA analysis

In this section, we summarise the analysis from Figure 2. We use the layer-wise feature analysis as described in Section 4.1 and compare features from each model and aligned phone representation. We can observe the changes in correlation measures for Wav2Vec2$_{\mathcal{S}\rightarrow\mathcal{T}}$ for *Hindi* (first plot) and IndicWav2Vec2$_{\hat{\mathcal{T}}\rightarrow\mathcal{T}}$ for English (second plot). We find that, in both cases, after PAT, the features are better correlated with the target language phonemes. This shows the domain adaptation capability of the PAT approach.

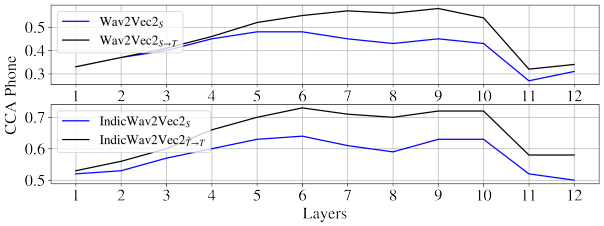


Figure 2: *CCA analysis on Wav2Vec2$_{\mathcal{S}}$ and IndicWav2Vec2$_{\hat{\mathcal{T}}}$*

6. Conclusions

This work presents the advantages of using adapter pre-training for efficient domain adaptation in Wav2Vec2-based ASR. We find that adapter pre-training can outperform standard adapter tuning, for adaptation in cases of domain mismatch - Wav2Vec2$_{\mathcal{S}\rightarrow\mathcal{T}}$. This result is validated across 9 *Indian* languages and 4 languages from *Fleurs*. Further, we find that matched case adaptation - IndicWav2Vec2$_{\hat{\mathcal{T}}\rightarrow\mathcal{T}}$ can also have minor improvements when compared to standard adapter tuning. Further, we also run experiments on various adapter dimensions and show that these results are consistent across adapter parameter sizes. Finally, we present the CCA analysis and show that adapter pre-training leads to better correlation with phonemes of the target language. The key takeaway from this work is that always pre-train adapters while performing low-resource domain mismatched adapter tuning for ASR. In the future, we plan to extend this work to other types of SSL models.

7. Acknowledgements

Authors thank the Bill and Melinda Gates Foundation and the Department of Science and Technology (DST), Govt of India for their support in this work.

8. References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, 2017.

[2] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Interspeech*, 2020.

[3] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006.

[4] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv:1211.3711*, 2012.

[5] Y. Higuchi, S. Watanabe, N. Chen, T. Ogawa, and T. Kobayashi, "Mask CTC: Non-Autoregressive End-to-End ASR with CTC and Mask Predict," in *Interspeech*, 2020.

[6] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *NeurIPS*, 2020.

[7] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli, "Unsupervised Cross-Lingual Representation Learning for Speech Recognition," in *Interspeech*, 2021.

[8] V. Pratap, A. Tjandra, B. Shi, P. Tomasello, A. Babu, S. Kundu, A. Elkahky, Z. Ni, A. Vyas, M. Fazel-Zarandi *et al.*, "Scaling speech technology to 1,000+ languages," *arXiv:2305.13516*, 2023.

[9] 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," *TASLP*, 2021.

[10] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised Pre-Training for Speech Recognition," in *Interspeech*, 2019.

[11] C. Gao, G. Cheng, T. Li, P. Zhang, and Y. Yan, "Self-supervised pre-training for attention-based encoder-decoder asr model," *TASLP*, 2022.

[12] J. Bai, B. Li, Y. Zhang, A. Bapna, N. Siddhartha, K. C. Sim, and T. N. Sainath, "Joint unsupervised and supervised training for multilingual asr," in *ICASSP*, 2022.

[13] Q.-S. Zhu, J. Zhang, Z.-Q. Zhang, M.-H. Wu, X. Fang, and L.-R. Dai, "A noise-robust self-supervised pre-training model based speech representation learning for automatic speech recognition," in *ICASSP*, 2022.

[14] V. S. Lodagala, S. Ghosh, and S. Umesh, "CCC-WAV2VEC 2.0: Clustering AIDED Cross Contrastive Self-Supervised Learning of Speech Representations," in *SLT*, 2023.

[15] A. Rouditchenko, S. Khurana, S. Thomas, R. Feris, L. Karlinsky, H. Kuehne, D. Harwath, B. Kingsbury, and J. Glass, "Comparison of Multilingual Self-Supervised and Weakly-Supervised Speech Pre-Training for Adaptation to Unseen Languages," in *Interspeech*, 2023.

[16] C. W. et al, "VoxPopuli: A Large-Scale Multilingual Speech Corpus for Representation Learning, Semi-Supervised Learning and Interpretation," in *ACL-IJCNLP*, 2021.

[17] T. J. et al, "Towards Building ASR Systems for the Next Billion Users," in *AAAI*, 2022.

[18] A. Gupta, H. S. Chadha, P. Shah, N. Chhimwal, A. Dhuriya, R. Gaur, and V. Raghavan, "Clsril-23: Cross lingual speech representations for indic languages," *arXiv:2107.07402*, 2021.

[19] T. Alumäe, J. Kong, and D. Robnikov, "Dialect Adaptation and Data Augmentation for Low-Resource ASR: Taltech Systems for the Madasr 2023 Challenge," in *ASRU*, 2023.

[20] 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 *ICML*, 2019.

[21] A. Üstün, A. Bérard, L. Besacier, and M. Gallé, "Multilingual Unsupervised Neural Machine Translation with Denoising Adapters," in *EMNLP*, 2021.

[22] A. Bapna and O. Firat, "Simple, Scalable Adaptation for Neural Machine Translation," in *EMNLP-IJCNLP*, 2019.

[23] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, "Towards a Unified View of Parameter-Efficient Transfer Learning," in *ICLR*, 2021.

[24] "AdapterFusion: Non-Destructive Task Composition for Transfer Learning, author=Pfeiffer, Jonas and Kamath, Aishwarya and Rücklé, Andreas and Cho, Kyunghyun and Gurevych, Iryna," in *EACL*, 2021.

[25] Y. M. et al., "UniPELT: A Unified Framework for Parameter-Efficient Language Model Tuning," in *ACL*, 2022.

[26] J. Pfeiffer, I. Vulić, I. Gurevych, and S. Ruder, "UNKs Everywhere: Adapting Multilingual Language Models to New Scripts," in *EMNLP*, 2021.

[27] E. J. Hu, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," in *ICLR*, 2021.

[28] Y.-L. Sung, J. Cho, and M. Bansal, "Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks," in *CVPR*, 2022.

[29] N. Morioka, H. Zen, N. Chen, Y. Zhang, and Y. Ding, "Residual adapters for few-shot text-to-speech speaker adaptation," *arXiv:2210.15868*, 2022.

[30] C.-P. Hsieh, S. Ghosh, and B. Ginsburg, "Adapter-Based Extension of Multi-Speaker Text-to-Speech Model for New Speakers," *arXiv:2211.00585*, 2022.

[31] T. Lee, M.-J. Lee, T. G. Kang, S. Jung, M. Kwon, Y. Hong, J. Lee, K.-G. Woo, H.-G. Kim, J. Jeong *et al.*, "Adaptable multi-domain language model for transformer asr," in *ICASSP*, 2021.

[32] Y. Zhao, C. Ni, C.-C. Leung, S. Joty, E. S. Chng, and B. Ma, "A Unified Speaker Adaptation Approach for ASR," in *EMNLP*, 2021.

[33] W. Hou, H. Zhu, Y. Wang, J. Wang, T. Qin, R. Xu, and T. Shinozaki, "Exploiting adapters for cross-lingual low-resource speech recognition," *TASLP*, 2021.

[34] B. Thomas, S. Kessler, and S. Karout, "Efficient adapter transfer of self-supervised speech models for automatic speech recognition," in *ICASSP*, 2022.

[35] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, 1999.

[36] S. Vander Eeckt and H. Van Hamme, "Using adapters to overcome catastrophic forgetting in end-to-end automatic speech recognition," in *ICASSP*, 2023.

[37] S. Kessler, B. Thomas, and S. Karout, "An adapter based pre-training for efficient and scalable self-supervised speech representation learning," in *ICASSP*, 2022.

[38] A. Conneau, M. Ma, S. Khanuja, Y. Zhang, V. Axelrod, S. Dalmia, J. Riesa, C. Rivera, and A. Bapna, "Fleurs: Few-shot learning evaluation of universal representations of speech," in *SLT*, 2023.

[39] J. K. et al, "Libri-Light: A Benchmark for ASR with Limited or No Supervision," in *ICASSP*, 2020.

[40] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *ICASSP*, 2015.

[41] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A Fast, Extensible Toolkit for Sequence Modeling," in *NAACL-HLT*, 2019.

[42] A. Pasad, J.-C. Chou, and K. Livescu, "Layer-wise analysis of a self-supervised speech representation model," in *ASRU*, 2021.