# Automatic Speech Recognition for low resource language (Pashto) using wav2vec Model

by

**NS Jawaria Tahir**

**(00000318287)**

Supervisor

**Assoc Prof Shibli Nisar, PhD**

A thesis submitted to the faculty of Electrical Engineering Department Military College of Signals, National University of Sciences and Technology, Islamabad, Pakistan as part of the requirements for the degree of MS in

Electrical Engineering

August

2023

# THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis written by Mis **NS Jawaria Tahir,** Registration No. **00000318287** of **Military College of Signals** has been vetted by undersigned, found complete in all respect as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial, fulfillment for award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the student have been also incorporated in the said thesis.

Signature: [illegible]

Name of Supervisor **Assoc Prof Shibli Nisar, PhD**

Date: ______________________

Signature (HoD): [illegible]

Brig
HoD
Dept of Elec Engg
Military College of Signals (NUST)
(NUST Campus)

Date: ______________________

Signature (Dean): [illegible]

Date: 10/8/23

Brig
Dean, MCS (NUST)
(Asif Masood, Phd)

## Abstract

Due to advancements in technology, speech recognition has grown to be one of the most important aspects of human-computer interaction. In recent years, a tremendous amount of research has been conducted in the area of speech signal processing. Particularly, the subject of Automatic Speech Recognition (ASR) technology has seen an increase in interest. ASR started out with basic systems that could only recognize a handful of sounds, but it has now developed into complex systems that not only can understand but also corresponds to human speech with ease. Major languages have access to a wealth of ASR research, however low resource languages are still underrepresented in this area of study. The foundation of this research work is Automatic Speech Recognition, specifically for Pashto, a low resource language. The creation of low resource data repositories to evaluate the newest ASR trends specifically for Pashto is needed for current advancement in ASR technology. As a result of its successful implementation, native speakers may interact with computers via voice commands in their native languages and take full advantage of the digitization boom. Research in this field will benefit academics by expanding existing huge corpora and generating cutting-edge ASR models for languages with limited resources. This research will also aid in identifying current, significant ASR difficulties in practical settings. Additionally, it will highlight the present shortcomings in traditional ASR systems. Also, this work aims to implement ASR for low resource language Pashto by making use of Facebook's most recent algorithm, wav2vec2, which is quite latest in ASR trends in recent years. After successful development of dataset, we have trained and fine-tuned our model to the latest trend of ASR. Almost 66% output accuracy with 37% WER (word error rate) is obtained at the output end of our model, which is quite an achievement for a low resource ASR system.

## Declaration

I certify that the work *“Automatic Speech Recognition for low resource language (Pashto) using wav2vec Model”* exhibited in this thesis has not been submitted in support of any other award or educational qualification either at this institution or elsewhere.

## Acknowledgements

All praises to Almighty ALLAH who showered his blessing and enabled me to complete such exploration work.

I would especially like to thank for an amazing support that my supervisors Assoc. Prof Shibli Nisar, PhD and Asst. Prof Abdul Wakeel, PhD have been, throughout the course of my thesis. They helped me a lot to able me in completing my MS Degree, they were all time readily available for guidance and to help me throughout my thesis.

Lastly, I would like to show gratitude to my family for all their care, love and support through my times of stress and excitement.

## Dedication

*I dedicate my work to,*

*my parents and my husband who always encouraged my higher education, for their prayers, love and motivation and sacrifices all along.*

*my supervisors for their support and patience during all the phases of my MS.*

# Table of Contents

## List of Figures

# List of Acronyms

| | |
|---|---|
| Pashto | ps |
| Automatic Speech Recognition | ASR |
| National Language Processing | NLP |
| Hidden Morkov Model | HMM |
| Deep Learning | DL |
| Neural Machine Translation | NMT |
| Deep Neural Network | DNN |
| Word Error Rate | WER |

*Chapter 1*

# INTRODUCTION

Speech recognition in Artificial Intelligence (AI) is the method used to translate spoken language into written format. The technology processes audio data and transforms it into words that may be used in various enterprises using machine learning and neural networks. The task of automatically identifying and transcribing spoken utterances into text is known as Automatic Speech Recognition (ASR). For many programs that don't operate on computers, which are becoming more and more widespread and famous, speech is a natural interface for them. With the aid of speech recognition technology, computers can take spoken audio, analyze it, and produce text from given input audio signal, Fig 1 and Fig 2

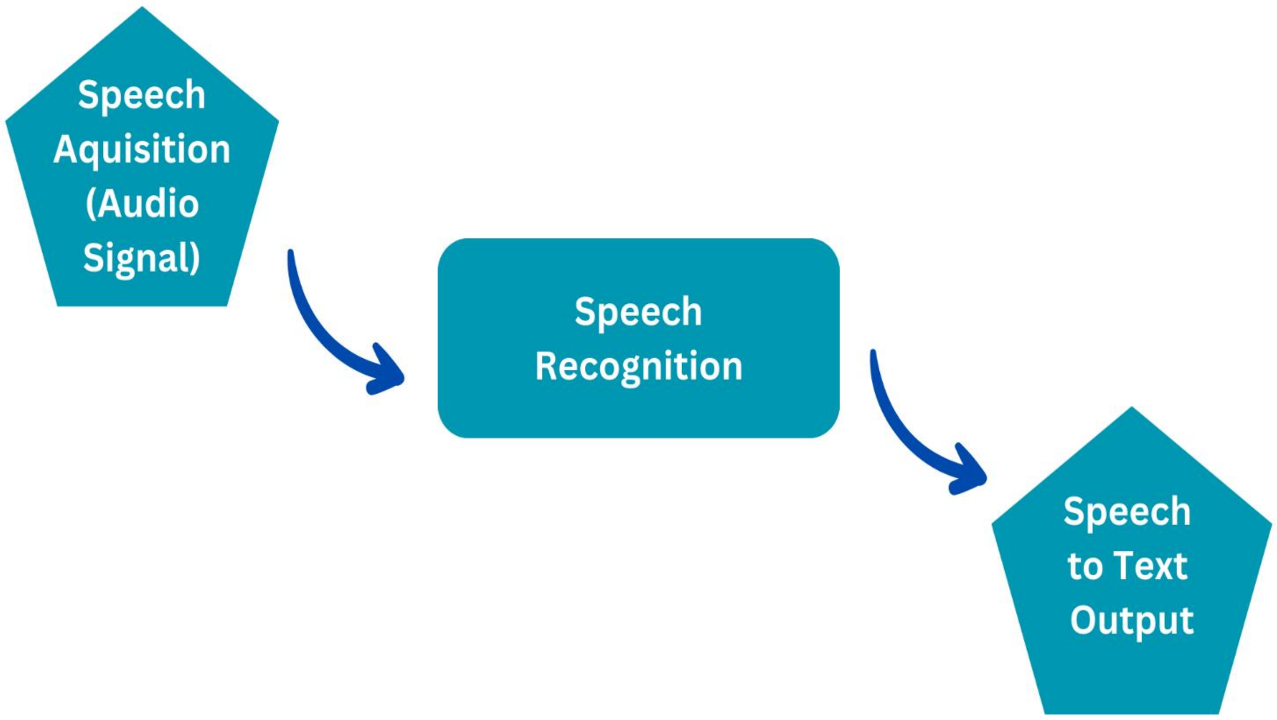


**Figure 1: General ASR System Block diagram**

Processing, interpreting, and understanding a speech signal has evolved into the key to many potent new technologies and techniques of communication, whether we are talking to robots, controlling digital devices, guiding the visually and audibly impaired or disable persons, or enabling hands-free technology. ASR plays a huge role in such advanced technologies.
Applications that support speech have dramatically increased in popularity in recent years. This is because it can be easily integrated with a variety of domain applications and voice over control systems. However, producing efficient ASR systems typically involves vast amounts of speech-to-text transcriptions in addition to sizable text corpora. languages with ample resources, such as English and Chinese, are generally free of this issue, where successful ASR implementations have been developed with excellent results [1].

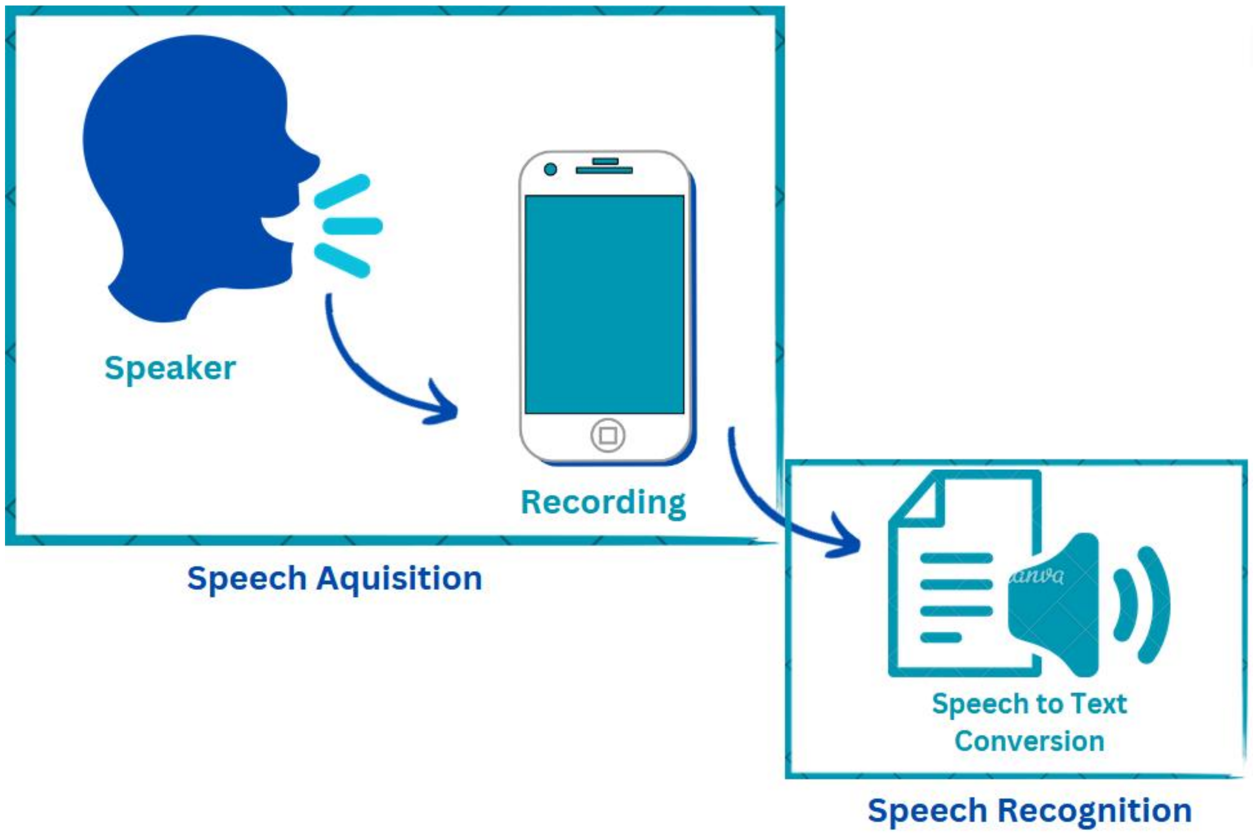


**Figure 2: Overview General ASR System**

Few studies are being done on some low-resource languages, such as regional Indian languages

[2]. Due to the accessibility of voice data in these languages, new trends and techniques for speech recognition tasks, including word2vec, deep learning end-to-end models, and traditional Hidden Markov model (HMM), have been tested [3]. However, only a few major languages are now supported by speech recognition technology. But regardless of voice data low resource languages are still far behind in ASR field.

## 1.1 ASR for Low Resource Language

ASR systems for low resource languages are available for a very few to none languages. The Pashto (ps) language also belongs to the very same category but it is supported by the Google Translation API's recognition engine for the Neural Machine Translation (NMT) model, however the Google Cloud Speech-to-Text domain does not have API support [4,5]. One can legitimately link Mozilla Common Voice to the most recent trend in creating ASR corpuses. Although Pashto has a population of around 50 million people worldwide [6, 15] who speak it in a variety of dialects, one could not locate it among the published languages, which is ironic given Common Voice's wonderful mission.

This language is widely used in various regions of Asia (Afghanistan and Pakistan etc.)[7]. Pashto is a low resource language, despite having a large population of speakers. Research in language technologies like audio and natural language processing (NLP) is lacking or limited for Pashto. The Pashto automatic speech recognition systems that are now in use are either nonexistent or have only been trained on digit-only data. Raw text from different genres (types of documents), such as books, scientific or academic papers, emails, social media posts, etc., is insufficient. There are no dictionaries, dependency tree datasets, semantic databases (like WordNet), or other lexical, syntactic, or semantic resources for this language. Despite having a significant population, the Pushto language group has no such systems in place. As a result, the advancement in the application of ASR couldn't be fully utilized.

There are very few financed projects that produce Pashto text and speech data. There isn't much work accessible to examine errors and increase accuracy in the existing limited work, which has a high word error rate. Thus, it is necessary to create a system of data preserving and data repositories for the language in order to address these problems. The objective of this research is to develop an automatic speech recognition system and first transcribed audio corpus for the Pashto language using latest ASR trends and technology.

Therefore, to achieve significant results for a robust Pashto ASR system, in this paper we worked on wav2vec 2.0 model which was developed in recent years and new to this industry. Its accuracy rate is said to be much more precise than any traditional ASR system. Thus, its implementation will open the door for low resource languages for advanced ASR systems and improvement in the related research areas.

## 1.2Problem Statements

Speech based Hands Free Technology has become the key to many powerful new technologies but its limited use in Pushto language has minimized its exploiting on large scale. To exploit the full potential of speech recognition technology, a robust ASR model should be developed to create transcribed dataset along more accuracy of speech recognition as compared to traditional ASR systems for the low resource Pashto language.

## 1.3 Objectives

Following are the objectives of this research.

- To create an Automatic Speech Recognition (ASR) system for low resource languages.
- To create a transcribe dataset for Pashto language in the backdrop of word2vec and E2E deep learning models of speech technology.
- Implementation of wav2vec model

## 1.4 Area of Application

The model can be applied to:

- Pashto-speaking virtual assistants.
- Voice-activated online banking.
- Voice bio-metric and identification systems.
- Online trading and E-commerce.
- User-friendly public transportation.

- Subtitling the podcasts.
- Applications in healthcare.
- Home networks and appliances.

## 1.5 Relevance to National needs:

There is a need to develop advance ASR and compiled Pashto vocabulary. Such effort has multipurpose implications in the areas of health, industries, and law enforcement sectors. The huge fraction of Pashto language population needs to exploit the boom of speech digitization. To preserve the language digitization, make use of computer/smart devices easy and provide means to practice latest development in ASR creation of speech corpus for Pashto language is a primary task. Research in this area will not only help researchers to create large corpuses but also open the doors for native speakers to step in era of advance technology and ASR systems.

## 1.6 Organization of thesis

The thesis chapters are organized in seven chapters. Chapter 1 is brief introduction to ASR Systems and need for more work in this field particularly low resource languages. Chapter 2 covers the summary of related work including literature review of some existing techniques. Chapter 3 describes the traditional methodology used for ASR systems and their drawbacks and loop holes are also briefly discussed in this chapter. Chapter 4 presents the modern ASR techniques. Chapter 5 covers the proposed system model. Chapter 6 discuss the simulation parameters, new findings and results. Chapter 7 presents the conclusion and future work in the field of advance ASR and Speech technology.

*Chapter 2*

# RELATED WORK

## 2.1 Literature Review

Speech recognition has progressed from recognizing a single word or just a couple of syllables to an entire language. Certainly, a lot has changed since the turn of the 20th century. As shown in Table 1, since the beginning of 1900s researchers already paved their way towards the modern speech recognition. In 1922, Radio Rex, the first toy to employ speech recognition, was developed. Basically, the initial speech recognition technologies prioritized numbers above words. IBM unveiled "Shoebox", a voice arithmetic device which could comprehend and reply to 16 English words.

**Table 1: Overview Speech Technology Trends**

| Year | System | Type | Size |
|---|---|---|---|
| **1922** | Radio Rex | Frequency Detector | 1 x word |
| **1962** | Shoe Box | Isolated Word recognition system | 16 x word |
| **1971-1976** | HARPY (CMU) | Connected speech | 1000 x word |
| **1980-1990** | HMM (Hidden Markov Model) | Statistical based on Markov chain | 1K x word |
| **1992-2002** | LVCSR (Large Vocabulary Continuous Recognition) | LVCSR based | 1K x word |

| | | | |
|---|---|---|---|
| **2006-2022** | Deep Learning | DNN based | 1Mn x word |

Later in mid 1900s, Harpy speech recognition was introduced which is based on finite state beam search system. Speech recognition vocabularies increased from a few hundred to several thousand words in the 1980s. One of the innovations was a statistical technique called "Hidden Markov Model (HMM)". Instead, relying just on words and searching for sound patterns, the HMM calculated the likelihood of the unknown sounds actually being words [9,12,17,32,35,36]. HMM is considered as an important and main traditional method for calculations and predictions in most of the speech recognition systems. Next Chapter 4 is totally based on HMM, so here we are skipping its details.

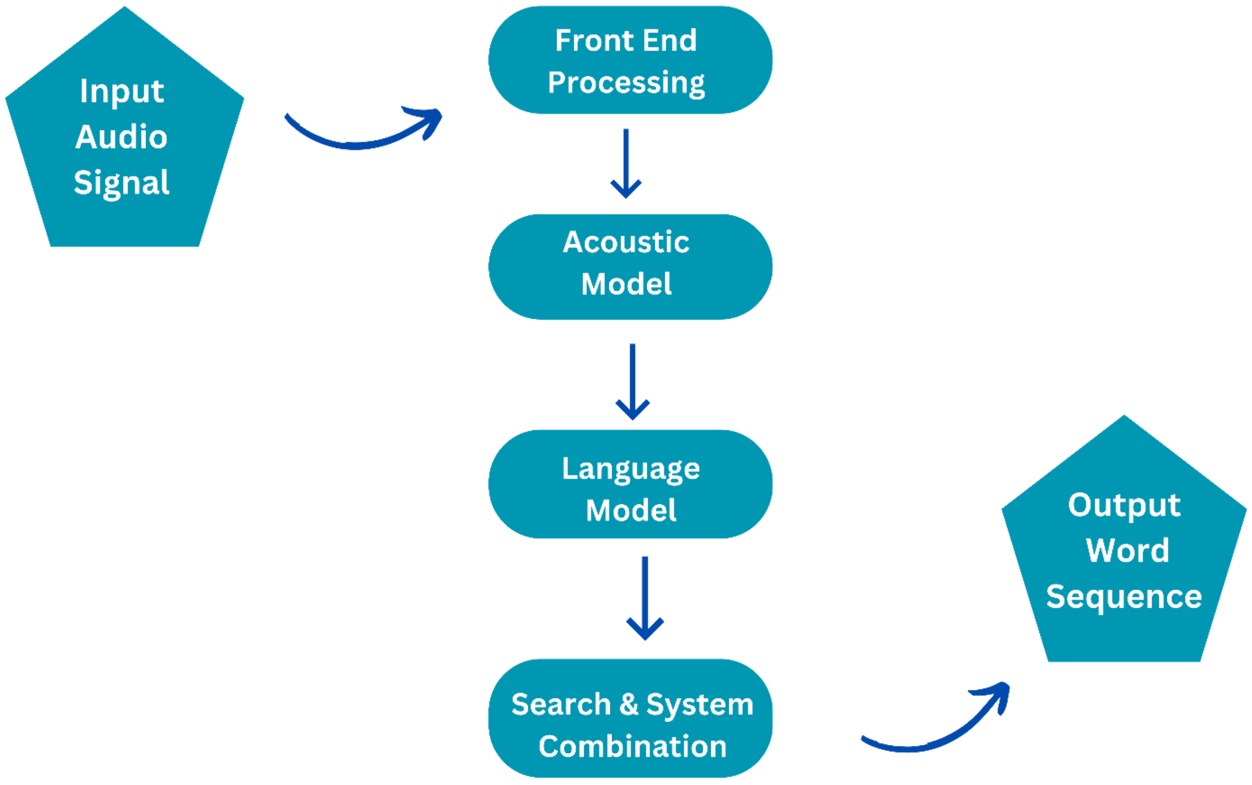


**Figure 3: General LVCSR Architect**

Researchers then shifted their interest towards HMM based Very Large Continuous Speech Recognition (LVCSR) systems [35], Fig 3 shows the general architect of LVCSR system. This system process the input audio signal by utilizing a huge language vocabolary unit to recognize the words in the input audio signal [36].

Given the extensive vocabulary size which is up to almost 100,000 words and phrases, it is impossible to evaluate every word combination directly, and the other issue in the system includes identification of the most likely word sequence from input signal as the word boundary information is not available in continuous speech, thus more computation difficulties are observed in this system.

The era of 2000s shifted the research path towards Machine Learning (ML), Deep Learning (DL) and Artificial Intelligence (AI).A variety of architectures that can create solutions forh several problems are implemented to represent Deep Learning. Speech Recognition using DL algorithems are now much efficient as compared to other ASR techniques as Deep learning approaches can be adjusted for transcribing in noisy situations as well as customized languages, accents, and dialects. Fig 4 represents general architect of Deep Learning.

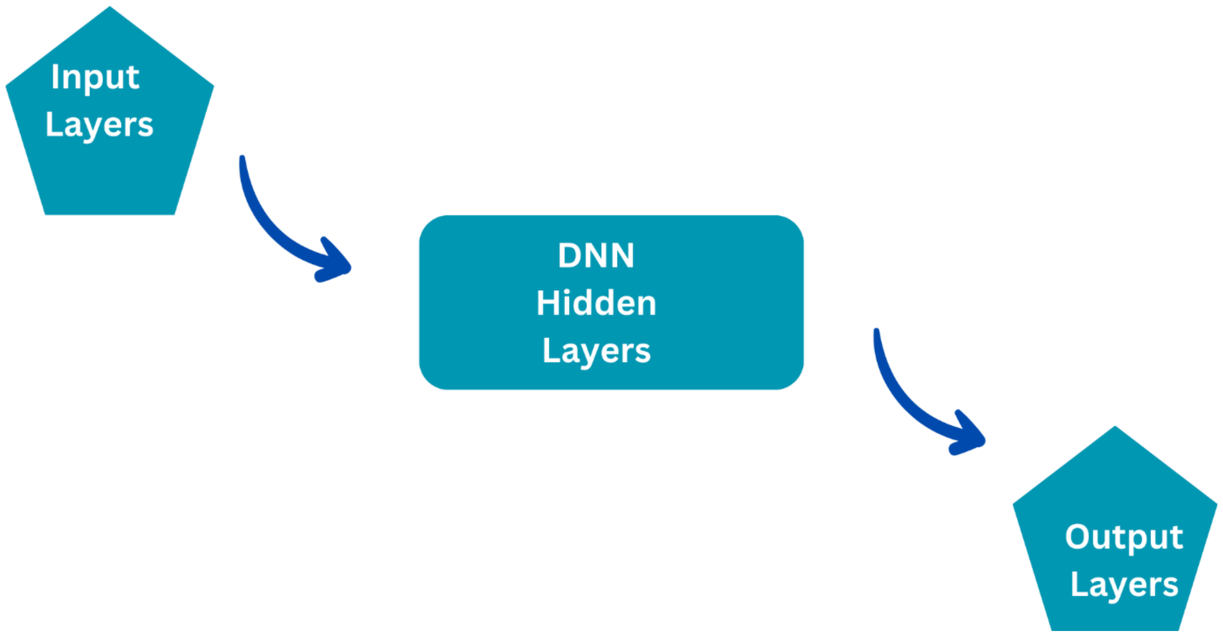


**Figure 4: General Architect for Deep Learning**

## 2.2 Work on various local languages

Due to the lack of resources like transcribed datasets, lexicons, or phonetic dictionaries, developing an ASR system for local languages is rather a difficult undertaking. Local languages including Punjabi [8,9,10], Gujrati [11], Urdu [12,13,14,15,16], Hindi [17,18,19,20], Marathi [21], Arabic [22,23], and Bengali [24] have had some recent work done on them, Fig 5. To the best of our knowledge very little or no work has been done for the Pashto speech recognition system including dataset [25].

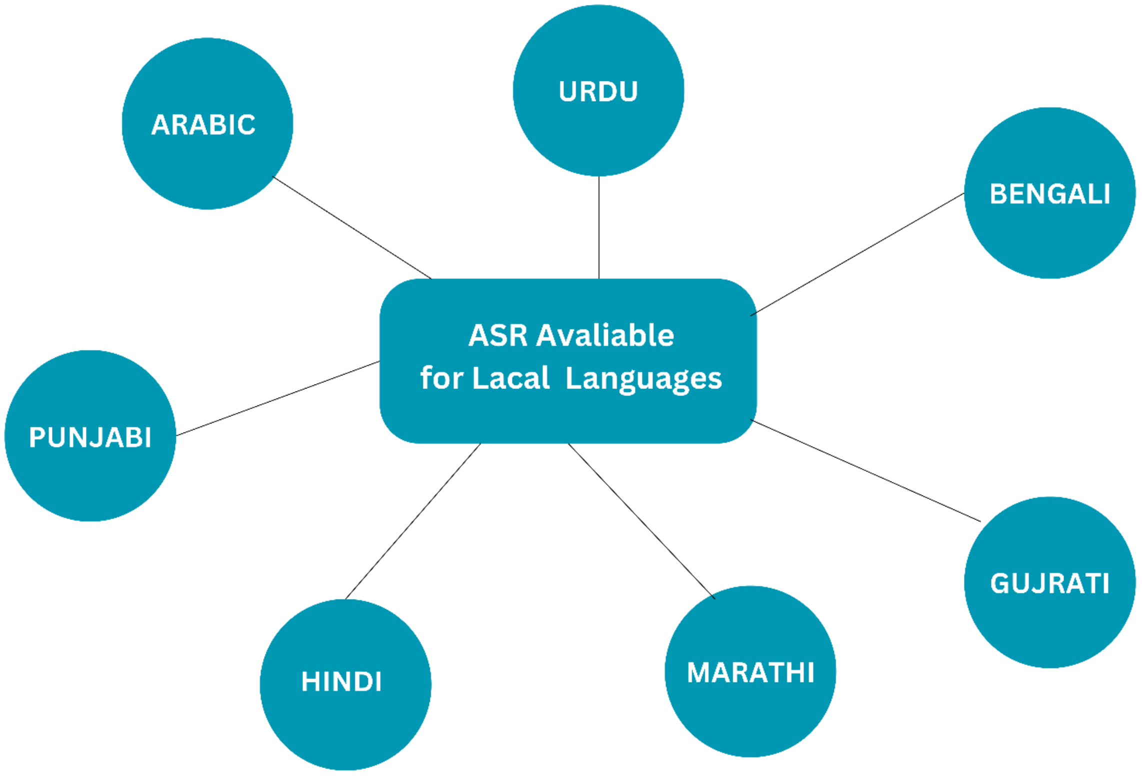


**Figure 5: Available ASR For Various Local Languages**

## 2.3 Work on low resource Pashto language

The establishment of an automatic voice recognition system for Pashto was done using a small Pashto vocabulary speech corpus [26]. Also, in another research a Pashto Spoken Digits database for automatic speech recognition was developed and Sony PCM-M 10 linear recorder was used to make the recordings, which were done in a silent setting, Fig 6. Mel Frequency Cepstral Coefficients (MFCC) were employed as a feature vector, and a classifier based on linear discriminant analysis (LDA) was used to categorize the data [27].

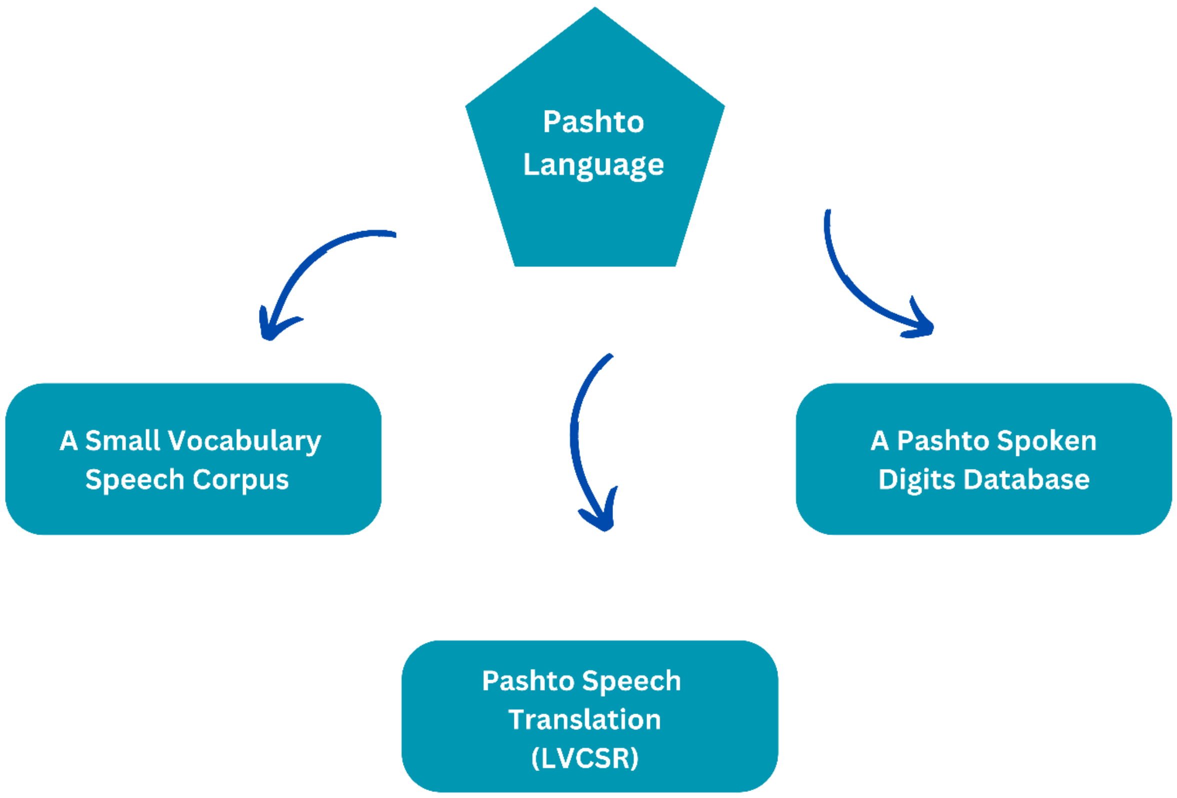

Pashto Language
A Small Vocabulary Speech Corpus
A Pashto Spoken Digits Database
Pashto Speech Translation (LVCSR)

**Figure 6: Work Done For Pashto ASR**

Another study showed the development of database and ASR of isolated Pashto spoken digits from Sefer (0) to Naha (9). For recording purposes, a Sony PCM-M 10 linear recorder was used, and Mel Frequency Cepstral Coefficients (MFCC) was employed to extract speech features. To the best of the author's knowledge, the K Nearest Neighbor (K-NN) classifier was used for the first time in Pashto to categorize speech features. The evaluation of the experimental results yields an overall average recognition accuracy of 76.8% [28].

The SCALE (Summer Camp for Applied Language Exploration) 2015 workshop on "Speech-to-text-translation for low-resource languages" summarized the progress of the LVCSR system as a component of the Pashto speech-translation system [29]. Without requiring manual pronunciations for every word in the training data, the researchers have attempted to examine two ways for bridging the performance gap between the grapheme and the phonetic approaches.

The first method relies on memorizing letter-to-sound rules from a limited number of Pashto manual pronunciations, whereas the second method relies on a hybrid phoneme/grapheme representation for recognition. Researchers demonstrate that both strategies outperform a comprehensive phonetic system using experimental results on spoken colloquial Pashto while only requiring manual pronunciations for a small subset of the words in the acoustic training data [30].

In the recent years, research work for ASR is shifted towards more advance methodologies like combination of HMM and MFCC, various AI algorithms. Table 3 shows some of the modern research stream for ASR systems.

**Table 3: Prominent Related Work**

| **S No** | **Research Work** | **Main Technology** | **Year** |
|---|---|---|---|
| **1** | Unsupervised automatic speech recognition: A review [33]. | MFCC, HMM, GMM | 2022 |
| **2** | Self-supervised end to end ASR, for low resource L2 Swedish [34]. | wav2vec | 2021 |
| **3** | Development of a large vocabulary for Pashto language automatic speech recognition system [35]. | HMM, MFCC | 2022 |
| **4** | Development of a large vocabulary Urdu automatic speech recognition system and error analysis methodology [36]. | HMM, MFCC | 2020 |
| **5** | Wav2vec 2.0: A framework for self-supervised learning of speech representations [37]. | wav2vec | 2020 |
| **6** | Pashto speech recognition with limited pronunciation lexicon [15]. | HMM | 2010 |

| | | | |
|---|---|---|---|
| **7** | Deep speech 2: End-to-end speech recognition in English and Mandarin. [1]. | HMM, DNN | 2016 |
| **8** | An ASR System for Spontaneous Urdu Speech [13] | HMM | 2010 |

## 2.4 Limitations Of Existing Techniques

For low resource languages existing techniques show certain limitations which are as follows:

- Environmental factors.
- Background noise.
- Accents and style of speaking.
- Nonsense errors.
- Spelling errors.
- Multiple detections.
- Homonyms.
- Unavailability of proper dataset for low-resource languages.
- Training Requirement of large corpus.
- Mandatary manual supervision for the creation of linguistic model for identification of specific terminologies and recognition of different accents and dialects.
- Additional acoustic model for detection of repeating patterns.
- Tuning of acoustic unit parameters for the control of noise factors, echoes and quality.
- Pre-requisite manually created phonetic dictionary for the implementation.

## 2.5 Motivation For Proposed Work

Limitations of existing techniques leads us to the proposed model (details in chapter 5) as it claims to surpass almost all the limitations faced during the implementation of existing techniques. Our proposed model (wav2vec2.0) is capable of training on 5 to 7 hours of data (small dataset) and outperforms the existing techniques who needs a proper continuous data set of hundreds of hours.

This AI based model is based on self-learning and do not require any manual supervision or language model for its performance. Also, it creates a dictionary of its own after self-learning. Additionally, it can also accurately transcribe in noisy environment and is capable of understanding the dialects, accents and multiple languages at the same time. Our model claims more accuracy with low word error rate (WER) as compared to the existing techniques. General comparison of proposed model and existing traditional models is given below:

**Table 4: Comparison Proposed Model Vs. Existing Traditional Models**

| **Models Used** | **Transcribed Audio Data Required** | **Lexicon (Phonetic Dictionary)** | **Language Model** | **Computational Power** | **Approach Used** | **Accuracy** |
|---|---|---|---|---|---|---|
| **Our Proposed Model (wav2vec2.0)** | 5 Hours | Not Required | Not Required | Low | Self Supervised | 66% |
| **Existing Traditional Models [35]** | 100 Hours | Required | Required | Very High | Supervised | 64 % |

# TRADITIONAL ASR TECHNIQUES

Automatic Speech Recognition (ASR) converts spoken words into written transcriptions. Traditional techniques used for speech recognition are mostly based on probability and prediction methods. Very popular technique is pipeline system of Hidden Markov Model of Markov chains.

## 3.1HMM based ASR System

As a time-based structure, speech signal can be recorded as a series of spectral vectors that span the audio frequency range. The raw speech signal can be considered as a piecewise stationary signal or a short time stationary signal. And Hidden Markov Model (HMM)not only supports but offers a straightforward foundation for creating such time-based models. General Architect for HMM based ASR is shown in Fig 7.

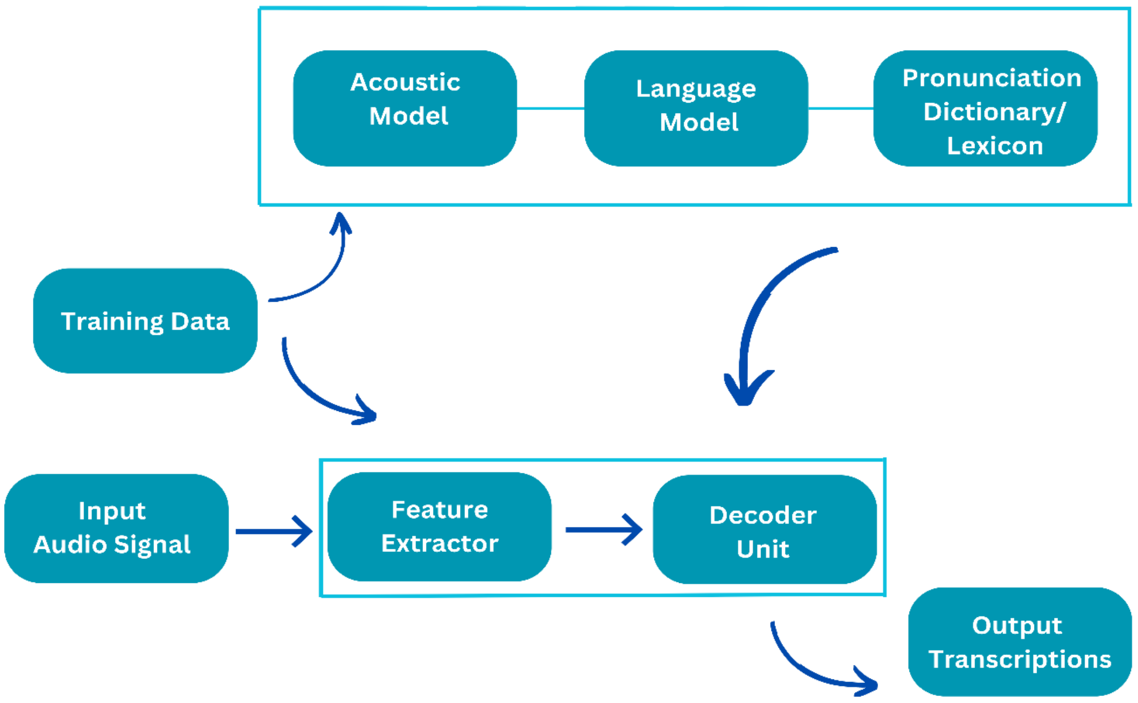


**Figure 7: General HMM Architect**

Raw audio signal is provided to the input end and after probabilistic computations of the system desired text output is achieved. Let’s discuss the working of HMM ASR by its unit-by-unit breakdown.

- **Acoustic Model:**

  This unit is probabilistic in nature as it uses Gaussian Mixture Model (GMM) to find the probability distribution of incoming phones and HMM for calculating transition probability. In simple words, it basically detects the repeating and recurring patterns in the input sound streams. After detection of such repeating patterns, this unit clusters them into small coherent sub units, which are then fed to decoder unit for further processing.

- **Language Model:**

  Language Model is also probabilistic in nature as it computes the probability of sequence of words in the given signal. This model also improves the accuracy of acoustic model by providing a huge linguistic knowledge as reference.

- **Lexical Model:**

  Lexical Model is basically a phonetic dictionary with pronunciations. Extracted and identified phones are overlapped with the data of lexicon with a statistical manner and spoken words are predicted.

- **Feature Extraction:**

  This unit extracts features from the raw audio signal. Extraction process is mostly executed by the combination of different mathematical techniques including Discrete Cosine Transform (DCT), Fast Fourier Transform (FFT) and Mel-Frequency warping, resulting in the discrete features in MFCC (Mel-Frequency Cepstral Coefficients) format.

- **Decoder Unit:**

  This unit receives the output of feature extractor, combined output of Acoustic, Lexical and Language models respectively and further process the data for training and decoding. Researchers use different approaches, algorithms and techniques for training and decoding purpose. Finally transcribed output texts are achieved at the output end after its processing from scoring toolset in the decoder unit.

## 3.2 Drawbacks of HMM based ASR System

There are certain loop holes in traditional ASR systems. Some significant drawbacks are listed below:

- Complex statistical computations.
- They make significant, potentially unfounded assumptions on the independent and stationary nature of the observations and states.
- HMMs are also constrained and insufficient since they can only accurately represent linear and discrete relationships between the observations and the states.
- Evident manual supervision is required by the system in the form of providing huge lexicon dictionaries and data vocabularies.
- Increased word error rate and precision issues.
- Significant time consumption
- Implementation of various cascaded techniques requiring more memory space.
- Less optimized method.
- HMM frequently uses a lot of unstructured parameters.
- Positional data are not taken into account.
- It has No memory; higher orders have some memory but do not explicitly use positional data.
- Handles insertions and deletions poorly.
- Environmental factors.
- Background noise.
- Accents and Style of speaking.
- Nonsense errors.
- Spelling errors.

- Multiple detections.
- Homonyms (Words with same spellings or same pronunciations).
- Unavailability of proper dataset for low-resource languages.
- Training Requirement of large corpus.
- Mandatary manual supervision for the creation of linguistic model.
- Additional acoustic model for detection of repeating patterns.

## 3.3Drawbacks related to the implementation of Speech Recognition System

The following are some of the difficulties in creating and implementing voice recognition pipelines in real-world settings:

- It is challenging for developers to utilize the speech recognition technology at various time due to lack of tools and kits that provide state-of-the-art ASR models.
- There are few customization options that let programmers adjust for jargon particular to a given domain and situation, as well as for numerous languages, dialects, and accents so that their applications understand and sound like them.
- Abundant real time restrictions related to the deployment and installments of these systems.
- Time consumption for system's computation is a headache for real time users of the speech technology.

*Chapter 4*

# MODERN ASR

Modern ASR systems are based on deep learning encoder-decoder architectures that process the input audio via a cascade of convolutional layers in order to produce a compact vector. The decoder then uses the input from the encoded vector to produce a string of characters. Fig 8 shows general architect of modern ASR.

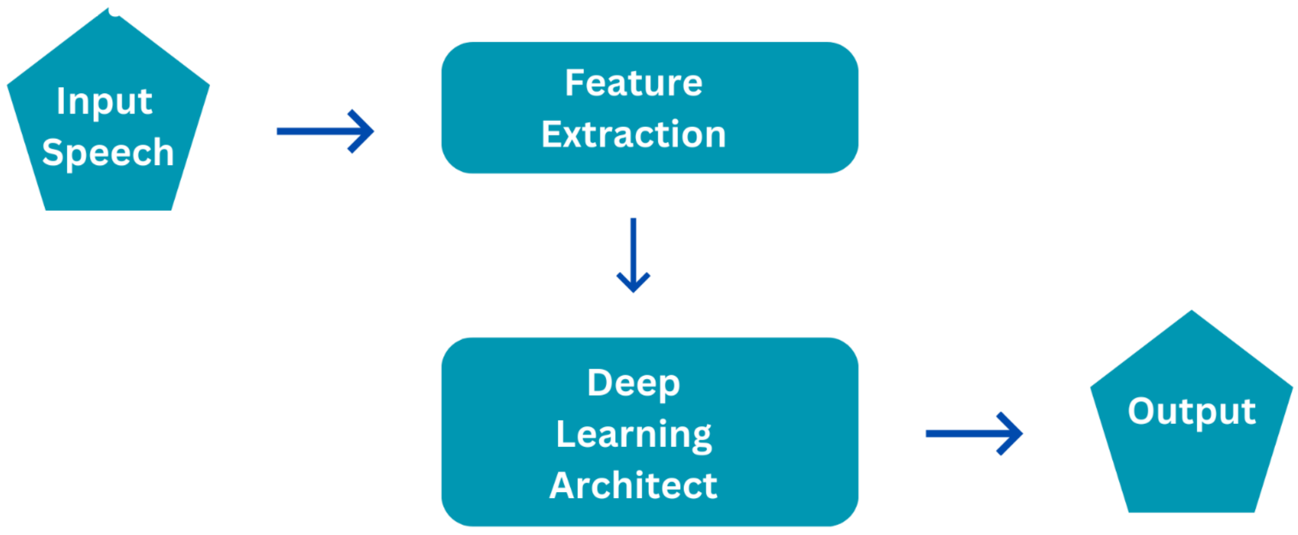


**Figure 8: General Architect of modern ASR.**

## 4.1 Growing Deep Learning

Deep learning techniques are very vast and trendy. Significant benefit of using DL is that the data pre-processing which is generally involved with machine learning is eliminated with deep learning. These algorithms can handle text and visual data that is unstructured and automate feature extraction, reducing the need for manual supervision. We can choose which DL technique are best suited to fulfil our requirements. Table 5 represents a brief overview of various DL techniques.

**Table 5: Growing Deep Learning (DL)**

| **DL Techniques** | **Overview** |
|---|---|
| Supervised Learning [9,12,17] | Unlabeled data => Label => Train |
| Self-Supervise Learning [3] | Isolated Word Recognition System |
| Seq-2-Seq Model<br>RNN based Seq-2-Seq [42] | Convert seqs of Type-A to seqs of Type B. |
| Transformers<br>Transformers XL [41] | Compute representations of input and output without using sequence-aligned RNNs or convolution |
| Google BERT [40] | Bidirectional Encoder Representations from Transformers |
| Wav2vec2.0<br>XLSR wav2vec [37,38] | Convolution+ Quantization + Transformers |

- **Supervised Learning:**

  It is distinguished by the way it trains algorithms to accurately classify data outcomes using labelled datasets. In most cases, supervised machine learning is used to categorize data or generate predictions [9,12,17]. Because labeled or tagged data is required, supervised machine learning uses a lot more resources and manual aid, Fig 9.

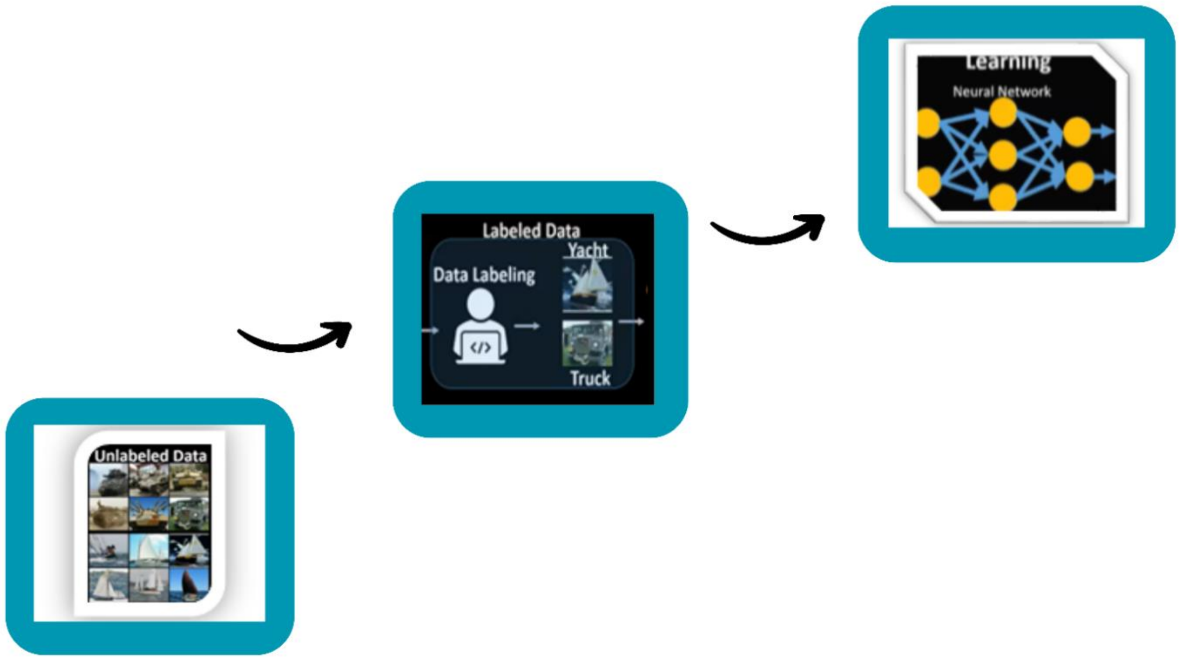


**Figure 9: Supervised Learning**

- **Unsupervised / Self Supervised Learning:**

In the self-supervised learning process, an algorithm trains itself to learn, pre-process and differentiate the input components, also known as Pretext learning or predictive learning. By automatically creating the labels, the unsupervised problem is converted into a supervised problem and thus eliminating the need for manual aid [3], Fig 10. Unsupervised learning is typically used to analyze dataset relationships.

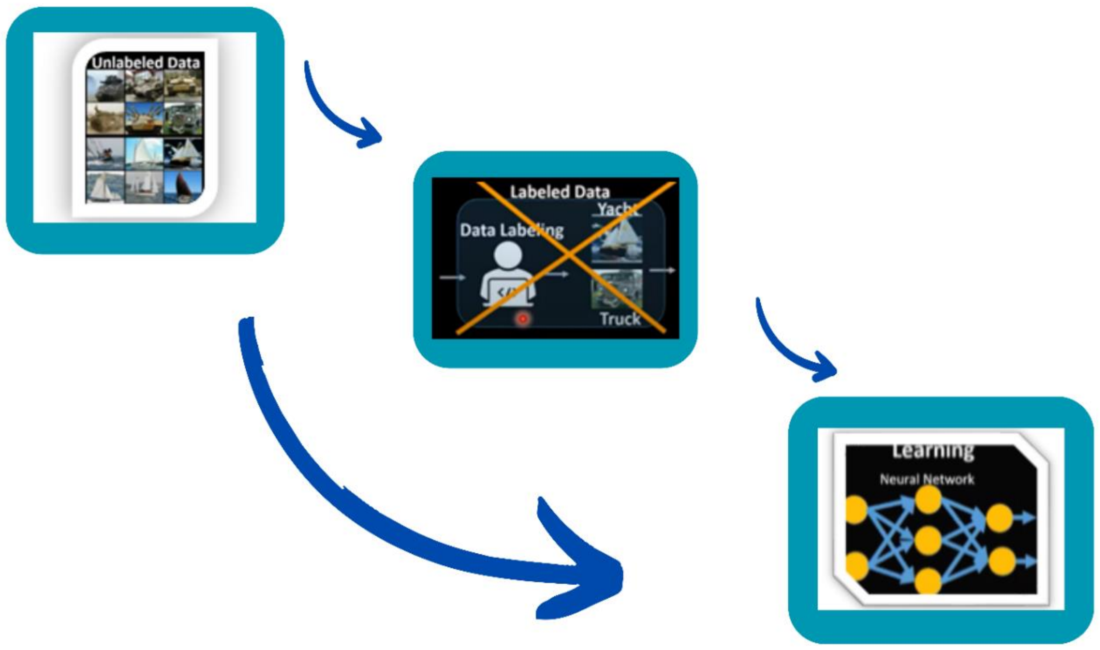


**Figure 10: Unsupervised Learning**

**Table 6: Brief comparison between Supervised and Unsupervised learning**

| **Machine Learning Type / System** | **Input Form** | **Computational Complexity level** | **Output Accuracy** |
|---|---|---|---|
| **Unsupervised Learning [9,12,17]** | Labelled | Higher Computation Complexity | Less Accuracy Rate |
| **Supervised Learning [3]** | Unlabeled | Simpler Computations | Higher Accuracy Rate |

Table 6, highlights the general difference between the working of both machine learning algorithms, supervised and unsupervised learning systems respectively.

- **Seq-2-SeqModel:**

  In National Language Processing (NLP), sequences of Type A are transformed into sequences of Type B using sequence-to-sequence models. For instance, translating sentences from one language to another is a sequence-to-sequence problem. From 2014 on-words, Recurrent Neural Network (RNN) based sequence-to-sequence models have gained a lot of popularity, Fig 11. The majority of data in today's world is in the form of sequences; these sequences can be audio, text, video, or number sequences [42].

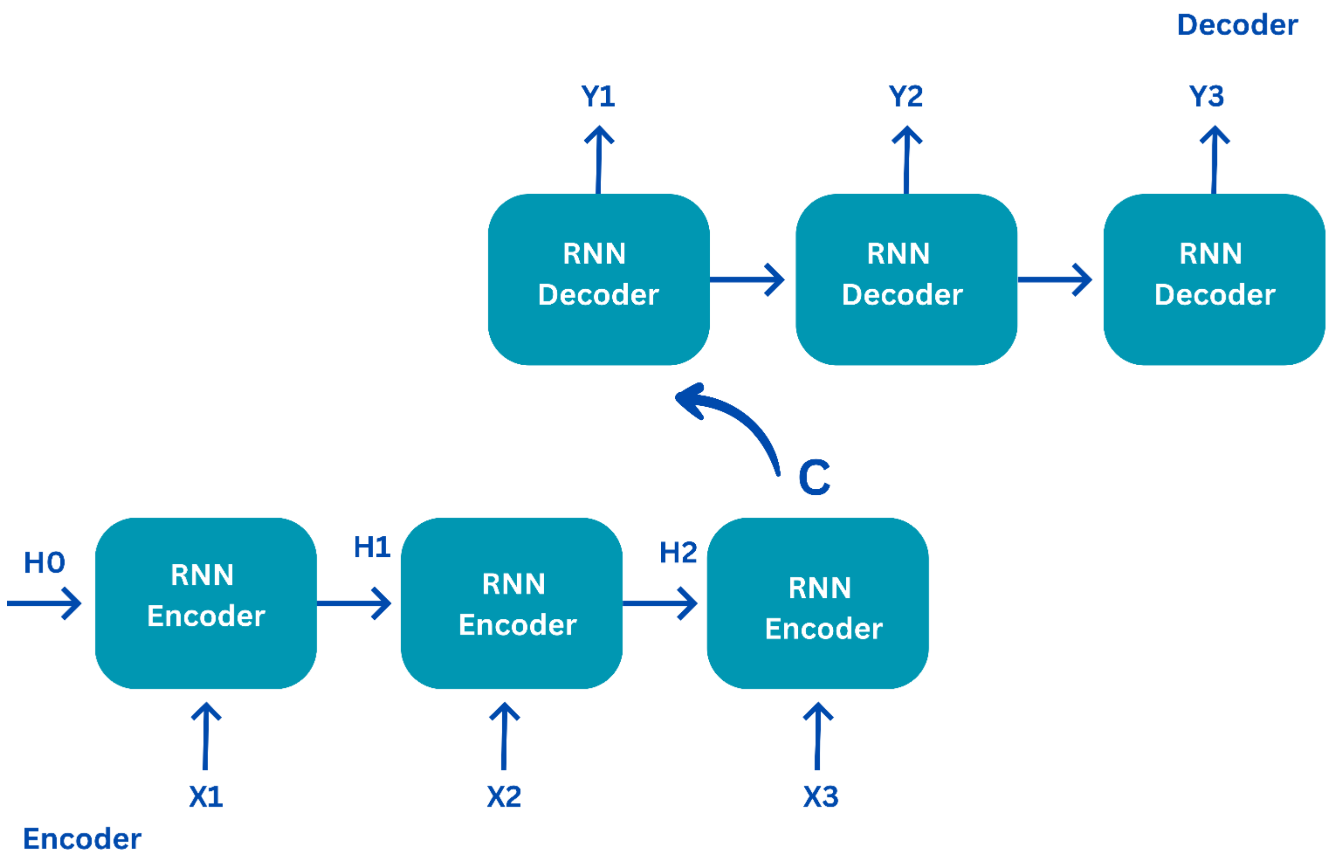


**Figure 11: Seq-2-SeqModel**

- **Transformers:**

  An innovative DL model called The Transformer in NLP tries to tackle sequence-to-sequence problems while skillfully managing long-range dependencies. Transforming input sequences into output sequences is referred to as "transduction" in this context. The goal of Transformer is to handle input and output relationships with complete attention and recurrence [41].

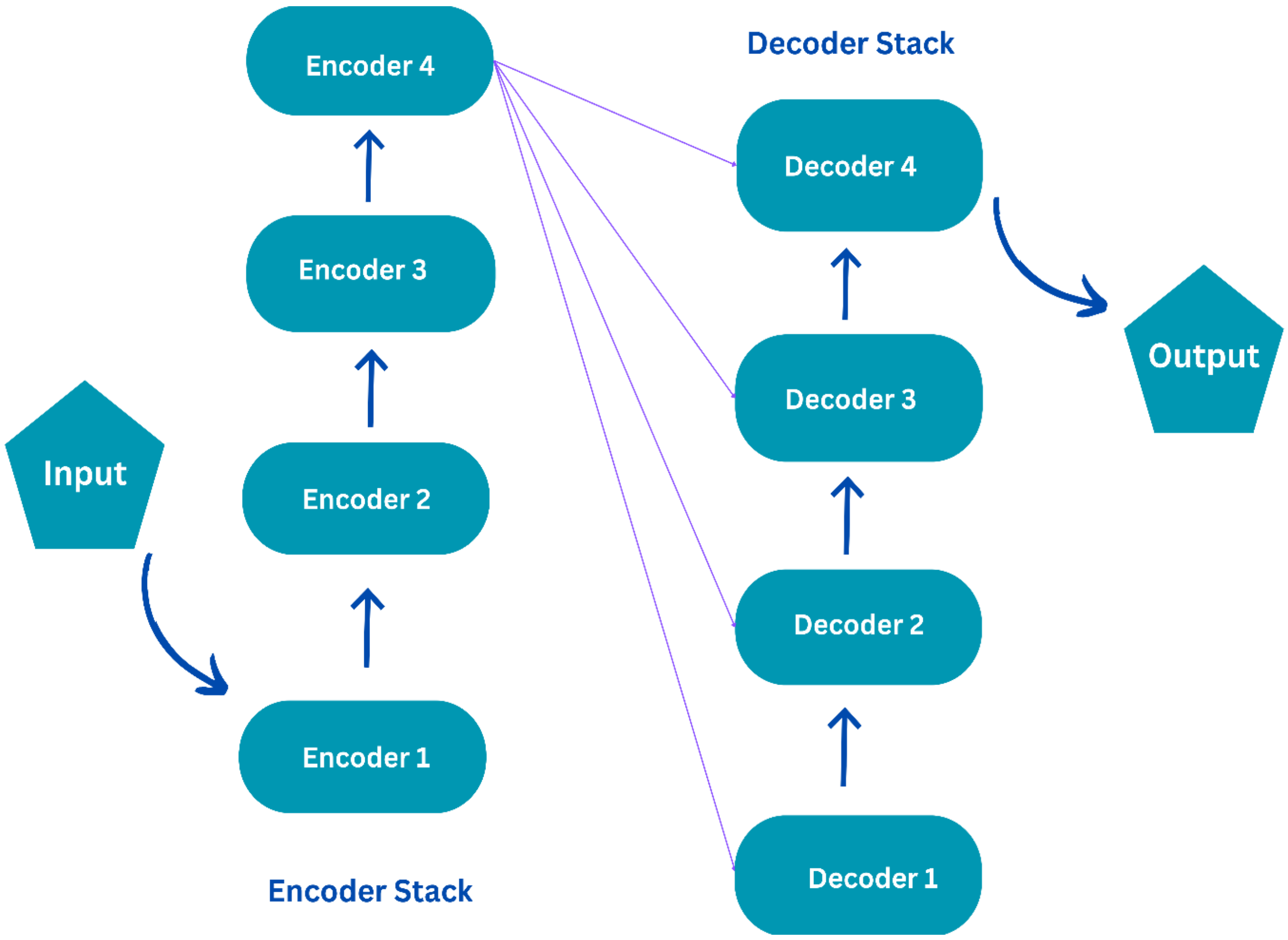


**Figure 12: General Architecture of Transformer**

Several identical encoders and decoders are layered on top of one another to form the encoder and decoder blocks and both blocks have equal number of stack units. The first encoder receives the input sequence's word embedding. These are then modified and transmitted to the following encoder. According to the diagram below, Fig 12, every decoder in the decoder-stack receives the output from the final encoder in the encoder stack. The Transformer is the first transduction model to calculate representations of its input and output purely utilizing self-attention without the use of convolution or sequence-aligned RNN.

Only text strings of a certain length can be handled by attention. Before being entered as input into the system, the text must be divided into a certain number of segments or chunks. Context is messed up by this text chunking. For instance, if a sentence is broken down from the middle, a lot of contexts is lost.

Therefore, the text is divided without regard for the phrase or any other semantic boundaries. Transformer-XL offers its remedy. The hidden state computed for the prior state as an additional context for the current segment during the training phase of Transformer-XL. Transformer-XL's recurrence technique handles the drawbacks of using a fixed-length context.

- **Transfer Learning**

Transfer learning (TL) is a machine learning (ML) research subject that focuses on using information learned while completing one job to complete another that is associated. For instance, the skills acquired while learning to identify simpler type of transport vehicles could be used when attempting to identify heavier ones. Fig 13 shows the general architect of Transfer Learning.

In Transfer learning, pre-trained models are frequently used as the foundation for deep learning tasks in computer vision and Natural Language Processing (NLP) because they save both time and money compared to developing neural network models from scratch and because they perform vastly better on related tasks. Fig 14 shows the strategy flow diagram of transfer learning (TL) [43].

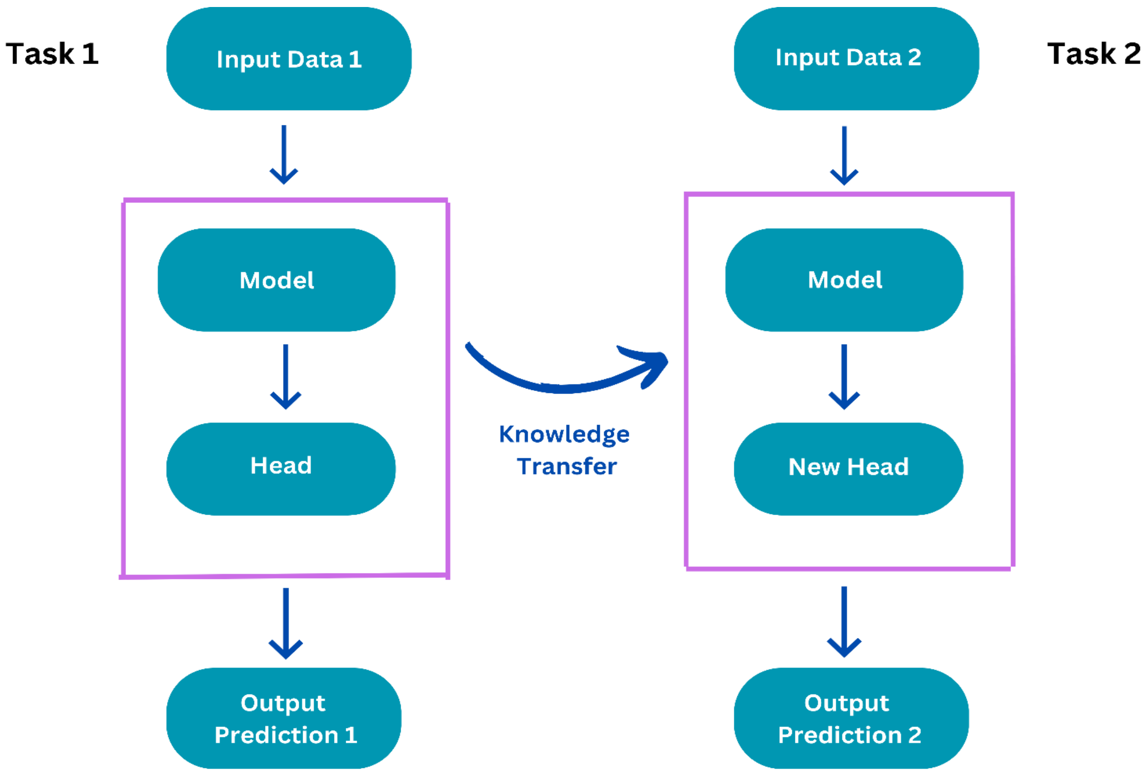


**Figure 13: General Architect of Transfer Learning (TL)**

.

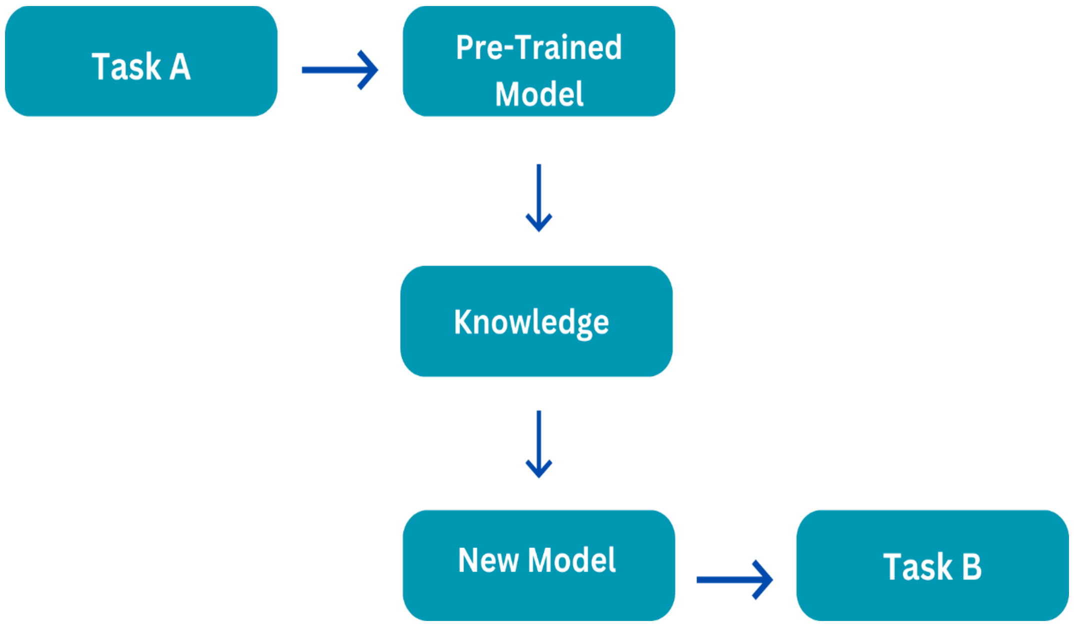


**Figure 14: Transfer Learning Strategy**

# PROPOSED METHODOLOGY

# WAV2VEC2.0/ XLSR MODEL

## 5.1 Wav2vec2.0

Wav2vec is a pretrained automatic speech recognition (ASR) model, a relatively novel idea in this industry that Facebook team Alexei Baevski, Michael Auli, and Alex Conneaut published in September 2020. Facebook AI showed Wav2Vec2 using the English ASR dataset LibriSpeech. Itis a uniqueness is reflected by its self-supervised training nature. Fig 15 shows general block diagram of the model.

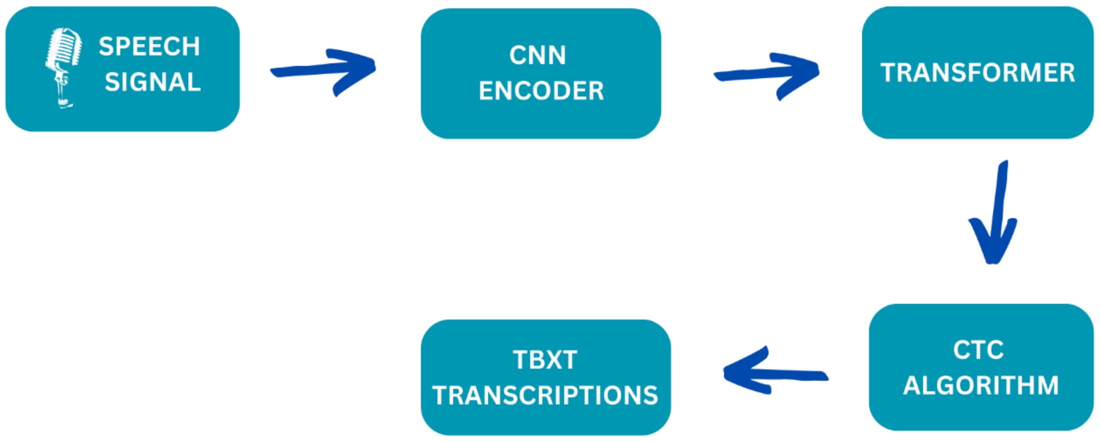


**Figure 15: General block diagram of wav2vec2 Model**

Pre-training a model using unlabeled data, which is always more accessible, is now possible with this method of training. This AI essentially trains the model not only to understand the distinction between original speech instances and modified phones, but also frequently performs this task hundreds of times for each second of audio, and forecasts the proper audio advanced in milliseconds. Fig:16 shows the general block diagram working steps of this Facebook AI.

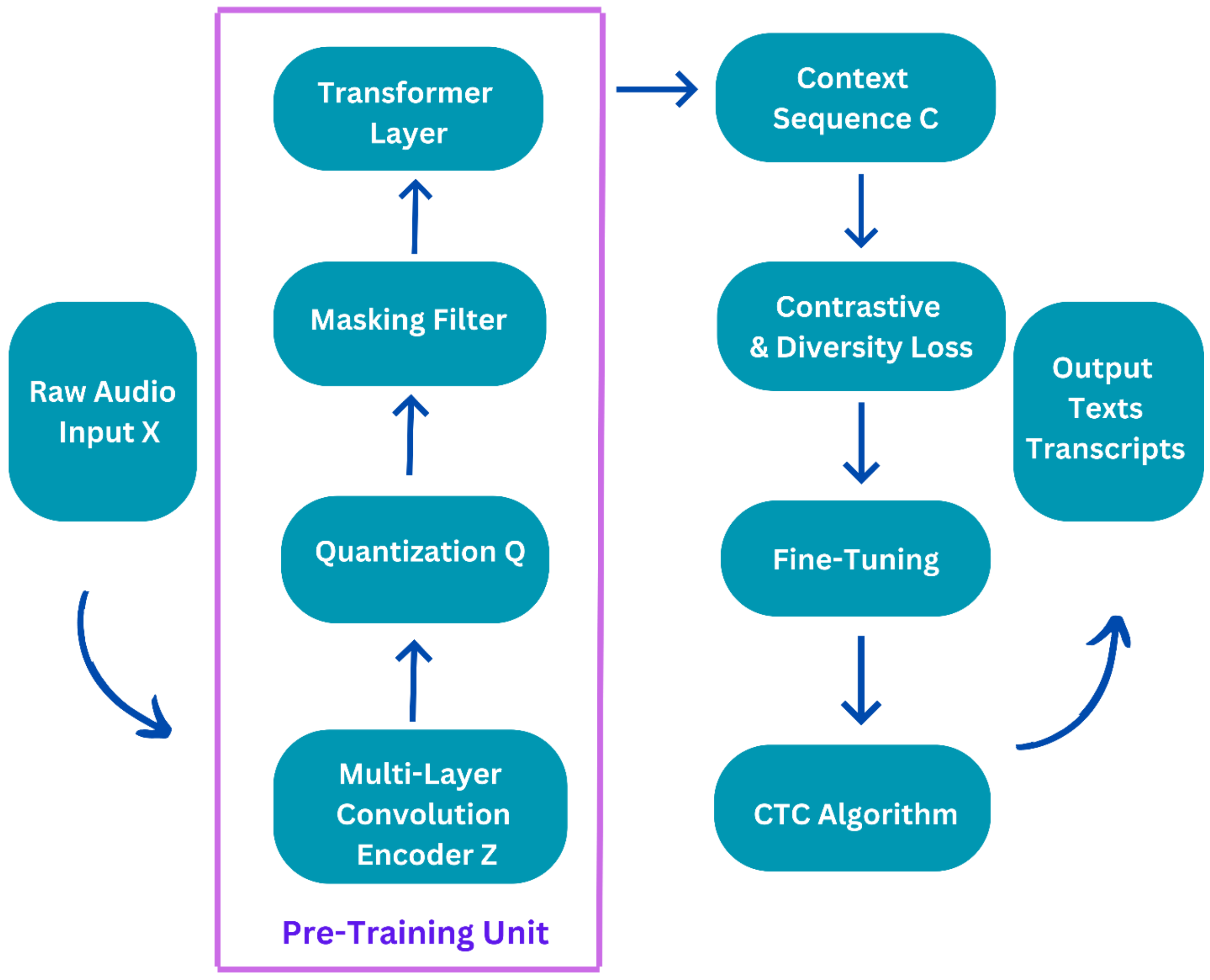


**Figure 16: Working Steps of wav2vec Model**

**Feature Encoder:**

As shown in Fig 17, this algorithm receives the audio input signal X which is fed to a multilayer convolution encoder Z. i.e.- many hidden layers of Convolution Neural Network

(CNN). Components of this encoder includes temporal convolution blocks, Normalization layer and Gaussian Error Linear Unit (GELU) activation function respectively.

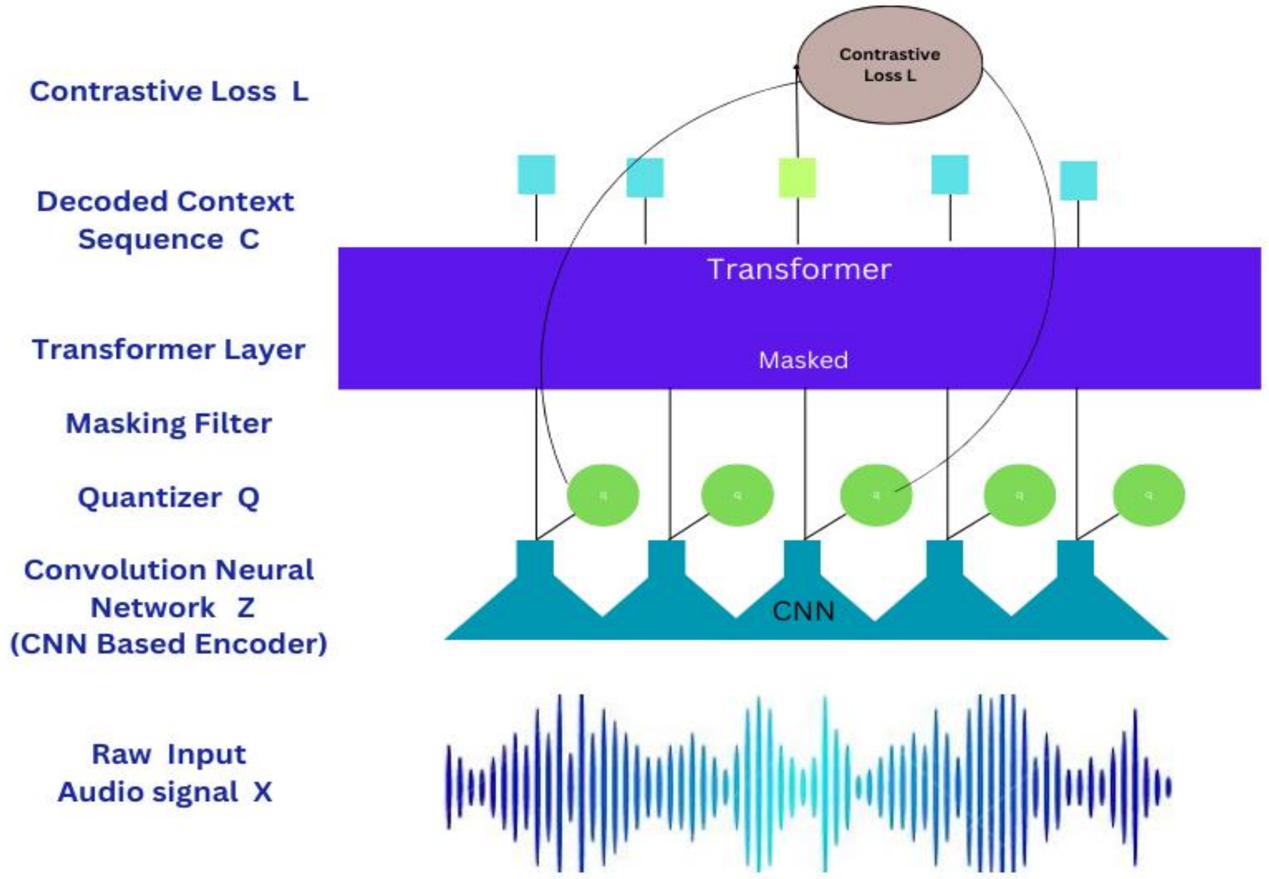


**Figure 17: wav2vec2 Model**

This encoder is essentially a feature extractor that converts the raw input signal X into a sequence of continuous audio signal vectors using a few convolutions, a GELU activation function that assigns input weights based on their probability under a gaussian distribution, and normalization to achieve zero mean and unit variance. Therefore, this encoder yields the Sequence of Latent Speech representations Z for T time steps.

$$Z = Z_1, Z_2, \ldots, Z_T. \quad (1)$$

In simple words the CNN-based Feature Encoder, with its convolutional layers, GELU activation, and normalization, is designed to learn hierarchical representations from the raw audio signals. The convolutional filters capture local patterns, the GELU activation introduces non-linearity, and the normalization helps in normalizing the representations. The resulting small continuous speech vectors can then be fed into subsequent layers or models for further processing or downstream tasks such as quantization, speech recognition or speaker identification.

**Quantization Model:**

Now these continuous audio vectors are digitized via quantization unit Q, yielding quantized audio vectors. To attain these quantized vectors, Apply the quantization model. For its implementation create a bunch of groups / codebooks and create a matrix ***e*** of size RxV**,** representing the input data in the quantization model.

$$e \in R^{V \times d/G} \quad (2)$$

the parameters are defined below:

- ***G*** = Number of groups / codebooks
- ***V*** = Number of entries / members in the group.
- ***d*** = Dimensions
- ***R*** = Total number of samples or instances in the dataset

In order to construct quantized vectors, for every iteration, select an entry / row randomly from every matrix, and then integrate the resulting concatenated vectors $e_1, \ldots, e_G$ as a single vector $R^d$. Now apply linear transformation on vector $R^d$ and obtain f- dimensional matrix vector $R^f$, which will give us the quantized vectors as we have only finite options to choose for each iteration.

$$q \in R^f \quad (3)$$

Model will take the feature encoder output latent speech representations Z and multiplied them with a matrix and turned them into a bunch of logits. In simple words, we are correcting the size of Z by mapping it on **l** logits (i.e. $Z \rightarrow l$).

Therefore, size of **Z** corresponds to the number of choices we have. So, first we will choose the group **G**, then a member / entry **V** in that group to yield the logits, which is given below:

$$l \in R^{G \times V} = \text{logits} \qquad (4)$$

Now the Gumbel SoftMax function will model the logits and convert the vectors into continuous probability distribution and models the randomness or noise factor. Therefore, probability of choosing group **g** and member **v** is given by SoftMax function as:

$$P_{(g,v)} = [\, \exp((\, l_{g,v} + n_v) / \tau\,)] \,/\, [\, \textstyle\sum_{k=1}^{v} \exp\,(\,(\, l_{g,k} + n_k) / \tau\,)] \qquad (5)$$

Where,

- $\tau$ = non-negative temperature parameter
- $n = -\log(-\log(u))$ = Gumbel Noise
- $u$ = uniform samples

So probability of selecting ith option is given by the ratio of exponential of modified logits and the sum of exponential of all the modified logits. Here $\tau$, the non-negative temperature parameter and by increasing its value, distribution gets prominently smoother. In forward route (from encoder to transformer), we choose the concatenated vectors $e_1, \ldots, e_G$ by looking the probabilities obtained using equation (5) the selecting the maximum or argmax to get the choice (codeword *i* ) to make i.e. choose the ith row of the matrix.

$$i = \arg\max_j P_{(g,\, j)} \qquad (6)$$

Now after all the iterations, quantized vectors are obtained.

$$Q = q_1, q_2, \ldots, q_T \qquad (7)$$

Q represents the finite quantized audio vectors of Latent Speech representations for T time steps.

**Masking:**

Now some of these quantized vectors are masked via masking filter and a new sequence of these vectors is fed to the input of transformer layer.

**Transformer Unit:**

Working of Transformer unit is as follows:

- **Positional encodings** are added to the input features using the convolution layer as a positional encoder, in order to incorporate the temporal information of the audio stream. The relative positions of the audio segments are disclosed by these encodings.
- **Transformer Encoder** has a stack of transformer encoder layers fig 12, receives the input features and their positional encodings after that. Multi-head self-attention systems and position-wise completely coupled feed-forward networks make up each encoder layer.
- **Multi Head Self Attention:** Each input feature vector can attend to every other vector in the sequence thanks to the self-attention mechanism, which captures contextual relationships. Different forms of relationships are captured using a variety of attention heads.
- **Feed-forward Networks:** The transformer then independently applies position-wise fully connected feed-forward networks to each feature vector following the self-attention phase. The representations are improved and changed as a result of this process.

- **Output:** The transformer encoder's ultimate output is a series of high-level contextualized representations, each of which contains data on the related input feature vector and its context.

$$C = c_1, c_2, \ldots, c_T \qquad (8)$$

C represents the Context Representation of capturing information from the sequence. Now this completes the pre-training loop. Basically, the multiple CNN layers and transformer constitute the training unit, Fig 18.

**Loss Function and Pre-Training:**

A significant contrastive loss L is observed at transformer output due to masking factors in the previous step. Now to back-propagate (from loss function to quantized vectors) to the derivative of the loss function obtained, we will approximate it with SoftMax and true gradient of the

Gumbel SoftMax outputs will smooth the distribution and choose discrete groups and codebook entries in an absolutely distinguishable manner.

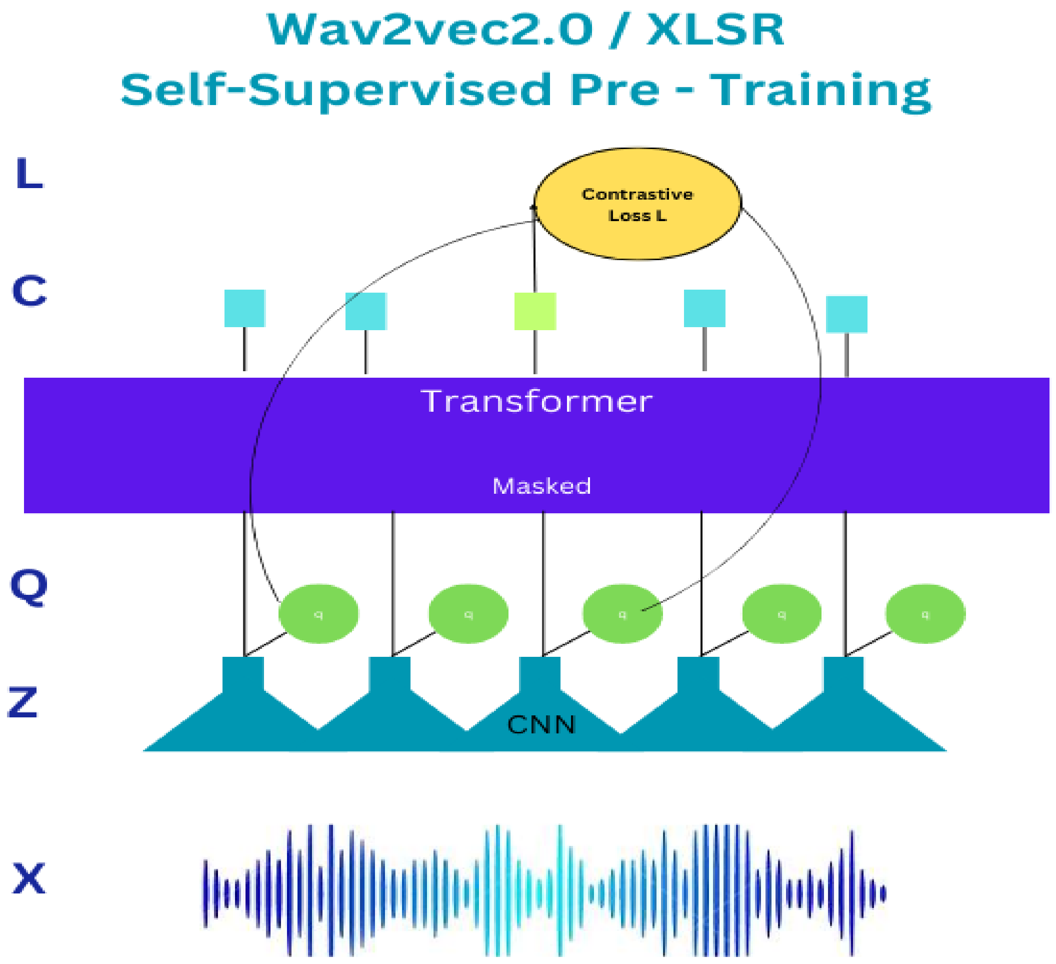


**Figure 18: wav2vec2 Model pre-training**

Pre-training involves completing a contrastive assignment $L_m$ that asks you to choose the real quantized latent speech representation for particular masked given time step from among a variety of distractor signals. To encourage the model to employ the codebook entries equally frequently, a codebook diversity loss $L_d$ is added with a tuned hyperparameter $\boldsymbol{\alpha}$.

$$L = L_m + \alpha L_d \tag{9}$$

where ,

- $L_m$ = Contrastive Loss
- $L_d$ = Diversity Loss
- $\alpha$ = Hyperparameter.

**Contrastive Loss:**

By encouraging the model to give the real representation $\mathbf{q_t}$, a higher similarity score than the distractors $\mathbf{q^{\sim}}$**,** this Contrastive Loss function $L_m$ aims to increase the model's capacity to recognize the correct quantized latent speech representation from the available options.

The loss $L_m$ is calculated using the contrastive loss equation by dividing the negative logarithm of the sum of the exponential similarity scores between the context representation $\mathbf{c_t}$ and the candidate representations $\mathbf{q^{\sim}}$ by **k** scaling factor. The importance of the representations with greater similarity scores is increased using the exponential function. Equation is given by:

$$L_m = -\log [ (\exp (sim( c_t + q_t) / k )) / \Sigma_{(q\sim \sim Qt)} (\exp (sim( c_t + q^{\sim}) / k ))] \tag{10}$$

Where,

- $c_t$ = output of the context network
- $q_t$ = true quantized latent speech representation at the masked time step T.
- $q^{\sim}$ **=** set of candidate representations, which includes the true representation $q_t$ and k distractors.
- ***k* =** scaling / temperature parameter. k parameter regulates how sharp the similarity scores are.The similarity scores are sharper for higher values of k and smoother for lower values.
- ***sim(a,b)*** = The cosine similarity between two vectors a and b is represented by the function sim(a, b). In this instance, it is calculated as the vectors' dot products divided by

their magnitudes. The resulting similarity value ranges from -1 to 1. To assess how similar two vectors are, the cosine similarity is frequently utilized.

**Diversity Loss:**

In order to promote the equitable use of the quantized codebook representations in each of the G codebooks, the diversity loss $L_d$ was introduced. This is implemented by increasing the entropy of the averaged SoftMax distribution (l) over the entries for each codebook $\bar{P}_g$ (This distribution indicates the probability of selecting each entry in the codebook*)* over a batch of utterances. The diversity loss $L_d$ is calculated as the negative sum of the entropy H, as given below:

$$L_d = \frac{1}{GV}\sum_{g=1}^{G} - H(\bar{P}_g) = \frac{1}{GV}\sum_{g=1}^{G}\sum_{v=1}^{GV}\bar{P}_{g,v}\log\bar{P}_{g,v} \qquad (11)$$

Where,

- $G$ = Number of groups / codebooks
- $V$ = Number of entries / members in the group.
- $H(\bar{P}_g)$ = It illustrates the averaged SoftMax distribution's entropy. The uncertainty or randomness in a probability distribution is measured by entropy. Entropy maximization facilitates a more uniform distribution, which promotes the equitable utilization of codebook entries.
- $\bar{P}_g$ = It depicts the gth codebook's averaged SoftMax distribution over the entries. The likelihood of choosing each entry in the codebook is represented by this distribution.

This loss encourages the model to use every entry in each codebook equally by maximizing entropy, encouraging variation in the choice of quantized representations. A balanced use of the quantized codebook is supported by this loss in order to prevent the model from favoring particular codebook entries or representations, increasing the diversity of the quantized representations and enhancing the overall performance of the quantization model.

**Fine-Tuning:**

The previously trained network is topped with a single linear layer for fine-tuning, which involves the training of the model on labelled audio data for subsequent tasks including voice recognition, speech interpretation, and audio categorization's exhibits substantial advancements above prior state-of-the-art outcomes in speaker/language identification, voice translation, and speech recognition. Following the adjustments, the model effectively recognizes, categorizes, and translates the desired input, Fig 19.

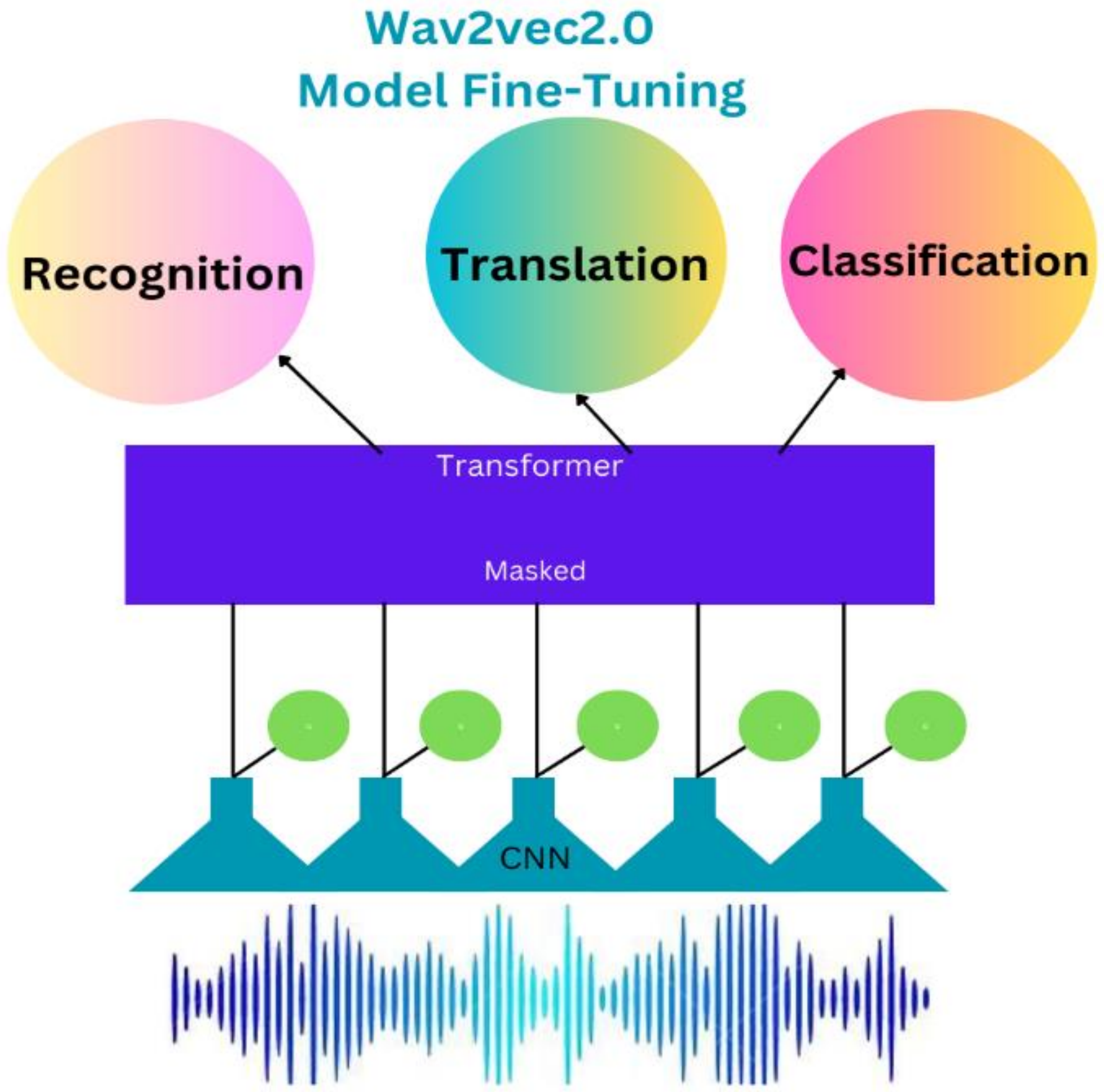


**Figure 19: wav2vec2 Model Fine-Tuning**

**Decoding:**

Once the transformer yields the output sequence, model starts to predicts the masked vectors. Now this process results in the output in chunks form. These letter chunks are then classified by Contrastive Task layer into respective output tokens. In the final step, these tokens are combined with Connectionist Temporal Classification (CTC) algorithm to constitute the final text transcriptions.

Generally, in CTC algorithm, the input consists of a series of observations, while the outputs may also include blank outputs along the series of labels. Because there are far more observations than labels, training is a challenging task. For instance, different time slices can correspond to a particular phoneme, can exist in spoken sounds. CTC forecast a probability distribution for each time step since there is uncertainty about the observed sequence's alignment with the goal labels. The continuous output of a CTC network, is adjusted during training to represent the likelihood of a label.

CTC does not try to learn timings and boundaries of words. If the only difference between two label sequences is how they align, blanks are not taken into account. There is an effective forward-backward algorithm for scoring, which is a non-trivial process given the variety of ways equivalent label sequences might occur. The neural network parameters can then be updated using the back-propagation technique using CTC scores.

To train recurrent neural networks (RNNs) like LSTM (Long Short-Term Memory) networks to handle sequence problems where the time is uncertain, Connectionist Temporal Classification (CTC) is a form of neural network output and it is associated with scoring function which is establishing a grade or score to evaluate the performance of the output. Therefore, in this way Word Error Rate (WER) is minimized and accurate results are received at the output end.

## 5.2 XLSR Wav2vec2.0

There is a multilingual wav2vec2 variant known as XLSR. Cross-lingual speech representations, or XLSR for short, refers to the model's capacity to acquire speech representations that are applicable to several different languages. Fig 20 highlights the model ability to work on multiple languages. This single, multilingual speech recognition model competes favorably with a number of potent standalone models.

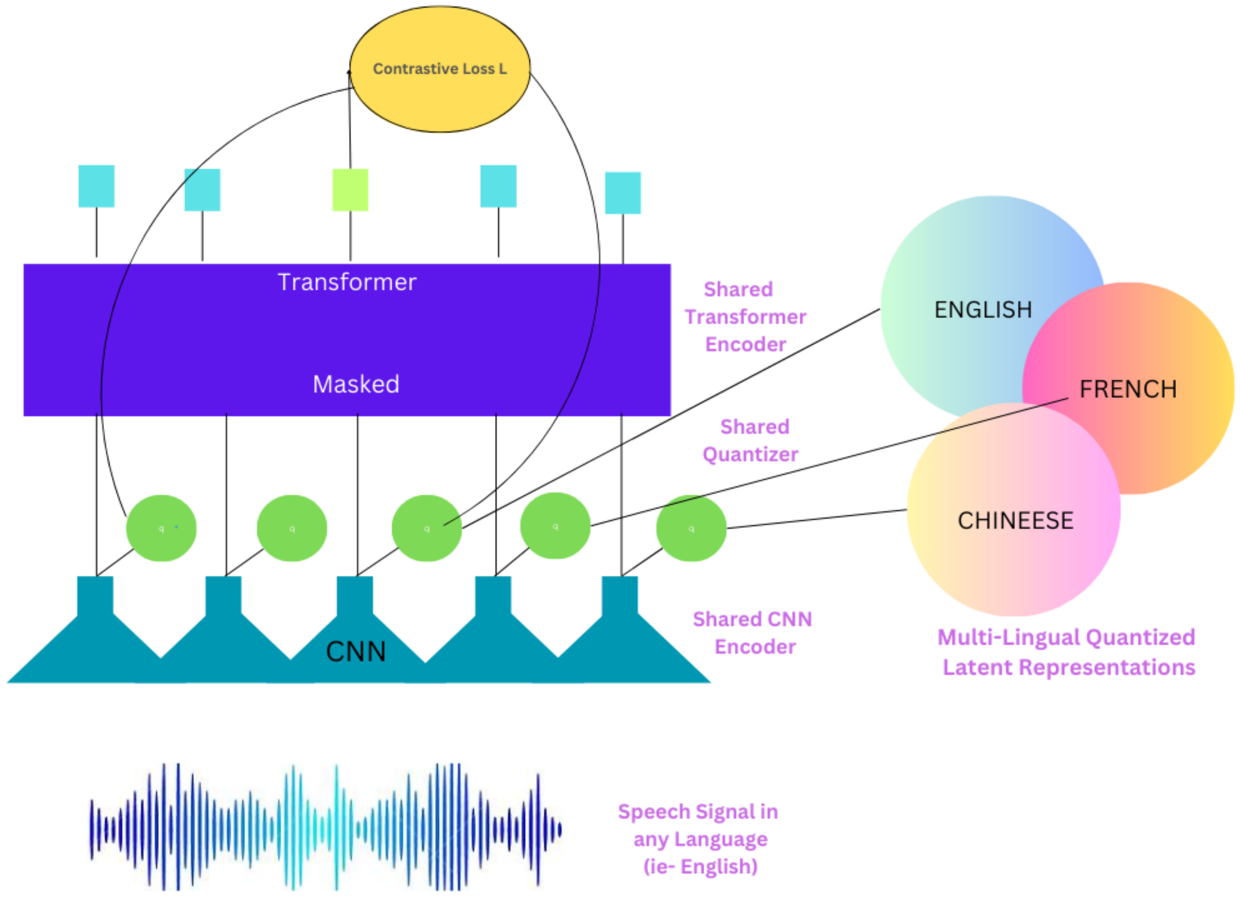

Contrastive Loss L
Transformer
Masked
Shared Transformer Encoder
Shared Quantizer
ENGLISH
FRENCH
CHINEESE
Multi-Lingual Quantized Latent Representations
CNN
Shared CNN Encoder
Speech Signal in any Language (ie- English)

**Figure 20: XLSR wav2vec2 Model**

The model was trained on labelled audio data for subsequent tasks such as voice recognition, speech interpretation, and audio categorization. This approach shows significant improvements over previous state-of-the-art results in speaker/language identification, voice translation, and speech recognition. The previously trained network is reinforced with a single linear layer for fine-tuning. As a result, the model XLSR gains the capacity to bridge discrete tokens between languages. The model successfully recognizes, classifies, and translates the desired input after the adjustments. Fig 21 elaborates the self-supervised pre-training from unlabeled dataset to the fine-tuning of the model.

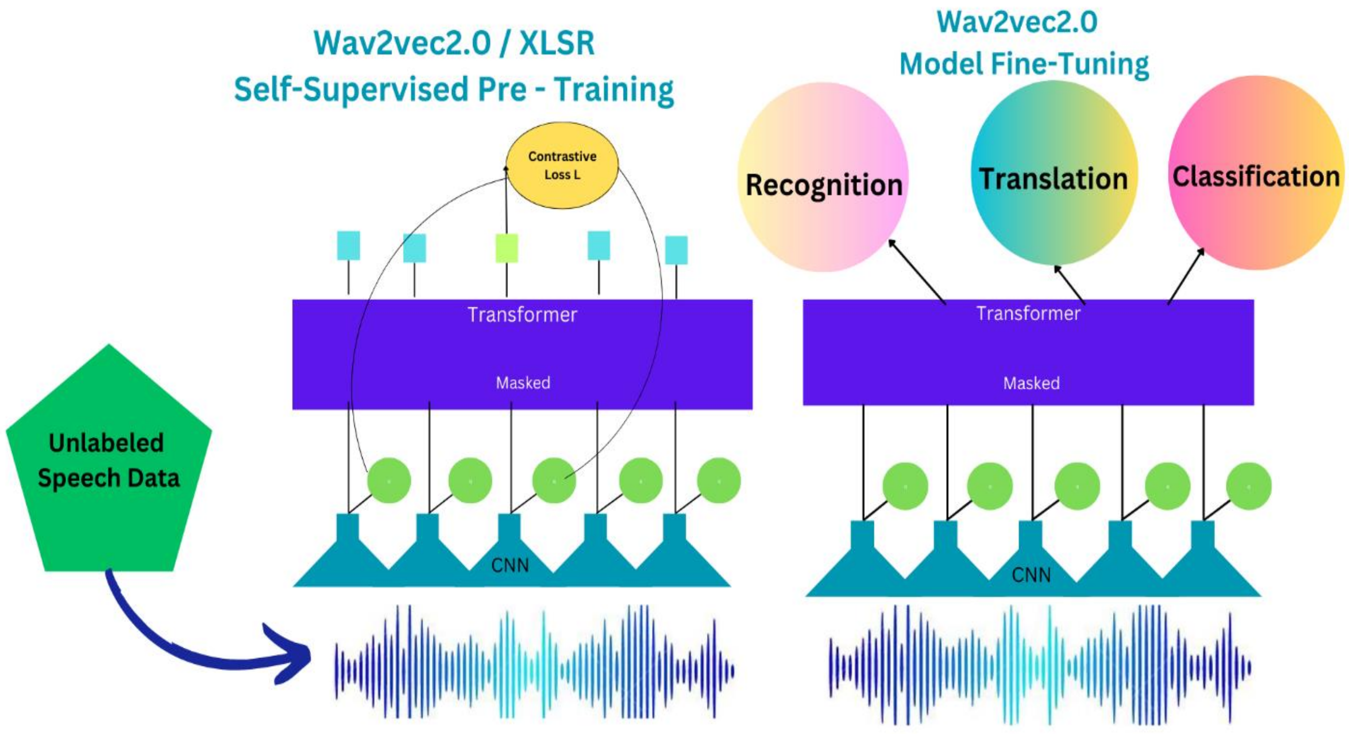


**Figure 21: Fine Tuning XLSR wav2vec2 Model**

*Chapter 6*

# SIMULATION AND RESULTS

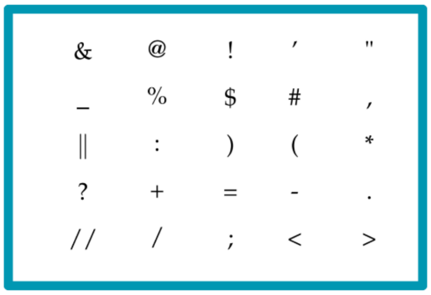
## 6.1 Dataset

ASR Implementation takes a sizable dataset, recorded from numerous speakers, to build a speaker-independent speech recognition system. The total duration of these recordings is roughly 100 hours. Approximately 5000 Pashto transcribed audio files with 70% male and 30 female audio representation. Participants belongs to various districts in Pakistan's Khyber Pakhtunkhwa province.

Written data was read aloud by transcribers and recorded in order to obtain transcribed audio recording data. Our study after listening to these recordings shows that we need approx. 10,000 words to complete an hour of recording. As a result, a substantial amount of textual information was needed to finish the recording work, and it was gathered from several sources. To gather textual information, Pashto literary works, blogs, and news websites were utilized. The information gathered came from current affairs, blog posts, documentaries, and articles that were featured in various newspapers.

A portion of the information was gathered through Facebook, a social media platform, in the form of posts and short tales. To create the input, this data had to be filtered and polished because it was in hundreds of lines.

The data that was gathered contained numerous errors or unrelated information. It included punctuation special characters and other symbols that are not a part of spoken language and cannot be recorded or trained, such as ",.,*, and @. These characters will be either deleted from the database or replaced with spaces while coding. Fig 22 shows the list of characters generating errors.



| | | | | |
|---|---|---|---|---|
| & | @ | ! | ’ | " |
| _ | % | $ | # | , |
| ‖ | : | ) | ( | * |
| ? | + | = | - | . |
| // | / | ; | < | > |

**Figure 22: List of Characters Removed**

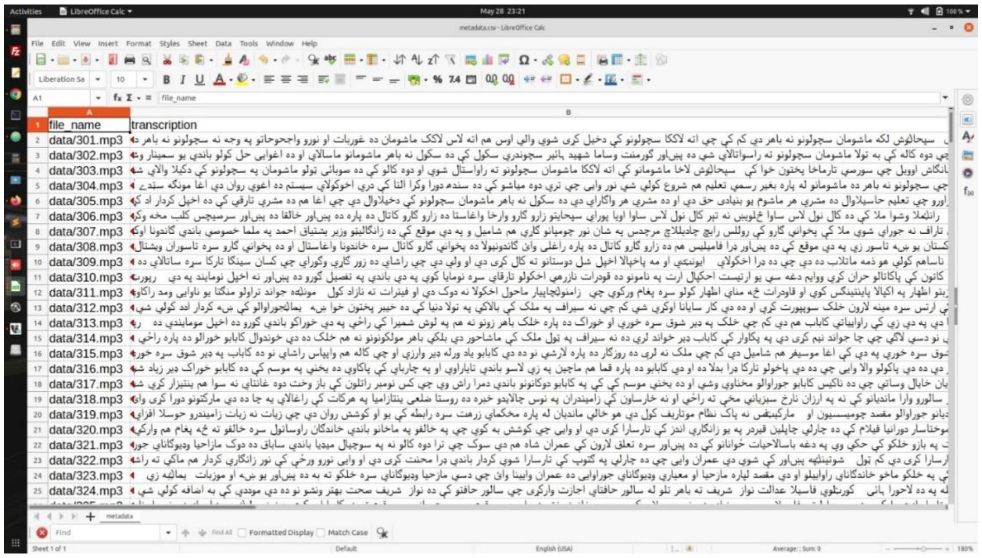
Participants recorded the audio files using the built-in software on their mobile devices. The participants had the option of selecting the gadget. Since the goal was to collect actual data, there was no dedicated lab setting for recording. Participants were not given any specific instructions other than to read as accurately as they could. Recording process was done in a very careful manner so that the speaker's speech shouldn't be replaced by noise thus minimizing errors and the noise factor was also normal. Depending on the participant's situation, several sampling rates were used for each recording. Later, all audio data was converted to a 16 kHz sampling rate, stereo channel format. Fig 22 represents some of the screen shots of compiled data set. After couple of training sessions, we successfully refined 5 to 7 hours of our dataset to 1000 audio files to train the model.

## 6.2 Word Error Rate (WER)

This statistic is used to gauge the output error of a voice recognition system. Three options for a word could exist during the recognition process. First off, a different word can be used in its place. This is referred to as "substitution". Second, it can be removed but not changed in any way. In the third place, a new word is added. The three different sorts of problems mentioned above raise the WER. The WER formula is given by:

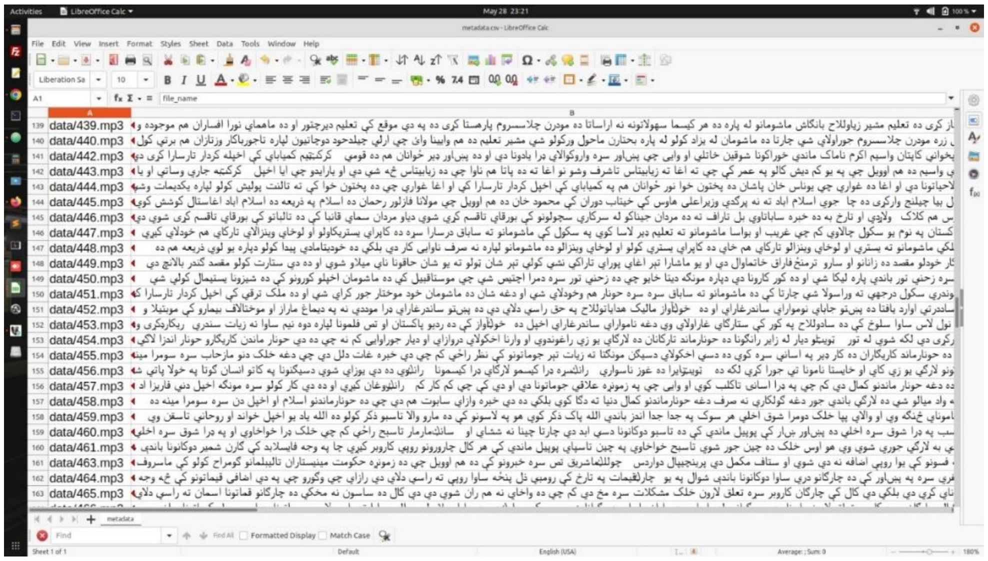
$$WER = (S + D + I)/100 \qquad (12)$$

In this case, S represents the substitution, D represents the deletion and I represent the Insertion. If the WER is 40%, it indicates that 4 words out of every 10 are added, removed, or replaced. Another possibility is that 3 words are substituted and 1 is either removed or added, among other conceivable combinations. It is feasible to achieve WER greater than 100% in this manner.



| | A | B |
|---|---|---|
| 1 | file_name | transcription |
| 2 | data/301.mp3 | كا سچلونو كي دخيل كرى شوي والي اوس هم اته لاس لاكك ماشومان ده غوربات او نورو واجحوحاتو په وجه نه سچلونو نه باهر د |
| 3 | data/302.mp3 | اور گورمنت وساما شهيد هائير سچوندري سكول كي ده سكول نه باهر ماشومانو ماسالاي او ده اغوايي حل كولو باندي يو سمينار و |
| 4 | data/303.mp3 | ماشومانو كي اته لاككا ماشومان سچلونو ته راواستال شوي او دوه كالو كي ده صوبائي تولو ماشومان په سچلونو كي دكيلا والاي ش |
| 5 | data/304.mp3 | كولي شي نور وابي چي تري دوه مياشو كي ده سندم دورا وكرا الثا كي دري اخوكولاي سيستم ده اغوي روان دي اغا مونگه ستدے |
| 6 | data/305.mp3 | ده مشري هر واكاراي دي ده سكول نه باهر ماشومان سچلونو كي دخيلاوال دي چي اغا هم ده مشري تارقي كي ده اخپل كردار اد ك |
| 7 | data/306.mp3 | لاس ساوا اويا پوراي سپحايتو زارو گارو وارخا واغاستا ده زارو گارو كاتال ده پاره ده پيښاور خالقا ده پيښاور سرصيچس كلب مخه وكړ |
| 8 | data/307.mp3 | مرچدس په شان نور چومپانو گاري هم شاميل و په دي موقع كي ده زانگاليتو وزير يشتياق احمد په ملما خصوصي باندي گاندونا اوك |
| 9 | data/308.mp3 | ده زارو گارو كاتال ده پاره راغلي واڼ گاندونيولا ده پخواني گارو كاتال سره خاندونا واغاستال او ده پخواني گارو سره تاسوران ويشتال |
| 10 | data/309.mp3 | او مه پاخپالا اخپل شل دوستانو ته كال كرى دي او ولي دي چي راشاي ده زور گاري وكوراي چي كسان سينگا تاركا سره ساتالاي ده |
| 11 | data/310.mp3 | په نامونو ده قودرات نازرهي اخكولو تارقاي سره نومايا كوي په دي باندي په تفصيل گورو ده پيښاور نه اخپل نومايند په دي ريور |
| 12 | data/311.mp3 | سره پغام وركوي چي زامنولچاپيار ماحول اخكولا نه دوك دي او فيترات ته نازاد كول مونڼه جواند تراولو منكتا يو ناوايي ومد راكاو |
| 13 | data/312.mp3 | وكري شي كم چي نه سيراف په ملک کي بالاكي په تولا دنيا كي ده خيبر پختون خوا بڼه يمالجوراوالو كي بڼه كردار ادد كولي شي |
| 14 | data/313.mp3 | [illegible] |
| 15 | data/314.mp3 | [illegible] |
| 16 | data/315.mp3 | [illegible] |
| 17 | data/316.mp3 | [illegible] |
| 18 | data/317.mp3 | [illegible] |
| 19 | data/318.mp3 | [illegible] |
| 20 | data/319.mp3 | [illegible] |
| 21 | data/320.mp3 | [illegible] |
| 22 | data/321.mp3 | [illegible] |
| 23 | data/322.mp3 | [illegible] |
| 24 | data/323.mp3 | [illegible] |
| 25 | data/324.mp3 | [illegible] |

Figure 23(a): Compiled dataset



| | A | B |
|---|---|---|
| 139 | data/439.mp3 | [illegible] |
| 140 | data/440.mp3 | [illegible] |
| 141 | data/442.mp3 | [illegible] |
| 142 | data/443.mp3 | [illegible] |
| 143 | data/444.mp3 | [illegible] |
| 144 | data/445.mp3 | [illegible] |
| 145 | data/446.mp3 | [illegible] |
| 146 | data/447.mp3 | [illegible] |
| 147 | data/448.mp3 | [illegible] |
| 148 | data/449.mp3 | [illegible] |
| 149 | data/450.mp3 | [illegible] |
| 150 | data/451.mp3 | [illegible] |
| 151 | data/452.mp3 | [illegible] |
| 152 | data/453.mp3 | [illegible] |
| 153 | data/454.mp3 | [illegible] |
| 154 | data/455.mp3 | [illegible] |
| 155 | data/456.mp3 | [illegible] |
| 156 | data/457.mp3 | [illegible] |
| 157 | data/458.mp3 | [illegible] |
| 158 | data/459.mp3 | [illegible] |
| 159 | data/460.mp3 | [illegible] |
| 160 | data/461.mp3 | [illegible] |
| 161 | data/463.mp3 | [illegible] |
| 162 | data/464.mp3 | [illegible] |
| 163 | data/465.mp3 | [illegible] |

Figure 23(b): Compiled dataset

## 6.3 Implementation

We can develop and run Python code in our browser using Colab, which is also known as the "Colaboratory". we can develop and run code in an interactive setting known as a Colab notebook. From our Google Drive account, including spreadsheets, Github, and many other places. Also, we can upload our own data into Colab notebooks.

By only a few lines of code, we can use Colab to import an image dataset, train an image classifier on it, and subsequently evaluate the model. No matter how powerful our system is, we can take advantage of Google technology, such as GPUs (Graphical Processing Units) and TPUs (Tensor Processing units), by using Colab, which run code on Google's server infrastructure. Therefore, we will implement Hugging face wav2vec2.0 model in colab by following these steps:

- First of all, we will mount our google drive to colab directories.
- Install Dataset , Transformer, and all the required liberaries including torchaudio (library of audio and signal processing), jiwer (a quick and easy Python package for measuring the word error rate (WER) along with the evaluation of an autonomous speech recognition system), librosa (to deal with audio and music-related studies), numba (to support numpy), and huggingface_hub transformers datasets.
- Import Python packages like torch , ( for tensor computation and strong GPU acceleration), NumPy (to deal with numbers) , pandas (a python module that offers quick, adaptable, and powerful data structures to work with labelled data), re (Using a string and a regular expression pattern, the search is performed within the string), json (A specified format called JavaScript Object Notation (JSON) is frequently used to transmit data as text across a network), random (for generation of random numbers), huggingface_hub (for interaction with hugging face), IPhython.display (for swift execution of a single line of Python code and for standard interpretation), random (built-in package that produces random numerals), and display, HTML (to open and read HTML files).

- Now load the data parameters, class labels voice data and evaluation metrices from the dataset. A float array representing the unprocessed waveform of the speech signal is what the speech model wav2vec2.0 uses to represent the signal. Since the wav2vec2 model was trained using Connectionist Temporal Classification (CTC), Wav2Vec2CTCTokenizer class must be used to decode the model output.
- For this purpose, use transformer to create the wav2vec2CTCTokenizer (for input tokens), wav2vec2ForCTC (for building the model by given inputs, specifying the model architecture and fine-tuning tasks), class Training Arguments (to set several options, including the outcome directory, testing and scoring method, and learning frequency), Trainer (machine learning and computational intelligence-based training facility), wav2vec2FeatureExtractor and wav2vec2Processor.
- Now load the required data class (solely envisioned to retain and hold data values that needs to be transferred among several components of a system) and field function (to give extra information like default parameters and testing guidelines) from data classes. Then load the functions like Any (for true/false conditions), Dict (also called dictionary representing a data structure of non-homogenous {key: value} pair), List (for holding several elements in a single parameter), Optional (an argument with default value) and Union (yields a set with all the values from the initial set as well as from the provided one) from the object's type analyzing and errors anticipating module named typing.
- Now login to hugging face account and load the audio and transcriptions files of dataset from the google drive.
- Wav2vec2Processor usually aids with normalization the data. To set up the trainer we further need to process the data, we will use data collector (tool for building interlinked computation pipelines) to build a batch. Padding and random masking can also be done where required to even out the data lengths for processing.
- This data collator applies distinct padding functions to input and labels for their subsequent interpretations. Since audio input and output are of distinct modes and should not be handled by the same padding function.
- Therefore, input and labels are returned in PyTorch tensors (a multidimensional matrix or array with only one data type's members) format by the tokenizer.

- Similar to the conventional data collators, the labels' tokens are padded with -100 so that they are not included in the calculation of the loss. And then in this way refined batch (script file) will be obtained.
- Now to get a sense of the transcriptions, creating a little function that displays a few random samples from the collection. The function will create an HTML file called "data.html" that contains a table with the samples from the dataset that were chosen at random.
- Numerous unusual symbols (like.'?: etc.)  are present in the transcribed data. Due to the fact that voice segments do not actually correlate to a distinct sound unit, it is considerably more difficult to classify these special symbols without any reference from a standard language framework.
- Additionally, they do not play any vital role to comprehend the meaning of a speech signal. Here we will just normalize the written content and eliminate all the symbols that do not add up to the semantics of a word and cannot actually be represented by a conventional voice. In order to remove them, firstly create an array of the characters to be removed.
- Now convert this array in a list and create a pattern by using regular expression (RegEx, a string of characters that creates a search pattern) i.e., regex filter for the list.
- Remove the special characters by mapping them with regex pattern and refined batch will be returned in its output.
- Now apply the Extract function (extract required data and form new function), which tends to integrate all the transcription strings in the batch and yields a new single concatenated string.
- By applying list function, a set of distinctive characters from the new string will be formed, which is then assigned to the vocab variable.
- Therefore, extract function will return a dictionary of concatenated string set list (all_test) and variable vocab containing list of all the unique characters.
- As an input, function prepare_dataset receives a data batch, which is often a portion of the total dataset, and passes the audio data from batch to wav2vec2processor's processor object.

- Now that the audio data has been tokenized and subjected to feature extraction, the processor transforms the unprocessed voice data into input parameters appropriate for the Wav2Vec2 model involving the sampling rate.
- Here, we extracted the first element of the batched output to obtain a single input value as the processed features are normally returned as a batched output (each component of the batched output corresponds to a single sample in the input batch).
- Therefore, the model can now receive this particular input value without any failure. Now apply the length function which will determine the length of the input values, which is then allocated to batch. When the process of training or evaluation starts, this length information will be helpful.
- To process the transcribed data, the code then calls processor to use the processor as a target processor.
- Each audio sample in the batch has a written text linked to it, which is stored in the batch field and here those linked transcripts are tokenized and transformed into model-compatible input Identities.
- Finally, batch receives the processed input Identities. To calculate the loss and metrics during training or evaluation, these labels will be used. The input values, their lengths, and label fields are now included in the updated batch that the function returns.
- Now by comparing the model's predictions to the labels, the compute metrics function determines how well the model performed in the evaluation phase of the training process.
- Basically, the anticipated logits are extracted from the pred object. These logits show the probability that the model estimates for each input token in the sequence.
- For each place in the input sequence, the token index with the highest probability is determined using the argmax function in this case. The estimated token Identities constitute as an outcome.
- The CTC (Connectionist Temporal Classification) loss function is extensively employed in ASR programs. Some input sequence tokens may be designated as padding tokens during training and excluded from the loss calculation.
- These padding tokens are frequently represented in the training data by a particular token ID, which is commonly -100. To maintain consistency and an accurate assessment of the

model's performance, code is basically substituting the padding token ID for -100 in the label token IDs. The situations where labels weren't taken into account during training, this step will also deal with it.

- The predicted token IDs are subsequently transformed back into their appropriate text representations by the processor. We get the finalized transcriptions as a result. In order to create their corresponding text representations, the label token IDs are decoded.
- Then, it is made sure that distinct tokens are not clustered together during decoding with the aid of argument statement.
- The word error rate between the reference transcriptions and the predicted transcriptions is calculated by the object wer_metric. The WER metric calculates the difference in word-level errors between the reference and forecasted texts. The function then generates a dictionary that contains the calculated Word Error Rate.
- Now the code eliminates special characters from the training and test datasets respectively prior to generate a vocabulary based on the dataset’s distinctive components and saves it as a JSON file to be deployed while training the ASR model.
- The Wav2Vec2CTCTokenizer class is now initialized, and code loads the tokenizer from the provided directory.
- The tokenizer can be configured using parameters (unknown, padding, and spaces) to alter the special tokens it returns.
- The name of the repository where the tokenizer will be pushed into the Hugging Face Model Hub is represented by the string value "wav2vec2-large-xls-r-300m-pashto-colab-test-6" in a variable. Here through the Model Hub, the tokenizer is posted and made accessible to the general public.
- Now Wav2Vec2FeatureExtractor object is initialized. The Wav2Vec2 model processes the feature vectors that are generated by the feature extractor from audio signals.
- The given parameters define the feature size (one), the sampling rate (16000 Hz) of the audio, the padding value (zero), the option to normalize the audio features, and the option to return attention masks paired with the features.
- After that, the Wav2Vec2Processor object is initialized. To preprocess the audio and text data for training or inference, the processor combines the feature extractor and tokenizer.

- The previously constructed feature_extractor object is set as the feature_extractor parameter, and the previously generated tokenizer is set as the tokenizer parameter.
- Now the dataset's first element, common_voice_train, is printed. It shows the related metadata for the initial data sample, which often include details like the location of the audio file, the text of the transcription, and other pertinent information.
- Then it prints the first data sample's audio file path, accesses the audio field in the metadata dictionary, and obtains the value corresponding to the path key.
- After the path values, sample's audio, related to metadata is printed, and the metadata dictionary's audio field is accessed and its contents are highlighted. The sample rate, duration, and other pertinent parameters of the audio file may also be included in this.
- The train audio files are loaded and converted into the Audio data type with a preset sampling rate of 16000 Hz using the casting procedure.
- Repeating the same procedure then with test audio files. And after that, the first sample's audio waveform is acquired at the set sampling rate (16000 Hz).
- A random number is generated that falls between 0 and the length of the common_voice_train dataset minus 1. It is applied to the dataset to choose a random index.
- Then it displays the "transcription" field from the common_voice_train dataset's randomly chosen data sample. The associated text transcription for that specific audio sample is also shown.
- It creates an audio widget that can play the audio waveform using the IPython Audio function by using the audio waveform data and the given16000 Hz sample frequency. The audio sample will be played on its execution.
- It outputs the intended text transcription for the audio sample that was chosen at random and then it displays the audio waveform array's shape for the audio sample that was chosen at random.
- The number of audio channels and the duration of the audio waveform are both disclosed. Then it indicates the number of samples per second in the audio waveform and the sampling rate of the randomly chosen audio sample.
- The sampling rate in this instance is 16000 Hz.

- The datasets of training and testing are preprocessed and then system creates a data collator to prepare the data for training. After that it loads the WER measure, for the process of evaluation.
- The model that was pretrained is finally loaded. It consists of a modified / fine-tuned Wav2Vec2 Farsi (Persian) language model.
- The dropout probability for the model's attention layers is set to 0.1, which means that 10% of the attention weights will be set at random to 0 during training to avoid over-fitting (when training data is more than testing data).
- By randomly zeroing out 10% of the hidden units during training, the model's dropout probability for hidden layers, which is set at 0.1, helps to regularize the model.
- Since there will be no dropout applied to the feature projections, the dropout probability for the feature projection layer is set to 0.0.
- The likelihood of masking specific time steps in the input during training is set to 0.05. In order to encourage the model to develop robust representations, on average 5% of the input time steps will be substituted with a masking token.
- The layer-drop probability is set to 0.1, which means that 10% of the model layers will be dropped at random during training and skipped during forward pass. This can also be regarded as regularization method.
- By ignoring the mismatched widths, the model will be able to deal with input sequences of different lengths during inference and training phases. When dealing with audio inputs of varying length, it will be quite be helpful.
- As a result, "mean" is selected as the reduction method for the CTC loss computation. It indicates that the total loss will be calculated by averaging the loss numbers at different time steps.
- Then it defines the ID of the padding token in the processor's tokenizer. By doing so, it guarantees that the model understands how to deal with padding tokens throughout both training and inference.
- Now set the arguments for the training to be used.

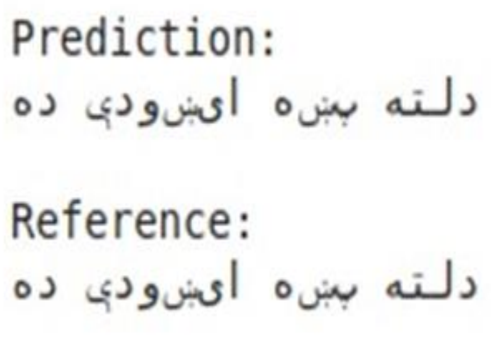
- We can adjust a number of training process variables, including batch size, learning rate, assessment approach, and logging frequency, among others, by defining these training arguments.
- For this purpose, first defines where the trained model and other training material will be saved in the output directory.
- To maximize training effectiveness, organize the training instances by input sequence lengths that are similar.
- The batch size per device (GPU) for training is set. In this scenario, one training sample will be processed by each GPU each batch.
- The amount of gradient accumulation steps is then specified here. Gradients are gathered over a number of steps, and then back-propagation is used to update the model's parameters. In this case, gradients are accumulated up until the update.
- Now the training's evaluation procedure is set. In this instance, evaluation is carried out at predetermined intervals determined by evaluation steps.
- Here we are setting epochs value, as it will conclude how many training epochs (passes of the dataset) will be made in total.
- In order to speed up training and use less memory, system supports the mixed-precision training, which uses lower precision (float16) for some computations.
- Now set some parameters including save step (how frequently the model checkpoints are saved during training), evaluation steps (the number of evaluations on the validation dataset will be carried out while training), logging steps (the number of times that training metrics and other data will be logged), learning rate (optimizer rate which regulates the step size while updating the model's parameters during training) and warm up steps (the number of steps for the learning rate scheduler).
- Now restricts the total number of stored checkpoints to be kept to two. Up to the given limit, only the most recent checkpoints will be retained.
- The training pipeline will be built up with the proper model, data collator, training arguments, evaluation dataset, and tokenizer by initializing the Trainer object.
- The Trainer object offers ways to train the model, assess its effectiveness, and store checkpoints as we go.

- In order to load the trained model, handle the input data, provide estimations, and contrast them with the transcription of the actual events, we will start the training process.
- Configure the trainer and train the model on specified dataset. In parallel, system will push all the related files to Hugging Face Model Hub.
- Using the repository where the model was pushed, load the optimized model from the Hugging Face Model Hub along with appropriate processor (tokenizer and feature extractor) for the model.
- Now the loaded processor is used to process the incoming audio data from the common_voice_test dataset's first sample.
- The processed input values are returned as a dictionary of PyTorch tensors.
- Now the system will run the loaded model with the input values that have been processed through it to produce the anticipated logits.
- By moving the argmax down the last dimension of the logit’s tensor, the predicted token IDs are extracted.
- The predictions for the first sample are retrieved using the [0] indexing.
- Now the expected transcription is obtained by utilizing the processor's decode technique to decode the predicted token IDs.
- And lastly, system displays the ground truth reference transcription from the common_voice_test dataset.

## 6.4 Results

For the low resource language Pashto, we have built the dataset and applied the wav2vec2 model successfully. After fine-tuning the model, we obtained findings that were about 66% accurate over all. The word mistake rate, which is around 37% (WER reduces with time and training), is used to assess the system's performance. The mapping between the training data set and the test data set yields very clear results.

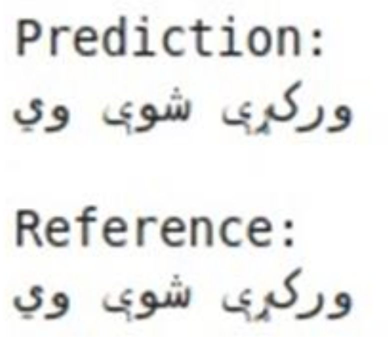
Despite the fact our WER for audio chunks and short lengths of speech is relatively comparable to published work in Pashto, the lower the error rate, the more precise this model will be with this volume of data. WER is calculated in accordance with all test data utterances. In order to lower WER, we noticed multiple data transcriptions and discovered that some mistakes were complex words rather than reading errors. Results are shown below:



```
Prediction:
دلـته ښه ایښودې ده

Reference:
دلـته ښه ایښودې ده
```

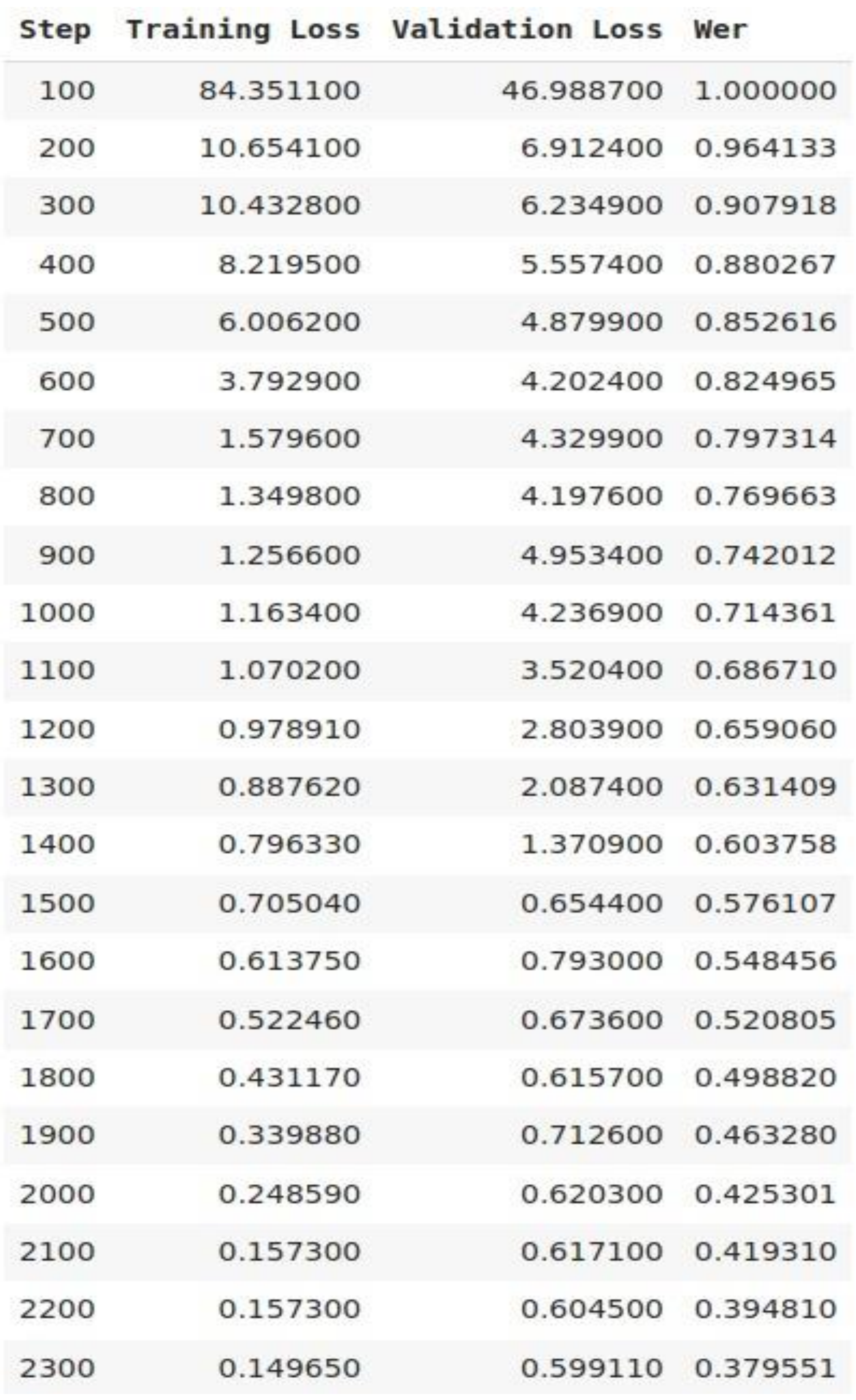
**Figure 24(a): Output Transcriptions**



```
Prediction:
ورکړې شوې وي

Reference:
ورکړې شوې وي
```

**Figure 24(b): Output Transcriptions**



| Step | Training Loss | Validation Loss | Wer |
|---|---|---|---|
| 100 | 84.351100 | 46.988700 | 1.000000 |
| 200 | 10.654100 | 6.912400 | 0.964133 |
| 300 | 10.432800 | 6.234900 | 0.907918 |
| 400 | 8.219500 | 5.557400 | 0.880267 |
| 500 | 6.006200 | 4.879900 | 0.852616 |
| 600 | 3.792900 | 4.202400 | 0.824965 |
| 700 | 1.579600 | 4.329900 | 0.797314 |
| 800 | 1.349800 | 4.197600 | 0.769663 |
| 900 | 1.256600 | 4.953400 | 0.742012 |
| 1000 | 1.163400 | 4.236900 | 0.714361 |
| 1100 | 1.070200 | 3.520400 | 0.686710 |
| 1200 | 0.978910 | 2.803900 | 0.659060 |
| 1300 | 0.887620 | 2.087400 | 0.631409 |
| 1400 | 0.796330 | 1.370900 | 0.603758 |
| 1500 | 0.705040 | 0.654400 | 0.576107 |
| 1600 | 0.613750 | 0.793000 | 0.548456 |
| 1700 | 0.522460 | 0.673600 | 0.520805 |
| 1800 | 0.431170 | 0.615700 | 0.498820 |
| 1900 | 0.339880 | 0.712600 | 0.463280 |
| 2000 | 0.248590 | 0.620300 | 0.425301 |
| 2100 | 0.157300 | 0.617100 | 0.419310 |
| 2200 | 0.157300 | 0.604500 | 0.394810 |
| 2300 | 0.149650 | 0.599110 | 0.379551 |

**Figure 25: Output Progress**

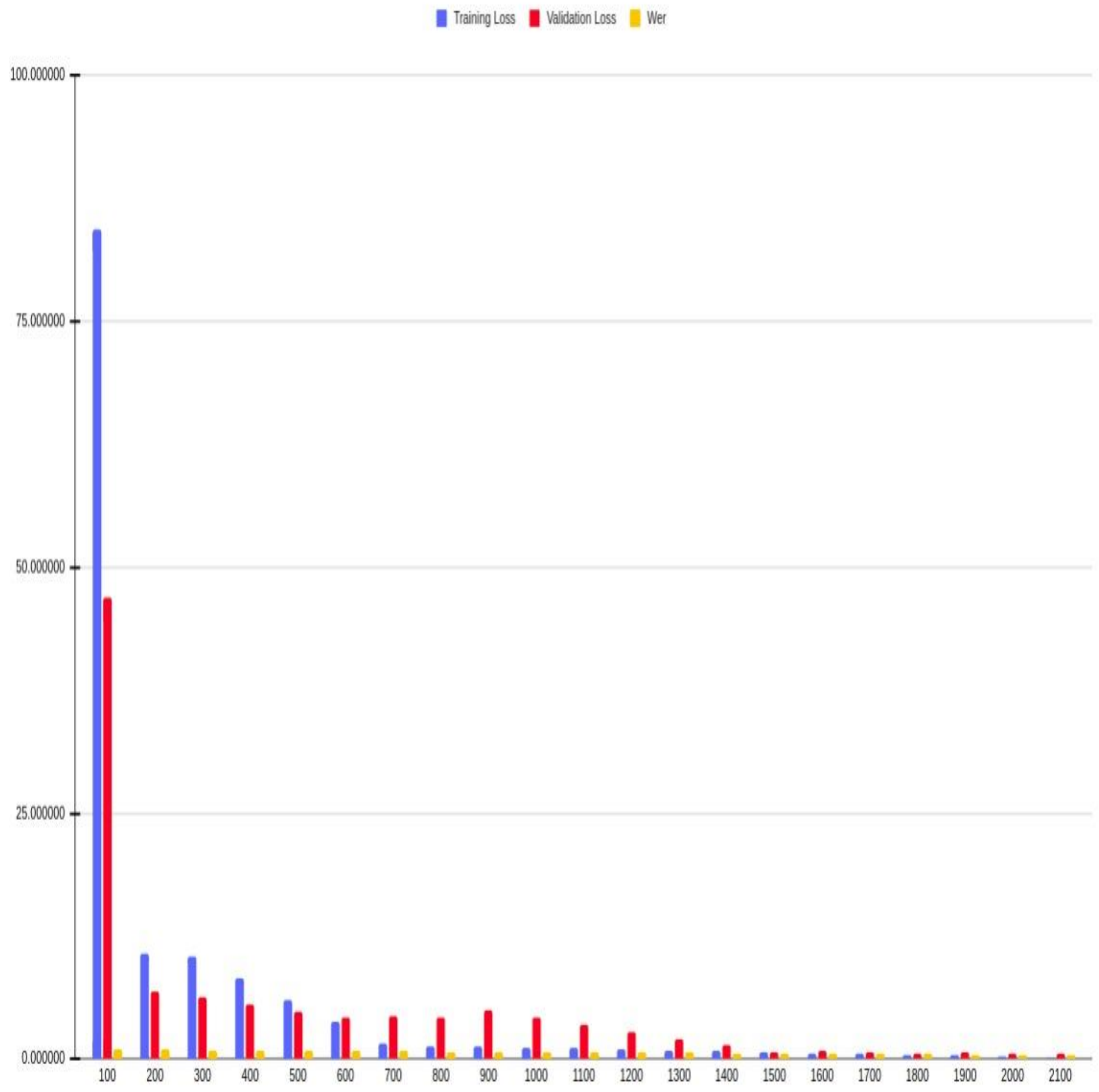
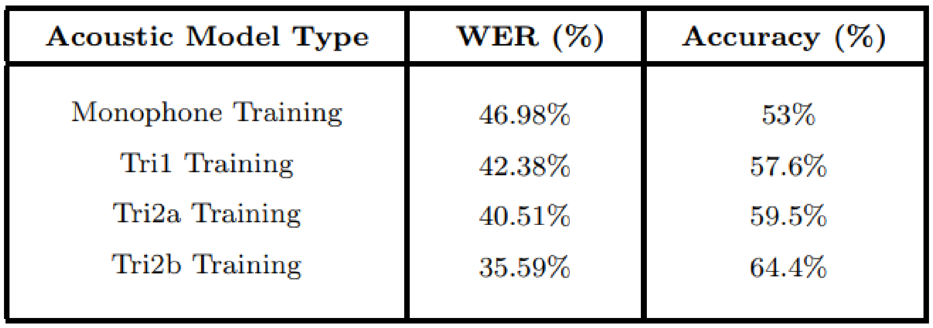


**Figure 26: Bar Chart of Output Parameters**

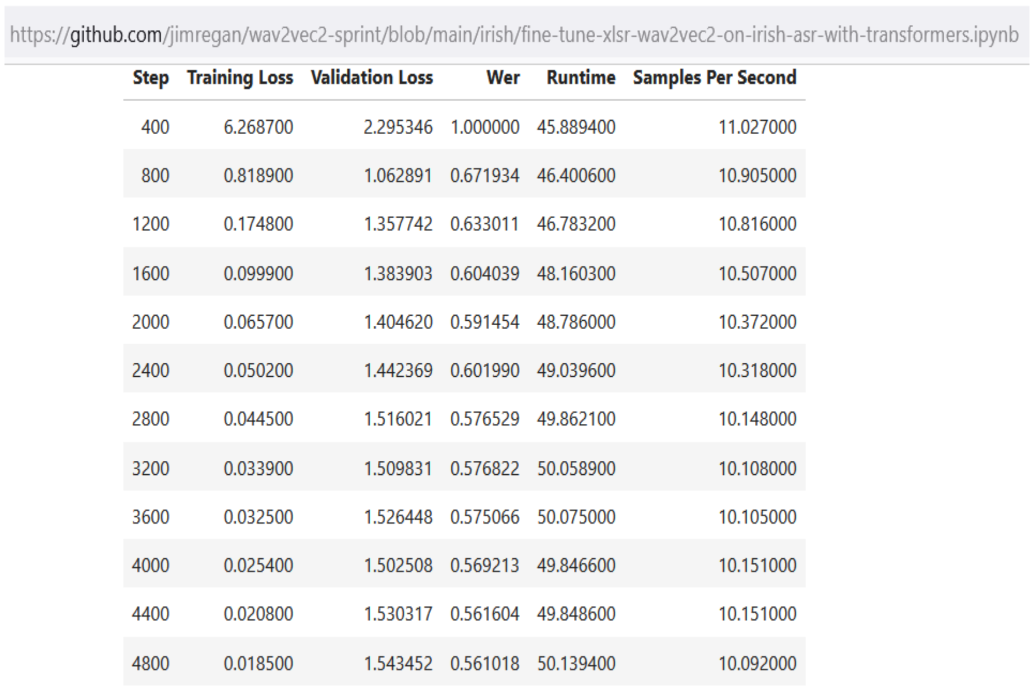
## 6.5 Output Comparisons

Comparison of our low resource wav2vec2.0 Pashto output results with the output of the traditional Pashto ASR system as well as wave2vec2.0 high resource (English) and low resource (Irish) languages is made to validate the effectiveness of the proposed result Fig 27,28,29 shows the output results of HMM based traditional Pashto ASR system, wave2vec2.0 low resource Irish ASR system and wave2vec2.0 high resource English ASR system respectively. Detailed comparison analysis is concluded in section 6.6 below.



| Acoustic Model Type | WER (%) | Accuracy (%) |
|---|---|---|
| Monophone Training | 46.98% | 53% |
| Tri1 Training | 42.38% | 57.6% |
| Tri2a Training | 40.51% | 59.5% |
| Tri2b Training | 35.59% | 64.4% |

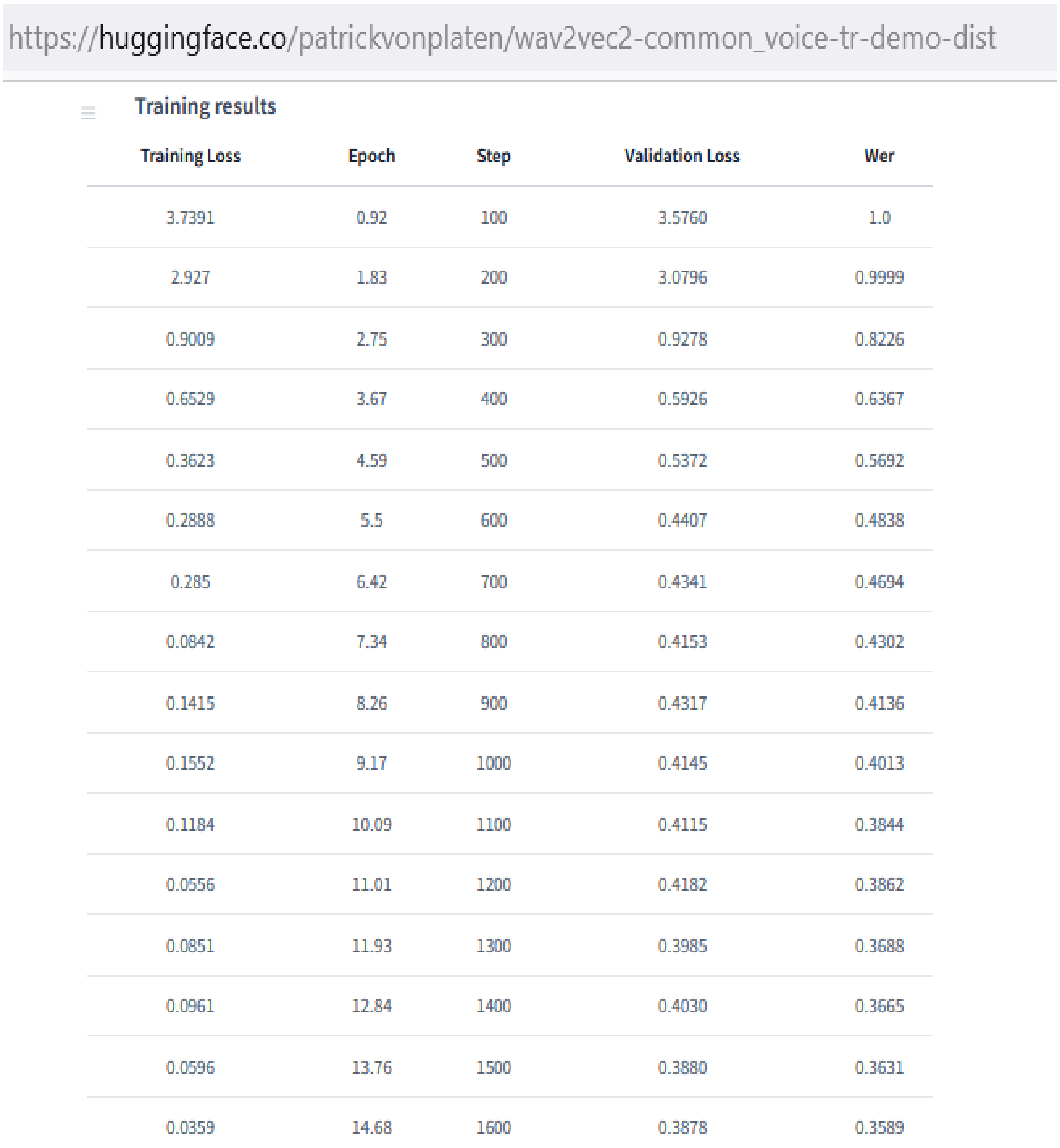
Figure 27: HMM Based Pashto ASR Output [35]



https://github.com/jimregan/wav2vec2-sprint/blob/main/irish/fine-tune-xlsr-wav2vec2-on-irish-asr-with-transformers.ipynb

| Step | Training Loss | Validation Loss | Wer | Runtime | Samples Per Second |
|---|---|---|---|---|---|
| 400 | 6.268700 | 2.295346 | 1.000000 | 45.889400 | 11.027000 |
| 800 | 0.818900 | 1.062891 | 0.671934 | 46.400600 | 10.905000 |
| 1200 | 0.174800 | 1.357742 | 0.633011 | 46.783200 | 10.816000 |
| 1600 | 0.099900 | 1.383903 | 0.604039 | 48.160300 | 10.507000 |
| 2000 | 0.065700 | 1.404620 | 0.591454 | 48.786000 | 10.372000 |
| 2400 | 0.050200 | 1.442369 | 0.601990 | 49.039600 | 10.318000 |
| 2800 | 0.044500 | 1.516021 | 0.576529 | 49.862100 | 10.148000 |
| 3200 | 0.033900 | 1.509831 | 0.576822 | 50.058900 | 10.108000 |
| 3600 | 0.032500 | 1.526448 | 0.575066 | 50.075000 | 10.105000 |
| 4000 | 0.025400 | 1.502508 | 0.569213 | 49.846600 | 10.151000 |
| 4400 | 0.020800 | 1.530317 | 0.561604 | 49.848600 | 10.151000 |
| 4800 | 0.018500 | 1.543452 | 0.561018 | 50.139400 | 10.092000 |

Figure 28: Facebook wave2vec2 Low Resource Irish ASR Output [39]



https://huggingface.co/patrickvonplaten/wav2vec2-common_voice-tr-demo-dist

Training results

| Training Loss | Epoch | Step | Validation Loss | Wer |
|---|---|---|---|---|
| 3.7391 | 0.92 | 100 | 3.5760 | 1.0 |
| 2.927 | 1.83 | 200 | 3.0796 | 0.9999 |
| 0.9009 | 2.75 | 300 | 0.9278 | 0.8226 |
| 0.6529 | 3.67 | 400 | 0.5926 | 0.6367 |
| 0.3623 | 4.59 | 500 | 0.5372 | 0.5692 |
| 0.2888 | 5.5 | 600 | 0.4407 | 0.4838 |
| 0.285 | 6.42 | 700 | 0.4341 | 0.4694 |
| 0.0842 | 7.34 | 800 | 0.4153 | 0.4302 |
| 0.1415 | 8.26 | 900 | 0.4317 | 0.4136 |
| 0.1552 | 9.17 | 1000 | 0.4145 | 0.4013 |
| 0.1184 | 10.09 | 1100 | 0.4115 | 0.3844 |
| 0.0556 | 11.01 | 1200 | 0.4182 | 0.3862 |
| 0.0851 | 11.93 | 1300 | 0.3985 | 0.3688 |
| 0.0961 | 12.84 | 1400 | 0.4030 | 0.3665 |
| 0.0596 | 13.76 | 1500 | 0.3880 | 0.3631 |
| 0.0359 | 14.68 | 1600 | 0.3878 | 0.3589 |

**Figure 29: Facebook (wav2vec2-large-xlsr-53) High Resource English ASR Output [38]**

## 6.6 Output Comparisons Analysis

Here, at this point we can observe that our proposed model wav2vec2.0 outperforms the traditional HMM Pashto ASR (64% accuracy and 35% WER using 100 hour of audio dataset for training) in terms of accuracy 66% and successfully achieved WER of 37% (using approximately 5 hours of training data). By comparing our output results with the output of the same model (wav2vec2.0) with other low resource language (Irish) ASR system, we find out that our accuracy and WER is very promising and outstanding as compared to it, i.e.,50% WER. Also, comparison of our model wav2vec2.0 with the same model (wav2vec2.0) with high resource language (English) ASR system, we find out that WER is approximately 35%. Therefore, we can say that the proposed model yields very satisfying output results which can get more better and precise with further fine-tuning of model and additional refining of dataset.

**Table 7: Comparison Analysis**

| **Model Vs. Output** | **HMM Traditional Model (Low Resource Pashto 100 hours)** | **Proposed Model Wav2vec2.0 (Low Resource Pashto 5-7 hours)** | **Wav2vec2.0 (Low Resource Irish)** | **Wav2vec2.0 (High Resource English)** |
|---|---|---|---|---|
| **WER (Word Error Rate)** | 35.5% (64% Accuracy) | 37% (66% Accuracy) | 50% | 35% |

*Chapter 7*

# CONCLUSION AND FUTURE WORK

## 7.1 Conclusion

Our prediction can undoubtedly be used to identify the transcription for sentence chunks / short phrases, but it is not yet 100% perfect. Although, accuracy rate of 66% with 37% of WER is achieved successfully. For short length audio inputs, the model outperforms the traditional ASR pipelines with accurate transcriptions exhibiting high accuracy rate with low WER. The model's overall performance can certainly be enhanced by giving it more time to train, spending more effort on data preparation, and especially by utilizing a language model for decoding. Adding more varied information, such as Pashto literature, prose, and tales in data set will be beneficial. We also make available to the general public our work for comparing input speech and recognized output speech. Given that Pashto is the second-most widely used temporary language in Pakistan, there is a lot more room for speech recognition research in this language. Future research will ultimately aim to create a trustworthy and precise Pashto speech recognition system for a range of linguistic and acoustic circumstances.

## 7.2 Future Work

For a variety of speech-related applications like keyword identification and acoustic event recognition, wav2vec offers superior audio data representations. The use of AI to proactively identify and identify harmful information and protect users on internet platforms may be improved with more research in this area. However, the wider ramifications of this study are connected to teams at Facebook AI and the larger AI community's pursuit of self-supervised training methods. We have contributed our bit to this endeavor by optimizing the wav2vec model for Pashto, a low resource language. Self-supervision is advancing research in practically every area of the science, not only speech. Through continued accessible collaborative science, we can move closer to a time when unlabeled training data is the norm rather than the exception. The aim of this research is to create and enhance speech recognition-based systems, as well as to grow the attention of researchers in this area of study. ASR for low-resource Pashto is thus simply a first step towards the creation of more sophisticated Pashto applications and technology. Our next target for the future is to more fine-tune and train our model for the achievement of higher accuracy with lower WER along with more optimization and robustness. Further make this system applicable for real time.

# REFERENCES

[1] Deep speech 2: End-to-end speech recognition in English and Mandarin, 33rd International Conference on Machine Learning, ICML 2016.

[2] https://arxiv.org/pdf/2102.04889v1.pdf

[3] Combining Spectral and Self-Supervised Features for Low Resource Speech Recognition and Translation Apr 2022 https://arxiv.org/pdf/2204.02470v2.pdf

[4] https://cloud.google.com/speech-to-text/docs/languages

[5] https://cloud.google.com/translate/docs/languages

[6] https://en.wikipedia.org/wiki/Pashtuns#:~:text=The%20total%20population%20 of%20the, census%20in%20Afghanistan%20since%201979

[7] https://reader.elsevier.com/reader/sd/pii/S2405844020302176token=EF94929A98 D4C66D088F6F745A569B1878E48669B5E7AAC7A8034E92EB58A3E4B1B0C 4010FDFCD21B821EF2D2EC99F1F&originRegion=eu-west 1&origin Creation

[8] M. Dua, R.K. Aggarwal, V. Kadyan, S. Dua, Punjabi automatic speech recognition using HTK, IJCSI Int. J. Comput. Sci. Issues 9 (4) (2012), 1694-0814.

[9] K. Sharma, P. Singh, Speech recognition of Punjabi numerals using synergic HMM and DTW approach, Indian J. Sci. Technol. 8 (27) (2015).

[10] Y. Kumar, N. Singh, An automatic spontaneous live speech recognition system for Punjabi language corpus, Int. J. CTA 9 (20) (2016) 9575–9595

[11] H.B. Chauhan, B.A. Tanawala, Comparative study of MFCC and LPC algorithms for Gujrati isolated word recognition, Int. J. Innovat. Res. Comput. Commun. Eng. 3 (2) (2015) 822–826.

[12] J. Ashraf, N. Iqbal, N.S. Khattak, A.M. Zaidi, March. Speaker independent Urdu speech recognition using HMM, in: Informatics and Systems (INFOS), 2010 the 7th International Conference on, IEEE, 2010, pp. 1–5.

[13] A.A. Raza, S. Hussain, H. Sarfraz, I. Ullah, Z. Sarfraz, An ASR system for spontaneous Urdu speech, Proc. Orient. COCOSDA (2010) 24–25.

[14] H. Ali, A. Jianwei, K. Iqbal, Automatic speech recognition of Urdu digits with optimal classification approach, Int. J. Comput. Appl. 118 (9) (2015).

[15] Rohit Prasad et al. “Pashto speech recognition with limited pronunciation lexicon”. In: IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE,2010.

[16] M. Qasim, S. Nawaz, S. Hussain, T. Habib, Urdu speech recognition system for district names of Pakistan: development, challenges, and solutions, in: Coordination and Standardization of Speech Databases and Assessment Techniques (O-COCOSDA), 2016.

[17] S. Sinha, S.S. Agrawal, A. Jain, Continuous density Hidden Markov Model for Hindi speech recognition, GSTF J. Comput. 3 (2) (2018).

[18] A. Sharma, M.C. Shrotriya, O. Farooq, Z.A. Abbasi, Hybrid wavelet based LPC features for Hindi speech recognition, Int. J. Inf. Commun. Technol. 1 (3-4) (2008)373–381.

[19] S. Ranjan, Exploring the discrete wavelet transform as a tool for Hindi speech recognition, Int. J. Comput. Theor. Eng. 2 (4) (2010) 642

[20] K. Kumar, R.K. Aggarwal, A. Jain, A Hindi speech recognition system for connected words using HTK, Int. J. Comput. Syst. Eng. 1 (1) (2012) 25–32.

[21] S. Kayte, M. Mundada, D.C. Kayte, Implementation of Marathi language speech databases for large dictionary, IOSR J. VLSI Signal Process. 5 (2015) 40–45e.

[22] N. Hammami, M. Bedda, N. Farah, Probabilistic classification based on Gaussian copula for speech recognition: application to Spoken Arabic digits, in: Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), 2013, IEEE, 2013.

[23] M. Hassine, L. Boussaid, H. Massouad, Hybrid techniques for Arabic Letter recognition, Int. J. Intell. Inf. Syst. 4 (1) (2015) 27–34.

[24] G. Muhammad, Y.A. Alotaibi, M.N. Huda, Automatic speech recognition for Bangla digits, in: Computers and Information Technology, 2009. ICCIT'09. 12th International Conference on, IEEE, 2009, December, pp. 379–383.

[25] Z. Ali, A.W. Abbas, T.M. Thasleema, B. Uddin, T. Raaz, S.A.R. Abid, Database development and automatic speech recognition of isolated Pashto spoken digits using MFCC and k-NN, Int. J. Speech Technol. 18 (2) (2015) 271–275.

[26] file:///C:/Users/MAJORS~1/AppData/Local/Temp/06330501.pdf.

[27] https://www.uetpeshawar.edu.pk/TRP-G/Dr.Nasir-Ahmad TRP/Conferences/2012/Pashto%20Spoken%20Digits%20Database%20for%20the%20 Automatic%20Speech%20Recognition%20Research.pdf.

[28] https://www.dline.info/jcl/fulltext/v5n2/3.pdf.

[29] 1706.00321.pdf (arxiv.org).

[30] Pashto speech recognition with limited pronunciation lexicon - MOAM.INFO or Pashto speech recognition with limited pronunciation lexicon | IEEE Conference Publication | IEEE Xplore.

[31] Irfan Ahmed, The development of isolated words Pashto automatic speech recognition system,18th ICAC, IEEE. 2012.

[32] Asadullah, Automatic Urdu Speech Recognition using Hidden Markov Model, 2016 International Conference on Image, Vision and Computing (ICIVC). 2016.

[33] Hanan Aldarmaki, Asad Ullah, Nazar Zaki, Unsupervised automatic speech recognition: A review, Speech Communication, 2022, DOI: 10.48550/arXiv.2106.04897.

[34] Ragheb Al-Ghezi, Yaroslav Getman, Aku Rouhe, Raili Hilden, Mikko Kurimo, Self-supervised end to end ASR, for low resource L2 Swedish, INTERSPEECH, 2021.

[35] Muhammad Rehman Gul, Shibli Nisar, Muhammad Tariq, Development of a large vocabulary for Pashto language automatic speech recognition system, 2022.

[36] Adnan Ali, Muhammad Ali Tahir, Hassan Nazeer Chaudhry, Development of a large vocabulary Urdu automatic speech recognition system and error analysis methodology, 2020.

[37] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, Michael Auli, Wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020.

[38] https://huggingface.co/patrickvonplaten/wav2vec2-common_voice-tr-demo-dist

[39] https://github.com/jimregan/wav2vec2-sprint/blob/main/irish/fine-tune-xlsr-wav2vec2-on-irish-asr-with-transformers.ipynb

[40] https://cloud.google.com/ai-platform/training/docs/algorithms/bert-start

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, Attention is all you need, arXiv:1706.03762v5.

[42] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, Sequence to sequence learning with neural networks.

[43] F. Zhuang, A Comprehensive Survey on Transfer Learning, IEEE, 2021.

[44] W.Han,Z.Zhang, Y.Zhang, J.Yu, C.-C.Chiu, J. Qin, A.Gulati, R. Pang, and Y. Wu. Contextnet: Improving convolutional neural networks for automatic speech recognition with global context. arXiv, 2020.

[45] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. arXiv, abs/1611.01144, 2016.

[46] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. IEEE Trans. Pattern Anal. Mach. Intell., 33(1):117–128, Jan. 2011.

[47] D. Jiang, X. Lei, W. Li, N. Luo, Y. Hu, W. Zou, and X. Li. Improving transformer-based speech recognition using unsupervised pre-training. arXiv, abs/1910.09932, 2019.

[48] J. Kahn et al. Libri-light: A benchmark for asr with limited or no supervision. In Proc. of ICASSP, 2020.

[49] K. Kawakami, L. Wang, C. Dyer, P. Blunsom, and A. van den Oord. Learning robust and multilingual speech representations. arXiv, 2020.

[50] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In Proc. of ICLR, 2015.

[51] A. Laptev, R. Korostik, A. Svischev, A. Andrusenko, I. Medennikov, and S. Rybin. You do not need more data: Improving end-to-end speech recognition by text-to-speech data augmentation. arXiv, abs/2005.07157, 2020.

[52] M. P. Lewis, G. F. Simon, and C. D. Fennig. Ethnologue: Languages of the world, nineteenth edition. Online version: http://www.ethnologue.com, 2016.

[53] A. H. Liu, T. Tu, H. yi Lee, and L. shan Lee. Towards unsupervised speech recognition and synthesis with quantized speech representation learning. arXiv, 2019.

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

[55] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, and H. Ney. Rwth asr systems for librispeech: Hybrid vs attention. In Interspeech 2019, 2019.

[56] C. J. Maddison, D. Tarlow, and T. Minka. A* sampling. In Advances in Neural Information Processing Systems, pages 3086–3094, 2014.

[57] I. Misra and L. van der Maaten. Self-supervised learning of pretext-invariant representations. arXiv, 2019.

[58] A. Mohamed, D. Okhonko, and L. Zettlemoyer. Transformers with convolutional context for ASR. arXiv, abs/1904.11660, 2019.

[59] M. Ott, S. Edunov, D. Grangier, and M. Auli. Scaling neural machine translation. In Proc. of WMT, 2018.

[60] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli. fairseq: Afast, extensible toolkit for sequence modeling. In Proc. of NAACL System Demonstrations, 2019.

[61] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: an asr corpus based on public domain audio books. In Proc. of ICASSP, pages 5206–5210. IEEE, 2015.

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