Exploring Parameter-efficient Adapters for Low-resource Automatic Speech Recognition

Ahnaf Mozib Samin

Supervisor:
Dr. Andrea DeMarco, University of Malta, Malta

Co-supervisor:
Dr. Claudia Borg, University of Malta, Malta
Dr. Shekhar Nayak, University of Groningen, The Netherlands

September, 2023

Submitted in partial fulfilment of the requirements for the degree of Master of Science in Human Language Science and Technology.

L-Università ta' Malta
Faculty of Information & Communication Technology

Acknowledgements

I would like to express my heartfelt gratitude to my supervisors, Dr. Shekhar Nayak from the University of Groningen, and Dr. Andrea DeMarco and Dr. Claudia Borg from the University of Malta. Their constant guidance, support, and invaluable feedback have been instrumental in shaping this thesis. Dr. Andrea DeMarco's insightful research directions concerning parameter-efficient automatic speech recognition at the very beginning of this work have been particularly beneficial.
I am indebted to the German Research Center for Artificial Intelligence (DFKI) for their generous cluster support, and I extend my thanks to the Center for Information Technology at the University of Groningen for granting access to the Hábrók high-performance computing cluster.
My participation in the Erasmus Mundus Language and Communication Technologies (EM-LCT) program has provided me with an extraordinary opportunity to grow myself. I express my sincere gratitude to the Erasmus+ program of the European Commission for the full-funded scholarship during my master's program. These two years in the EM-LCT program have not only enriched me with cutting-edge knowledge in natural language processing but also exposed me to diverse European cultures, which I will forever cherish. My thanks go to the fellow students and esteemed faculty members of EM-LCT.
I am deeply thankful to Dr. Marc Tanti from the University of Malta for his exceptional *Deep Learning for NLP* course, where I not only gained proficiency in the PyTorch framework but also acquired crucial insights into deep learning models, proving invaluable for this thesis work. I also wish to acknowledge my Bachelor of Science thesis supervisor, Prof. Dr. Shahidur Rahman from Shahjalal University of Science and Technology, Bangladesh, whose early introduction to research laid the foundation for my achievements in my master's program.
Lastly, my gratitude knows no bounds for my parents, whose constant support has been the cornerstone of my journey. Their guidance and care during my early years has opened doors to the opportunities I have encountered till now. I also express my thanks to my newborn niece, Amayra Sehrish Ahmed, whose presence during the thesis writing added joyful moments to this endeavor.

Abstract

Parameter-efficient adapter modules have been leveraged in pre-trained speech models for speech processing tasks such as automatic speech recognition (ASR) in recent years. An adapter, integrated into these pre-trained speech models, typically consists of two feed-forward layers that are trained while keeping the pre-trained backbone frozen. Despite their emergence for ASR, a comprehensive exploration of adapters remains lacking, leaving several research questions unanswered.
In this thesis, we employ adapter-based tuning on two state-of-the-art pre-trained models, XLS-R and MMS, and compare it with the complete fine-tuning approach. Our study investigates the data requirements for adapter-tuning and reveals that while adapters are unsuited for few-shot learning, they exhibit competitive performance compared to full fine-tuning when at least 10 hours of labeled speech data are available. We also demonstrate that exploiting the larger XLS-R model with 2 billion parameters for adapter-tuning exhibits superior performance than fine-tuning the entire XLS-R 2B model. This phenomenon likely arises due to the susceptibility of larger models to overfitting during full fine-tuning, a challenge effectively circumvented by training only the adapters while leveraging the pre-trained knowledge.
Moreover, our experiment reveals that more pre-training data might be helpful for the adapter-tuning to work well. Additionally, we perform separate experiments on transfer learning with adapters and scaling the adapter modules with more feed-forward layers, yielding valuable insights. To the best of our knowledge, this exhaustive study is pioneering in its exploration of adapters for ASR, contributing significant insights to this evolving technology.

Contents

List of Figures **vi**

List of Tables **vii**

1 Introduction **1**

2 Background **4**

2.1 Automatic Speech Recognition . . . 4

2.2 Sequence-to-sequence models and Transformer . . . 6

2.2.1 Encoder and Decoder . . . 7

2.2.2 Attention . . . 8

2.2.3 Feed-forward networks . . . 11

2.2.4 Positional encoding . . . 11

2.3 Wav2vec 2.0 . . . 12

2.3.1 Architecture of Wav2vec 2.0 . . . 12

2.3.2 Cross-Lingual Speech Recognition - XLS-R . . . 15

2.3.3 Massively Multilingual Speech - MMS . . . 17

2.4 Issues with the Complete Fine-tuning approach . . . 18

2.5 Evaluation Metrics . . . 20

3 Literature Review **21**

3.1 Adapters in Natural Language Processing . . . 21

3.2 Adapters in Speech Processing Domain . . . 22

3.3 ASR for Bengali and Maltese . . . 26

3.3.1 Bengali . . . 26

3.3.2 Maltese . . . 27

4 Datasets **28**

4.1 Languages . . . 28

4.2 Selection and Preparation of Datasets 29
4.2.1 Bengali 29
4.2.2 English 30
4.2.3 Maltese 30

5 Methodology **32**
5.1 Architecture of the network 32
5.1.1 Inserting Adapters into Transformer 32
5.1.2 Designing Adapters 33
5.2 Experimental Setup 35

6 Results and Discussion **39**
6.1 Effect of Train set Size on Full Fine-tuning vs Adapter-tuning 39
6.2 Larger pre-trained models tend to benefit from adapter-tuning 43
6.3 Adapter-tuning-based ASR for different languages 44
6.4 Transfer Learning Across Adapters 46
6.5 Scaling adapters with more fully connected layers 47

7 Conclusion **49**
7.1 Ethical Considerations 50
7.2 Limitations 50
7.3 Future Work 51

References **52**

List of Figures

2.1 Architecture of Transformer. This figure is taken from Vaswani et al. (2017) . . 8
2.2 The left figure shows the scaled dot product attention. The right figure depicts the multi-head self-attention. These figures are taken from Vaswani et al. (2017). 10
2.3 The architecture of wav2vec 2.0 Baevski et al. (2020) 12

5.1 The left figure shows the architecture of the wav2vec 2.0 combined with two adapter modules. The two adapter modules are located after multi-head self-attention (MHSA) and feed-forward sub-layer, respectively. Add & Norm refers to the residual connection and layer normalization component. FC corresponds to the fully connected feed-forward network. The right figure depicts a bottleneck adapter with down-projection and up-projection layers, GeLU activation, layer normalization, and residual connection. 34

6.1 Word error rates (%) reported on the test sets after increasing the number of fully connected layers in each adapter module, inserted into XLS-R 2B. MT, BN, and EN corresponds to Maltese, Bengali, and English, respectively. 48

List of Tables

2.1 Details for XLS-R 0.3B, XLS-R 1B, and XLS-R 2B models Babu et al. (2021). FF dimension refers to the inner dimension of the feed-forward network within the Transformer block. #languages means the number of languages used for pre-training. Same datasets with an equal number of hours are used for pre-training these three models 16

4.1 The datasets are split into train, dev and test sets. BN, MT, and EN refer to Bengali, Maltese, and English, respectively. For English, each of dev and test sets has clean and other (noisy) versions. 31

5.1 Some of the hyper-parameters used for full fine-tuning and adapter-tuning with wav2vec 2.0 37

6.1 **Evaluation of full fine-tuning (FT) and adapter-tuning with XLS-R and MMS models for low-resource ASR in terms of WERs (%).** We study pre-trained ASR models with a varied number of trainable parameters (0.3B, 1B, and 2B). Three languages, namely Maltese (MT), Bengali (BN), and English (EN) from three diverse language groups are chosen. Results are reported on both dev and test sets. CTC-based greedy decoding is used. 41

6.2 **Evaluation of full fine-tuning (FT) and adapter-tuning with XLS-R and MMS models for low-resource ASR in terms of CERs (%).** We study pre-trained ASR models with a varied number of trainable parameters (0.3B, 1B, and 2B). Three languages, namely Maltese (MT), Bengali (BN), and English (EN) from three diverse language groups are chosen. Results are reported on both dev and test sets. CTC-based greedy decoding is used. 42

6.3 Evaluation of full fine-tuning and adapter-tuning with XLS-R 2B for a moderately large amount of data of 200 hours from SUBAK.KO for Bengali (BN) and 360 hours from LibriSpeech for English (EN). Results are reported in terms of WERs (%) on both dev and test sets. CTC-based greedy decoding is used. . . . 43

6.4 Number of hours of English, Maltese, and Bengali untranscribed speech data used for pre-training XLS-R Babu et al. (2021) 45

6.5 Pre-training the adapters within XLS-R 2B with the source language using a 50-hour-long train set and then fine-tuning those adapters with the target language using the same amount of train subset. The language modeling head dedicated to the source language is removed from the pre-trained XLS-R 2B. Results are reported in terms of WERs (%) on the dev and test sets. *source language* -> *target language* refers to the fact that the transfer of knowledge occurs from the source language to the target language. For comparison, the WER for the standard adapter-tuning without transfer learning across adapters is also provided. BN, MT, and EN refer to Bengali, Maltese, and English, respectively. 46

1 Introduction

Automatic Speech Recognition (ASR) has already made significant strides for high-resource languages, such as English, Mandarin, French, etc., due to the availability of large-scale transcribed speech corpora and abundant computational resources Ardila et al. (2020); Bu et al. (2017); Gulati et al. (2020). However, there exist more than 7000 languages in the world, the vast majority of which are either low-resource or zero-resourced languages Dunbar et al. (2021); Meng and Yolwas (2022). The majority of these languages are facing the risk of getting extinct due to the declining number of speakers in the community and the lack of representation in digital media. The performance of ASR for low-resource languages remains sub-optimal and there is an ongoing demand to preserve the cultural and linguistic heritage with the help of artificial intelligence (AI)-enabled technology.

Pre-trained self-supervised speech models such as wav2vec 2.0 Baevski et al. (2020), HuBERT Hsu et al. (2021), etc. are state-of-the-art technology that can enhance ASR performance for low-resource languages by leveraging knowledge from high-resource languages. These models employ self-supervised learning techniques, in which the model generates speech representations from the raw audio input by predicting masked speech or solving contrastive tasks. After pre-training on massive unlabeled datasets, these models can be fine-tuned on specific downstream speech processing tasks, such as automatic speech recognition (ASR), speaker verification, etc. leading to improved performance compared to training from scratch Wang et al. (2021b). Experiments indicate that even fine-tuning with only 10 minutes of labeled speech data, wav2vec 2.0 models achieve impressive ASR performance Baevski et al. (2020).

Although fine-tuning has been widely regarded as the best-performing strategy thus far, it has some significant limitations. Fine-tuning large pre-trained models involves updating the majority of the model's parameters, making it a parameter-inefficient and computationally expensive approach Pfeiffer et al. (2021). This may not be well-suited for resource-constrained scenarios. Moreover, these models occupy substantial storage space, posing challenges for developing offline ASR systems for mobile devices. Additionally, when dealing with multiple tasks or languages, fine-tuning necessitates a complete re-training of the pre-trained model for each new task or language, leading to catastrophic forgetting during sequential fine-tuning Pfeiffer et al. (2021). Choosing the sequence of languages or tasks for fine-tuning also presents a non-trivial decision-making challenge.

To tackle these challenges, adapter modules were initially introduced for computer vision Rebuffi et al. (2017). Adapters are typically small two-layer feedforward networks that are inserted into large pre-trained models Houlsby et al. (2019). Unlike full fine-tuning, only the adapter parameters are updated, while the remaining model is kept frozen. Consequently, the number of trainable parameters is significantly reduced, making the self-supervised pre-trained language modeling approach highly parameter-efficient. For each new task, a task-specific adapter can be trained and integrated into the pre-trained model, eliminating the need for complete re-training. Additionally, updating only the task-specific adapters prevents the occurrence of catastrophic forgetting Pfeiffer et al. (2021).

While adapters are now widely used for computer vision and natural language processing (NLP), they have received limited attention in speech processing tasks until recently Gállego et al. (2021); Thomas et al. (2022). Notably, Thomas et al. (2022) conducted one of the first and most prominent studies exploring adapters for ASR, to the best of our knowledge. They introduced bottleneck adapters, consisting of two feed-forward layers, to the wav2vec 2.0 English Base model. The adapters were trained for English and French ASR, using only 10-hour train sets from the LibriLight and CommonVoice (CV) corpora, respectively. The study concluded that adapter-tuning achieved comparable accuracy to full fine-tuning, however, with the advantage of updating only 9.2% of the parameters.

Since there are limited works such as Thomas et al. (2022) and Pratap et al. (2023), etc. for ASR modeling using adapter-tuning, there exist substantial research gaps that necessitate a comprehensive study of this method. Some mid/high-resource languages have more than 200 hours of labeled speech data, while the majority of low-resource languages have access to less than 20 hours. It still remains an open question what the minimum training data size must be for adapter-tuning to perform on par with complete fine-tuning. Conversely, can updating only the compact adapter modules improve the performance of ASR in mid/high resource scenarios? Moreover, by leveraging a larger multilingual pre-trained model and adapter-tuning approach, is it possible to improve ASR performance, particularly for languages with limited resources? Additionally, is it advantageous to pre-train the adapter with a source language and then fine-tune it on the target language to enable transfer learning?

This study aims to address the aforementioned research questions by conducting a comprehensive investigation of adapters for ASR, with a particular focus on the low-resource aspect. Through this research, we aim to reduce computational complexity while simultaneously improving ASR performance, ensuring the representation and preservation of low-resource languages in the field of technology.

The contribution of this work is mainly four-fold:

- We explore the applicability of the adapter-tuning approach for ASR under various resource-constraint scenarios, including extremely low-resource, low-resource, and medium/high-resource conditions. In this investigation, we select three languages from diverse language groups: English (West Germanic), Bengali (Indo-Aryan), and Maltese (Semitic). English represents a high-resource language, Bengali stands as a mid-resource language, and Maltese is classified as a low-resource language. To conduct the comparison, we create multiple training subsets and evaluate the performance of both full fine-tuning and adapter-tuning methods. To the best of our knowledge, no existing study has examined the data requirements for ASR utilizing the adapter-tuning approach.
- To leverage the potential of multilingual, pre-trained self-supervised speech models, we incorporate adapter modules into two state-of-the-art models, namely XLS-R Babu et al. (2021) and MMS Pratap et al. (2023). Additionally, we investigate whether employing a larger pre-trained model with a higher number of parameters enhances the performance of the adapter-tuning approach for ASR.
- In our experiments, we implement transfer learning with adapters to assess the potential benefits of utilizing pre-trained adapter weights for performance improvement. For this purpose, we pre-train the adapters on a source language and subsequently fine-tune them for the target language.
- Although bottleneck adapters with a two-layer feedforward network are commonly used as an adapter architecture Houlsby et al. (2019); Thomas et al. (2022), we expand the adapter module by increasing the number of fully connected layers and assess its impact on performance for the three languages.

2 Background

In this chapter, we are going to present the fundamental concepts in speech recognition as well as the advanced deep learning algorithms utilized for speech processing. Section 2.1 provides the background concepts on speech recognition models. Then, the detailed description of sequence-to-sequence models and transformers is provided in Section 2.2. We discuss the Wav2vec 2.0, which is the state-of-the-art transfer learning-based model in speech processing, and the multi-lingual pre-trained XLS-R and MMS models in Section 2.3. The issues with the traditional approaches of transfer learning are discussed in Section 2.4. Finally, we describe the widely-used evaluation metrics for ASR systems in Section 2.5.

2.1 Automatic Speech Recognition

Automatic Speech Recognition (ASR) is a speech processing task that converts spoken language into written text. There are numerous applications for this technology, including transcription services, voice assistants, voice-controlled systems, etc. ASR has seen a huge improvement in high-resource languages such as English, Mandarin, etc. getting closer to human-level speech recognition. However, the performance of even high-resource ASR drops when the speech utterance contains highly deviant dialect or accent, there is background noise, the rate of speech is fast, the microphone is far from the speaker, multiple speakers talk simultaneously, the out-of-vocabulary word is present, a new domain is introduced during inference (for example, air traffic control commands are presented to a model trained for general-purpose speech recognition), code-switching between languages occurs, etc. There are a larger number of languages, considered low-resource, in which ASR performance is still quite inadequate. Computational modeling of ASR has not been possible for a much wider selection of languages with zero resources.

An ASR system is developed following mainly three steps such as input feature extraction, acoustic modeling, and decoding.

Input feature extraction In the feature extraction step, speech signals are transformed into a more compact and representative feature space, such as Mel Frequency Cepstral Coefficients (MFCCs), which capture important characteristics of the audio. To compute MFCCs, the raw signal is windowed into shorter frames and then converted from the time domain to the frequency domain using a fast Fourier transform (FFT). After that, the Mel scale is obtained by applying filter banks. After taking the logarithm, and applying discrete cosine transform, the MFCCs are extracted. Some other feature extraction methods include linear predictive coding (LPC), perceptual linear prediction (PLP), relative spectral-perceptual linear prediction (RASTA-PLP), etc. In recent years, neural representations from pre-trained models are the state-of-the-art features for speech processing tasks Baevski et al. (2020) (See Section 2.3 for details).

Acoustic modeling The next step is acoustic modeling which maps the input features to the output units. Previously, one of the widely used approaches for developing ASR is the Gaussian Mixture Model (GMM)- Hidden Markov Model (HMM) based system Wang et al. (2019). With the advent of neural networks, GMM-HMM-based ASR models are replaced and large vocabulary continuous speech recognition (LVCSR) systems are developed with the help of a large amount of ASR data Graves et al. (2013). Neural Networks such as recurrent neural networks, convolutional neural networks, Transformer, etc. are used to model the relationship between the extracted features and the speech units. The models are trained on large amounts of transcribed speech data to estimate the probabilities of observing specific features given a speech unit. Connectionist Temporal Classification (CTC) is a widely used loss function used for ASR Graves et al. (2006).

Decoding The final step is decoding. During decoding, the ASR system searches for the most likely sequence of text outputs given the context. An external language model is commonly incorporated into the acoustic model to perform additional error correction. N-gram language modeling is a widely used and effective approach in ASR Samin et al. (2021). An N-gram language model provides the probability of a token e.g. word, character, etc. based on its preceding $N-1$ tokens. Typically, for word-based N-gram modeling, $N = 3, 4, 5$ and for character-based modeling $N = 10, 15, 20$. While N-gram modeling is simple and highly effective for ASR, it cannot capture long-range dependencies. Neural network-based language models (recurrent, convolution, and Transformer-based language models, etc.) are found to be better than N-gram modeling Likhomanenko et al. (2019).

Transfer learning has acquired significant popularity in ASR in recent years Baevski et al. (2020); Schneider et al. (2019). Transfer learning enables models to utilize knowledge from related tasks or domains to enhance their performance on the objective task. It has been demonstrated to be effective in ASR, especially with the rise of deep learning architectures such as Transformer.

Transfer learning in automatic speech recognition requires pretraining a neural network on a large amount of untranscribed data from a related task, such as a large-scale speech recognition corpus. Since untranscribed data is easier to gather, this approach eliminates the need for large-scale annotated speech datasets. This pre-trained network is then fine-tuned using a smaller, task-specific labeled target dataset. Fine-tuning adapts the pre-trained model to the domain of interest by modifying the network parameters using the domain-specific dataset. wav2vec 2.0 Baevski et al. (2020), HuBERT Hsu et al. (2021), Whisper Radford et al. (2023) are examples of pre-trained models that are used for numerous speech processing tasks. More details about the transfer learning approach can be found in the subsequent sections.

2.2 Sequence-to-sequence models and Transformer

The core of cutting-edge speech technology is sequence-to-sequence (seq2seq) modeling and Transformer.

Seq2seq models Seq2seq models are a group of neural network architectures that can process sequential data and produce output sequences. At the core of seq2seq models, there are two main components namely, encoder and decoder. The input embedding is passed to the encoder which processes it through a number of neural network layers and provides a fixed-length context vector. This context vector is a hidden state of the last layer of the encoder and captures the condensed representation of the entire input embedding. The context vector is then used as input to the decoder. The decoder is a neural network and generates the output sequence. The architecture of the encoder and decoder can vary, depending upon the task and choice of the practitioner.

Recurrent neural network (RNN) was among the widely used architectures as encoders and decoders due to its ability to process sequential data in each time step by taking into account the output from the previous time step. However, RNNs suffer from the gradient vanishing problem, which means that the gradients become vanishingly small, and thus, the weights of the network are not updated. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are popular variants of RNNs that address the vanishing gradient problem of the base RNN. However, there exist some issues with LSTM/GRU. First, RNN and its variants process the data sequentially and as a result, it requires an excessive amount of time to train the models, particularly when dealing with large-scale datasets. Second, RNN-type models follow the Markov property, which assumes that each state is only dependent on the previously seen state. However, the Markov property does not work for natural language and speech processing since there can be long-range dependencies in a sentence or speech utterance.

To fix the later issue, an attention mechanism is proposed that can capture a global context for each element in the input sequence, allowing the model to attend to relevant parts of the input during the decoding process. In addition to the recurrent units in the encoder, attention is coupled with it to provide a better representation of the context vector for each decoding time step. There are several variations of attention. In classical RNN-based seq2seq models, it is generally a weighted sum of the hidden states. The weights are learned by a multi-layer perceptron, also termed the attention layer. The performance of machine translation improved substantially by applying attention to the seq2seq models.

Transformer Vaswani et al. (2017) replaced the recurrent or convolutional units from the seq2seq models by introducing only the attention mechanism to the encoder and decoder. The architecture is referred to as Transformer. This work revolutionized deep learning research by enabling parallel processing of sequential data, capturing long-range dependencies, and mitigating vanishing gradients, etc. Because of parallel processing in Transformer, large-scale language models can be trained with huge amounts of data in less amount of time. Thus, the issue of sequential training with RNNs is addressed. RNN-based encoder-decoder models cannot perform well with long sentences, however, Transformer solves this issue by applying only the attention mechanism. Figure 2.1 presents the Transformer-based encoder-decoder architecture Vaswani et al. (2017). There are several core components in Transformer including the encoder, decoder, scaled dot product attention, multi-head self-attention, positional embedding, feedforward neural networks, layer normalization, and residual connection.

2.2.1 Encoder and Decoder

The encoder contains N number of layers, where N is set to 6 in the work of Vaswani et al. (2017) (See Figure 2.1). There are two main sub-layers in each encoder layer, namely, the multi-head self-attention (MHSA) and the fully connected feed-forward network. After each of these two sub-layers, there is a residual connection followed by layer normalization Ba et al. (2016). Given the input x to a sub-layer, the final output from the sub-layer is $LayerNorm(x + SubLayer(x))$. The output of each sub-layer, as well as the embedding layer, is $d_{model} = 512$

The decoder is composed of N layers, where $N = 6$ in the work of Vaswani et al. (2017). Apart from the multi-head self-attention and fully connected feed-forward network similar to the encoder, there is an additional multi-head self-attention sub-layer inserted between the existing sub-layers to perform the attention mechanism over the output of the encoder. There are residual connections followed by layer normalization around each sub-layer. To ensure that prediction at position i only depends on the known previous positions at less than i, the self-attention sub-layers are slightly modified by masking and offsetting the output embeddings by one position.

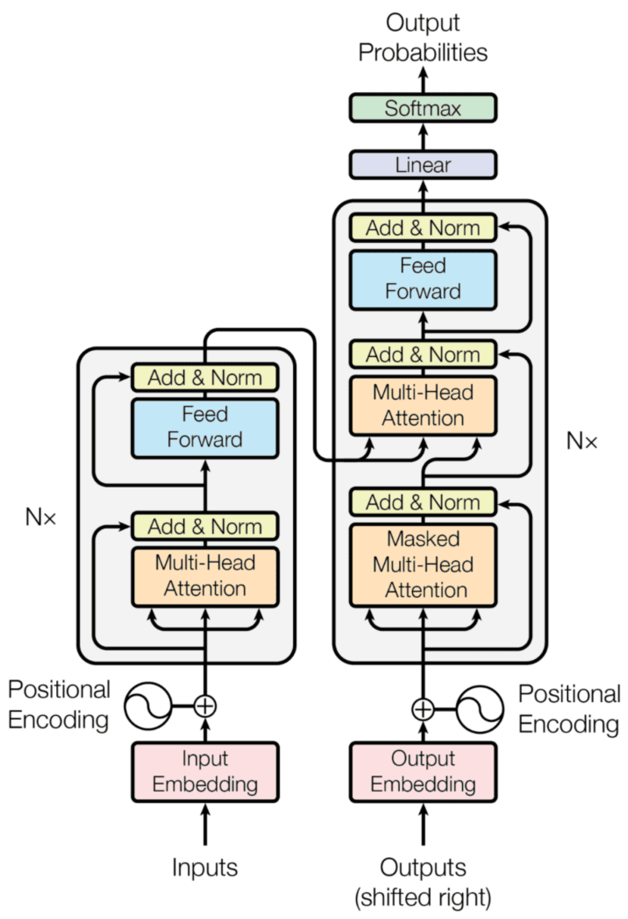

Figure 2.1: Architecture of Transformer. This figure is taken from Vaswani et al. (2017)

2.2.2 Attention

The attention mechanism allows a neural network model to focus on certain parts of the input data that are deemed to be the most important given the context while providing less importance to other parts. The idea is inspired by human cognition, where we attend to only specific information and ignore redundant information. In the context of NLP, using attention, the model can weigh the importance of each token (e.g. word, character, etc.) given the sentence. There are two main classes of attention, namely self-attention and dynamic attention. Self-attention operates on a single sequence of elements, typically represented as a matrix of embeddings. For each element (token) in the sequence, self-attention computes its relationship with all other elements in the sequence, generating an attention score for each pair of elements. On the other hand, dynamic attention relates an input sequence to an output sequence. Vaswani et al. (2017) utilized the self-attention mechanism in Transformer, mainly because of the reduced complexity of self-attention per layer.

From the input embedding vector, three vectors, namely query, key, and value, are prepared. Query represents the current element of the input sequence for which the attention score is computed, the key represents the elements of the input sequence that works as context and are used to compute the attention score, and finally, value presents the associated information for each key element. There are multiple ways to obtain these vectors from the input embedding. For example, one of them is by splitting the input embedding into three fixed-sized vectors or another way is to use the same input vector as the query, key, and value and linearly transform them using three linear layers. These three vectors work as projections of the same input sequence. After defining the query, key, and value, augmented with the positional information, they are passed to the multi-head self-attention sub-layer (See Figure 2.1). Using an attention function within the multi-head self-attention sub-layer, the query and a set of key-value pairs are mapped to the context vector that contains the relevant information for the query. Attention scores can be computed in several ways including scaled dot-product attention proposed by Vaswani et al. (2017), additive attention using a feed-forward network, etc.

Scaled dot product attention Figure 2.2 shows the operations to perform scaled dot product attention. A query and a key have the same dimension of d_k and a value has a dimension of d_v. For each query with all keys, the dot products are computed and then scaled by a factor of $\frac{1}{d_k}$. Then, a softmax function is applied to obtain the probability distribution of each query with all the keys. Finally, there is a dot product computed between the output of the softmax function and all values.

The same procedures can be done for all queries at a time using matrix multiplication. Suppose, Q , K, and V contain all queries, keys, and values, accordingly. The attention scores are computed as follows,

$$attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

Compared to additive function, scaled dot product attention is much faster and oc-

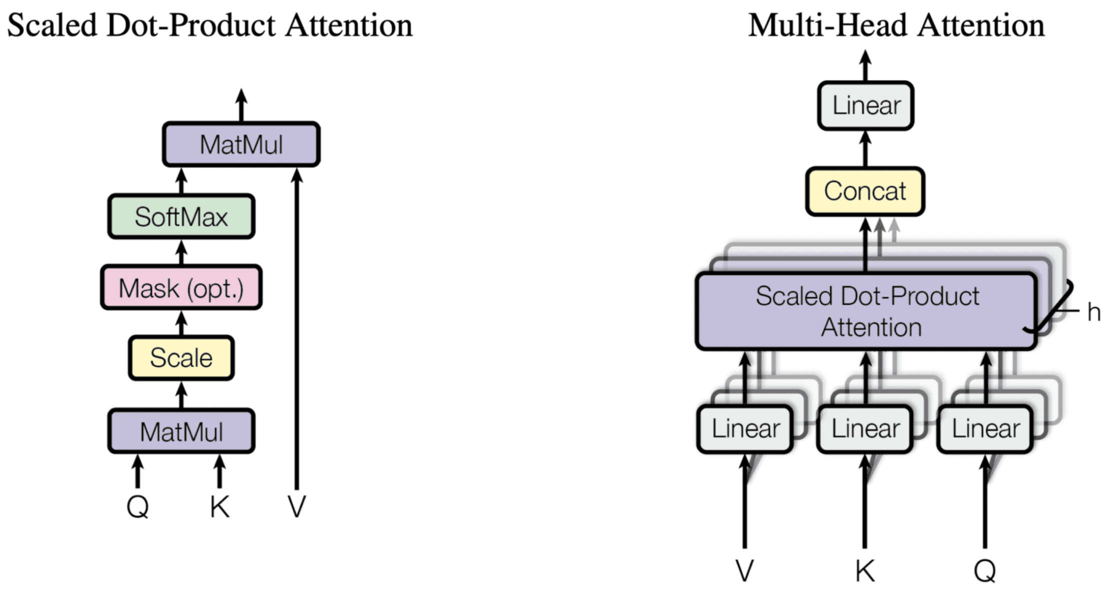

Figure 2.2: The left figure shows the scaled dot product attention. The right figure depicts the multi-head self-attention. These figures are taken from Vaswani et al. (2017).

cupies less space in the memory, thanks to the efficient matrix multiplication library. For small values of d_k, both attention mechanisms perform similarly. However, it has been seen that for larger value of d_k, additive attention is a better approach than the regular dot product attention without scaling Britz et al. (2017). By introducing the scaling factor, however, the performance of attention can be improved.

Multi-head self-attention Instead of performing scaled dot product attention a single time for a query and a set of keys and values, the same operation can be done multiple times. More specifically, each of the query, key, and value vectors can be passed to h number of linear layers to get h linear projections of the same embedding. The scaled dot product attentions are then computed h times. Finally, the resultant context vectors are concatenated and passed to a linear layer to obtain a final context vector. This method is referred to as multi-head self-attention, in which h is the number of heads. Experimental evaluation suggests that multi-head self-attention performs better than a single-head self-attention function. In the work of Vaswani et al. (2017), $h = 8$ is used. To ensure that the total computational complexity is kept the same as the regular single-head self-attention, the dimension of query and keys, d_k, and the dimension of values d_v of each head are reduced in the case of multi-head self-attention.

2.2.3 Feed-forward networks

There is a fully-connected feed-forward network in each layer of the encoder and decoder of the Transformer network. There are two linear layers in the feed-forward network, applied to each position separately, and in between the two linear transformations, there is a rectified linear unit (ReLU) activation function. In the work of Vaswani et al. (2017), the dimension of the input and output of the feed-forward network is $d_{model} = 512$ while the inner dimension of the feed-forward network is $d_{ff} = 2048$.

2.2.4 Positional encoding

In contrast to recurrent neural networks (RNNs), the Transformer does not comprehend the sequential order of the input data. Positional encoding is a technique used in the Transformer architecture to provide the model with information about the relative or absolute positions of tokens in the input sequence. Typically, the positional encoding is represented as a vector with the same size as the input embeddings. Before feeding the input embeddings into the encoder, positional encoding is added to the input embeddings. This technique is especially crucial for tasks involving sequential data, such as natural language processing, where the placement of words in a sentence can substantially alter its meaning.

There are numerous methods for computing positional encoding. Using trigonometric functions, such as the sine and cosine functions, to generate the encoding is a common method. Following is the formula for positional encoding in the work of Vaswani et al. (2017):

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{2.2}$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{2.3}$$

where, $PE_{(pos,2i)}$ represents the $i - th$ dimension of the positional encoding for the element at position "pos" and d_{model} is the dimension of the input embeddings. By using sinusoidal functions with varying frequencies and phases, positional encoding introduces unique representations for each position in the sequence.

2.3 Wav2vec 2.0

2.3.1 Architecture of Wav2vec 2.0

Wav2vec 2.0 adopts a self-supervised training approach by masking input frames in the latent space and utilizing a contrastive loss function Baevski et al. (2020) (See Figure 2.3). This method allows the model to learn intrinsic speech representations. These learned representations have been successfully applied in various speech processing tasks such as ASR, language identification, keyword spotting, speaker verification, and speech translation Fan et al. (2020); Li et al. (2021); San et al. (2021).

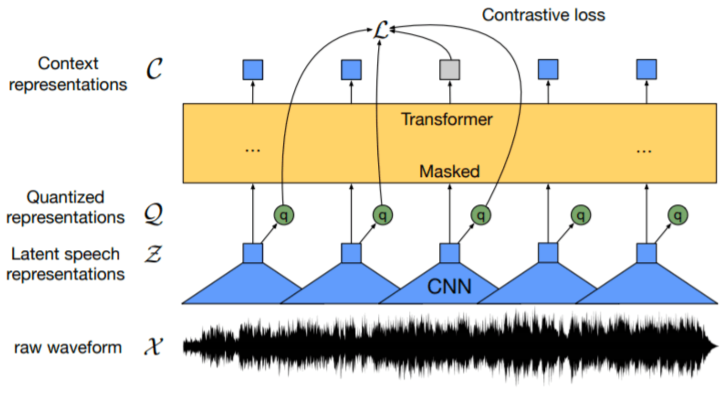

Figure 2.3: The architecture of wav2vec 2.0 Baevski et al. (2020)

Wav2vec 2.0 consists of several core components including the feature encoder, quantization module, contextual network, and objective function.

Feature encoder In Wav2Vec 2.0, raw audio X is taken as input and processed through a feature encoder composed of seven convolutional blocks. The objective of the feature encoder is to reduce the dimensionality of the raw audio signals. At first, the raw audio is normalized to a zero mean and unit variance. Each convolutional block consists of a temporal single-dimensional convolutional layer followed by a layer normalization Ba et al. (2016), and a Gaussian Error Linear Unit (GELU) activation function Hendrycks and Gimpel (2016). There are 512 channels at each convolutional layer. The stride size is (5,2,2,2,2,2,2) and the kernel width is (10,3,3,3,3,2,2). The output of the feature encoder is the latent representation Z that is a sequence of vectors $z_0, z_1, z_2, z_3, ..., z_T$ every 20 milliseconds. Here, T is the number of time steps determined by the total stride. The feature encoder encompasses a total receptive field equivalent to 400 samples or 25 ms of audio. The latent representation Z is passed as input to the contextual network.

Quantization module The output of the feature encoder Z is discretized to a finite set of speech representations from the continuous space via product quantization Jegou et al. (2010). One of the challenges of speech processing is that the speech signal is continuous. Written texts contain discrete tokens such as characters, byte-pair encoding, unigram, etc. Despite the continuous nature, speech signals can be labeled using phonemes. However, human supervision is required for labeling, which is not possible for self-supervised learning. Previous studies indicate that discretized speech representation leads to superior performance Baevski et al. (2019b).

In the product quantization procedure, there are multiple codebooks, each containing a fixed number of discretized speech representations (entries). In the wav2vec 2.0, G codebooks are available, each with V entries. The model chooses one entry e among the V entries from each of the G codebooks and finally concatenates the G number of entries. The resulting vector is the concatenation of $e_1, e_2, ..., e_G$, and a linear transformation is applied to obtain the quantized representation q. There are 2 codebooks and each codebook contains 320 entries. Thus, the possible number of entries can be 102,400 discretized speech representations from two codebooks.

To make the process of choosing entries from the codebooks fully differentiable, Gumbel softmax is utilized Maddison et al. (2015). A non-negative temperature is introduced to add the randomness effect.

Contextual network The latent representations from the feature encoder are fed as input to the contextual network composed of Transformer layers. In the wav2vec 2.0, the same Transformer encoder is used like Vaswani et al. (2017) except for relative positional embeddings are used instead of the absolute positional embeddings. To obtain the relative positional embeddings, the convolutional layer is utilized similarly to Baevski et al. (2019a); Mohamed et al. (2019); Wu et al. (2018). The output of the convolution followed by GELU is added to the input and then layer normalization is applied. Unlike seq2seq Vaswani et al. (2017), wav2vec 2.0 is a Transformer-based encoder-only model. The detailed Transformer encoder is described in section 2.2.

There are two variants of wav2vec 2.0, namely the base model and large model, proposed by Baevski et al. (2020). The base model contains 12 Transformer blocks while the large one has 24 Transformer blocks. Moreover, for the base model, the dimension of the model is 768, the inner dimension of the feed-forward network is 3072, and the number of attention heads is 8. In contrast, the large model has a dimension of 1024, inner dimension of the feed-forward network is 4096, and number of attention heads is 16.

Training objective There are two objective functions used to pre-train the wav2vec 2.0. Let, the total loss be L, the contrastive loss be L_m, and the diversity loss be L_d. So, the total loss L is computed as,

$$L = L_m + \alpha L_d \tag{2.4}$$

where α is an optimized hyper-parameter.

Contrastive loss is the key objective to learn the speech representations in wav2vec 2.0. Before feeding the output of the feature encoder to the contextual network, a proportion of the output or time steps are randomly masked and replaced by the same trained feature vector, Z'_m for all masked time steps. More specifically, a certain proportion of time steps, p are randomly chosen as the starting indices, and the consecutive M time steps are masked for every chosen index. The masked speech input vectors are fed to the contextual network, which provides the output $c_1, c_2, ..., c_t, ..., c_T$. For each masked time-steps, K quantized distractors are uniformly sampled from the other masked time-steps of the same utterance. The model then measures the cosine similarity between the context vector c_t and both the actual quantized target q_t and all quantized distractors $\tilde{q}$. The contrastive loss is then used to promote similarity with the actual quantized target and to penalize similarity with the quantized distractors. Contrastive loss is computed as follows,

$$L_m = -log \frac{exp(sim(c_t, q_t)/k)}{\sum_{\tilde{q} \sim Q_t} exp(sim(c_t, \tilde{q})/k)} \tag{2.5}$$

where similarity score is defined as, $sim(a, b) = a^T b/||a||||b||$. $K = 100$ distracters are used for wav2vec 2.0.

In the pre-training phase, an additional loss, termed as diversity loss L_d, is introduced alongside the contrastive loss to ensure the model utilizes all codewords with equal frequency. This is achieved by maximizing the entropy of the Gumbel-Softmax distribution, which prevents the model from consistently favoring a small subset of codebook entries over others.

Fine-tuning Pre-trained models are fine-tuned for the down-stream task such as automatic speech recognition by adding a linear layer (language modeling head) with the same size of the number of tokens. Fine-tuning is optimized by using a CTC loss function Graves et al. (2006). To prevent overfitting and improve the overall performance, a modified version of SpecAugment masks time-steps and channels during fine-tuning Park et al. (2019). It has been demonstrated that with only 10 minutes of labeled data, wav2vec 2.0 achieves a WER of 4.8% and 8.2% on the LibriSpeech test clean and other sets, respectively. This model achieves state-of-the-art performance on TIMIT gar (1993) and LibriSpeech Panayotov et al. (2015) benchmarks.

2.3.2 Cross-Lingual Speech Recognition - XLS-R

While wav2vec 2.0 Baevski et al. (2020) is pre-trained with a massive amount of monolingual English unlabelled speech data, collecting large-scale unlabelled speech data is not possible for many low-resource languages. In NLP, multi-lingual BERT Kenton and Toutanova (2019), XLM-R Conneau et al. (2020), etc. models show substantial improvement in performance for many low-resource languages. Motivated by this, a larger cross-lingual wav2vec 2.0 model called XLS-R (Cross-Lingual Speech Recognition) has been released Babu et al. (2021); Baevski et al. (2020). XLS-R is pre-trained on 436K hours of publicly available untranscribed speech data from 128 languages. Specifically, this multilingual model leverages 372K hours of parliament speech in 23 languages from VoxPopuli Wang et al. (2021a), 50K hours of read book in 8 languages from Multi-lingual LibriSpeech (MLS) Pratap et al. (2020), 7K hours of read speech in 60 languages from CommonVoice Ardila et al. (2020), 6.6K hours of YouTube speech in 107 languages from VoxLingua107 Valk and Alumäe (2021), and 1K hours of phone conversations in 17 languages from BABEL Gales et al. (2014).

Among the languages considered, approximately 24 are categorized as high-resource languages, with data exceeding 1,000 hours per language. Almost all of these high-resource languages are of European origin, with the exception of Kinyarwanda, an African language. Additionally, there are 17 mid-resource languages, each possessing more than 100 hours but less than 1,000 hours of data. This group includes Catalan, Persian, Turkish, Russian, Basque, etc. The largest group consists of 88 low-resource languages, each having less than 100 hours of speech data. Among the 436K hours of untranscribed speech data that is used to pre-train XLS-R, there are 69493 hours of English data and 9120 hours of Maltese data. However, the pre-trained XLS-R has seen only 100 hours of Bengali data during its pre-training stage. There is a huge imbalance in terms of the number of hours of data used for pre-training XLS-R for a large number of the 128 languages.

The same wav2vec 2.0 architecture is leveraged to pre-train XLS-R Baevski et al. (2020). The model is available in three variants with parameter counts of 0.3 billion (B), 1B, and 2 B. The detail of the model architecture of each of these three pre-trained models is shown in Table 2.1.

XLS-R improves the ASR performance as seen in five benchmarks including BABEL, MLS, Common Voice, VoxPopuli, and LibriSpeech Babu et al. (2021). BABEL contains noisy phone conversations and presents challenges for ASR modeling. On BABEL, XLS-R 2B sets state-of-the-art WERs in five languages including Assamese, Tagalog, Swahili, Lao, and Georgian. MLS consists of speech data for eight European languages including English, German, Dutch, French, Spanish, Italian, Portuguese, and Polish. XLS-R achieves

Table 2.1: Details for XLS-R 0.3B, XLS-R 1B, and XLS-R 2B models Babu et al. (2021). FF dimension refers to the inner dimension of the feed-forward network within the Transformer block. #languages means the number of languages used for pre-training. Same datasets with an equal number of hours are used for pre-training these three models

Parameter	**XLS-R 0.3B**	**XLS-R 1B**	**XLS-R 2B**
#Transformer blocks	24	48	48
Model dimension	1024	1024	1920
FF dimension	4096	4096	7680
Attention heads	16	16	16
#Params	317M	965M	2162M
#languages	128	128	128

competitive WERs with only 10 hours of labeled data compared to the work of Pratap et al. (2020) that utilized the full datasets. Interestingly, XLS-R 2B, on average, is outperformed by XLS-R 1B. CommonVoice is a read-speech corpus, thus it is a comparatively easier benchmark. However, with the setup equivalent to few-shot learning, where there is only 1 hour of labeled data available, the phoneme recognition performance is measured for 10 languages. XLS-R yields state-of-the-art phoneme error rates (PER) for all languages. XLS-R 2B achieves slightly lower PER on average, however, there are some languages for which XLS-R 1B performs better. For the Kyrgyz language, the three XLS-R models obtain lower PERs compared to the existing best-performing cross-lingual speech models including XLSR-53 and XLSR-10, among others Conneau et al. (2021). It is worthwhile to mention that there are only 11 hours of Kyrgyz data used for pre-training the XLS-R models. It demonstrates that XLS-R is especially beneficial for developing ASR applications for low-resource languages.

The VoxPopuli corpus contains approximately 1,800 hours of labeled speech data in 14 different languages. The data ranges from 543 hours for the English language to 35 hours for the Slovakian language. In addition, the dataset includes approximately 400K hours of unlabeled speech. This corpus represents a scenario in which a substantial quantity of unlabeled data from the same domain as the labeled data is accessible. XLS-R has been compared with the work of Wang et al. (2021a). XLS-R 0.3B achieves superior performance over the previous work on all 14 languages while XLS-R 1B exhibits even better performance. XLS-R 2B, however, is not evaluated on the VoxPopuli corpus.

On the three subsets (10 minutes, 1 hour, and 10 hours) of the English LibriSpeech corpus, the performance of XLS-R 0.3B and XLS-R 1B is compared with the wav2vec 2.0

English base model Baevski et al. (2020). wav2vec 2.0 English base model is pre-trained using the 60K hours of unlabelled LibriVox data and contains 300M trainable parameters. XLS-R 0.3B is outperformed by the mono-lingual English base wav2vec 2.0 in each subset. This happens because of the capacity dilution and the interference problem of the multi-lingual problem, as argued by Babu et al. (2021). By increasing the model capacity with XLS-R 1B, the performance improves, particularly with 10 minutes of labeled data where XLS-R 1B achieves the lowest WERs. For 10 hours of labeled data, however, the wav2vec 2.0 English model yields lower WERs. Although the authors argue that with a higher capacity model, the interference problem can be avoided and performance can be improved, the XLS-R 2B model is not evaluated to test the hypothesis.

Apart from ASR, XLS-R is fine-tuned and then evaluated for speech translation, language identification, and speaker identification Babu et al. (2021).

2.3.3 Massively Multilingual Speech - MMS

There are over 7000 languages in the world, however, the current speech technology has covered only more than 100 languages Babu et al. (2021). To fill the research gap of other languages, the Massively Multilingual Speech (MMS) project expands the number of languages for many speech processing tasks Pratap et al. (2023). A large-scale single wav2vec 2.0 is pre-trained for 1406 languages to ensure better multi-lingual representations. This model is termed MMS. The project also released a fine-tuned checkpoint of MMS after finetuning it with data from 1107 languages (44.7K hours of annotated speech data, termed as MMS-lab) specifically for ASR. The same number of languages has been used to finetune MMS for speech synthesis. For language identification, MMS is fine-tuned for 4017 languages. The datasets are collected from publicly available religious speeches and self-supervised wav2vec 2.0 is pre-trained to build MMS.

There are two variations of MMS models with 0.3B and 1B parameters, respectively. The MMS 0.3B model contains 24 transformer blocks. The dimension of this transformer model is 1024. The inner dimension of the feed-forward network is 4096. There are 16 heads for the self-attention block. The number of trainable parameters is 317M. The larger MMS model with 1B parameters comprises 48 transformer blocks with a dimension of 1024. Similar to the MMS 0.3B, the feed-forward sublayer within the contextual network has an inner dimension of 4096. There are 16 heads for the multi-head self-attention blocks. There are 965M trainable parameters in the MMS 1B model. For both variants, the usual wav2vec 2.0 architecture with the feature encoder and the quantization module remains the same. These two variants are pre-trained on the same datasets e.g. MMS-lab-U (55K hours), FLEURS, Vox Propuli-400K (371K hours), Multi-lingual Librispeech (50K hours), CommonVoice (8.8K hours), Vox Lingua - 107 (5.3K hours), and BABEL (1K hours).

The information about the number of hours for each language in the pre-trained MMS is not revealed, to the best of our knowledge. Thus, it is uncertain whether or not MMS encountered balanced hours of speech data for each language. However, the experimental evaluation suggests that MMS outperforms Whisper, a weakly supervised large-scale speech model, on the FLEURS benchmark Pratap et al. (2023); Radford et al. (2022). Additionally, it has been reported that compared to XLS-R 300M and XLS-R 1B, MMS achieves lower CERs on the 61 FLEURS languages after fine-tuning both with the MMS-lab data. Notably, for evaluation, after fine-tuning the model with a multi-lingual dataset, language-specific adapters are leveraged.

The ASR performance has also been evaluated by dividing the languages based on demographic locations. Among the 1,107 languages, there are 335 Asian, 136 South American, 144 North American, 41 European, 363 African, and 88 Pacific languages. It has been demonstrated that the character error rate is the highest for African languages and the lowest for South American ones. While the character error rates are almost similar for Asian, European, and Pacific languages, the performance of North American languages is sub-optimal.

2.4 Issues with the Complete Fine-tuning approach

Transfer learning and sequence-to-sequence (seq-to-seq) models are the current state-of-the-art approaches in NLP, computer vision, and speech domains Baevski et al. (2020); Devlin et al. (2018). The basic idea is to pre-train a large language model with billions of parameters on a huge amount of unlabeled data leveraging self-supervised learning, and then, fine-tune the model for the down-stream task with rather a smaller labeled dataset. While fine-tuning the pre-trained language model, either the entire model's parameters are updated or most of them are updated using the back-propagation algorithm. On the other hand, zero-shot learning is another approach for which the model is not trained, rather, the input is directly given to the model utilizing prompt engineering during inference Liu et al. (2023). For few-shot learning, the model parameters can be updated by providing a few examples, or a prompt, consisting of several examples, can be given as input to the model. It has been observed that fine-tuning the pre-trained language model for the downstream task results in better performance compared to zero-shot or few-shot approaches Brown et al. (2020). Despite the performance boost by leveraging fine-tuning approach, there are key issues with completely fine-tuning the pre-trained language models.

The number of parameters of the state-of-the-art pre-trained language models is becoming higher with the advancement of computational resources, enriched datasets, and self-supervised or unsupervised approaches. For example, the state-of-the-art language model, GPT-3 (Generative Pre-trained Transformer 3) consists of 175B parameters. The cross-lingual pre-trained XLS-R has as many as 2B parameters. Fine-tuning all the parameters of these models results in taking up gigabytes of space on a computer. This makes it implausible to store the models on both mobile devices and servers. As a result, the benefit of transfer learning-based ASR technologies cannot be utilized for offline mobile devices, given that internet coverage is not well-established in a major portion of the world till now e.g. mainly in African and Asian countries.

In addition to that, fine-tuning separate models for each task/language is not practical considering the number of languages in the world and the possible number of NLP tasks. On top of that, the issue is more prevalent in the field of speech processing since there can be numerous dialects and accents. Maintaining separate ASR models dedicated to each dialect/accent might bring better performance, however, is extremely computationally expensive. Likewise, personalized ASR systems cannot be implemented with the current technology of fine-tuning the entire models from the computational perspective.

As we see that fine-tuning separate models per task/language is not possible, there are also major issues with fine-tuning a single pre-trained model for multiple tasks. There are two main approaches for transfer learning when it comes to multiple tasks or languages. First, pre-trained language models can be fine-tuned on various tasks sequentially one after another. However, there is an issue of catastrophic forgetting by sequentially fine-tuning several tasks. Catastrophic forgetting refers to the problem in which the pre-trained language model only performs well on the last task it is fine-tuned for and forgets about the preceding tasks. Thus, sequential fine-tuning performs worse when there are two sequential tasks Phang et al. (2018); Pruksachatkun et al. (2020). Moreover, it is a non-trivial task to decide the order in which the pre-trained language model is going to be fine-tuned.

Second, the pre-trained language model can be fine-tuned on several tasks simultaneously with a multi-task learning objective Ruder (2017, 2019); Zhang and Yang (2021). More specifically, the weighted sum of training losses from multiple tasks is back-propagated to update the weights of the pre-trained language model. Although multi-task learning has improved the performance of many NLP tasks, it is not possible to add a new task to the existing fine-tuned model without complete re-training Stickland and Murray (2019). Moreover, Lee et al. (2017) found that multi-task learning suffers from under-fitting in high-resource tasks and over-fitting in low-resource tasks. In addition, there is no way to share weights for different tasks in multi-task learning.

In short, fine-tuning multiple pre-trained language models for each task/language is a computationally expensive approach. Likewise, fine-tuning a single pre-trained model for numerous tasks/languages also faces difficulties due to catastrophic forgetting.

2.5 Evaluation Metrics

ASR systems can be evaluated using various standard evaluation metrics including word error rate (WER), character error rate (CER), phoneme error rate (PER), sentence error rate (SER), token (e.g. BPE, Unigram, etc.) error rate (TER). Among them, WER is the most widely used evaluation metric for ASR. WER is calculated by comparing the predicted output with the reference transcript and defined as the following,

$$WER = \frac{S + D + I}{N} \tag{2.6}$$

where, S is the number of substitutions (predicted output contains misinterpreted words), D is the number of deletions (predicted output misses the words), I is the number of insertions (predicted output contains additional words), and N is the total number of words in the reference transcript.

The lower the WER the better the ASR model performs. The concept of WER is inspired by the Levensthein distance. Levenshtein distance between two strings is the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into the other. However, WER works on the word level instead of the character level. Although being a reliable evaluation metric, WER possesses some drawbacks. For example, if a single character within a word is wrong, then the whole word is considered incorrect. Thus, it cannot properly evaluate if the predicted word is incorrect due to just a single character error or multiple characters error. Furthermore, WER does not take the important words into consideration since semantics is not incorporated in measuring WER. Typically, WER is expressed in percentages. WER can be greater than 100% because N is a fixed number, while the summation of S, D, and I can be larger than N.

CER is another standard evaluation metric used for ASR. CER is similar to WER, but it operates at the character level instead of words. It measures the percentage of character-level errors in the ASR output compared to the reference transcript. Like WER, lower CER values indicate better performance.

In this work, we utilize both WER and CER as our evaluation metrics to benchmark full fine-tuning and adapter-tuning approaches.

3 Literature Review

In this chapter, we review the existing literature dedicated to parameter-efficient transfer learning with the help of adapter modules. We provide past research on adapters, applied in natural language processing tasks in Section 3.1. After that, we report the existing work on adapters in the speech processing domain in Section 3.2.

3.1 Adapters in Natural Language Processing

An adapter is usually a two-layer feed-forward network with a down and up projection although the architecture of the adapters can be modified Houlsby et al. (2019). This methodology has been experimented with for several NLP tasks including neural machine translation Bapna et al. (2019); Philip et al. (2020), commonsense reasoning, sentiment analysis, etc Pfeiffer et al. (2021). By introducing these trainable parameters for each task, we can easily add or remove any task-specific adapters and so the re-training issue for new tasks in multi-task learning can be avoided. In the adapter approach, the pre-trained model is frozen as it is and only the adapter modules are fine-tuned. Because of this, we can get rid of the catastrophic forgetting of pre-trained language models. Adapters can be trained in single-task Bapna et al. (2019); Houlsby et al. (2019); Pfeiffer et al. (2020) as well as multi-task setup Stickland and Murray (2019). The performance of adapters in NLP is either on par with or slightly below compared to full-finetuning Houlsby et al. (2019); Pfeiffer et al. (2021).

Pfeiffer et al. (2021) introduced the AdapterFusion technique by adding a fusion layer on top of adapters. The fusion layer applies an attention mechanism to different adapters and thus it enables transfer learning across adapters. It has been found that AdapterFusion achieves superior performance than both full-finetuning and adapter without fusion approaches. Rücklé et al. (2020) proposed the AdapterDrop technique that reduces computational complexity by removing adapters from lower transformer layers. More recent methods such as AdapterBias have been investigated to make pre-trained language models more computationally efficient Fu et al. (2022).

The placement of adapters is investigated by Houlsby et al. (2019). In each Transformer layer, they place two adapters after multi-head self-attention and feed-forward layers. Bapna et al. (2019) and Stickland and Murray (2019) inserted the adapters only after the feed-forward layer within the Transformer layer. There have been several works in NLP to find out the specific layers in which adapters are best suited. Pfeiffer et al. (2021) analyzed the attention plots from layers 1, 7, 9, and 12 and found that adapters inserted into the last Transformer layer perform worse in the case of the AdapterFusion approach. They hypothesized that the last Transformer layers are closer to the head (e.g. linear classification layer) and thus adding adapters to them acts like fine-tuning the heads. It has been also found that low-resource tasks benefit from high-resource tasks with the AdapterFusion approach Pfeiffer et al. (2021).

3.2 Adapters in Speech Processing Domain

Although there have been numerous recent works on adapters in NLP and computer vision, few works exist in the speech processing domain with adapters. Thomas et al. (2022) implemented the adapter approach with wav2vec 2.0 English base model and built ASR models for English and French. Their adapter comprises two feed-forward layers, GELU activation function Hendrycks and Gimpel (2016), and layer normalization Ba et al. (2016) similar to our adapter architecture (See Section 5.1), however, their adapter is slightly different from us since there is no residual connection in their adapter. They found that for English ASR, the adapter approach performs slightly worse than full fine-tuning, and for French ASR, adapter-tuning achieves a comparatively lower word error rate. The benefit of using adapters is that by updating only 9.2% of the parameters, they achieved similar performance compared to full fine-tuning. However, in this work, only the wav2vec 2.0 English model has been utilized instead of exploring the possibility of cross-lingual transfer learning with XLS-R and adapters Babu et al. (2021). Moreover, their study does not reveal the training data requirement of the adapter-tuning as well as the effect of stacking more feed-forward layers and the impact of transfer learning within adapters.

Adapters are also utilized for ASR in the MMS model Pratap et al. (2023). At first, the pre-trained MMS 1B cross-lingual model is fine-tuned on the labeled MMS-lab data, which is 44.7K hours long and covers 1,107 languages. The performance of this fine-tuned cross-lingual model is further enhanced by incorporating language-specific adapters. The adapters are inserted in each Transformer block after the feed-forward network. An adapter module comprises layer normalization, two feed-forward layers for up-projection and down-projection, ReLU activation function. The inner dimension of the feed-forward network is 16. After adding the language-specific adapters, only additional 2M parameters (2% of the total parameters) are introduced for each language. Only the adapters and a randomly initialized language modeling head are trained while keeping the rest of the model frozen. Without the language-specific adapters, the fine-tuned cross-lingual model achieved 24.8% WER on the FLEURS-54 benchmark Conneau et al. (2023), however, after adding the adapters, the WER is reduced to 18.7%.

Google Universal Speech Model (USM), has a convolution-augmented transformer (Conformer) Gulati et al. (2020) working as an encoder, then either a CTC Graves et al. (2006), RNN transducer (RNN-T) Graves (2012) or a Listen, Attend, and Spell (LAS) Chan et al. (2016), and finally the language modeling head Zhang et al. (2023). This model was pre-trained with 12M hours of unlabeled data covering more than 300 languages. In some experiments, two language-specific parallel residual adapters are inserted into the Conformer blocks of the Google USM model. By training only 2.3% of the total parameters, the ASR performance was slightly worse than fine-tuning the entire model. However, the computational cost was reduced significantly with no requirement of fine-tuning the entire model for each language/domain. This work also evaluated adapters, inserted into Google USM, for speech translation.

Le et al. (2021) applied the adapter tuning approach for multilingual speech translation by using multilingual BART (mBART) as the pre-trained model. Adapters can be incorporated into the transformer backbone in either a serial or parallel fashion. Suppose, x is the input, function f represents a layer (for example, a whole encoder layer or a sub-layer such as multi-head self-attention, feed-forward network), and y is the output of function f. Consider that the adapter layer is represented by a function g. The adapted output can be computed as,

$$y_{serial} = g(f(x)) \tag{3.1}$$

$$y_{parallel} = f(x) + g(x) \tag{3.2}$$

So, for serial adapters, the output of the layer f is modified with the help of adapter modules while in the case of serial adapters, the output of the adapter g is added to the output of the layer f. Le et al. (2021) utilized the typical bottleneck adapters with layer normalization, two feed-forward layers, ReLU activation, and a residual connection. The authors explored both serial and parallel adapters and found that serial adapters perform better than parallel ones for speech translation. Their final conclusion on adapters is in line with Eeckt et al. (2022); Thomas et al. (2022) stating that the adapters are parameter efficient but results are on par with full fine-tuning.

Chen et al. (2023) utilized three pre-trained models such as HuBERT Hsu et al. (2021), wav2vec 2.0 Baevski et al. (2020), and DeCoAR2 Ling and Liu (2020) for the adapter tuning approach. They investigated adapters for several tasks including ASR, keyword spotting, phoneme recognition, etc, and explored several types of adapters such as Houlsby Houlsby et al. (2019), AdapterBias Fu et al. (2022), Weighted-sum, LoRa Hu et al. (2021), etc. Their finding was that Houlsby et al. (2019) adapter is overall better in terms of both performance and efficiency while the weighted sum is the most efficient approach in speech processing. LoRA adapters obtain the lowest accuracy as found in the experiment.

Tomanek et al. (2021) implemented residual adapters for atypical and accented speech recognition tasks with two state-of-the-art ASR architectures, namely Neural Network Transducers (RNN-T) and Transformer Transducers (T-T). The architecture of the residual adapters follows the same as Bapna et al. (2019) where layer normalization is applied to the input and this is followed by a down-projection feed-forward layer, a ReLU activation, and an up-projection feed-forward layer. They demonstrated that using dedicated adapters for atypical speech and accented speech recognition leads to reduced WERs compared to the unadapted models. Moreover, adapters are able to achieve similar performance to fine-tuning with significantly less number of parameters. Similar to Tomanek et al. (2021), Sathyendra et al. (2022) utilized adapters in Transducers and showed that contextual adapter-based training outperforms full fine-tuning for personalized speech recognition. Kannan et al. (2019) utilized the same residual adapters similar to Tomanek et al. (2021) and applied it to the multi-lingual RNN-T models and presented significant ASR improvement over the monolingual RNN-T models using nine Indic languages.

Adapters have been used for multi-domain ASR in a Transformer-based language model and this approach achieved a lower word error rate than a single music domain ASR model Lee et al. (2021). Their adapters follow the bottleneck structure with down-projection and up-projection.

For multiple tasks in the speech domain, bottleneck adapters with layer normalization, down-projection layer, ReLU, up-projection layer, and skip connection components were utilized by Eeckt and Van Hamme (2023). They used two types of encoders such as Transformer and Conformer to incorporate the adapters. In the transformer model, adapters are added between the multi-head self-attention and the feed-forward sub-layers. As for the Conformer model, adapters are inserted between the convolutional network and the last feed-forward network. The dimension of the adapter is only 32. While adapters being trained and the pre-trained backbone being frozen is a typical adapter-tuning approach, the authors extend the idea by proposing a two-stage approach. In the first stage, task-specific adapters are trained for a new task as usual (the rest of the pre-trained model is frozen). During the second stage, a ten-times smaller learning rate is used to tune the entire model. This approach has been found to alleviate the catastrophic forgetting issue of the direct sequential fine-tuning approach. In this work, six tasks are defined as ASR modeling for six English dialects from the United States, England, Australia, India, Scotland, and Ireland collected from the CommonVoice dataset.

Since pre-trained models contain a large number of layers, it is a non-trivial decision to make which pre-trained neural layers should be selected to incorporate the lightweight adapter modules. To fill the research gap, Huang et al. (2023) proposed a two-stage algorithm and evaluate it on ASR and spoken language understanding (SLU) tasks. In the initial phase, the model is trained for a specific downstream task using additional shallow, learnable layers and weight parameters. These enable a weighted summation of layer outputs in the context of self-supervised learning (SSL). This preliminary training identifies the most crucial layers based on their respective weights. In the subsequent stage, adapters are introduced into these essential layers, ensuring both performance preservation and efficacy in neural architecture search. The outcomes of the experiments conducted on the CommonVoice dataset reveal an impressive 20.6% absolute improvement in Word Error Rate (WER) for the Welsh language. This improvement is in contrast to the conventional approach of inserting adapter modules into the topmost layers without conducting a search. Additionally, in the context of the SLURP SLU task, the proposed method attains a notable 4% enhancement in intent accuracy compared to the conventional baseline.

Foundation models are trained on huge amounts of broad data that ensure better generalization and robustness to domain shifting. Domain-specific adapters are leveraged in foundation ASR models and the experimental result suggests that with a greatly reduced number of parameters, the same performance compared to full fine-tuning can be attained Li et al. (2023a). The same benefit of adapters in speech foundation models is confirmed by the work of Huo et al. (2023).

Adapters are utilized in personalized speech recognition in a multi-turn dialog setting Chang et al. (2023). Alexandridis et al. (2023) proposed gated contextual adapters for parameter-efficient contextual biasing in ASR transducer models.

Recently, Otake et al. (2023) proposed a novel adapter architecture for speech processing tasks by introducing two sets of adapters, namely, layer adapters (L-adapters) and encoder adapters (E-adapters). Within transformer-based self-supervised speech models, different speech representations are generated by distinct layers. L-adapters are able to extract relevant features from these diverse layer representations, thereby augmenting the relevance of subsequent tasks. In contrast, E-adapters improve the overall network's adaptability, resulting in a major improvement. By combining these two adapters, the performance improves compared to the bottleneck adapters.

Adapter-based training in pre-trained language models has recently been evaluated for other speech processing tasks besides ASR, including speaker verification Peng et al. (2022), text-to-speech Hsieh et al. (2022); Morioka et al. (2022), speech understanding Li et al. (2023b), and speech translation Gállego et al. (2021); Le et al. (2021).

3.3 ASR for Bengali and Maltese

While most of the state-of-the-art ASR algorithms are evaluated on English benchmarks, there is little research done in Bengali and Maltese.

3.3.1 Bengali

For a long time, the ASR research on Bengali was restricted to isolated spoken word recognition and digit recognition using small datasets Sultana et al. (2021). There have been recent developments of publicly available large-scale annotated Bengali speech corpora for large vocabulary continuous speech recognition (LVCSR) Sultana et al. (2021). There are mainly two standard Bengali accents namely Bangladeshi standard Bengali and West Bengal (a state in India) standard Bengali. For Bangladeshi standard Bengali, there are 229-hour-long large Bengali ASR Training Data (LB-ASRTD) crowd-sourced by Google Kjartansson et al. (2018) and 241-hour-long SUBAK.KO corpus Kibria et al. (2022). Both of them are read speech corpora. BABEL, a 215-hour-long telephone conversation-based speech corpus, is released in 2016 and contains West Bengal standard Bengali speeches. The Bengali part from the open source Common Voice (CV) dataset contains 1800 hours of recorded data among which 68 hours are validated Ardila et al. (2020).

Samin et al. (2021) performed a character-wise analysis of the LB-ASRTD corpus to evaluate its strengths and weaknesses using two ASR models such as DeepSpeech 2 and deep CNN Amodei et al. (2016). Ahmed et al. (2020) proposed an iterative algorithm to automatically develop weakly supervised speech corpora using existing ASR models. The resulting corpus contains around 1000 hours of transcribed Bengali speech data. However, the dataset is not validated under human supervision. More recently, Samin et al. (2023) developed BanSpeech, a 6-hour-long multi-domain Bengali ASR benchmark that contains both read speech and spontaneous speech from 13 diverse domains. They benchmarked several Bengali ASR models such as a self-supervised wav2vec 2.0, a weakly supervised Whisper, and a fully supervised CNN using BanSpeech. The requirement of a dedicated spontaneous speech corpus is suggested in the work. Shahgir et al. (2022) reported the performance of wav2vec 2.0 after fine-tuning it with the around 400-hour-long CV dataset.

3.3.2 Maltese

Earlier work on Maltese speech recognition was limited to isolated word recognition, particularly digit recognition Falzon (1993). Calleja (2003) explored HMM for acoustic modeling using mobile telephone conversation-based speech. The author reported a high word error rate (WER) of 96.8%. Several efforts were made to build Maltese speech datasets Gatt and Čéplö (2013); Vella and Farrugia (2006); Vella et al. (2010).

In 2020, an 8-hour-long MASRI-HEADSET corpus was developed Mena et al. (2020a). The recordings were collected from different geographical locations all over the Maltese Islands. Using the Kaldi neural networks (NN) recipe, a WER of 10.56% has been obtained on the test set. Several data augmentation techniques such as unsupervised training, multi-lingual training, and utilization of text-to-speech models to augment with synthesized data have been investigated from a low-resource perspective with Maltese DeMarco et al. (2023). The performance of data-augmented ASR for low-resource Maltese was found to be promising.

4 Datasets

This chapter provides an overview of the languages and datasets used in this work. Moreover, the preparation of datasets to train ASR models is described in this chapter.

4.1 Languages

We choose three languages such as Bengali, English, and Maltese from three diverse language groups with an aim to ensure that our findings are language-independent as much as possible.

Bengali, also known as Bangla, is an Indo-Aryan language in the Indo-European language family and spoken mainly in Bangladesh and India. There are around 235 million native speakers along with 40 million second language (L2) speakers of Bengali, making it the seventh most spoken language by the number of speakers in the world. Although a huge number of speakers, Bengali can be categorized as a mid-resource language for the automatic speech processing task. With recent developments in ASR datasets, there are more than 1000 hours of publicly available annotated speech datasets in Bengali. However, there is still a scarcity of research on Bengali despite possessing many challenges with developing ASR due to its complex morphological parsing. Bengali is a low-resource language for many natural language and speech processing tasks such as keyword spotting, automatic forced alignment, etc, to name a few.

English is a West Germanic language in the Indo-European language family and it is the most widely spoken language in the world with around 372.9 million native speakers and 1.08 billion L2 speakers. Most of the state-of-the-art ASR algorithms are benchmarked with English data and the performance of the English ASR models is getting closer to human-level speech recognition, thanks to the abundance of datasets.

Maltese is a Semitic language from the Afro-Asiatic language family. There are around 530,000 number of native speakers in Malta and this is the only official Semitic language of the European Union. Unlike some European languages such as English, German, French, etc, Maltese is a low-resource language with only around less than 80 hours of ASR data.

Similar to Bengali, Maltese is also a morphologically-rich language (MRL), presenting challenges to building robust ASR systems.

4.2 Selection and Preparation of Datasets

To build robust ASR systems, a high-quality annotated speech dataset is a prerequisite. Here, annotation refers to the process that the speech utterances must be labeled with the corresponding written texts, both of which will be used to train the ASR models in a supervised way.

4.2.1 Bengali

We use SUBAK.KO, a 241-hour long annotated speech dataset for Bengali ASR Kibria et al. (2022). There are 229 hours of clean read speech and 12 hours of broadcast speech in SUBAK.KO, making it a mostly read speech corpus. The speech utterances are collected in a noise-proof studio from 33 native Bangladeshi Bangla male speakers, 28 native Bangladeshi Bangla female speakers, and 2 L2 speakers. To ensure regional accent variability in SUBAK.KO, the speakers are selected from all eight divisions from Bangladesh. 40 text domains are considered in SUBAK.KO to meet the principal corpus design criteria e.g. reception and production criteria Atkins et al. (1992). The details of SUBAK.KO can be found in the work of Kibria et al. (2022).

The standard SUBAK.KO train, dev, and test set contain 200.3 hours, 20.5 hours, and 20.3 hours of speech, respectively. We use the same splits in order to ensure reproducibility and enable comparison between previous approaches. For our experiments, we create several subsets by splitting the 200-hour-long train set. There are 5 subsets including 10 minutes, 1 hour, 10 hours, 20 hours, and 50 hours long subsets. Random sampling is performed to create the subsets. To make sure that the results of a subset are unbiased due to random sampling, we include the samples of the smaller subset to the larger subset e.g. samples of the 10-minute subset to the 1-hour subset, the samples of the 1-hour subset to the 10-hour subset, and so on. The train, dev, and test sets are speaker-dependent, meaning the same speaker can be found in all three sets due to random sampling. The samples of the 12-hour-long broadcast speech can also be found in the three splits. There are no standard noisy dev and test benchmarks for SUBAK.KO as of July 2023.

4.2.2 English

For English, we use the widely used LibriSpeech, containing around 1000 hours of annotated speech data and the LibriLight benchmarks. LibriSpeech is obtained from the LibriVox project which has read speech samples from audiobooks. LibriSpeech has three standard subsets with 100 hours, 360 hours, and 500 hours of data, respectively. While developing LibriSpeech, the speech samples of the audiobooks are automatically aligned and segmented, and noisy transcripts are filtered out to ensure a clean read speech corpus for English ASR. LibriSpeech contains a large number of both male and female speakers.

LibriLight is derived from the LibriSpeech corpus and it has three standard supervised subsets with 10 min, 1 hour, and 10 hours of data. In our experiments, we use the standard 10 min, 1-hour, and 10 hours subsets from LibriLight. Babu et al. (2021) conducted the experiments on XLS-R and Thomas et al. (2022) investigated the adapter tuning for speech recognition using these subsets. Moreover, we split the 100-hour LibriSpeech subset into two subsets containing 20 hours and 50 hours of data by random sampling. Thus, similar to Bengali subsets, we have 5 subsets for English.

We use the widely used LibriSpeech dev and test benchmarks for evaluation. Each of the dev and test sets contain a clean set and a noisy (other) set. The dev clean set and dev other set consist of 5.4 hours and 5.3 hours of speech, respectively. There are 5.4 hours and 5.1 hours of speech in the test clean and test other sets, accordingly.

4.2.3 Maltese

For experimenting with Maltese, we combine several Maltese datasets including Common Voice (CV) dataset (10 hours), MASRI-HEADSET (6.4 hours), MEP (1.2 hours), Tube (13.20 hours), MERLIN (19.4 hours), and Parlament (2.3 hours) to create the train set. Our final training dataset amounts to 50 hours of data. CV is an initiative led by Mozilla to build transcribed speech datasets for ASR through crowd-sourcing for as many as 112 languages as of July 2023. The campaign is still ongoing, resulting in a total of 28,118 recorded hours, among which 18,652 hours are validated. The MASRI-HEADSET corpus, developed at the University of Malta, contains a total of 8 hours of read speech. The dataset is recorded from 25 speakers (13 female, 12 male). The speakers are selected from geographical locations all over the Maltese Islands. Parliament speech corpus is created by downloading high-quality videos from the Maltese Parliament website. During annotation, only read speeches are kept while filtering out the spontaneous ones. The dataset is not gender-balanced with 23 male speakers and three female speakers.

The dev and test sets have 2.3 hours and 1.5 hours of speeches, accordingly. The test set is composed of two subsets including Test MASRI (1 hour) and Test CV (0.5 hour).

Similar to Bengali and English ASR experiments, we prepare 5 subsets for Maltese with 10 min, 1 hour, 10 hours, 20 hours, and 50 hours of speech. The subsets are generated by performing random sampling. To mitigate any unwanted bias due to random sampling, we always include the smaller subset in the larger ones, similar to what we did for Bengali subsets.

Table 4.1 presents the summary of the key dataset statistics for this work.

Table 4.1: The datasets are split into train, dev and test sets. BN, MT, and EN refer to Bengali, Maltese, and English, respectively. For English, each of dev and test sets has clean and other (noisy) versions.

Lang	**Language Group**	**Datasets**	**train set length**	**dev set**		**test set**	
				clean	**other**	**clean**	**other**
BN	Indo-Aryan	SUBAK.KO Kibria et al. (2022)	200.3	20.5	-	20.3	-
MT	Semitic	Common Voice Ardila et al. (2020) MASRI-HEADSET Mena et al. (2020b) MEP Tube MERLIN Parlament	52.5	2.3	-	1.5	-
EN	West Germanic	LibriSpeech Panayotov et al. (2015)	960.9	5.4	5.3	5.4	5.1

5 Methodology

In this chapter, we explain the architecture of the wav2vec 2.0 incorporating the adapter modules and provide details on the experimental setup.

5.1 Architecture of the network

5.1.1 Inserting Adapters into Transformer

Figure 5.1 shows the architecture of the adapter-based wav2vec 2.0 model. In the left figure, two adapter modules are inserted into each Transformer block in the contextual network while keeping the core architecture of wav2vec 2.0 as it is. At first, the raw input signal is passed to the feature encoder composed of 7 convolutional blocks. The output of the feature encoder is at first masked at the proportion rate of p and then fed as input to the Transformer-based contextual network. The architectures of the feature encoder and the quantization module follow the regular wav2vec 2.0 model Baevski et al. (2020) (See details in section 2.3). Each transformer block contains several sub-blocks/sub-modules. MHSA refers to the multi-head self-attention block that leverages scaled dot product attention and 16 attention heads. Then, dropout is applied to the output of MHSA. After that, the first adapter module is inserted. Add & Norm module represents the residual connection and the layer normalization. Formally, let the input to the MHSA be x, and the output of the first adapter module be a. So, the Add & Norm output is defined as,

$$Add\&Norm = LayerNorm(x + a) \tag{5.1}$$

The task of the Add & Norm layer is adding the input of the MHSA module and the output of the adapter module and then passing the added vector through layer normalization. The output of the add & norm block is given as input to the single fully connected feed-forward network. Then, the second and final adapter module is inserted. Finally, similar to the previous residual connection and layer normalization, the output of the first layer normalization is added to the output of the second adapter, and layer normalization is applied to the resultant vector to obtain the final output vector from the transformer block.

There can be N number of transformer blocks where N can be either 24 or 48 depending upon the specific wav2vec 2.0 model. For example, XLS-R with 0.3B parameters contains 24 transformer blocks while both XLS-R models with 1B parameters and 2B parameters have 48 transformer blocks. There are 48 transformer blocks in the MMS 1B model.

At the end of the network, a linear classifier, also referred to as language modeling head, with the dimension equal to the number of token sets is added. The linear classifier is trained using the CTC loss function Graves et al. (2006).

During adapter-tuning, the feature encoder, MHSA, and FC blocks are kept frozen while the two adapter modules from each transformer block along with the layer normalization layers and the language modeling head are trained. It reduces the number of total trainable parameters of the wav2vec 2.0 by a large margin.

5.1.2 Designing Adapters

On the right side from Figure 5.1, the architecture of the bottleneck adapter is displayed. In the standard bottleneck adapter, there are two fully-connected feed-forward networks. There is the first fully connected layer that acts as a down-sampler to project the Transformer model dimension to a specific lower dimension. This process is called down-projection. The projected lower dimension is also referred to as the inner dimension of the feed-forward network. Then, a Gaussian error linear unit (GELU) activation function is added Hendrycks and Gimpel (2016). The second fully connected layer works as an up-sampler to project to the original Transformer model dimension. The inner dimension of both of the fully connected layers is kept at 256. These fully connected layers are initialized with almost zero values. Finally, there is a residual connection that adds the output of the second fully connected feed-forward layer to the original input representation to the adapter and passed through layer normalization to get the final output. Let the Transformer model dimension be d_m and the output of the second FC layer is f_m. Both d_m and the f_m have the same dimension of m. The output of the Add & Norm layer is computed as,

$$AdapterOutput = LayerNorm(d_m + f_m) \tag{5.2}$$

The architecture of the adapters and the position of them at the transformer block can be modifiable. We use this specific setting of the standard bottleneck adapters positioned at these exact positions following the existing works in NLP and computer vision and

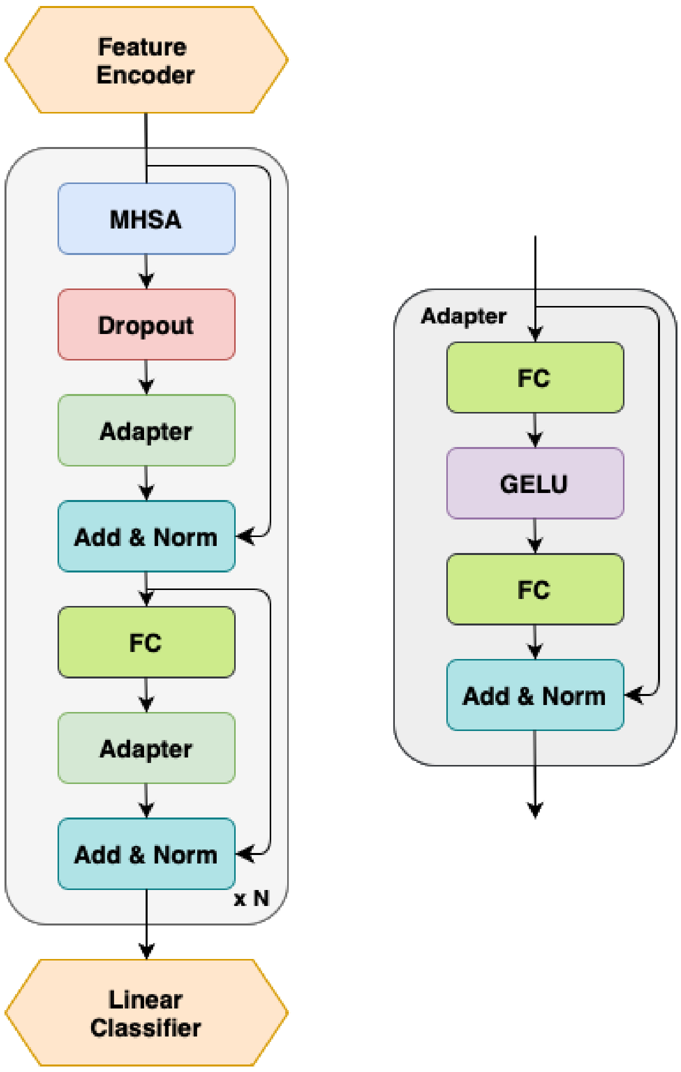

Figure 5.1: The left figure shows the architecture of the wav2vec 2.0 combined with two adapter modules. The two adapter modules are located after multi-head self-attention (MHSA) and feed-forward sub-layer, respectively. Add & Norm refers to the residual connection and layer normalization component. FC corresponds to the fully connected feed-forward network. The right figure depicts a bottleneck adapter with down-projection and up-projection layers, GeLU activation, layer normalization, and residual connection.

call it our baseline adapter architecture. However, as an ablation study, we scale the number of fully connected layers within an adapter module to evaluate its impact on overall performance.

5.2 Experimental Setup

There exists the original wav2vec 2.0 implementation released by PyTorch and the HuggingFace wav2vec 2.0 implementation which provides an API that makes the training procedure much easier. We modify the HuggingFace implementation of wav2vec 2.0 to run the experiments with the adapter-tuning approach. For full fine-tuning, we also make use of the HuggingFace implementation. We investigate four wav2vec 2.0 models including XLS-R 300M, XLS-R 1B, XLS-R 2B, and MMS 1B. Each of these four models is both fully fine-tuned and adapter-tuned to compare the benefits and drawbacks of these two methods. Moreover, we use five subsets (10 min, 1 hour, 10 hours, 20 hours, and 50 hours) in each of the Maltese, Bengali, and English datasets and employ the two approaches for a thorough investigation.

We choose the XLS-R model for our experiments since XLS-R has been wide-recognized and successfully applied for cross-lingual ASR for many low-resource languages as well as the high-resource ones Babu et al. (2021). Moreover, XLS-R offers models with different capacities including 0.3B, 1B, and 2B parameters. XLS-R is pre-trained in as many as 128 languages and all three languages in our experiments such as English, Bengali, and Maltese are present during XLS-R pre-training. However, the training data distribution for each of these languages is imbalanced in XLS-R. This pre-trained model enables a solid baseline to compare with the previous approaches.

We also leverage the pre-trained MMS model which has been released very recently at the time of this thesis. MMS covers comparatively a wider selection of languages of 1,406 languages, making this speech model more cross-lingual. MMS offers a model capacity of 1B parameters. By choosing MMS for our experiments, we can evaluate this pre-trained model with the widely-used XLS-R model, both of which are cross-lingual.

The audio files of the three languages are sampled at 16K Hertz (Hz). We use the waveform audio file format (in short, wav) to store the audio files. The English LibriSpeech dataset originally contains audio files in FLAC (free lossless audio codec) format, which we convert to wav format after downloading them. The Maltese and Bengali datasets are pre-processed in wav format, so we do not require to convert them. Mono channel is used and a bit-rate of 256 is set for the audio files. The absolute paths of the audio files along with the corresponding transcription files are saved in a tab-separated values (tsv) file. These labeled audio-text pairs from the tsv file are used to train the ASR models.

A token set, consisting of the symbols of a language, is prepared for each language before initiating the full fine-tuning/adapter-tuning. There are 54 tokens, 113 tokens, and 32 tokens for Maltese, Bengali, and English, respectively. Four special tokens such as <*unk*> (unknown), <*pad*> (padding), <*s*> (starting of a sentence), and </*s*> (ending of a sentence) are included to these token sets. We use orthographic tokens (characters) as tokens in our experiments. Bengali is a morphologically rich language with a considerably large number of characters. Moreover, in our Bengali dataset, a small number of words are transcribed in English to represent the code-switching scenario that is highly observed in modern Bengali. And so we include the English alphabet in our Bengali token list. As a result of these two reasons, the number of tokens for Bengali is much higher than the Maltese and English token sets.

For the fully fine-tuning approach, we start fine-tuning the model with a learning rate of 3e-5 and keep the first 300 steps as warm-up steps. With the adapter-tuning approach, we use a learning rate of 5e-5 and set the first 400 steps as warm-ups. The maximum number of steps is set to 150K for both full fine-tuning and adapter-tuning. We finish training a model if no improvement is seen for the 10 consecutive epochs in both methods. The best model is chosen based on the WER. The lower the WER, the better the performance of the ASR system is.

In the original XLS-R works, it has been reported that a very large batch size is highly effective in reducing the WER Babu et al. (2021). They use an effective batch size of roughly 0.44 hours for XLS-R 0.3B and XLS-R 1B and 1.06 hours for XLS-R 2B. However, the infrastructure needed to run the experiments with such a very high batch size is implausible for many of the low-resource languages. Due to our limitation to the access of very high computational resources and also to depict the low-resource scenario, we run our experiments with smaller batch size. We use a batch size of 4 samples for both full fine-tuning and adapter-tuning. Gradients are accumulated for every two steps and so the effective batch size is 8 in both approaches. A very rough estimate of our effective batch size is 64 seconds considering that the length of each sample is 8 seconds.

To reduce memory usage, we use mixed precision training (FP16) for both full fine-tuning and adapter-tuning. We set the seed as 100 to ensure the reproducibility of our experiments. Table 5.1 summarizes some of the important hyper-parameters used to run the experiments.

We always run the experiments with a single GPU. Several models of GPUs are used such as A100 80 GB, A100 40 GB, RTX 3090 24 GB, and RTX A6000 48 GB. The train set is used for training the models, the dev set to monitor the losses and WERs during training and the test set for final evaluation. For English models, only the dev clean set is

Table 5.1: Some of the hyper-parameters used for full fine-tuning and adapter-tuning with wav2vec 2.0

Hyper-parameter	Full Fine-tuning	Adapter-tuning
learning rate	3e-5	5e-5
batch size	4	4
gradient accumulation	2	2
max steps	150K	150K
warm up steps	300	-
early stopping patience	10	10
lr scheduler type	cosine	-
fp16	true	true
seed	100	100
criterion	CTC	CTC
sample rate	16K	16K

used as the dev set.

We use greedy search decoding using CTC to obtain the output tokens. Although it has been studied that external language models (e.g. N-gram language model or neural network-based language model) and beam search decoding can greatly help improve the performance of the ASR models Samin et al. (2021), we do not incorporate them since external language models can have its own biases reflecting on the results. Moreover, the goal of this study is to assess the adapter-tuning approach in comparison with full fine-tuning for ASR. Thus, beam search decoding is not within the scope of this work.

Code implementation There exists the publicly available code implementing the wav2vec 2.0 architecture by Fairseq authors and the HuggingFace team. This standard code is utilized in both XLS-R and MMS models. The only difference between these two models is the pre-training data. XLS-R covers 128 languages while MMS is pre-trained on a larger amount of data covering 1,406 languages. This code contains several classes written in PyTorch, implementing the feature encoder, quantization module, feature projection, attention, feed-forward, positional conv embedding, and encoder layer, among others. To implement the adapter-tuning approach in the wav2vec 2.0 architecture, modification is required in the *Wav2Vec2Encoder* class. As depicted in Figure 5.1, two adapters are placed after the multi-head self-attention, and the feed-forward component, respectively. For that, within the *Wav2Vec2Encoder* class, at first, two bottleneck adapters are initiated in the *init* function, and in the *forward* function, they are incorporated at the specified places. During the initialization and specification of adapter architecture, PyTorch-provided pre-defined implementations of feed-forward networks, layer norm, and GELU non-linear activation function are used.

PyTorch provides the *requires_grad* variable which can be either *True* or *False*. When the value is set to *True*, the parameters are updated using back-propagation, otherwise, the parameters are kept frozen. Using this variable, parameters of different components e.g. feature encoder, encoder layer, quantization module, etc. can be either updated or frozen. For the parameters of adapters, and layer normalization, this variable is set to *True*. For the rest of the components of wav2vec 2.0, the parameters are frozen. There exist pre-defined functions in the wav2vec 2.0 code that enables this feature of either updating or freezing the parameters. We add such a function dedicated to the adapter module.

6 Results and Discussion

In this chapter, we are going to present the results of our experiments with fine-tuning and adapter-tuning using wav2vec 2.0 for ASR. Moreover, we briefly discuss the results in order to explain the key findings to facilitate research for parameter-efficient ASR. This chapter is divided into multiple sections for better convey of our experimental outcomes.

6.1 Effect of Train set Size on Full Fine-tuning vs Adapter-tuning

We observe that both full fine-tuning and adapter-tuning offer distinct advantages depending on the dataset size (see Table 6.1). In this experiment, we create five training sets containing 10 minutes, 1 hour, 10 hours, 20 hours, and 50 hours of data for each of the languages, Maltese, Bengali, and English. We explore both full fine-tuning and adapter-tuning techniques using various XLS-R models (0.3B, 1B, and 2B) and the MMS 1B model.

In cases of extremely low-resource scenarios, where the training data is limited to around 10 minutes or 1 hour, we find that fine-tuning significantly outperforms adapter-tuning. This trend holds across all languages and model sizes. This specific setting also depicts few-shot learning since we fine-tune the models on only 10 minutes or 1 hour of labeled speech. Thus, adapter-tuning is not advantageous for few-shot learning.

However, as we move to moderately low-resource scenarios, with 10 hours or more data, we observe that the performance of adapter-tuning is on par with the fine-tuning approach while remarkably reducing the number of trainable parameters. For English ASR, the best-performing model fine-tuned with 10 hours of labeled speech data achieves a WER of 7.6% on the LibriSpeech test clean set using the adapter-tuning approach, while the fully fine-tuned wav2vec 2.0 model on the same English data achieves the best WER of 8.3%. Conversely, with Maltese and Bengali, full fine-tuning of the wav2vec 2.0 architecture exhibits slightly better performance than adapter-tuning.

As we increase the dataset size to 20 hours and 50 hours, the WER gap between fine-tuning and adapter-tuning is still minimal and both approaches perform competitively.

Table 6.3 presents the WERs for Bengali and English in a mid-resource scenario in which 200 hours and 360 hours of train data are present. We leverage the XLS-R 2B model for both fine-tuning and adapter-tuning. Adapter-tuning yields substantially lower WERs than fine-tuning on both SUBAK.KO and LibriSpeech dev and test sets. Therefore, adapter-tuning is well-suited to develop ASR models using a considerably large amount of data.

The character error rates (CERs) of the fully fine-tuning and adapter-tuning approaches for different training subsets and model sizes are shown in Table 6.2. We observe that the CER evaluation follows the same trend as the WER evaluation.

One of the notable benefits of adapters is the significant reduction in trainable parameters. For instance, with XLS-R 0.3B, the number of trainable parameters decreases from 315M to 26M through adapter tuning. Similarly, for XLS-R 1B, XLS-R 2B, and MMS 1B, the trainable parameters reduce to only 64M from the original 962M, 2B, and 962M, respectively. As a result, the storage space of the models also decreases to approximately 4.9 GB for adapter tuning compared to 9.1 GB for full fine-tuning.

In short, we observe that fine-tuning is a better approach for extremely low-resource cases. However, in a moderately low-resource setting, adapter-tuning performs competitively to full fine-tuning while substantially reducing the number of trainable parameters. Given a considerably large amount of labeled speech data, adapter-tuning clearly exhibits much better performance.

Overall, adapter-tuning shows a trade-off where it achieves comparable WERs with significantly reduced trainable parameters and model storage size, making it a promising approach in certain low-resource scenarios such as 10 hours to 50 hours of Maltese and Bengali data. As for error rate, for English, we obtain even lower WERs using an adapter-based approach. The performance of adapter-tuning improves with a large amount of data as seen in Table 6.3. However, when the training data is limited to 1 hour or less, fine-tuning is still the best technique.

Table 6.1: **Evaluation of full fine-tuning (FT) and adapter-tuning with XLS-R and MMS models for low-resource ASR in terms of WERs (%).** We study pre-trained ASR models with a varied number of trainable parameters (0.3B, 1B, and 2B). Three languages, namely Maltese (MT), Bengali (BN), and English (EN) from three diverse language groups are chosen. Results are reported on both dev and test sets. CTC-based greedy decoding is used.

Dataset size	Approach	Model	Trainable params	MT		BN		EN			
				dev	test	dev	test	dev clean	dev other	test clean	test other
10 min	FT	XLS-R 0.3B	315M	59.8	63.6	72.4	70.2	40.3	50.8	39.3	48.4
		XLS-R 1B	962M	68.1	70.5	71.0	70.4	37.8	48.7	**36.1**	45.7
		XLS-R 2B	2B	61.9	62.9	72.5	69.9	41.5	51.4	39.4	49.0
		MMS 1B	962M	56.2	**60.5**	67.2	**64.2**	37.6	46.2	36.5	**43.4**
	adapter	XLS-R 0.3B	26M	98.9	98.9	94.1	93.7	100.0	100.0	100.0	100.0
		XLS-R 1B	64M	91.0	90.3	90.4	89.9	97.7	100.0	98.3	100.0
		XLS-R 2B	64M	93.7	93.5	88.1	87.6	91.1	97.5	91.8	96.2
		MMS 1B	64M	89.2	89.0	100.0	100.0	100.0	100.0	100.0	100.0
1 hour	FT	XLS-R 0.3B	315M	37.1	**43.1**	48.7	46.2	17.5	27.2	16.7	25.5
		XLS-R 1B	962M	44.0	48.1	45.3	**44.3**	16.5	27.2	**15.5**	24.9
		XLS-R 2B	2B	40.5	43.9	50.5	47.5	19.4	29.9	17.9	27.6
		MMS 1B	962M	37.9	43.5	47.0	**44.3**	17.4	26.5	16.1	**23.9**
	adapter	XLS-R 0.3B	26M	60.9	65.4	63.8	63.4	85.7	93.0	86.0	92.2
		XLS-R 1B	64M	60.8	63.4	57.8	56.9	38.2	52.1	38.2	53.3
		XLS-R 2B	64M	98.7	98.2	66.7	66.2	24.0	35.6	24.2	36.3
		MMS 1B	64M	58.8	61.9	59.3	58.5	34.7	48.9	34.9	49.4
10 hours	FT	XLS-R 0.3B	315M	19.6	**27.8**	22.2	20.1	9.4	18.4	8.7	17.2
		XLS-R 1B	962M	20.2	28.6	20.8	19.8	9.1	19.1	8.3	17.4
		XLS-R 2B	2B	26.5	31.1	22.3	20.2	11.1	22.5	10.1	20.1
		MMS 1B	962M	28.2	35.0	20.8	**18.8**	9.6	18.7	8.7	16.7
	adapter	XLS-R 0.3B	26M	29.0	34.8	27.6	26.9	10.5	21.5	10.7	21.6
		XLS-R 1B	64M	23.9	29.4	21.6	21.0	9.1	18.2	9.3	18.3
		XLS-R 2B	64M	22.6	31.0	26.7	25.7	7.5	15.4	**7.6**	**15.7**
		MMS 1B	64M	28.9	36.1	28.9	28.4	9.1	17.8	9.2	18.1
20 hours	FT	XLS-R 0.3B	315M	17.4	26.0	16.8	15.2	7.5	17.7	7.1	16.3
		XLS-R 1B	962M	18.8	26.2	19.3	18.6	7.8	19.8	7.1	18.2
		XLS-R 2B	2B	23.1	28.2	15.5	**13.7**	8.2	19.6	7.4	18.2
		MMS 1B	962M	18.3	26.5	15.5	13.9	8.2	18.6	7.5	16.5
	adapter	XLS-R 0.3B	26M	21.7	28.2	17.8	17.4	7.8	19.7	8.0	19.8
		XLS-R 1B	64M	20.6	26.5	25.4	25.0	6.6	16.4	6.8	16.7
		XLS-R 2B	64M	17.1	**25.6**	16.3	15.8	5.8	14.3	**6.0**	**14.9**
		MMS 1B	64M	24.1	30.2	18.7	18.0	7.2	16.2	7.5	16.2
50 hours	FT	XLS-R 0.3B	315M	15.4	24.5	13.9	12.4	6.1	15.5	5.8	14.1
		XLS-R 1B	962M	12.3	**21.1**	12.1	**10.9**	6.5	18.0	6.0	16.2
		XLS-R 2B	2B	18.4	24.4	21.8	19.8	7.0	18.9	6.4	17.3
		MMS 1B	962M	12.7	21.5	13.5	12.1	6.7	16.5	6.0	14.8
	adapter	XLS-R 0.3B	26M	19.1	26.2	14.7	14.6	6.3	16.5	6.4	16.2
		XLS-R 1B	64M	18.0	24.9	13.3	12.9	5.0	12.9	**5.1**	12.7
		XLS-R 2B	64M	15.4	23.9	11.6	11.3	5.0	12.7	5.3	12.9
		MMS 1B	64M	23.4	29.9	13.8	13.2	5.6	12.9	5.5	**12.6**

Table 6.2: **Evaluation of full fine-tuning (FT) and adapter-tuning with XLS-R and MMS models for low-resource ASR in terms of CERs (%).** We study pre-trained ASR models with a varied number of trainable parameters (0.3B, 1B, and 2B). Three languages, namely Maltese (MT), Bengali (BN), and English (EN) from three diverse language groups are chosen. Results are reported on both dev and test sets. CTC-based greedy decoding is used.

Dataset size	Approach	Model	Trainable params	MT		BN		EN			
								dev		test	
				dev	test	dev	test	clean	other	clean	other
10 min	FT	XLS-R 0.3B	315M	17.1	19.9	25.8	23.5	13.6	20.1	12.8	18.4
		XLS-R 1B	962M	26.0	28.1	24.0	23.0	13.1	20.2	11.9	17.9
		XLS-R 2B	2B	18.6	18.5	25.1	22.7	14.2	20.8	12.9	18.8
		MMS 1B	962M	15.5	18.1	22.9	20.6	12.4	17.8	11.5	15.9
	adapter	XLS-R 0.3B	26M	42.3	43.8	43.5	43.1	100.0	100.0	100.0	100.0
		XLS-R 1B	64M	35.5	36.3	36.3	35.6	50.7	57.4	50.5	57.1
		XLS-R 2B	64M	32.3	34.1	34.2	33.3	38.2	46.1	38.1	45.7
		MMS 1B	64M	34.7	35.7	85.7	85.4	49.0	56.1	48.9	55.0
1 hour	FT	XLS-R 0.3B	315M	10.0	12.6	15.0	13.1	5.5	10.5	5.0	9.3
		XLS-R 1B	962M	15.1	16.8	14.1	13.0	5.2	10.7	4.6	9.2
		XLS-R 2B	2B	11.5	12.6	15.9	13.8	6.1	11.8	5.3	10.2
		MMS 1B	962M	10.3	12.8	14.6	12.8	5.5	10.3	4.8	8.6
	adapter	XLS-R 0.3B	26M	18.5	21.4	21.1	20.1	32.3	40.0	32.2	40.3
		XLS-R 1B	64M	18.4	20.0	18.2	17.2	12.4	20.4	12.4	20.8
		XLS-R 2B	64M	39.9	40.9	20.6	19.6	7.4	13.3	7.4	13.5
		MMS 1B	64M	17.9	19.6	18.3	17.3	11.4	19.4	11.4	19.3
10 hours	FT	XLS-R 0.3B	315M	5.5	7.8	6.7	5.3	3.0	7.4	2.7	6.5
		XLS-R 1B	962M	6.6	8.9	6.2	5.2	3.0	7.8	2.5	6.6
		XLS-R 2B	2B	7.9	8.9	6.9	5.5	3.6	9.2	3.1	7.7
		MMS 1B	962M	8.4	10.8	6.6	5.1	3.1	7.7	2.7	6.4
	adapter	XLS-R 0.3B	26M	8.4	10.4	8.2	7.2	3.3	8.5	3.3	8.3
		XLS-R 1B	64M	7.1	8.6	6.9	6.1	2.9	7.1	2.9	7.0
		XLS-R 2B	64M	6.3	8.8	7.8	6.8	2.3	5.9	2.3	5.9
		MMS 1B	64M	8.8	10.7	8.2	7.5	2.8	6.9	2.9	6.8
20 hours	FT	XLS-R 0.3B	315M	4.9	7.5	5.4	4.2	2.5	7.1	2.2	6.1
		XLS-R 1B	962M	6.0	8.1	5.9	5.0	2.5	8.2	2.2	7.1
		XLS-R 2B	2B	7.1	8.4	5.2	3.9	2.7	7.9	2.3	7.1
		MMS 1B	962M	5.2	7.7	5.2	3.9	2.7	7.5	2.3	6.2
	adapter	XLS-R 0.3B	26M	6.5	8.4	5.7	4.9	2.5	7.7	2.5	7.7
		XLS-R 1B	64M	6.1	7.7	7.5	6.8	2.0	6.3	2.1	6.3
		XLS-R 2B	64M	4.8	7.3	5.2	4.5	1.8	5.5	1.8	5.7
		MMS 1B	64M	7.2	9.0	5.9	5.0	2.3	6.3	2.3	6.0
50 hours	FT	XLS-R 0.3B	315M	4.3	6.7	4.6	3.5	2.0	6.1	1.7	5.3
		XLS-R 1B	962M	4.2	6.6	4.3	3.3	2.1	7.5	1.8	6.3
		XLS-R 2B	2B	5.4	6.9	6.2	4.8	2.2	7.7	1.9	6.6
		MMS 1B	962M	3.7	6.1	4.7	3.8	2.2	6.7	1.8	5.6
	adapter	XLS-R 0.3B	26M	5.7	7.5	4.8	4.3	1.9	6.3	1.9	6.1
		XLS-R 1B	64M	5.4	7.0	4.5	3.8	1.5	4.9	1.5	4.7
		XLS-R 2B	64M	4.3	6.7	4.4	3.8	1.5	4.8	1.5	4.7
		MMS 1B	64M	6.7	8.6	4.4	3.5	1.7	5.0	1.6	4.6

Table 6.3: Evaluation of full fine-tuning and adapter-tuning with XLS-R 2B for a moderately large amount of data of 200 hours from SUBAK.KO for Bengali (BN) and 360 hours from LibriSpeech for English (EN). Results are reported in terms of WERs (%) on both dev and test sets. CTC-based greedy decoding is used.

Language	**train set length (hr)**	**Model**	**Approach**	**dev set**		**test set**	
				clean	**other**	**clean**	**other**
BN	200	XLS-R 2B	fine-tuning	18.8	-	16.3	-
			adapter-tuning	8.1	-	6.9	-
EN	360	XLS-R 2B	fine-tuning	6.4	17.9	5.8	15.7
			adapter-tuning	3.5	9.4	3.7	9.4

6.2 Larger pre-trained models tend to benefit from adapter-tuning

From Table 6.1, it is evident that the fully fine-tuned XLS-R model with 2B parameters yields relatively high WERs across all three languages and various sizes of training datasets. More specifically, the fully fine-tuned XLS-R 2B is outperformed by either of the fully fine-tuned XLS-R 0.3B, XLS-R 1B, or MMS 1B for all languages and dataset sizes in our study. For example, on the LibriSpeech test clean set, XLS-R 2B gets WERs of 39.4%, 17.9%, 10.1%, 7.4%, and 6.4% after fully fine-tuned with 10 minutes, 1 hour, 10 hours, 20 hours, and 50 hours of labeled training data, respectively. On the other hand, XLS-R 1B with lower model capacity achieves 36.1%, 15.5%, 8.3%, 7.1%, and 6.0% WERs, outperforming the XLS-R 2B with the same five training subsets, accordingly, on the LibriSpeech test clean set.

To further investigate this issue, we refer to the original work by Babu et al. (2021), where several experiments are conducted using the LibriSpeech benchmark and fine-tuned XLS-R 2B. However, in that study, the performance of the English LibriSpeech ASR was reported only for the English mono-lingual wav2vec 2.0 LV-60K (0.3B), XLS-R 0.3B, and XLS-R 1B models, with no results presented for the XLS-R 2B model. Although the authors argue that models with larger capacity can avoid the interference problem of the pre-trained models and achieve lower WERs, as seen for XLS-R 1B performing better than XLS-R 0.3B, this hypothesis is not further evaluated by fine-tuning an XLS-R 2B model Babu et al. (2021). Moreover, it is commonly assumed in the community that larger pre-trained models achieve superior performance Brown et al. (2020), but our empirical findings demonstrate the opposite in the case of XLS-R 2B.

In contrast, employing the adapter-tuning approach enables the XLS-R with 2B parameters to achieve the lowest WERs across several dataset sizes except for the extremely low-resource ones (e.g. 10 min or 1 hour). For instance, with the English 10-hour train subset, the XLS-R 2B outperforms all other fully fine-tuned and adapter-tuned models, achieving a WER of 7.6% on the LibriSpeech test clean benchmark. While we observe significant WER differences between full fine-tuning and adapter-tuning for smaller pre-trained models with 0.3B and 1B parameters, the XLS-R 2B, combined with adapter modules, consistently demonstrates superior performance compared to other models in most cases. Particularly for datasets with 10 hours or more of English and Maltese data, XLS-R 2B with adapters consistently exhibits lower WERs than the fully fine-tuned XLS-R 2B model. The benefit of a larger XLS-R model is also strengthened by our experiments with large-scale Bengali and English training data where the adapter-tuning still outperforms fine-tuning by a large margin (See Table 6.3).

This finding is noteworthy since the improved performance of the larger pre-trained model when supplemented with adapters, comes with a substantial reduction in trainable parameters (from 2B to 64M). We propose the hypothesis that adapters act as regularizers in large pre-trained speech models, effectively leveraging the potential of these models.

6.3 Adapter-tuning-based ASR for different languages

In this section, we examine the WERs of both full fine-tuning and adapter-tuning for three languages: Maltese (MT), Bengali (BN), and English. These languages are selected because they represent distinct language families, with Maltese being Semitic, Bengali Indo-Aryan, and English West Germanic. Additionally, English is considered a high-resource language, while Bengali and Maltese are considered mid-resource and low-resource languages, respectively.

Analyzing Table 6.1, we observe a consistent trend in Maltese and English, in which the performance of adapter-tuning is comparable with full fine-tuning in most cases. Specifically, with 10 hours or more of English data, adapter-tuning achieves lower WERs on the LibriSpeech dev and test sets compared to full fine-tuning. For Maltese, we find competitive WERs between full fine-tuning and adapter-tuning. Interestingly, the XLS-R 2B model demonstrates its potential, particularly in Maltese and English. However, for Bengali, full fine-tuning achieves better performance than adapter-tuning across all training dataset sizes. Furthermore, the performance of the XLS-R 2B model, when combined with adapters, is sub-optimal for all Bengali training sets, except for the 50-hour one.

Table 6.4: Number of hours of English, Maltese, and Bengali untranscribed speech data used for pre-training XLS-R Babu et al. (2021)

Language	**ISO**	**XLS-R pre-training data**
English	en	69493 hours
Maltese	mt	9120 hours
Bengali	bn	100 hours

To investigate the probable reason for the comparatively poorer performance of adapter-tuning in Bengali compared to English and Maltese, we refer to Table 6.4, which displays the number of hours of English, Maltese, and Bengali data used for pre-training the XLS-R models Babu et al. (2021). It reveals that XLS-R was pre-trained with 69,493 hours and 9,120 hours of data for English and Maltese, respectively. However, for Bengali, only 100 hours of data were used for pre-training XLS-R. We argue that the inadequate representation of Bengali in the pre-trained XLS-R model, compared to English and Maltese, might be the reason for the sub-optimal and less consistent performance of adapter-tuning in Bengali. However, when we utilize more labeled Bengali data (200 hours) for adapter-tuning, the performance enhances by a large margin than full fine-tuning (See Table 6.3).

Apart from the amount of pre-training data, there is a key difference between Bengali and the other two languages (English and Maltese). Bengali contains 50 alphabets including 39 consonants and 11 vowels. On the other hand, there are 30 alphabets (24 consonants and 6 vowels) in Maltese. Likewise, English contains 26 alphabets, among which there are 21 consonants and 5 vowels. The complexity of the language modeling head of the Bengali ASR system is comparatively much higher compared to the complexities of the English and Maltese ASR systems.

The WERs of English ASR are much lower than Maltese and English ASR. Due to complex morphological parsing, it is a non-trivial task to build ASR systems for Maltese and Bengali. The performance of ASR for these two languages is supposed to be further improved by integrating an external language model, as indicated by a previous study Samin et al. (2021). Since using language models for decoding is out of scope in our research questions, we do not conduct experiments in that direction.

6.4 Transfer Learning Across Adapters

In this experiment, we pre-train the XLS-R 2B using the adapter-tuning approach with a source language at first. During this stage, only the adapter modules along with the layer normalization and language modeling head are trained while the rest of the XLS-R is kept frozen. After that, we remove the language modeling head from the pre-trained XLS-R since the source language and the target language contain a varied number of tokens. Then, we use the pre-trained adapters of XLS-R 2B and fine-tune them with the adapter-tuning approach for the target language. For both the source language and target language, 50-hour-long train sets have been used.

Table 6.5: Pre-training the adapters within XLS-R 2B with the source language using a 50-hour-long train set and then fine-tuning those adapters with the target language using the same amount of train subset. The language modeling head dedicated to the source language is removed from the pre-trained XLS-R 2B. Results are reported in terms of WERs (%) on the dev and test sets. *source language* -> *target language* refers to the fact that the transfer of knowledge occurs from the source language to the target language. For comparison, the WER for the standard adapter-tuning without transfer learning across adapters is also provided. BN, MT, and EN refer to Bengali, Maltese, and English, respectively.

Language	**Transfer learning across adapters**	**dev set**		**test set**	
		clean	**other**	**clean**	**other**
BN	No	11.6	-	11.3	-
	EN -> BN	14.8	-	13.8	-
	MT -> BN	14.5	-	13.6	-
MT	No	15.4	-	23.9	-
	BN -> MT	14.0	-	22.7	-
	EN -> MT	15.3	-	24.1	-
EN	No	5.0	12.7	5.3	12.9
	BN -> EN	4.9	12.8	4.9	12.9
	MT -> EN	4.7	12.6	4.7	12.6

Our hypothesis is that by pre-training the adapters with a source language at first, we can enhance their performance of adapter-tuning for the target language by enabling transfer learning within adapters. Since the adapters are pre-trained with an equal amount of data in the case of all three languages, we can observe whether or not a specific source language and target language set benefits each other.

Table 6.5 presents the performance of transfer learning across adapters. Without pre-training the adapters and by directly performing adapter-tuning, Bangla ASR obtains

11.3% WER on the test set. Using English and Maltese as source languages, the WER increases to 13.8% and 13.6%, respectively.

Maltese ASR yields the lowest WER of 22.7% when the adapters are pre-trained with Bengali data. Using English as the source language, however, the performance of the Maltese ASR drops to 24.1% WER. The direct adapter-tuning method gets a WER of 23.9% WER.

As for English ASR, using Maltese as the source language, we obtain the lowest WERs of 4.7% and 12.6% on test clean and test other sets, respectively. By pre-training the adapters with Bengali data, we achieve slightly lower WERs of 4.9% and 12.9% compared to the 5.3% and 12.9% WERs from the direct adapter-tuning approach on the test clean/other sets, respectively.

Overall, by pre-training the adapter modules with a source language, we observe a slight improvement in performance in the case of Maltese and English. However, the Bengali ASR performance degrades by pre-training the adapters with a source language. We argue that initializing the adapter with a set of trained weights on a source language has the potential to perform better than starting the adapter tuning from scratch. However, the benefit might be better observed when the source language and the target language are closely related to each other by origin. Since Bengali, Maltese, and English come from different language sub-families, the performance can be further improved by selecting a closely related source language. The reason why transfer learning in adapters for Bengali ASR is not performing as expected might be linked to the fact that XLS-R has seen only 100 hours of unlabelled data during the pre-training stage (See Table 6.4).

6.5 Scaling adapters with more fully connected layers

The standard bottleneck adapter, widely used in computer vision and NLP contains two fully connected (FC) feedforward layers and a GELU activation function in between them. At the end of the adapter module, the original input representations to the adapter and the output of the final FC layer are added via a residual connection and passed through layer normalization to provide the final adapter output. The architecture of bottleneck adapters is described in detail in Section 5.1. We increase the FC layers in each adapter block to see the impact. The inner dimension of the FC layers is 256. After each FC layer, the GELU activation function is added. In this way, we design adapters with 2 layers, 4 layers, 6 layers, and 8 layers of feedforward networks for comparison.

Figure 6.1 represents the outcome of scaling the adapters with more feed-forward layers. We conduct experiments in Maltese, Bengali, and English datasets. For Maltese,

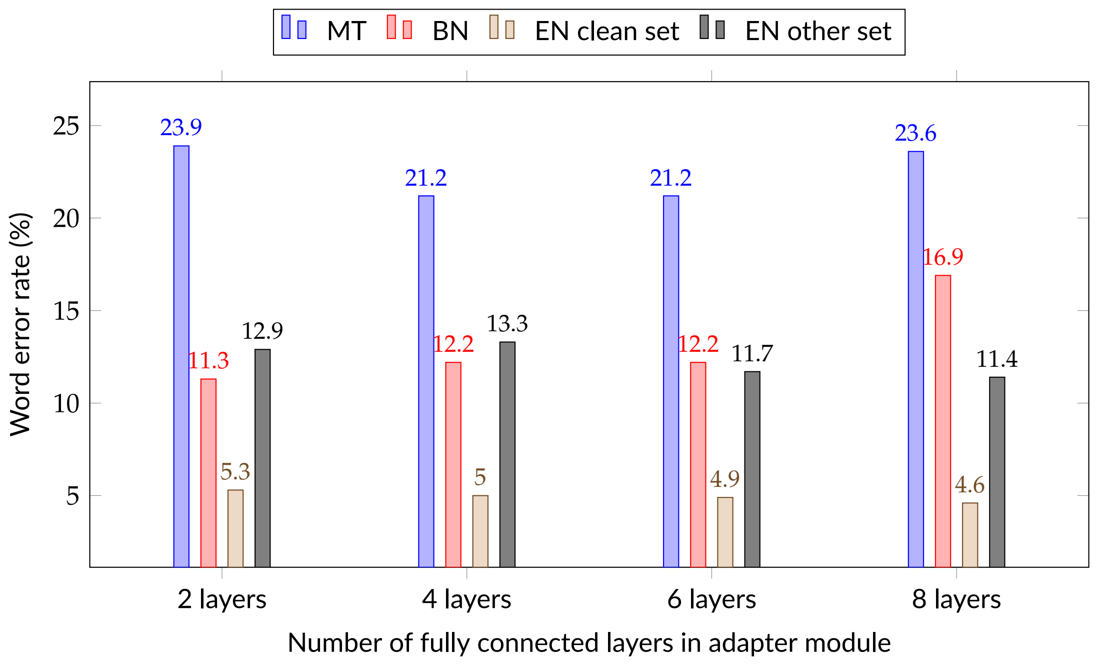

Figure 6.1: Word error rates (%) reported on the test sets after increasing the number of fully connected layers in each adapter module, inserted into XLS-R 2B. MT, BN, and EN corresponds to Maltese, Bengali, and English, respectively.

using either 4 layers or 6 layers of the feed-forward network provides much lower WERs than using a standard two-layer bottleneck adapter. The WERs are comparable in the case of using 2 FC layers and 8 FC layers. Likewise, the overall performance of adapter-tuning improves for English ASR by stacking more feed-forward layers as seen on the LibriSpeech test clean and other sets. In contrast, for Bengali, adding more FC layers drops the adapter-tuning performance. Thus, the benefit of scaling adapter is not consistent for different languages, probably because of the amount of unlabelled data seen during pre-training. For example, XLS-R is pre-trained with a huge amount of unlabelled English and Maltese data while for Bengali, there are only 100 hours of data.

Overall, stacking 6 fully connected feed-forward network for the adapter architecture is optimal for all three languages. Notably, by increasing the FC layers in adapters, the number of trainable parameters gets higher, although not significantly (e.g. from 64M for 2 FC layers to 102M for 6 FC layers).

7 Conclusion

This work presents a thorough study of parameter-efficient adapters for large pre-trained speech models for the first time, to the best of our knowledge. We train adapters with feed-forward layers inserted into the large cross-lingual pre-trained speech models such as XLS-R and MMS while keeping most of the parameters of the pre-trained models frozen. Then, we compare the WERs and CERs of the approach with the regular complete fine-tuning method. We perform our experiments in three languages including Bengali, Maltese, and English from distinct language groups such as Indo-Aryan, Semitic, and West Germanic, respectively. The XLS-R has different model capacities with 0.3B, 1B, and 2B parameters, whereas MMS has only a single variant with 1B parameters

Data requirement for adapters: For an exhaustive study of adapters, we first look into the data requirement of this approach to perform on par with the complete fine-tuning. With only 10 minutes or 1 hour of labeled data, depicting the few-shot scenario, the adapter-tuning approach performs quite poorly compared to the full fine-tuning. As we increase the data amount to 10 hours or more, the performance of adapter-tuning competes with the full fine-tuning, while achieving even lower WERs in several cases. In moderately high-resource settings with more than 200 hours of data, adapter-tuning outperforms full fine-tuning by a large margin.

Larger models with adapters are better: We achieve a substantial performance improvement by leveraging pre-trained models with larger capacities (XLS-R 2B) for the adapter-tuning strategy. We argue that larger pre-trained models tend to get overfitted after full fine-tuning. Adapters are able to circumvent this bottleneck of the larger pre-trained models by working similarly to a regularizer.

Adapters are parameter-efficient: Notably, the promising performance of adapters can be achieved using as little as 2.96% of the total trainable parameters. Moreover, the storage these models take up in computers also reduces by nearly half. These findings are crucial to build low-resource ASR systems since it enables deploying parameter-efficient ASR models on offline mobile devices.

Adapters work well for languages with more pre-training data: We observe that adapter-tuning for Maltese and English ASR models follows the same trend in different experiments while the performance of Bengali ASR models tends to be different at times. For example, with 10 to 20 data, Bangla ASR struggles to perform better with adapter tuning. The pre-training data for Bangla in XLS-R is quite low compared to the other two languages. This might contribute to the performance of adapter-based Bangla ASR models.

Transfer learning in adapters can often be beneficial: We pre-train the adapters with a source language and then fine-tune them on the target language to enable transfer learning in adapters. The pre-trained weights for adapters are found to be beneficial for Maltese and English while not advantageous for Bengali.

Scaling adapters with more feed-forward layers: At last, we scale the capacities of adapters by stacking more feed-forward layers. The regular adapter architecture consists of two feed-forward layers for down-projection and up-projection. We find that using six feed-forward layers in the adapter is optimal across the three languages. However, this benefit comes with a slight increase in the number of trainable parameters (from 64M to 102M). While Maltese and English ASR show a similar trend of overall decreasing WERs as the number of feed-forward layers increases, Bengali ASR is found to be the opposite, possibly due to the pre-training dataset size.

Both Maltese and Bengali lack computational models for speech recognition. We intend to make all our trained ASR models in Bengali, Maltese, and English available for public use.

7.1 Ethical Considerations

ASR datasets may comprise speech utterances that could potentially encompass gender bias, political bias, particular religious perspectives, racism, private information, or objectionable content. While it might be unfeasible for us to manually identify these aspects, we opt to utilize publicly accessible datasets that have been employed by other researchers, thus offering a dependable foundation. These datasets ensure a substantial presence of speech utterances from various genders, ensuring a significant level of representation if not absolute parity.

7.2 Limitations

While this work provides novel findings applying adapters for ASR, there exist some limitations. In our experiments with pre-training adapters on the source language, we use three languages (Bengali, Maltese, and English) that derive from distinct language groups. However, using closely-related language pairs, more performance gain is expected as observed in similar studies Baevski et al. (2020). Due to the limited scope, we restrict our experimentation to only the selected three languages.

We empirically find out that in extremely low-resource cases, adapter-tuning struggles to perform adequately. However, this study does not propose a methodology that can enable adapters to perform well in such extreme cases. For many close to zero-resourced languages, this could have been highly applicable.

7.3 Future Work

Our immediate future work would be investigating the transfer learning within parameter-efficient adapter modules leveraging closely-related languages. We intend to utilize Arabic (close to Maltese), Assamese (close to Bengali), and German (close to English) as the source languages for pre-training the adapters.

We use feed-forward networks as the backbone of the adapter architecture. It would be interesting to incorporate different architectures including recurrent units, convolutional layers or attention mechanisms, etc. into the adapters and observe whether it can reduce the error rate more while still being parameter-efficient.

Our bigger vision is to perform several tasks simultaneously using a single language model and introduce prompt engineering to accomplish different tasks during inference in a zero-shot/few-shot setting. Although there have been some initial works towards this direction for text input-based NLP Brown et al. (2020); Pfeiffer et al. (2021), this is still far from being implemented in the field of speech processing. Multi-tasking can be possible by training task-specific adapters on ASR, speaker identification, emotion recognition, etc., and incorporating them into a pre-trained speech model such as wav2vec 2.0 Baevski et al. (2020). Contemporary adapter-based multi-tasking techniques such as AdapterFusion Pfeiffer et al. (2021) can be leveraged in the speech models to reach our goal.

References

Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93:27403, 1993.

Ahmed, S., Sadeq, N., Shubha, S. S., Islam, M. N., Adnan, M. A., and Islam, M. Z. Preparation of bangla speech corpus from publicly available audio & text. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 6586–6592, 2020.

Alexandridis, A., Sathyendra, K. M., Strimel, G. P., Chang, F.-J., Rastrow, A., Susanj, N., and Mouchtaris, A. Gated contextual adapters for selective contextual biasing in neural transducers. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10095322.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182. PMLR, 2016.

Ardila, R., Branson, M., Davis, K., Kohler, M., Meyer, J., Henretty, M., Morais, R., Saunders, L., Tyers, F., and Weber, G. Common voice: A massively-multilingual speech corpus. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4218–4222, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL `https://aclanthology.org/2020.lrec-1.520`.

Atkins, S., Clear, J., and Ostler, N. Corpus design criteria. *Literary and linguistic computing*, 7(1):1–16, 1992.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Babu, A., Wang, C., Tjandra, A., Lakhotia, K., Xu, Q., Goyal, N., Singh, K., von Platen, P., Saraf, Y., Pino, J., et al. Xls-r: Self-supervised cross-lingual speech representation learning at scale. *Interspeech*, 2021.

Baevski, A., Auli, M., and Mohamed, A. Effectiveness of self-supervised pre-training for speech recognition. *arXiv preprint arXiv:1911.03912*, 2019a.

Baevski, A., Schneider, S., and Auli, M. vq-wav2vec: Self-supervised learning of discrete speech representations. *arXiv preprint arXiv:1910.05453*, 2019b.

Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.

Bapna, A., Arivazhagan, N., and Firat, O. Simple, scalable adaptation for neural machine translation. *arXiv preprint arXiv:1909.08478*, 2019.

Britz, D., Goldie, A., Luong, M.-T., and Le, Q. Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1451, 2017.

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

Bu, H., Du, J., Na, X., Wu, B., and Zheng, H. Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline. In *2017 20th conference of the oriental chapter of the international coordinating committee on speech databases and speech I/O systems and assessment (O-COCOSDA)*, pages 1–5. IEEE, 2017.

Calleja, S. Maltese speech recognition over mobile telephony. 2003.

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4960–4964. IEEE, 2016.

Chang, F.-J., Muniyappa, T., Sathyendra, K. M., Wei, K., Strimel, G. P., and McGowan, R. Dialog act guided contextual adapter for personalized speech recognition. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10094707.

Chen, Z.-C., Fu, C.-L., Liu, C.-Y., Li, S.-W. D., and Lee, H.-y. Exploring efficient-tuning methods in self-supervised speech models. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 1120–1127. IEEE, 2023.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, É., Ott, M., Zettlemoyer, L., and Stoyanov, V. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, 2020.

Conneau, A., Baevski, A., Collobert, R., Mohamed, A., and Auli, M. Unsupervised cross-lingual representation learning for speech recognition. *Proc. Interspeech 2021*, 2021.

Conneau, A., Ma, M., Khanuja, S., Zhang, Y., Axelrod, V., Dalmia, S., Riesa, J., Rivera, C., and Bapna, A. Fleurs: Few-shot learning evaluation of universal representations of speech. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 798–805. IEEE, 2023.

DeMarco, A., Mena, C., Gatt, A., Borg, C., Williams, A., and van der Plas, L. Analysis of data augmentation methods for low-resource maltese asr, 2023.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dunbar, E., Bernard, M., Hamilakis, N., Nguyen, T. A., de Seyssel, M., Rozé, P., Rivière, M., Kharitonov, E., and Dupoux, E. The zero resource speech challenge 2021: Spoken language modelling. In *Interspeech 2021-Conference of the International Speech Communication Association*, 2021.

Eeckt, S. V. and Van Hamme, H. Using adapters to overcome catastrophic forgetting in end-to-end automatic speech recognition. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10095837.

Eeckt, S. V. et al. Using adapters to overcome catastrophic forgetting in end-to-end automatic speech recognition. *arXiv preprint arXiv:2203.16082*, 2022.

Falzon, G. Isolated word speech recognition. B.S. thesis, University of Malta, 1993.

Fan, Z., Li, M., Zhou, S., and Xu, B. Exploring wav2vec 2.0 on speaker verification and language identification. *arXiv preprint arXiv:2012.06185*, 2020.

Fu, C.-L., Chen, Z.-C., Lee, Y.-R., and Lee, H.-y. Adapterbias: Parameter-efficient token-dependent representation shift for adapters in nlp tasks. *arXiv preprint arXiv:2205.00305*, 2022.

Gales, M. J., Knill, K. M., Ragni, A., and Rath, S. P. Speech recognition and keyword spotting for low-resource languages: Babel project research at cued. In *Fourth International workshop on spoken language technologies for under-resourced languages (SLTU-2014)*, pages 16–23. International Speech Communication Association (ISCA), 2014.

Gállego, G. I., Tsiamas, I., Escolano, C., Fonollosa, J. A., and Costa-jussà, M. R. End-to-end speech translation with pre-trained models and adapters: Upc at iwslt 2021. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 110–119, 2021.

Gatt, A. and Čéplö, S. Digital corpora and other electronic resources for maltese. In *Corpus linguistics*, pages 96–97. UCREL Lancaster, 2013.

Graves, A. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

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

Graves, A., Mohamed, A.-r., and Hinton, G. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013. doi: 10.1109/ICASSP.2013.6638947.

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., et al. Conformer: Convolution-augmented transformer for speech recognition.*Proc. Interspeech 2020*, $pages 5036--5040, 2020$.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

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

Hsieh, C.-P., Ghosh, S., and Ginsburg, B. Adapter-based extension of multi-speaker text-to-speech model for new speakers. *arXiv preprint arXiv:2211.00585*, 2022.

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

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

Huang, J., Ganesan, K., Maiti, S., Min Kim, Y., Chang, X., Liang, P., and Watanabe, S. Findadaptnet: Find and insert adapters by learned layer importance. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10095392.

Huo, Z., Sim, K. C., Li, B., Hwang, D., Sainath, T. N., and Strohman, T. Resource-efficient transfer learning from speech foundation model using hierarchical feature fusion. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10096799.

Jegou, H., Douze, M., and Schmid, C. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

Kannan, A., Datta, A., Sainath, T. N., Weinstein, E., Ramabhadran, B., Wu, Y., Bapna, A., Chen, Z., and Lee, S. Large-scale multilingual speech recognition with a streaming end-to-end model.*Proc. Interspeech 2019*, $pages 2130--2134, 2019$.

Kenton, J. D. M.-W. C. and Toutanova, L. K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

Kibria, S., Samin, A. M., Kobir, M. H., Rahman, M. S., Selim, M. R., and Iqbal, M. Z. Bangladeshi bangla speech corpus for automatic speech recognition research. *Speech Communication*, 136:84–97, 2022.

Kjartansson, O., Sarin, S., Pipatsrisawat, K., Jansche, M., and Ha, L. Crowd-sourced speech corpora for javanese, sundanese, sinhala, nepali, and bangladeshi bengali. 2018.

Le, H., Pino, J., Wang, C., Gu, J., Schwab, D., and Besacier, L. Lightweight adapter tuning for multilingual speech translation. *arXiv preprint arXiv:2106.01463*, 2021.

Lee, J., Cho, K., and Hofmann, T. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378, 2017.

Lee, T., Lee, M.-J., Kang, T. G., Jung, S., Kwon, M., Hong, Y., Lee, J., Woo, K.-G., Kim, H.-G., Jeong, J., et al. Adaptable multi-domain language model for transformer asr. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7358–7362. IEEE, 2021.

Li, B., Hwang, D., Huo, Z., Bai, J., Prakash, G., Sainath, T. N., Chai Sim, K., Zhang, Y., Han, W., Strohman, T., and Beaufays, F. Efficient domain adaptation for speech foundation models. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023a. doi: 10.1109/ICASSP49357.2023.10096330.

Li, X., Wang, C., Tang, Y., Tran, C., Tang, Y., Pino, J., Baevski, A., Conneau, A., and Auli, M. Multilingual speech translation from efficient finetuning of pretrained models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 827–838, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.68. URL https://aclanthology.org/2021.acl-long.68.

Li, Y., Mehrish, A., Bhardwaj, R., Majumder, N., Cheng, B., Zhao, S., Zadeh, A., Mihalcea, R., and Poria, S. Evaluating parameter-efficient transfer learning approaches on sure benchmark for speech understanding. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023b. doi: 10.1109/ICASSP49357.2023.10095656.

Likhomanenko, T., Synnaeve, G., and Collobert, R. Who needs words? lexicon-free speech recognition. *Proc. Interspeech 2019*, pages 3915–3919, 2019.

Ling, S. and Liu, Y. Decoar 2.0: Deep contextualized acoustic representations with vector quantization. *arXiv preprint arXiv:2012.06659*, 2020.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

Maddison, C. J., Tarlow, D., and Minka, T. A* sampling, 2015.

Mena, C., Gatt, A., DeMarco, A., Borg, C., van der Plas, L., Muscat, A., and Padovani, I. Masri-headset: A maltese corpus for speech recognition. *arXiv preprint arXiv:2008.05760*, 2020a.

Mena, H., Daniel, C., Gatt, A., DeMarco, A., Borg, C., van der Plas, L., Muscat, A., and Padovani, I. MASRI-HEADSET: A Maltese corpus for speech recognition. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 6381–6388, Marseille, France, May 2020b. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.784.

Meng, W. and Yolwas, N. A review of speech recognition in low-resource languages. In *2022 3rd International Conference on Pattern Recognition and Machine Learning (PRML)*, pages 245–252. IEEE, 2022.

Mohamed, A., Okhonko, D., and Zettlemoyer, L. Transformers with convolutional context for asr. *arXiv preprint arXiv:1904.11660*, 2019.

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

Otake, S., Kawakami, R., and Inoue, N. Parameter efficient transfer learning for various speech processing tasks. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10096311.

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015. doi: 10.1109/ICASSP.2015.7178964.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., and Le, Q. V. Specaugment: A simple data augmentation method for automatic speech recognition. *Interspeech 2019*, 2019.

Peng, J., Stafylakis, T., Gu, R., Plchot, O., Mošner, L., Burget, L., and Černockỳ, J. Parameter-efficient transfer learning of pre-trained transformer models for speaker verification using adapters. *arXiv preprint arXiv:2210.16032*, 2022.

Pfeiffer, J., Rücklé, A., Poth, C., Kamath, A., Vulić, I., Ruder, S., Cho, K., and Gurevych, I. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779*, 2020.

Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., and Gurevych, I. Adapterfusion: Non-destructive task composition for transfer learning. In *16th Conference of the European Chapter of the Associationfor Computational Linguistics, EACL 2021*, pages 487–503. Association for Computational Linguistics (ACL), 2021.

Phang, J., Févry, T., and Bowman, S. R. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*, 2018.

Philip, J., Berard, A., Gallé, M., and Besacier, L. Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4465–4470, 2020.

Pratap, V., Xu, Q., Sriram, A., Synnaeve, G., and Collobert, R. Mls: A large-scale multilingual dataset for speech research■*Proc. Interspeech 2020*, $pages 2757 - -2761,$ 2020.

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

Pruksachatkun, Y., Phang, J., Liu, H., Htut, P. M., Zhang, X., Pang, R. Y., Vania, C., Kann, K., and Bowman, S. R. Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work? *arXiv preprint arXiv:2005.00628*, 2020.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*, 2022.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR, 2023.

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

Rücklé, A., Geigle, G., Glockner, M., Beck, T., Pfeiffer, J., Reimers, N., and Gurevych, I. Adapterdrop: On the efficiency of adapters in transformers. *arXiv preprint arXiv:2010.11918*, 2020.

Ruder, S. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Ruder, S. *Neural transfer learning for natural language processing*. PhD thesis, NUI Galway, 2019.

Samin, A. M., Kobir, M. H., Kibria, S., and Rahman, M. S. Deep learning based large vocabulary continuous speech recognition of an under-resourced language bangladeshi bangla. *Acoustical Science and Technology*, 42(5):252–260, 2021.

Samin, A. M., Kobir, M. H., Rafee, M. M. S., Ahmed, M. F., Hasan, M., Ghosh, P., Kibria, S., and Rahman, M. S. Investigating self-supervised, weakly supervised and fully supervised training approaches for multi-domain automatic speech recognition: a study on bangladeshi bangla, 2023.

San, N., Bartelds, M., Browne, M., Clifford, L., Gibson, F., Mansfield, J., Nash, D., Simpson, J., Turpin, M., Vollmer, M., et al. Leveraging pre-trained representations to improve access to untranscribed speech from endangered languages. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 1094–1101. IEEE, 2021.

Sathyendra, K. M., Muniyappa, T., Chang, F.-J., Liu, J., Su, J., Strimel, G. P., Mouchtaris, A., and Kunzmann, S. Contextual adapters for personalized speech recognition in neural transducers. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8537–8541. IEEE, 2022.

Schneider, S., Baevski, A., Collobert, R., and Auli, M. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

Shahgir, H., Sayeed, K. S., and Zaman, T. A. Applying wav2vec2 for speech recognition on bengali common voices dataset.

arXiv preprint arXiv:2209.06581, 2022.

Stickland, A. C. and Murray, I. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, pages 5986–5995. PMLR, 2019.

Sultana, S., Rahman, M. S., and Iqbal, M. Z. Recent advancement in speech recognition for bangla: A survey. *Int. J. Adv. Comput. Sci. Appl*, 12(3):546–552, 2021.

Thomas, B., Kessler, S., and Karout, S. Efficient adapter transfer of self-supervised speech models for automatic speech recognition. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7102–7106. IEEE, 2022.

Tomanek, K., Zayats, V., Padfield, D., Vaillancourt, K., and Biadsy, F. Residual adapters for parameter-efficient asr adaptation to atypical and accented speech. *arXiv preprint arXiv:2109.06952*, 2021.

Valk, J. and Alumäe, T. Voxlingua107: a dataset for spoken language recognition. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 652–658. IEEE, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vella, A. and Farrugia, P.-J. Maltobi–building an annotated corpus of spoken maltese. *Proceedings of Speech Prosody 2006*, 2006.

Vella, A., Chetcuti, F., Grech, S., and Spagnol, M. Integrating annotated spoken maltese data into corpora of written maltese. In *Editors & Workshop Chairs*, page 83. Citeseer, 2010.

Wang, C., Rivière, M., Lee, A., Wu, A., Talnikar, C., Haziza, D., Williamson, M., Pino, J., and Dupoux, E. Voxpopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation. In *ACL 2021-59th Annual Meeting of the Association for Computational Linguistics*, 2021a.

Wang, D., Wang, X., and Lv, S. An overview of end-to-end automatic speech recognition. *Symmetry*, 11(8):1018, 2019.

Wang, Y., Boumadane, A., and Heba, A. A fine-tuned wav2vec 2.0/hubert benchmark for speech emotion recognition, speaker verification and spoken language understanding. *arXiv preprint arXiv:2111.02735*, 2021b.

Wu, F., Fan, A., Baevski, A., Dauphin, Y., and Auli, M. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*, 2018.

Zhang, Y. and Yang, Q. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2021.

Zhang, Y., Han, W., Qin, J., Wang, Y., Bapna, A., Chen, Z., Chen, N., Li, B., Axelrod, V., Wang, G., et al. Google usm: Scaling automatic speech recognition beyond 100 languages. *arXiv preprint arXiv:2303.01037*, 2023.