DOMAIN ADAPTATION BASED CONFORMER MODEL FOR AUTOMATIC SPEECH RECOGNITION

ASHFAQE AHMED (20I206)

HARIHARARAM S (20I218)

MOHAMMED NIHMATHULLAH M (20I226)

NAVIN PRAANAV P (20I230)

SANJJAY S (20I241)

BACHELOR OF TECHNOLOGY

Branch: INFORMATION TECHNOLOGY

of Anna University

April 2023

DEPARTMENT OF INFORMATION TECHNOLOGY

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

DOMAIN ADAPTATION BASED CONFORMER MODEL FOR AUTOMATIC SPEECH RECOGNITION

Bonafide record of work done by

ASHFAQE AHMED	(20I206)
HARIHARARAM S	(20I218)
MOHAMMED NIHMATHULLAH M	(20I226)
NAVIN PRAANAV P	(20I230)
SANJJAY S	(20I241)

BACHELOR OF TECHNOLOGY

Branch: INFORMATION TECHNOLOGY

of Anna University

April 2023

.....	.....
Dr.D.Karthika Renuka	**Dr.K.Umamaheswari**
Faculty guide	Head of the Department

Certified that the candidate was examined in the viva-voce examination held on ……

.....	.....
(Internal Examiner)	(External Examiner)

ACKNOWLEDGEMENT

We express our sincere thanks to **Dr.K.Prakasan**, Principal, and PSG College of Technology for providing an opportunity to take up this project work.

We are greatly indebted to **Dr.K.Umamaheswari**, Professor and Head of the Department of Information Technology for her kind support and encouragement were given to complete the project.

We express our humble and profound gratitude to our project guide & program coordinator **Dr.D.Karthika Renuka,** Professor, Department of Information Technology for her valuable guidance and timely advice to complete the project work successfully.

We express our sincere thanks to our tutor **Dr.R.Senthilprabha**, Assistant Professor (Sl. Gr.), Department of Information Technology, for her constant support and guidance that played a vitalrole in completing our project.

We express our gratitude to **Dr.L.Ashok Kumar**, Professor, Department of EEE, for providing the internship in the Center for Audio-Visual Speech Recognition (PSG-CAVSR).

We wish to record our gratitude to all the department faculty members for their support extended in the completion of our project.

PSG College of Technology -641 004

DST-ICPS Sponsored PSG Centre for Audio Visual Speech Recognition (CAVSR)

Coimbatore

10.04.2023

TO WHOMSOVER IT MAY CONCERN

This is to certify that the following students from 3rdyear, B.Tech Information Technology, PSG College of Technology, have successfully completed the project titled **"DOMAIN ADAPTATION BASED CONFORMER MODEL FOR AUTOMATIC SPEECH RECOGNITION"** as a part of their internship at PSG College of technology.

ASHFAQE AHMED (20I206)

HARIHARARAM S (20I218)

MOHAMMED NIHMATHULLAH M (20I226)

NAVIN PRAANAV P (20I230)

SANJJAY S (20I241)

Dr.L.Ashok Kumar

Principal Investigator

SYNOPSIS

Automatic Speech Recognition (ASR) is the ability to translate a dictation or spoken word to text. Speech Recognition is known as “automatic speech recognition” (ASR), or speech-to-text (STT). Speech recognition is the process of converting an acoustic signal, captured by a microphone or any peripherals, to a set of words. To achieve speech understanding linguistic processing can be used. The recognized words can be an end in themselves, as for applications such as commands & control data entry and document preparation. In society, everyone wishes to interact with each other and tries to convey their own message to others. The receiver of messages may get the exact and full idea of the senders or may get a partial idea or sometimes cannot understand anything out of it. In some cases, this may happen when there is some lacking communication (i.e when a child conveys a message, the mother can understand easily while others cannot). Speech recognition is the machine on the statement or command of human speech to identify and understand and react accordingly. It is based on the voice as the research object, it allows the machine to automatically identify and understand human spoken language through speech signal processing and pattern recognition. Speech recognition technology is the high-tech that allows the machine to turn the voice signal into theappropriate text or command through the process of identifying and understanding. Speech recognition is cross-disciplinary and involves a wide range. It has a very close relationship with acoustics, phonetics, linguistics, information theory, pattern recognition theory and neurobiology disciplines. With the rapid development of computer hardware and software and information technology, speech recognition technology is gradually becoming a key technology in computer information processing technology. The goal of automatic speech recognition is to provide a means forverbal human-to-machine communication. Basically, speech recognition is a pattern recognition problem. Speech Recognition Systems are generally classified as discrete or continuous systems that are speaker-dependent, independent, or adaptive. Domain adaptation is the machine learning approach that aims at the transfer of knowledge of a well-trained model to our target domain which usually is a low-resource domain. In our work we were successful in domain adapting a pre-trained conformer transducer model trained on Native English speakers’ speech data to that of our target Indian English domain and there was a significant improvement in the word error rate metric after implementing domain adaptation.

CONTENTS

CHAPTER **Page No.**

Acknowledgement..... (i)

Completion Certificate .....(ii)

Synopsis ..... (iii)

1. INTRODUCTION..... 7

1.1. Speech Recognition 7

1.2. Domain Adaptation 8

1.3. Automatic Speech Recognition 9

1.4. Deployment 10

1.5. Motivation 10

1.6. Scope of project 11

2. LITERATURE SURVEY.....12

2.1 Wav2Vec 2.0: Analyzing domain shift in self-supervised pre-training 12

2.2 Domain adaptation of low-resource target-domain models using well-trained ASR conformer models 12

2.3 Existing Systems for ASR 12

3. SYSTEM REQUIREMENTS.....15

3.1. Software Requirements 15

3.2. VSCode 15

3.3. Numpy 15

3.3.1 ND Array 16

3.4. Shutil 16

3.5. Librosa 17

3.6. PyTorch 17

3.7. Pandas 17

3.8. Nemo Toolkit 17

3.9. Hugging Face 18

3.9.1 Dataset 18

3.10. Flask 18

4. PROPOSED SYSTEM 20

4.1. Automatic Speech Recognition 20

4.1.1 Librispeech Corpus Dataset 20

4.1.2 NPTEL 2020 - Indian English Speech Dataset 21

4.1.3 Data Annotation & E-Learning Corpus 22

4.1.4 Wav2Vec2 22

4.1.5 Conformer Transducer 23

4.1.6 Evaluation 25

4.2. Domain Adaptation 26

4.2.1 Source Dataset 26

4.2.2 Target Dataset 27

4.2.3 Methodology 28

4.2.4 Conformer Domain Adaptation 29

5. IMPLEMENTATION AND RESULT ANALYSIS 31

5.1 Implementation of ASR Evaluation 31

5.1.1 Dataset Creation 31

5.1.2 Dataset Preprocessing 31

5.1.3 Evaluation of the model 32

5.1.4 Results and Analysis 32

5.2 Deployment of the Model 32

5.2.1 Setup 32

5.2.2 Frontend 33

5.2.3 Backend 34

5.2.4 Integration 34

5.3 Implementation of Domain Adaptation 35

5.3.1 Results and Analysis 35

CONCLUSION AND FUTURE ENHANCEMENTS 36

BIBLIOGRAPHY 37

APPENDIX - 1 40

CHAPTER 1

INTRODUCTION

1.1 SPEECH RECOGNITION

Speech recognition, also known as speech-to-text, refers to the capacity of a machine or program to recognize spoken words and translate them into text that can be read. Simple speech recognition software has a restricted vocabulary and can only identify words and phrases when spoken clearly. However, advanced software can handle natural speech, diverse accents, and multiple languages. The development of speech recognition has been the result of extensive research in fields such as computer science, linguistics, and computer engineering. Nowadays, many modern devices and text-based programs incorporate speech recognition features to provide users with a more convenient and hands-free experience.

Speech recognition and voice recognition are two different technologies and should not be confused:

- Speech recognition is used to identify words in spoken language.
- Voice recognition is a biometric technology for identifying an individual's voice.

Speech recognition systems use computer algorithms to process and interpret spoken words and convert them into text. A software program turns the sound a microphone records into written language that computers and humans can understand, following these four steps:

1. analyze the audio;
2. break it into parts;
3. digitize it into a computer-readable format; and
4. use an algorithm to match it to the most suitable text representation.

Speech recognition software must adapt to the highly variable and context-specific nature of human speech. The software algorithms that process and organize audio into the text are trained on different speech patterns, speaking styles, languages, dialects, accents and phrasings. The software also separates spoken audio from background noise that often accompanies the signal. To meet these requirements, speech recognition systems use two types of models:

- **Acoustic models**: These represent the relationship between linguistic units of speech and audio signals.
- **Language models:** Here, sounds are matched with word sequences to distinguish between words that sound similar.

1.2 DOMAIN ADAPTATION

Domain adaptation in ASR is an active area of research, and various techniques are being explored to improve the adaptability of ASR systems to different domains. It plays a critical role in enabling ASR systems to be more robust and accurate when applied to diverse real-world applications, such as transcription services, voice assistants, and speech-based communication systems. Overall, domain adaptation in ASR is essential for achieving high performance in ASR applications in different domains by adapting the ASR system to the specific characteristics of the target domain. Thus, it helps to bridge the gap between different domains and makes ASR systems more versatile and effective in real-world scenarios. However, it also presents challenges such as data scarcity, domain mismatch, and model complexity, which need to be carefully addressed in order to achieve optimal domain adaptation results. Properly implemented domain adaptation techniques can greatly enhance the performance of ASR systems and make them more practical and effective in a wide range of applications. In conclusion, domain adaptation is a crucial aspect of ASR that enables the system to adapt to diverse domains and deliver accurate and reliable results in real-world scenarios. Advances in domain adaptation techniques continue to push the boundaries of ASR performance and open up new possibilities for speech recognition applications in different domains. Overall, domain adaptation is an important research area that continues to evolve and contribute to the advancement of ASR technology. However, further research and development are needed to tackle the challenges associated with domain adaptation in ASR and make it more effective and practical for real-world applications. Nonetheless, domain adaptation is a promising direction for improving the robustness and versatility of ASR systems, and it is likely to remain an active area of research in the field of speech recognition for years to come. In summary, domain adaptation in ASR is a critical technique that enables ASR systems to adapt to different domains and deliver accurate results in diverse real-world applications, making it an important research area in the field of speech recognition. As ASR technology continues to advance, domain adaptation techniques are expected to play an increasingly important role in improving ASR performance and expanding the range of applications where ASR can be effectively deployed. Overall,

domain adaptation in ASR is a vital research area that is likely to continue to grow and evolve in the coming years, driving advancements in ASR technology and expanding the range of applications where ASR can be effectively used. So, the continued exploration and development of domain adaptation techniques in ASR

For example, an ASR system trained on clean, well-recorded speech may not perform well when applied to noisy or reverberant speech.

There are several approaches to domain adaptation in ASR. One common approach is to collect additional data from the target domain and use it to fine-tune the ASR system. This fine-tuning process helps the ASR system to adapt to the specific characteristics of the target domain by updating the model parameters based on the target domain data.

1.3 AUTOMATIC SPEECH RECOGNITION

ASR (Automatic Speech Recognition) is a technology that enables machines to recognize and interpret human speech. It is based on algorithms that analyze audio input and convert it into text output. ASR is used to transcribe spoken language into text, enabling users to search, analyze, and understand spoken content more easily. ASR is also used to enable voice-activated systems and virtual assistants, such as Siri, Alexa, and Google Assistant. These systems use ASR technology to interpret and respond to spoken commands and queries.ASR is an important technology in a wide range of industries and applications. In healthcare, it can be used to transcribe medical dictations, enabling doctors to record patient notes and diagnoses more efficiently. In the legal industry, it can be used to transcribe court proceedings and depositions. In customer service, it can be used to automate call centers and improve response times for customer queries. In education, it can be used to transcribe lectures and enable closed captioning for videos.ASR technology is constantly evolving and improving, thanks to advances in machine learning and artificial intelligence. Deep learning techniques have enabled ASR systems to achieve higher accuracy rates, and ongoing research is focused on developing systems that can handle more diverse accents and languages. The development of ASR technology is driven by the need for more intuitive and user-friendly interfaces, as well as the increasing demand for automation and data analysis. Overall, ASR is an exciting and rapidly evolving field that has the potential to revolutionize the way we communicate and interact with

technology, making it easier and more efficient for humans to interact with machines.

1.4 DEPLOYMENT

Deployment of a machine learning model is necessary to make it accessible and usable in real-world applications. A trained machine learning model is typically developed to solve a specific problem, such as image recognition, natural language processing, or recommendation. Deployment allows the model to be used in real-world scenarios, such as in production systems, applications, or services, to provide predictions or recommendations to end-users. Deploying a model makes it accessible to users who may not have the technical expertise or knowledge to train and use the model themselves. It allows non-technical users to benefit from the capabilities of the model without having to understand the intricacies of the underlying machine learning algorithms.Deploying a model enables it to be scaled to handle large amounts of data and concurrent requests. This is particularly important in scenarios where the model needs to handle high volumes of data, such as in online recommendation systems or real-time data processing applications. Deploying a model allows it to be integrated with other systems, applications, or services to enable seamless interactions between different components of a larger system. For example, a machine learning model can be integrated into a chatbot, a mobile app, or a web service to provide personalized recommendations or predictions. Deploying a model allows for easy maintenance and updates to the model. As new data becomes available, the model may need to be retrained or updated to ensure its accuracy and performance. Deployment makes it easier to update the model and keep it up-to-date with the latest data.

1.5 MOTIVATION

Automatic Speech Recognition (ASR) systems are highly sensitive to differences in acoustic conditions, accents, languages, or other factors, which can result in decreased performance when the system is applied to new domains. Domain adaptation techniques are needed to adapt ASR models to new domains and improve their accuracy and performance in real-world scenarios.

While there are various domain adaptation techniques proposed in the literature, there may be limitations in their effectiveness or applicability to ASR tasks. The project aims to explore and implement domain adaptation techniques specifically for the Conformer model, which is a recently proposed deep learning architecture for ASR and has shown potential for further improvement through domain adaptation.

ASR systems are widely used in applications such as voice assistants, transcription services, and speech recognition in various domains. Enhancing the accuracy and performance of ASR models through domain adaptation can lead to better user experience, increased usability, and improved overall system performance.

The project is motivated by the potential practical applications of domain adaptation techniques in real-world scenarios. By implementing and evaluating domain adaptation techniques for the Conformer model, the project aims to contribute to the advancement of ASR technology and its practical applications in diverse domains, such as multilingual ASR, accented speech recognition, or domain-specific ASR systems.

1.6 SCOPE OF THE PROJECT

The scope of ASR (Automatic Speech Recognition) is quite broad and encompasses a wide range of applications and technologies. ASR has the potential to transform the way we communicate and interact with technology. The primary application of ASR is to transcribe spoken words into text, which has numerous applications ranging from transcribing meetings and lectures to creating subtitles for videos and television shows. ASR systems are widely used in real-world scenarios such as voice assistants, transcription services, call centers, and other applications. However, the performance of ASR models can degrade when applied to new domains due to differences in acoustic conditions, accents, languages, or other factors.

Domain adaptation techniques are essential to improve the accuracy and robustness of ASR systems in diverse domains, enabling better human-machine interaction and user experience. Conventional ASR models, trained on large amounts of labeled data, may not perform well in domain-specific or data-scarce scenarios. Domain adaptation techniques can bridge the gap between the availability of labeled data and the need for accurate ASR in specific domains, allowing for more effective and efficient ASR deployment in practical applications. Domain adaptation can provide cost-effective solutions for adapting ASR models to new domains without the need for extensive retraining on domain-specific data. can be especially beneficial in scenarios where collecting large amounts of labeled data for each domain may not be feasible. Domain adaptation in ASR is an active research area that aims to advance the state-of-the-art in ASR technology. Implementing and evaluating domain adaptation techniques can contribute to the development of new approaches, insights, and innovations in the field of ASR, leading to improved performance and capabilities of ASR system

CHAPTER 2

LITERATURE SURVEY

Several papers and applications on existing systems for Automatic Speech Recognition have been studied as a reference for this project. The details that have been inferred from those applications and papers are discussed below.

2.1 WAV2VEC 2.0: ANALYZING DOMAIN SHIFT IN SELF-SUPERVISED PRE-TRAINING

Using target domain data during pre-training significantly enhances performance in various scenarios. Pre-training on unlabeled in-domain data lessens the gap between in-domain and out-of-domain labeled data by 66%-73% in competitive setups. It is easier to obtain unlabeled target domain data than labeled data, making this approach more practical. Additionally, pre-training on multiple domains enhances generalization performance on unseen domains. These findings have practical implications for improving model performance in real-world applications.

2.2 DOMAIN ADAPTATION OF LOW-RESOURCE TARGET-DOMAIN MODELS USING WELL-TRAINED ASR CONFORMER MODELS

Our study focuses on adapting Automatic Speech Recognition (ASR) models to low-resource target-domain data, given a well-trained model on a large dataset. We propose using embeddings from the encoder layers of the well-trained ASR model as features for a downstream target-domain Conformer model, rather than relying on the decoder. We conduct experiments to determine the optimal encoder layer for feature extraction and whether to freeze or update the encoder layers. We also show that applying Spectral Augmentation on the proposed features enhances target-domain performance. Our approach yields a significant relative improvement of 30% and 50% over baselines for LibriSpeech-100-clean and WSJ target-domain data, respectively

2.3 EXISTING SYSTEMS FOR ASR

Jamal, Norezmi, et al. "Automatic speech recognition (ASR) based approach for speech therapy of aphasic patients: A review." *AIP Conference Proceedings*. Vol. 1883. No. 1. AIP Publishing LLC, 2017.[3]

This paper provides an overview of the use of automatic speech recognition (ASR) in speech therapy for individuals with aphasia. ASR technology is being increasingly researched as a means of providing accurate, real-time evaluation of speech input from individuals with speech disorders. The review

highlights the challenges faced by ASR in accurately recognizing speech input, such as phoneme recognition, speech continuity, and environmental factors. The paper also examines recent developments in ASR technology and its potential use in improving the symptoms of aphasia.

Alharbi, Sadeen, et al. "Automatic speech recognition: Systematic literature review." *IEEE Access* 9 (2021): 131858-131876.[4]

This paper provides a systematic review of automatic speech recognition (ASR) technology over the last six years. The review highlights the evolution of ASR from simple systems to sophisticated natural language systems. It identifies major challenges and research gaps in ASR and covers 82 relevant articles from five research databases using the PRISMA protocol. The results of the review offer insights into research trends and suggest new directions for future research in ASR technology.

Neri, Ambra, Catia Cucchiarini, and Helmer Strik. "Automatic speech recognition for second language learning: How and why it actually works." *Proc. ICPhS*. 2003.[5]

This paper reviews studies and reviews on the usability of Automatic Speech Recognition (ASR) technology for training pronunciation in the second language (L2). The paper suggests that criticism of ASR technology is often based on limited familiarity with the technology and broader courseware design matters. The analysis considers actual problems of state-of-the-art ASR technology and suggests ways to employ ASR for pedagogically sound and reliable courseware development.

Baevski, Alexei, Steffen Schneider, and Michael Auli. "vq-wav2vec: Self-supervised learning of discrete speech representations." *arXiv preprint arXiv:1910.05453* (2019).[6]

We propose vq-wav2vec to learn discrete representations of audio segments through a wav2vec-style self-supervised context prediction task. The algorithm uses either a gumbel softmax or online k-means clustering to quantize the dense representations. Discretization enables the direct application of algorithms from the NLP community which require discrete inputs. Experiments show that BERT pre-training achieves a new state of the art on TIMIT phoneme classification and WSJ speech recognition.

Sadhu, Samik, et al. "Wav2vec-c: A self-supervised model for speech representation learning." *arXiv preprint arXiv:2103.08393* (2021).[7]

Wav2vec-C introduces a novel representation learning technique combining elements from wav2vec 2.0 and VQ-VAE. Our model learns to reproduce quantized representations from partially masked speech encoding using a contrastive loss in a way similar to Wav2vec 2.0. However, the quantization process is regularized by an additional consistency network that learns to reconstruct the input features to the wav2vec 2.0 network from the quantized representations in a way similar to a VQ-VAE model. The proposed self-supervised model is trained on 10k hours of unlabeled data and subsequently used as the speech encoder in a RNN-T ASR model and fine-tuned with 1k hours of labeled data. This work

is one of only a few studies of self-supervised learning on speech tasks with a large volume of real far-field labeled data. The Wav2vec-C encoded representations achieves, on average, twice the error reduction over baseline and a higher codebook utilization in comparison to wav2vec 2.0

Baevski, Alexei, et al. "wav2vec 2.0: A framework for self-supervised learning of speech representations." *Advances in neural information processing systems* 33 (2020): 12449-12460.[8]

This paper presents the wav2vec 2.0 method which can learn powerful speech representations without transcriptions and outperform semi-supervised methods. The method masks speech input in the latent space and uses a contrastive task over jointly learned quantized representations. Experiments show that wav2vec 2.0 achieves state-of-the-art performance with minimal labeled data and pre-training on unlabeled data.

Gulati, Anmol, et al. "Conformer: Convolution-augmented transformer for speech recognition." *arXiv preprint arXiv:2005.08100* (2020).[9]

This work proposes Conformer, a convolution-augmented transformer model for Automatic Speech Recognition (ASR), combining the strengths of both Transformer and Convolutional Neural Network (CNN) based models. Conformer achieves state-of-the-art performance on the widely used LibriSpeech benchmark, with or without an external language model. The model also shows competitive performance with a small parameter size of 10M.

Guo, Pengcheng, et al. "Recent developments on espnet toolkit boosted by conformer." *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.[10]

This paper discusses recent advancements in the ESPnet toolkit, specifically the Conformer architecture, and its performance in various end-to-end speech processing tasks including ASR, ST, SS, and TTS. The study highlights the benefits and training tips for the Conformer, which outperforms current state-of-the-art Transformer models. The authors plan to release pre-trained models and open-source recipes to reduce the burden of preparing research environments requiring high resources.

Zeineldeen, Mohammad, et al. "Conformer-based hybrid ASR system for Switchboard dataset." *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022. [11]

The use of conformer architecture in hybrid ASR models, which has not been explored before. The authors present a competitive conformer-based hybrid model training recipe with different training methods and downsampling techniques for efficient training. The experiments conducted on Switchboard 300h dataset show that the proposed conformer-based hybrid model achieves competitive results and outperforms the BLSTM-based hybrid model significantly on Hub5'01 test set.

CHAPTER 3

SYSTEM REQUIREMENTS

All computer software requires the presence of certain hardware components and other software resources to function properly. Device specifications begin to riseover time as demand for more computing power and resources in newer versions of software increases.

3.1 SOFTWARE REQUIREMENTS

Operating System: Windows 10/11

Coding Language: Python

Python is a high-level, general-purpose programming language that is widely used for web development, scientific computing, data analysis, artificial intelligence, and many other applications.

Packages: nemo-toolkit, pytorch-lightning, soundfile, librosa, shutil, torch, numpy

IDE: Colab, Visual Studio Code

3.2 VSCODE

Visual Studio Code, often referred to as VSCode, is a popular open-source integrated development environment (IDE) developed by Microsoft. It is a lightweight yet powerful tool that provides an extensive set of features for editing, debugging, and building software applications. VSCode is highly customizable and offers support for various programming languages, including Python, JavaScript, TypeScript, C++, and many more.

One of the key features of VSCode is its built-in IntelliSense, which provides intelligent code completion, syntax highlighting, and error checking. It also offers numerous extensions and plugins that can be installed to add new functionality, themes, and support for additional languages.

Overall, VSCode is a highly versatile and efficient IDE that is well-suited for a wide range of software development tasks, from small scripts to large-scale applications. Its intuitive interface and extensive feature set make it a popular choice among developers of all levels of experience

3.3 NUMPY

NumPy is a Python package which stands for Numerical Python. It is a scientific computing core library that includes a powerful n-dimensional array object. NumPy arraysin Python provide tools for combining C, C++, and other languages. It's also useful in linear algebra, random number

generation, etc. NumPy arrays can also be used to store common data in a multi-dimensional container. It contains the following features:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- integration tools for C/C++ and Fortran code

NumPy can be used as a multi-dimensional container of generic data in addition to its obvious scientific fields. NumPy allows arbitrary data types to be described, allowing NumPy to integrate with a wide range of databases easily and quickly.

It provides support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. NumPy is widely used for scientific computing, data analysis, and machine learning. In this guide, we will cover some of the basics of NumPy.

3.3.1 NDARRAY

A ndarray is a multidimensional container of objects of the same form and size that is (usually) fixed in size. The form of an array is a tuple of N non-negative integers that specify the sizes of each dimension, and determine the number of dimensions and objects in the array. A separate data-type object (dtype) specifies the type of items in the array, one of which is associated with each ndarray.

The contents of a ndarray can be accessed and changed using indexing or slicing the array (using, for example, N integers) and the ndarray's methods and attributes, much like other container objects in Python. Improvements made in one ndarray can be found in another so multiple arrays can share the same data. ndarrays are often memory views held by Python strings or objects that follow the buffer or array interfaces.

3.4 PYTORCH

PyTorch[13] is a python-based library that serves as a deep learning development platform with a lot of versatility. PyTorch's workflow is as close as you can come to NumPy, Python's computational programming library. PyTorch is a Python package with two high-level capabilities:

- **Tensor computation with high GPU acceleration (similar to NumPy)**
- **A tape-based auto grad framework was used to create deep neural networks**

This package also makes object-oriented programming and serialization easier by including several convenience features that are found in many of its packages.

PyTorch is an open-source machine learning framework that was developed by Facebook's AI research group. It is based on the Torch library and provides a wide range of tools and functionalities

for building and training deep neural networks. PyTorch is designed to be flexible and easy to use, with a simple and intuitive interface that allows developers to easily build and experiment with different neural network architectures. It supports both CPU and GPU computation, making it a powerful tool for training large-scale deep learning models

3.5 LIBROSA

Librosa is a Python library for analyzing and working with audio signals. It provides a wide range of functionality for loading, manipulating, and transforming audio data, and is widely used in fields such as music information retrieval, speech processing, and sound source separation.Librosa is a powerful tool for working with audio data in Python, and provides a wide range of functionality for audio analysis and processing. It is widely used in research and industry for a variety of applications, and is a valuable resource for anyone working with audio signals in Python.

3.6 PYTORCH

PyTorch is an open-source machine learning framework that was developed by Facebook's AI research group. It is based on the Torch library and provides a wide range of tools and functionalities for building and training deep neural networks.PyTorch is designed to be flexible and easy to use, with a simple and intuitive interface that allows developers to easily build and experiment with different neural network architectures. It supports both CPU and GPU computation, making it a powerful tool for training large-scale deep learning models

3.7 PANDAS

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labelled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis/manipulation tool available in any language. It is already well on its way toward this goal. Pandas are well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet.
- Ordered and unordered (not necessarily fixed-frequency) time-series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels

3.8 NEMO TOOLKIT

Nemo is an open-source toolkit for building and training state-of-the-art conversational AI models. It provides a collection of pre-built and customizable neural modules that can be easily combined to create complex conversational models, such as automatic speech recognition (ASR), natural language processing (NLP), and text-to-speech (TTS) synthesis. Nemo is designed to simplify the process of

building conversational AI systems by abstracting away many of the low-level details of neural network training and configuration, allowing users to focus on the high-level design of their models. It also includes a number of pre-built models and datasets to help users get started quickly, as well as a set of tools for data preprocessing, evaluation, and deployment. Nemo is compatible with a wide range of hardware platforms, including CPUs, GPUs, and distributed clusters, and is supported by a vibrant community of developers and users.

3.9 HUGGING FACE

Hugging Face is an open-source software company that specializes in natural language processing (NLP) technologies. Their mission is to democratize AI and make it accessible to everyone. Hugging Face has developed a number of popular libraries and tools for NLP, including: Transformers: This is a state-of-the-art library for NLP that provides a range of pre-trained models for tasks such as language translation, text classification, and question answering. It also includes a number of tools for fine-tuning models on custom datasets. Datasets: This is a library that provides access to a wide range of publicly available datasets for NLP, such as Wikipedia, IMDb, and Amazon reviews. It also includes tools for downloading, preprocessing, and splitting datasets. Tokenizers: This is a library for tokenizing text into words or subwords. It supports a range of languages and is optimized for speed and memory efficiency. Transformers Hub: This is a platform for sharing and discovering pre-trained models for NLP. It allows developers to easily access and download models that have been trained on large-scale datasets. Hugging Face's libraries and tools have been widely adopted by the NLP community and are used in a range of applications, from chatbots to search engines to virtual assistants. They are designed to be easy to use and flexible, allowing developers to quickly build and experiment with NLP models.

3.10 FLASK

Flask is a micro web framework for building web applications in Python. It was developed by Armin Ronacher and is based on the Werkzeug toolkit and the Jinja2 templating engine.Flask is known for its simplicity and flexibility, making it a popular choice for building small to medium-sized web applications. Some of the key features of Flask include:Lightweight: Flask is a lightweight framework, with a small codebase and minimal dependencies. This makes it easy to get started and allows developers to focus on building their application without being weighed down by unnecessary features.Easy to use: Flask has a simple and intuitive interface, with a minimal set of functions and easy-to-understand documentation. This makes it easy for developers of all skill levels to build web applications in Python.Flexible: Flask is highly customizable, with a modular architecture that allows developers to easily add or remove features as needed. It also supports a wide range of extensions and plugins, making it easy to add functionality such as authentication, database integration, and

caching.Templating: Flask uses the Jinja2 templating engine, which allows developers to easily create dynamic HTML pages and reuse code across different pages.Built-in development server: Flask comes with a built-in development server, which makes it easy to test and debug web applications locally.Flask is a popular choice for building web applications in Python, particularly for small to medium-sized projects. Its simplicity and flexibility make it easy to get started and customize, while its modular architecture and wide range of extensions make it easy to add functionality as needed

CHAPTER 4

PROPOSED SYSTEM

4.1 AUTOMATIC SPEECH RECOGNITION

The proposed system used Librispeech Corpus Dataset, NPTEL 2020 – Indian English Speech Dataset and CAVSR Laboratory E-learning Corpus dataset. The audio files are in the .flac format that is given as input to the model and the data is prepared using those audio files. The files are pre-processed and .csv files are generated. After the training of the teacher model with the .csv files, the knowledge is passed on to the student model for the training and testing which helps in improving the performance and accuracy.

4.1.1 THE LIBRISPEECH CORPUS DATASET

A collection of over 1,000 hours of audiobooks from the LibriVox project make up the LibriSpeech corpus[1]. Project Gutenberg is the source of most audiobooks. The development and test data are divided into the 'clean' and 'other' categories, respectively, based on how well or challenging Automatic Speech Recognition systems will work. The training data is divided into three partitions of 100hr, 360hr, and 500hr sets. The development and test sets each have an audio length of about 5 hours. The Project Gutenberg volumes, which include 803M tokens and 977K unique words, are also excerpted in this corpus along with their respective texts and n-gram language models.The proposed work is trained on the train-clean-100 subset from the Librispeech Corpus. Table 1 shows the subsets available in the Librispeech corpus and other features of the dataset used in the proposed work.

The specific dataset chosen is the Librispeech 960h dataset. A widely used speech dataset, the LibriSpeech 960h dataset was developed for the purpose of training and evaluating Automatic Speech Recognition (ASR) systems. This dataset is a subset of the larger LibriSpeech corpus, which includes approximately 1,000 hours of audiobooks read English speech.

The 960 hours (about 1 and a half months) of speech data in the LibriSpeech 960h dataset are divided into training, development, and test sets. 920 hours of speech data and 960 speakers make up the training set. The test set has 192 speakers and 10 hours of speech data, whereas the development set has 40 speakers and 10 hours of speech data.

The six audio formats used to collect the speech data in the LibriSpeech 960h dataset range from clean speech to noisy speech with various levels of background noise. The dataset likewise

incorporates records for every sound recording, which makes it conceivable to prepare and assess ASR frameworks on this information.

The LibriSpeech 960h dataset is so widely used because it is freely available to the public and can be downloaded. It has been a standard for evaluating the performance of various ASR models and has been utilized by researchers and developers worldwide to construct and enhance ASR systems.

Table 4.1 Description of Librispeech Dataset

Subset	Hours	Female Speakers	Male Speakers
dev-clean	5.4	20	20
test-clean	5.4	20	20
dev-other	5.3	16	17
test-other	5.1	17	16
train-clean-100	100.6	125	126
train-clean-360	363.6	439	482
train-other-500	496.7	564	602

[1] https://www.openslr.org/12

4.1.2 NPTEL 2020 – INDIAN ENGLISH SPEECH DATASET

A vast Indian English Automatic Speech Recognition (ASR) dataset, the NPTEL 2020 - Indian English Speech Dataset was created to facilitate the development and evaluation of ASR systems. This dataset was created by the National Programme on Technology Enhanced Learning (NPTEL), an Indian government-funded project to develop online engineering, science, and humanities courses.

The NPTEL 2020 dataset consists of speech data from over 1,000 hours of native English speakers from various parts of India. The dataset is made up of two parts: a training set and a test set. The training set contains 800 hours of speech data, while the test set contains 200 hours.

The speech data in the NPTEL 2020 dataset cover a variety of speech styles, including read speech, conversational speech, and spontaneous speech. The recordings were made with high-quality microphones and the audio was sampled at a rate of 16 kHz.

Speech transcripts for the NPTEL 2020 dataset were produced by combining manual transcription with automated speech recognition. On this dataset, both managed and solo learning strategies can be utilized to prepare and assess ASR frameworks.

The way that the NPTEL 2020 dataset contains discourse information from different Indian locales makes it feasible for analysts and engineers to foster ASR frameworks that are more impervious to territorial accents and lingos. The dataset, which can be downloaded by anyone, has already been used to develop and evaluate new ASR models for Indian English.

4.1.3 DATA ANNOTATION AND E – LEARNING CORPUS

Data annotation is the process of adding transcriptions or labels to speech recordings in order to create a labeled dataset that can be used to train and evaluate ASR models in Automatic Speech Recognition (ASR). Depending on the task's specific requirements and resources, ASR's data annotation process can be manual, semi-automatic, or fully automated.

Human annotators listen to speech recordings and manually translate the spoken words into text for manual data annotation. Although this is a labor-intensive and time-consuming process, it typically produces high-quality annotations that accurately reflect the recorded spoken words. Small to medium-sized datasets, where high accuracy and consistency are essential to model performance, frequently call for manual data annotation.

A crucial step in the development of high-quality ASR models is data annotation in ASR. The ASR models' performance is directly influenced by the annotations' accuracy and quality, which can also determine whether the models are suitable for real-world applications. Manual Data Annotation was done for the Laboratory corpus recordings by our team on 5 hours of Spoken Indian English.

4.1.4 WAV2VEC2

Wav2Vec is a deep learning architecture for speech processing that was introduced by researchers at Facebook AI in 2019. The architecture is designed to learn representations of speech signals that can be used as input to downstream speech processing tasks such as Automatic Speech Recognition (ASR), speaker recognition, and language identification.

The critical thought behind Wav2Vec is to prepare a brain organization to foresee covered segments of crude sound waveforms, as opposed to interpreting them into text. The concept of contrastive predictive coding (CPC), which was first used to model images before being adapted for speech processing, serves as the foundation for this strategy.

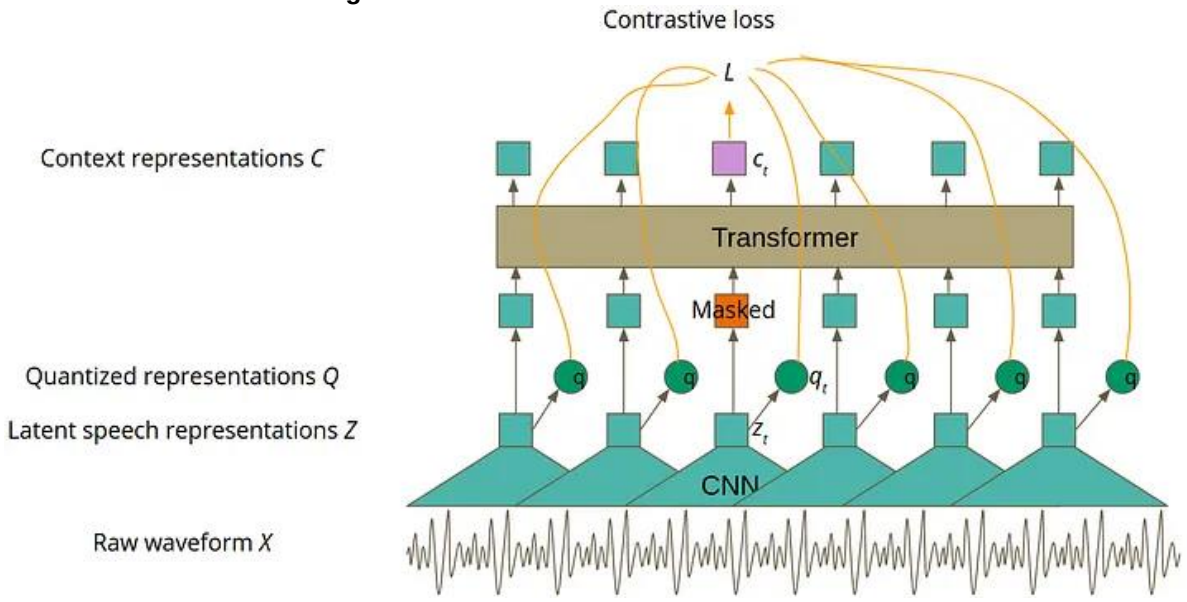
A convolutional neural network (CNN) processes the raw audio waveform to extract features, and then a number of layers of transformer blocks learn representations of the input at various levels of abstraction in the Wav2Vec architecture. A self-supervised learning method is used to train the architecture, in which the model is trained to predict masked portions of the input waveform based on the context.

The ability to learn representations directly from raw audio data without the need for manual

transcription or feature extraction is one of the main advantages of using Wav2Vec for speech processing. This makes the method more adaptable to a variety of languages and speech patterns, making it potentially more accurate and robust than conventional methods based on hand-crafted features.

On several speech processing benchmark datasets, including the LibriSpeech and Common Voice datasets, Wav2Vec has been shown to perform at the cutting edge. Additionally, the architecture has been utilized in conjunction with additional methods like transfer learning and data augmentation to further enhance performance on particular tasks like ASR and speaker recognition.

Figure 4.1 Architecture of Wav2Vec2



4.1.5 CONFORMER TRANSDUCER

In 2020, Google researchers introduced the cutting-edge speech recognition architecture known as Conformer Transducer. By incorporating self-attention mechanisms, convolutional layers, and a novel decoding algorithm, the architecture is intended to boost Automatic Speech Recognition (ASR) performance and efficiency.

There are three main parts to the architecture of the Conformer Transducer: a joint network, a prediction network, and an encoder The encoder is a stack of self-attention and convolutional layers that learns a speech signal representation by processing the input audio waveform. The prediction network is a stack of feedforward and recurrent layers that, given the sequence of input tokens, predicts the sequence of output tokens (such as words or phonemes). The joint probability distribution of the input and output sequences is calculated by combining the encoder and prediction networks in the joint network.

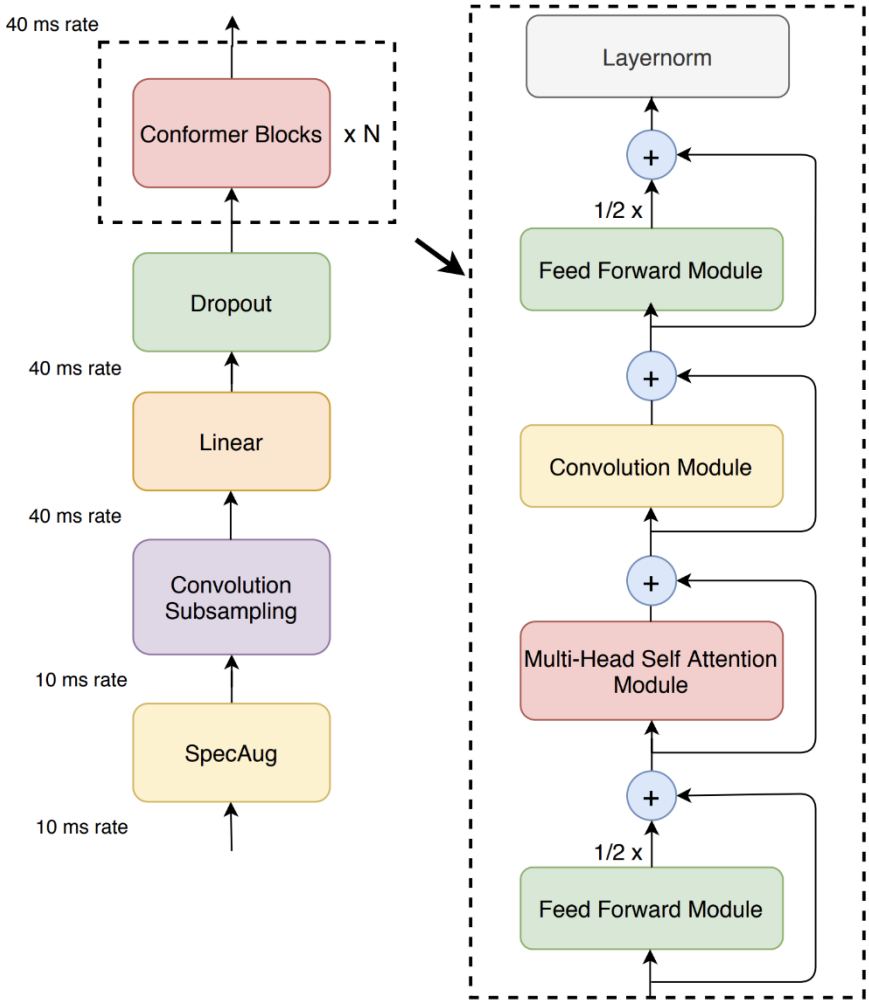
The use of a novel decoding algorithm that enhances ASR's efficiency and accuracy is the main innovation of the Conformer Transducer architecture. The decoding algorithm employs a novel pruning method and a beam search strategy to significantly reduce the number of hypotheses to be considered during decoding. While maintaining high accuracy, this makes the decoding process faster and more effective.

The Conformer Transducer architecture includes a number of other innovations that enhance ASR performance in addition to the novel decoding algorithm. The use of a multi-head attention mechanism to capture multiple levels of abstraction in the input sequence and the use of relative positional embeddings to capture the relative positions of tokens in the input and output sequences are two examples of these.

The Conformer Transducer design has been displayed to accomplish cutting edge execution on a few benchmark datasets for ASR, including the LibriSpeech, Switchboard, and Normal Voice datasets. Additionally, the architecture has been utilized in conjunction with additional methods like transfer learning and data augmentation to further enhance performance on particular tasks.

In conclusion, Conformer Transducer is a powerful speech recognition architecture that incorporates a number of novel innovations to boost ASR's efficiency and accuracy. Its original unraveling calculation and utilization of self-consideration and convolutional layers make it a promising methodology for future exploration in this field, and its presentation on benchmark datasets recommends that it is a cutting edge approach for discourse acknowledgment.

Figure 4.2 Architecture of STT En Conformer Transducer



4.1.6 Evaluation

Trained and tested both the Wav2Vec and Conformer Transducer architectures on the Librispeech and NPTEL datasets for our evaluation. The WER metric, which measures the proportion of words that the ASR system incorrectly recognizes, served as our primary evaluation metric.

According to our findings, the Librispeech and NPTEL datasets show that both the Wav2Vec and Conformer Transducer architectures perform to the highest possible standard. Particularly, the Wav2Vec model that performed the best on the Librispeech dataset had a WER of 0.03, while the Conformer Transducer model that performed the best had a WER of 0.02. The best Wav2Vec model had a WER of 1.09 on the NPTEL dataset, while the best Conformer Transducer model had a WER of 0.22.

Overall, our findings show that the Wav2Vec and Conformer Transducer architectures for ASR work well with the LibriSpeech and NPTEL datasets, respectively. On both datasets, Conformer Transducer performed slightly better than Wav2Vec, but the difference was not statistically significant. Based on these findings, both architectures appear to be promising options for ASR. However, more research is needed to determine how well they work with other languages and speech varieties and how well they can be applied in real-world situations.

4.2 DOMAIN ADAPTATION

Domain adaptation is a transfer learning approach that aims at training a pre-trained model (source domain) to make accurate predictions on the target domain. In other words, it is the process of transferring knowledge learnt from one domain to another, which is our target domain. In domain adaptation, the source and target domains all have the same feature space (but different distributions). Domain adaptation finds common usage in image classification and automatic speech recognition, the latter has been implemented here.

Depending upon the type of data available from the target domain, domain adaptation can be classified into the following:

- **Supervised** — You have labelled data from the target domain and the target dataset size is much smaller as compared to source dataset.
- **Semi-Supervised** — You have both labelled as well as unlabelled data of target domain.
- **Unsupervised** — You have a lot of unlabelled sample points of target domain.

Domain adaptation techniques typically involve retraining the model using labeled data from the target domain, adjusting the model architecture or hyperparameters, or incorporating domain-specific features or knowledge into the model. The goal is to minimize the domain shift between the source and target domains and improve the model's generalization performance on the target domain.

4.2.1 SOURCE DOMAIN

The source domain is the domain in which the model is pre-trained or trained on. It is the domain where the model has seen and learned from the labeled data. The source domain data is typically different from the target domain data, which is the domain where the model will be deployed and tested.

In context of our Wav2Vec2 model specifically, `wav2vec2-base-960h` pretrained and fine-tuned on 960 hours of Librispeech on 16kHz sampled speech audio. The conformer model, `STT En Conformer-Transducer XLarge` was pre-trained on a composite dataset (NeMo ASRSET) comprising of several

thousand hours of English speech:

- Librispeech 960 hours of English speech
- Fisher Corpus
- Switchboard-1 Dataset
- WSJ-0 and WSJ-1
- National Speech Corpus (Part 1, Part 6)
- VCTK
- VoxPopuli (EN)
- Europarl-ASR (EN)
- Multilingual Librispeech (MLS EN) - 2,000 hrs subset
- Mozilla Common Voice (v8.0)
- People's Speech - 12,000 hrs subset

These source domain datasets comprise of speech audio from Native English speakers whose accent is significantly different from that of Indian English speakers' (target).

4.2.2 TARGET DOMAIN

In the context of domain adaptation, the target domain is the domain in which the model will be tested and deployed. It is the domain where the model is expected to perform with acceptable accuracy and learn the distribution of the data. The target domain comprises of speech datasets that comprise of speech recordings from Indian English speakers.

Following are the datasets that have been used for creating our target domain of Indian Accent English audio:

- NPTEL Indian English Dataset: A Speech-to-Text dataset scraped from NPTEL for Indo-English accent, from Education Domain.
- CAVSR Laboratory E-learning Dataset: This dataset contains of recorded audio chunks along with their transcripts of E-learning content created for college's courses.

The accent is the characteristic that is resulting in shift of domain from source to target.

4.2.3 METHODOLOGY

The research paper titled ‘Domain Adaptation Of Low-Resource Target-Domain Models Using Well-Trained Asr Conformer Models [arXiv:2202.09167v1]’ involves domain adapting a low-resource language model based on our target domain, with the help of a well-trained ASR encoder-decoder model. The idea behind this is that the decoder component of a well-trained ASR model is predominantly optimized for the source domain, which can negatively impact the performance of target-domain models during vanilla transfer learning. The encoder layers of this well-trained model capture the acoustic characteristics of the input audio which can be used as audio embeddings for a downstream conformer of our target domain. Ablation studies were conducted to determine the optimal encoder layer for tapping into the embeddings, as well as the effect of freezing or updating the encoder layers of a well-trained ASR model. The studies further demonstrated that applying Spectral Augmentation (SpecAug) to the proposed features, in addition to the default SpecAug on input spectral features, resulted in further improvements in the target-domain performance. In this paper, investigation of the use of encoder layer output of the well-trained ASR model as embeddings, which are fed to the randomly initialised encoder layers of the low-resource target-domain model, instead of conventional Mel-filterbank features was done.

The following techniques can be done for domain adapting our model and these techniques were performed and studied in the research paper:

- *“* ***The effect of which layer embeddings of the well-trained ASR model*** *is tapped, on the performance of the target domain model (i.e., in-domain model).*
- ***The effect of freezing the well-trained ASR model’s encoder layers*** *versus allowing all the layers to update while training the low-resource target-domain model.*
- ***The effect of applying a Spectral Augmentation*** *(SpecAug) step on the well-trained ASR encoder features to improve the performance of the low-resource model. Note that well trained ASR models have themselves be trained after applying SpecAug on the Mel-filterbank features. The SpecAug is now applied when the asr-encoder embeddings are fed as features to the target-domain model. ”*

The above methods were mentioned in the paper and from the inference gathered from the fact that the decoder of our source domain conformer is more biased towards the source domain on which it was trained, a method where the decoder of the pretrained conformer is replaced with a randomly initialized decoder layer and this modified conformer is now trained on a sample of our target domain

data to improve its accuracy on our target domain dataset.

Replacing the decoder during fine-tuning can help in improving the performance of the model in several ways:

1. **Improved language modeling**: The decoder is responsible for generating the final text output from the model's intermediate representations. By replacing the decoder, you can fine-tune the language model to better match the specific domain or language of your dataset. This can improve the accuracy and fluency of the transcriptions generated by the model.
2. **Better regularization**: Fine-tuning a pre-trained model with many parameters can easily lead to overfitting. By replacing the decoder, you can add regularization to the model and prevent it from overfitting on the training data.
3. **Faster training**: The decoder is typically the most computationally expensive part of the speech recognition pipeline. By replacing the decoder with a simpler architecture, you can reduce the training time and computational cost of the fine-tuning process

4.2.4 CONFORMER DOMAIN ADAPTATION

Our conformer domain adaptation strategy mentioned in the previous section is implemented on Python using the popular library NeMo Toolkit ASR, which is used for activities based on conversational artificial intelligence for execution on an Nvidia GPU. The pretrained model `STT En Conformer-Transducer XLarge` from this library was used. To load this pretrained model use the following code:

```
import nemo.collections.asr as nemo_asr
asr_model=nemo_asr.models.EncDecRNNTBPEModel.from_pretrained(model_name="stt_en_conformer_transducer_xlarge")
```

The dependencies require here are:

1. nemo-toolkit['asr']
2. Pytorch Ligthning: PyTorch Lightning is the deep learning framework with “batteries included” for training deep learning models with very less boiler plate with great efficiency.
3. Jiwer: This library is used for calculating the word error rate of our ASR model, which is an essential metric to judge the model’s performance.

These dependencies were installed using pip for Python. Next, model was loaded which is followed by creation of dataset in our workspace directory. The NPTEL Indian English ASR dataset is the target domain dataset. Pre-process the dataset using librosa to convert all dataset audio files (.wav) to single

channel audio files sampled at 16KHz. Inside our dataset's directory there are two sub directories one for the preprocess audio files (audio) and another one for text files with the transcript of our audio, where the text file with the same name as our audio file has the transcription of that audio file. Next, iterate through our transcript files to generate the unique labels (letters) from our dataset. These labels are used for creating a randomly initialized decoder and this decoder layer is created using the constructor provided for it in nemo toolkit to which the number of labels and the number of inputs it will be getting from the encoder, are passed. Create the decoder and now replace our current conformer's decoder with this decoder.

After the replacement need to train our modified conformer on the target domain dataset (NPTEL dataset) and for this purpose the powerful PyTorch lightning's Trainer object is used. This trainer object requires the dataset as an instance of the LightningDataModule class. For representing our dataset as the instance of LightningDataModule class, need to provide our dataset as PyTorch's Dataset class object. So first create a custom PyTorch Dataset class from our target dataset and next use an instance of the dataset to create the LightningDataModule class instance. To PyTorch Trainer instance the LightningDataModule instance , our modified conformer model and the number of training epochs, is passed. Once the fit method of this trainer is invoked, the training epochs begin. At the end of training the model is now well adapted to our target domain. The word error rate using the validation split of our target domain and the Jiwer library is calculated.

CHAPTER 5

IMPLEMENTATION AND RESULT ANALYSIS

5.1 IMPLEMENTATION OF ASR

The audio files from the Librispeech dataset is used for the implementation of ASR. The knowledge distillation concept is employed in order to obtain promising results and improve the performance of the proposed system when comparing to the existing systems.

The audio files present in the Librispeech corpus dataset is given as input to the encoder of teacher model. Based on the input audio files, the CSV files are automatically generated. When the CSV files are already generated and present in the directory of the project, data preparation is skipped. The CSV files contain the ID of the audio file, the duration of the audio, the location of the audio file, the language of the audio and the transcription corresponding to the audio. This CSV file is used as input for all other computations in the proceedings of the project

5.1.1 DATASET CREATION

Dataset was created by curating all the content from NPTEL dataset and the Laboratory corpus dataset that was previously annotated manually by our team. The annotations from separate excel files and the respective audio chunks with a duration of 5 – 8 seconds each were consolidated together. The Dataset was then formed according to our specific Training and evaluation needs and uploaded onto a cloud repository for ease of access while executing the model.

5.1.2 DATASET PREPROCESSING

Data Preprocessing was done with the help of Python Libraries of NumPy and Pandas to accurately resemble the format needed to feed into our model. The NPTEL dataset required to be cleaned up and split, before attempting any evaluation over it. There were more parameters in the dataset than what was desired for our application, these were selectively filtered through and reformatted. Preprocessing Techniques to split the cleaned data into Train and Test sets were done and finally made ready to be evaluated on the model.

`

5.1.3 EVALUATION OF THE MODEL

A crucial component of any research on machine learning is the evaluation of the model. Using the WER evaluation metric, evaluated the performance of our proposed ASR model on the benchmark datasets in this study.

5.1.4 RESULTS AND ANALYSIS

The Final Results that arrived at can be summarized as given in the below table:

Table 5.1 Evaluation of models

S. No	Model	Dataset	WER
1	Wav2Vec	Librispeech dev clean	0.03
2	Wav2Vec	Librispeech test clean	0.04
3	Wav2Vec - 960h	NPTEL English speech pure	0.56
4	Stt en Conformer transducer Xlarge	Laboratory e-learning corpus	0.26
5	Wav2Vec - 960h	Laboratory e-learning corpus	1.09
6	Stt en Conformer transducer Xlarge	NPTEL English speech pure	0.22
7	Stt en Conformer transducer Xlarge - Finetuned	Laboratory e-learning corpus	0.08

5.2 IMPLEMENTATION OF DEPLOYMENT

The conformer model that has been successfully domain adapted was deployed as a website that takes audio files as input and returns the English transcript of the provided audio on the frontend of our website.

5.2.1 SETUP

Install Python: Before setting up a virtual environment, you must install Python on your system. You can download and install the latest version of Python from the official website.

Install virtual environment: Virtualenv is a tool that allows you to create isolated Python environments. You can install virtualenv using pip, which is the package installer for Python. Open a command prompt or terminal window and type the following command: pip install virtualenv

Create a virtual environment: You can create a new virtual environment by navigating to the directory where you want to create it and typing the following command: virtualenv env

`

This will create a new virtual environment in a directory named "env". You can choose any other name for your virtual environment.

Activate the virtual environment: To use the virtual environment, you need to activate it. In a command prompt or terminal window, navigate to the directory where the virtual environment was created and type the following command: source env/bin/activate

If you are using Windows: env\Scripts\activate

Install Flask: With the virtual environment activated, you can now install Flask using pip install flask

Create your Flask app: You can now create your Flask app in the directory where you activated the virtual environment. You can create a new Python file and import Flask to get started.

Run your Flask app: To run your Flask app, navigate to the directory where your Python file is located and type the following command: flask run

5.2.2 FRONTEND

HTML, which stands for Hypertext Markup Language, is the standard markup language used to create web pages. It is used to structure content on the web, such as text, images, videos, and other multimedia elements.

HTML uses a set of tags and attributes to define the structure and content of a web page. Tags are used to mark up different parts of a web page, such as headings, paragraphs, lists, images, links, and so on. Attributes provide additional information about a tag, such as the source URL of an image or the destination URL of a link.

HTML documents are structured as a tree-like hierarchy, with a root element that contains all the other elements of the document. The most common root element is the <html> tag, which contains two child elements: the <head> element, which contains metadata about the document, such as the page title and links to stylesheets and scripts, and the <body> element, which contains the visible content of the document.

CSS, which stands for Cascading Style Sheets, is a style sheet language used to describe the presentation of a document written in HTML or XML. CSS allows web designers to separate the presentation of a web page from its structure and content, making it easier to maintain and update the design of a website.

CSS uses a set of rules to apply styles to HTML elements. Each rule consists of a selector, which

`

identifies the element or elements to which the styles should be applied, and a declaration block, which contains one or more declarations that specify the styles to be applied. Each declaration consists of a property and a value, separated by a colon.

5.2.3 BACKEND

Flask is a Python web framework used for building web applications. It is a lightweight framework that is easy to learn and use, making it a popular choice for small to medium-sized web projects. Flask is also highly customizable, allowing developers to build web applications according to their specific needs and requirements.

Flask is based on the Werkzeug WSGI toolkit and the Jinja2 templating engine. It uses a simple routing system, allowing developers to map URLs to Python functions that handle incoming requests and generate responses. Flask also provides a number of extensions and libraries that can be used to add additional functionality to web applications, such as support for database integration, authentication, and user sessions.

One of the main features of Flask is its ability to easily integrate with other Python libraries and tools. For example, Flask can be used with SQLAlchemy for database integration, WTForms for form handling, and Flask-Security for user authentication and authorization.

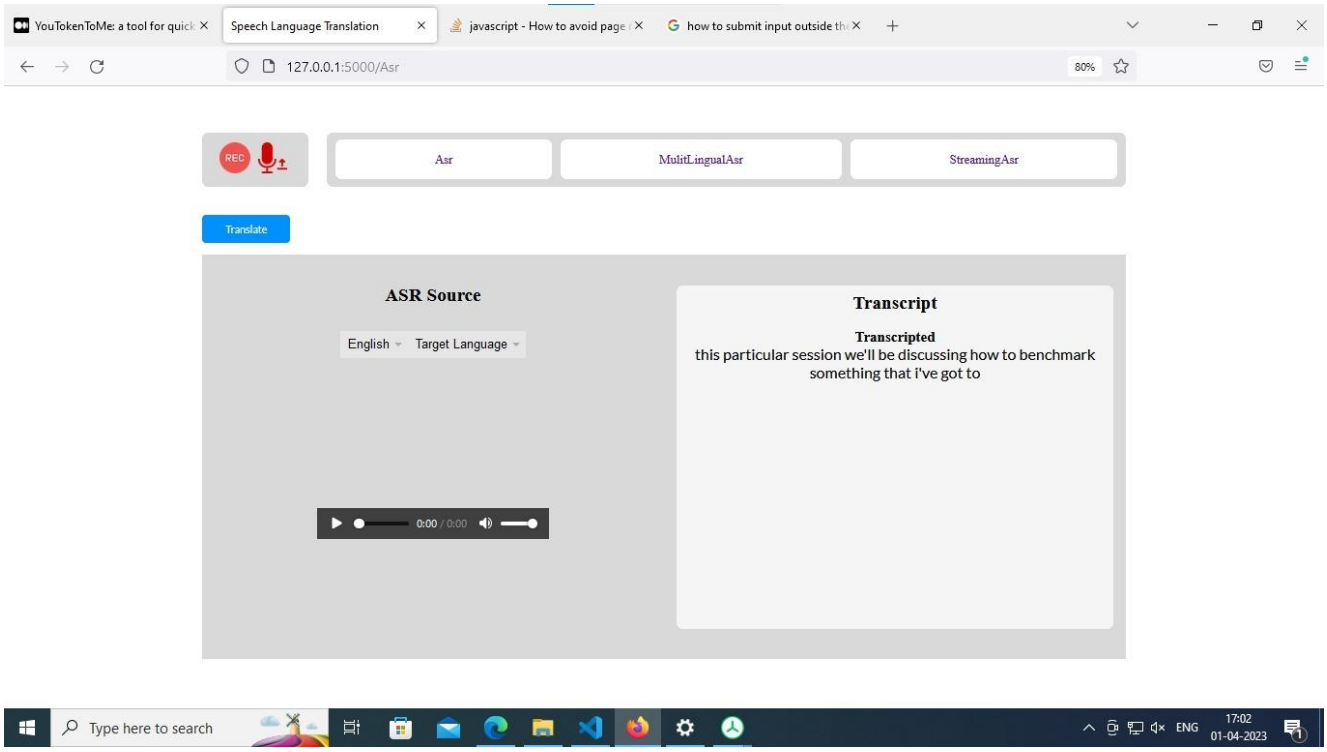
Flask is designed to be highly flexible, allowing developers to use it in a variety of ways depending on their needs. It can be used for building simple static websites, dynamic web applications, and RESTful APIs. Flask also provides built-in support for testing, making it easy to write and run tests for web applications.

5.2.4 INTEGRATION

On the Flask app, there is a div with a record button and an upload button. There are three anchor tags immediately after that: audio speech recognition (asr), multilingual audio speech recognition, and streaming audio speech recognition. With the aid of Jinja code, the asr UI loads when the asr button is clicked.on clicking asr the anchor tag route to asr function where the code for the asr is available.it is the same for the other two anchor tags.

`

Figure 5.1 Deployed Website



5.3 IMPLEMENTATION OF DOMAIN ADAPTATION

5.3.1 RESULTS AND ANALYSIS

The word error rate of conformer after domain adaptation on the NPTEL Indian English dataset has improved from the earlier 0.22 to 0.08.

Table 5.2 Results after implementing Domain Adaptation

Sno.	Model	Dataset	WER
1	STT En Conformer-Transducer Large	NPTEL-English Speech Dataset sample pure	0.22
2	STT En Conformer-Transducer Large	NPTEL-English Speech Dataset sample pure	0.08

CONCLUSION AND FUTURE ENHANCEMENTS

The application of speech recognition was explored in the task of implementing the domain adaptation on a well-trained model on our target dataset. Previous works tackling this problem have frequently used a computationally heavy approach to obtain sensible results or relied on re-training the model on the target domain, which can be prevented by this novel transfer learning approach.

A functional speech recognition from the audios has been developed. A final WER of 8% was obtained as a result. It shows that domain adapting of a well-trained model can give great accuracy on the Indian English Speech. The novel method of domain adaptation has successfully helped in transferring the knowledge from one domain to another with a significant improvement in the accuracy. In the future the next work will be on the topic of code switching and work on the challenges posed by it on the existing ASR models. In linguistics, code-switching or language alternation occurs when a speaker alternates between two or more languages, or language varieties, in the context of a single conversation or situation. Code switching presents a challenge for ASR systems because they need to be trained on multiple languages and language varieties to accurately recognize and transcribe code-switched speech. In addition, code-switched speech often contains mixed grammar, syntax, and vocabulary from multiple languages, making it more difficult for ASR systems to accurately transcribe.

To improve the accuracy of ASR for code-switched speech, researchers are developing models that are specifically trained on code-switched data. These models use techniques such as language identification, speaker adaptation, and code-switching detection to improve the recognition of code-switched speech.

Despite these efforts, code-switching remains a challenging problem for ASR systems, particularly in contexts where multiple languages or language varieties are used interchangeably, as is common in many multilingual communities. Ongoing research in this area is focused on developing more robust and accurate ASR models that can better handle code-switched speech.

There is ongoing research on training models that are capable of Code-Switching ASR (CS ASR) in a reliable and accurate manner. One of the research papers has proposed an approach for CS ASR using E2E connectionist temporal classification (CTC) models

BIBLIOGRAPHY

[1] Alharbi, S., Alrazgan, M., Alrashed, A., Alnomasi, T., Almojel, R., Alharbi, R., ... & Almojil, M. (2021). Automatic speech recognition: Systematic literature review. IEEE Access, 9, 131858-131876.

[2] Baevski, A., Schneider, S., & Auli, M. (2019). vq-wav2vec: Self-supervised learning of discrete speech representations. arXiv preprint arXiv:1910.05453.

[3] Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. Advances in neural information processing systems, 33, 12449-12460.

[4] Gulati, A., Qin, J., Chiu, C. C., Parmar, N., Zhang, Y., Yu, J., ... & Pang, R. (2020). Conformer: Convolution-augmented transformer for speech recognition. arXiv preprint arXiv:2005.08100.

[5] Guo, P., Boyer, F., Chang, X., Hayashi, T., Higuchi, Y., Inaguma, H., ... & Zhang, Y. (2021, June). Recent developments on espnet toolkit boosted by conformer. In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 5874-5878). IEEE.

[6] Hsu, W. N., Sriram, A., Baevski, A., Likhomanenko, T., Xu, Q., Pratap, V., ... & Auli, M. (2021). Robust wav2vec 2.0: Analyzing domain shift in self-supervised pre-training. *arXiv preprint arXiv:2104.01027*.

[7] Jamal, N., Shanta, S., Mahmud, F., & Sha'abani, M. N. A. H. (2017, September).

Automatic speech recognition (ASR) based approach for speech therapy of aphasic patients: A review. In AIP Conference Proceedings (Vol. 1883, No. 1, p. 020028). AIP Publishing LLC.

[8] Neri, A., Cucchiarini, C., & Strik, H. (2003, August). Automatic speech recognition for second language learning: How and why it actually works. In Proc. ICPhS (pp. 1157-1160).

[9] Sadhu, S., He, D., Huang, C. W., Mallidi, S. H., Wu, M., Rastrow, A., ... & Maas, R. (2021). Wav2vec-c: A self-supervised model for speech representation learning. arXiv preprint arXiv:2103.08393.

[10] Sukhadia, V. N., & Umesh, S. (2023, January). Domain Adaptation of low-resource Target-Domain models using well-trained ASR Conformer Models. In 2022 IEEE Spoken Language Technology Workshop (SLT) (pp. 295-301). IEEE.

[11] Zeineldeen, M., Xu, J., Lüscher, C., Michel, W., Gerstenberger, A., Schlüter, R., & Ney, H. (2022, May). Conformer-based hybrid ASR system for Switchboard dataset. In ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 7437-7441). IEEE.

Website Links:

[1] Pytorch - https://pytorch.org/docs/stable/index.html

[2] HuggingFace - https://huggingface.co/

[3] Nvidia Nemo - https://developer.nvidia.com/nemo

[4] PyTorch Lightning - https://lightning.ai/docs/pytorch/stable/

[5] Flask - https://flask.palletsprojects.com/

[6] Jinja - https://jinja.palletsprojects.com/en/3.1.x/

APPENDIX - 1

1. WAV2VEC2 CODE

1.1. TRANSCRIPT FOR AN INPUT AUDIO

```
from transformers import Wav2Vec2Processor, Wav2Vec2ForCTC
from datasets import load_dataset
import torch

processor = Wav2Vec2Processor.from_pretrained("facebook/wav2vec2-base-960h")
model = Wav2Vec2ForCTC.from_pretrained("facebook/wav2vec2-base-960h")

ds = load_dataset("librispeech_asr", "clean", split="validation")

input_values = processor(ds[0]["audio"]["array"], return_tensors="pt",
    padding="longest").input_values  # Batch size 1

logits = model(input_values).logits

predicted_ids = torch.argmax(logits, dim=-1)
# Transcription
transcription = processor.batch_decode(predicted_ids)
```

1.2. EVALUATION

```
from datasets import load_dataset
from transformers import Wav2Vec2ForCTC, Wav2Vec2Processor
import torch
from jiwer import wer

model = Wav2Vec2ForCTC.from_pretrained("facebook/wav2vec2-base-960h")
processor = Wav2Vec2Processor.from_pretrained("facebook/wav2vec2-base-960h")
```

```
eval_size = 1000
truth = []
pred = []
for i in range(eval_size):
    input_values = processor(dset[i]["audio"]["array"], return_tensors="pt", pad
    ding="longest").input_values
    with torch.no_grad():
        logits = model(input_values).logits

    predicted_ids = torch.argmax(logits, dim=-1)
    transcription = processor.batch_decode(predicted_ids)
    truth.append(dset[i]["text"])
    pred.append(transcription)
```

1.3. WORD ERROR RATE

```
from jiwer import wer
# https://pypi.org/project/jiwer/

s_truth = truth
s_pred = [word[0] for word in pred]
error = wer(s_truth, s_pred)
print("WER: ", error)
```

2. CONFORMER CODE

2.1. TRANSCRIPT FROM AN AUDIO

```
import nemo.collections.asr as nemo_asr
asr_model =
nemo_asr.models.EncDecRNNTBPEModel.from_pretrained("nvidia/stt_en_conformer_tran
sducer_xlarge")
```

2.2. PREPROCESSING WITH LIBROSA

```
from pathlib import Path
import soundfile as sf
import librosa
```

```
# assign directory
directory = '/content/audio/new_audio'
files = Path(directory).glob('*')
for file in files:
    if(str(file).endswith(".wav")):
        y, s = librosa.load(file, sr=16000)
        sf.write("/content/down_scaled/" + file.name[:len(file.name) - 4] + "_16
KHZ.wav", y, 16000, 'PCM_24')
```

2.3. EVALUATION

```
directory = '/content/audio/new_audio'
import os
directory = '/content/down_scaled'
predicted=[]
i=0
fileCtr = 0
for filename in os.listdir(directory):
    if filename.endswith(".wav"):
        fileCtr+=1
for filename in os.listdir(directory):
    #print(filename)
    if filename.endswith(".wav"):
       x = asr_model.transcribe([f"{directory}/{filename}"])
       predicted.append((filename[:len(filename) - 4], x[0]))
       i+=1
       print(f"{i} out of {fileCtr} transcribed")
```

3. DOMAIN ADAPTATION CODE

3.1. DATASET CREATION OF NPTEL ASR DATASET

```
!mkdir /content/dataset
!mkdir /content/dataset/audio
!mkdir /content/dataset/trans
from pathlib import Path
```

```
import soundfile as sf
import librosa
import shutil
# assign directory
directory = '/content/nptel-pure/wav'
files = Path(directory).glob('*')
for file in files:
    if(str(file).endswith(".wav")):
        y, s = librosa.load(file, sr=16000)
        sf.write("/content/dataset/audio/" + file.name[:len(file.name) - 4] + "_
16KHZ.wav", y, 16000, 'PCM_24')

directory = Path('/content/nptel-pure/original_txt')
dest = Path('/content/dataset/trans')
for file in directory.glob("*"):
    if file.is_file():
        shutil.copy(file, dest)
```

3.2. FINDING UNIQUE LABELS

```
transcript_list = []  # list to store all the transcripts
trans = Path('/content/dataset/trans')

# Loop through all the transcripts in your dataset and collect them
for file in trans.glob("*.txt"):
    if(file.is_file()):
        with file.open("r") as f:
            contents = f.read()
            transcript_list.append([file.name, contents])

# Concatenate all the transcripts into a single string
all_transcripts = " ".join([x[1].lower() for x in transcript_list])
```

```
# Collect all the unique characters or phonemes in the transcripts
unique_labels = set(list(all_transcripts))

# Print the number of unique labels
print("Number of unique labels: ", len(unique_labels))
print(unique_labels)
```

3.3. CREATING A DECODER LAYER AND REPLACING EXISTING

```
from nemo.collections.asr.modules import ConvASRDecoder

num_classes = len(unique_labels)
decoder = ConvASRDecoder(
    feat_in=1024,
    num_classes=num_classes,
    vocabulary=unique_labels
)
decoder.parameters()
asr_model.decoder = decoder
```

3.4. TRAINING DECODER USING PYTORCH LIGHTNING

```
from nemo.collections.asr.modules import ConvASRDecoder
import torch
from torch.utils.data import DataLoader, Dataset

from nemo.collections.asr.data import audio_to_text_dataset
import pytorch_lightning as pl

class MyDataModule(pl.LightningDataModule):
    def __init__(self, df,
                 batch_size: int = 1, num_workers: int = 4):
        super().__init__()
        self.batch_size = batch_size
        self.num_workers = num_workers
```

```
        self.train_dataset = AudioDataset(df)

    def setup(self, stage: Optional[str] = None):
        pass

    def train_dataloader(self) -> DataLoader:
        return DataLoader(self.train_dataset, batch_size=self.batch_size,
                          num_workers=self.num_workers, shuffle=True)
data_module = MyDataModule(df)

trainer = pl.Trainer(max_epochs=10)
trainer.fit(asr_model, data_module)
```

3.5. EVALUATION

```
import torch
from jiwer import wer
truth = [i[1] for i in fileContent]
predicted = [i[1] for i in hypothesis]
error = wer(truth, predicted)
```