TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

AUTOMATIC SPEECH RECOGNITION AND CLASSIFICATION OF NEPALI SPEECH

Submitted by:
Aayush Acharya (074BCT501)
Kanchan Poudel (074BCT515)
Pranab Mishra (074BCT525)
Prasiddha Bhandari (074BCT527)

A PROJECT SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE IN COMPUTER ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
LALITPUR, NEPAL

April 18, 2022

This work was done by ***Aayush Acharya***, ***Kanchan Poudel***, ***Pranab Mishra*** *and* ***Prasiddha Bhandari*** *as their Bachelor's thesis at* ***Pulchowk Campus, Institute of Engineering(IOE)***, *in collaboration with* ***Nepal Applied Mathematics and Informatics Institute for research(NAAMII)***. *They were affiliated as* ***Research Interns at NAAMII*** *during the period.* ***Dr. Basanta Joshi*** *supervised the project as supervisor from the University. Similarly,* ***Dr. Bishsesh Khanal*** *supervised the project as supervisor from* ***NAAMII***.

LETTER OF APPROVAL

The undersigned certify that they have read and recommended to the Institute of Engineering for acceptance, a project report entitled **AUTOMATIC SPEECH RECOGNITION AND CLASSIFICATION OF NEPALI SPEECH** submitted by **Aayush Acharya**, **Kanchan Poudel**, **Pranab Mishra** and **Prasiddha Bhandari** in partial fulfillment of the requirements for the Bachelor's Degree in Computer Engineering.

Supervisor: **Dr. Basanta Joshi**, Assistant Professor
Department of Electronics and Computer Engineering
Pulchowk Campus, Institute of Engineering

Internal Examiner: **Dr. Surendra Shrestha**, Associate Professor
Department of Electronics and Computer Engineering
Pulchowk Campus, Institute of Engineering

External Examiner: **Dr. Suresh Pokharel**, Director
IT Program
Presidental Business School

DATE OF APPROVAL:

LETTER OF DEPARTMENTAL CLEARANCE

The undersigned certifies that the project entitled **AUTOMATIC SPEECH RECOGNITION AND CLASSIFICATION OF NEPALI SPEECH** submitted by **Aayush Acharya**, **Kanchan Poudel**, **Pranab Mishra** and **Prasiddha Bhandari** in partial fulfillment of the requirements for the Bachelor's Degree in Computer Engineering has been accepted by the Department of Electronics and Computer Engineering, Pulchowk Campus.

Prof. Dr. Ram Krishna Maharjan
Head of Department
Department of Electronics and Computer Engineering
Pulchowk Campus, Institute of Engineering
Lalitpur, Nepal

COPYRIGHT

ACKNOWLEDGEMENT

First of all, we would like to extend our sincere gratitude towards the **Department of Electronics and Computer Engineering, IOE Pulchowk Campus**, for providing us with this opportunity to hone our programming and analytical skills. Due to this opportunity, we gained insights on designing a furnished solution to a real-world problem. While academic projects, in general, aim to enhance our understanding more than creating a solution, providing the opportunity to do this major project allows us to take our raw skills and abilities to the next step by creating a platform to innovate an impactful solution. We praise and respect the department's initiative in indulging the students in this self-practising method of education to familiarize us more with tackling real-world problems while also enhancing our capabilities to work as a team. The opportunity to undertake this project will also lay a strong groundwork for us to pursue future projects in the field of software development and artificial intelligence. We are also thankful for receiving support and guidance through any difficulties we encountered in the course of this project.

We would like to give special gratitude to **Dr. Basanta Joshi**, our project supervisor, whose constant guidance, and precious encouragement helped us pursue this project. We are thankful his invaluable suggestions and constructive criticisms.

We would like to express our gratitude to **Nepal Applied Mathematics and Informatics Institute for research (NAAMII)** for the project idea and the opportunity for collaboration in this project. Special thanks to **Dr. Bishesh Khanal**, Director of the organization, for guiding us throughout the project with valuable feedback and suggestions.

Our special gratitude to all of our friends and seniors who have directly and indirectly helped us during this project by sharing their experience and ideas. Lastly, special thanks to our family members for their constant support and encouragement.

Authors
Aayush Acharya
Kanchan Poudel
Pranab Mishra
Prasiddha Bhandari

ABSTRACT

The swift rise of internet users in Nepal has proliferated the amount of content uploaded by the day. The advent of many platforms has evolved from simple text based communications allowing the users to share content in the form of audio and video. While this has provided the users with flexibility, it has also made it difficult to search for relevant content on the internet. This is more so the case for low resource languages like Nepali, where speech recognition systems have not evolved as it has for resource rich languages like English. This project proposes a semi-supervised Automatic Speech Recognition (ASR) and classifier in the form of a web interface which classifies audio/video based on its content. The shortcomings of low resource language is mitigated in supervised approach by using wav2vec 2.0, a self-supervised framework for learning contextualized representations. The english wav2vec 2.0 and Cross Lingual Speech Representations for Indic Languages (CLSRIL-23) model trained on indic languages is pretrained and finetuned on Open Speech and Language Resources (OpenSLR) dataset to create a Nepali ASR. The CLSRIL-23 model outperforms the wav2vec 2.0 model in most test sets achieveing a word error rate of 23.05 in OpenSLR female dataset. For classification, Multilingual Representations for Indian Languages (MuRIL) model, a Bidirectional Encoder Representations from Transformers (BERT) based model trained on indic languages is utilized to classify the transcriptions generated among a set of categories. The MuRIL model achieves 92.06 percent accuracy. The overall Machine Learning (ML) pipeline is encompassed in a web application whereby classified and transcribed videos can be searched on the basis of their content. All the implemented code and related files are available at: *https://github.com/akppprojects/asrtcnl*.

Keywords : Automatic Speech Recognition, Deep Learning, Natural Language Processing, Speech Processing, Text Classification, Transformer

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1D 1 Dimensional. 33

AI Artificial Intelligence. 1, 2, 9, 18, 38

ANN Artificial Neural Networks. vii, 1, 9, 10

API Application Programming Interface. 29, 38

ASR Automatic Speech Recognition. v, vii, ix, 1–4, 6, 22, 23, 28–30, 33, 34, 38–44, 46, 48, 52, 60–63

BBC British Broadcasting Corporation. 7

BERT Bidirectional Encoder Representations from Transformers. v, 7, 12, 20, 34

CER Character Error Rate. 45, 47, 49–52

CLSRIL-23 Cross Lingual Speech Representations for Indic Languages. v, viii, 7, 33, 34, 46, 48, 49, 51, 52, 60–62

CNN Convolutional Neural Networks. vii, 6, 7, 9, 10, 12, 40, 47

CPU Central Processing Unit. 22

CTC Connectionist Temporal Classification. vii, xii, 6, 14–17, 30, 37, 41, 61

CUDA Compute Unified Device Architecture. 38

DL Deep Learning. vii, 1, 3, 6, 9, 12

GAN Generative Adversarial Network. 7

GELU Gaussian Error Linear Unit. 33

GMM Gaussian Mixture Model. 6

GPT Generative Pretrained Transformer. 12

GPU Graphics Processing Unit. 22, 38, 46, 47, 62

GRU Gated Recurrent Unit. 6

HMM Hidden Markov Model. 1, 6

HTTP HyperText Transfer Protocol. 29

IDE Integrated Development Environment. 22

JS JavaScript. 29

KHz Kilo Hertz. 32, 33, 37, 39

LM Language Model. vii, xiv, 18, 20, 42, 48, 50–54, 57, 76

LPC Linear Predictive Coding. 6

LSTM Long Short-Term Memory. 3, 6, 7, 20

M Million. 34

MFCC Mel Frequency Cepstral Co-efficients. 6

ML Machine Learning. v, vii, 1, 3, 5, 6, 9, 11, 19, 30, 50

MLM Masked Language Modelling. 34

ms millisecond. 33

MuRIL Multilingual Representations for Indian Languages. v, ix, xii, 7, 8, 29, 30, 34, 35, 43, 61, 62

NLP Natural Language Processing. vii, 1–4, 6, 7, 12, 18–20

NoSQL Not only Structured Query Language. 22

OpenSLR Open Speech and Language Resources. v, ix, xii, xiv, 35–37, 42, 46, 48, 50–52

PC Personal Computer. 22

REST Representational State Transfer. 29, 38

RNN Recurrent Neural Networks. vii, 1, 3, 6, 7, 11, 20

SQL Structured Query Language. 22

SVM Support vector machines. 19

TF-IDF Term Frequency - Inverse Document Frequency. 7, 19

TLM Transistional Language Modelling. 34

UI User Interface. 38

UML Unified Modelling Language. viii, 25

URL Uniform Resource Locator. 21, 28–30

WER Word Error Rate. xiv, 33, 44, 47–52, 75

1. INTRODUCTION

The amount audio and video content uploaded on the internet is proliferating day by day. With that, it has become more difficult for people to filter genuine content for their consumption. This has become a very common problem in Nepali online media portals. Misleading titles, click-baits in audio/video contents have become unavoidable in social media and streaming sites. This demands for techniques to analyze and classify audio contents automatically to aid content filtering. There have been efforts to develop audio classification systems mostly using aspects other than speech content, like music, environmental noise, etc. But there doesn't seem to have been any notable work to develop an end to end system for speech content classification. So for now, the most appropriate approach for this seems to be conversion of speech to text using Automatic Speech Recognition (ASR), followed by text classification using Natural Language Processing (NLP) techniques.

ASR is one of the most important applications of speech processing where human audio data or speech is converted to it's corresponding text, which can further be applied to tasks like speech to text translation, virtual assistant, voice recognition, speech emotion recognition, speech content classification, etc. In general, processing and analysis of human speech has always been one of the major goals of computer language processing. The rapid development of Artificial Intelligence (AI) has also aided a lot to the development of speech processing technologies. Various techniques ranging from Hidden Markov Model (HMM), Artificial Neural Networks (ANN), Recurrent Neural Networks (RNN), etc. to transformers today, have been widely used for speech processing. In comparison to traditional machine learning techniques where feature engineering and extraction require a lot of manual effort, the use of Deep Learning (DL) has led to development of models that learn complex features automatically and has thus provided an effective approach for speech processing.

As of now, the most typical approach to solve the task of ASR using Machine Learning (ML) has been supervised approach, which requires a large number of transcribed text corresponding to the speech data. The amount of labelled data the supervised algorithms require are usually a bottleneck, especially for low resource Languages like Nepali. Unsupervised learning algorithm that need no labelled data and self supervised learning algorithms that need very less amount of labelled data to solve ML problems existed from really long ago. But the adaptation of those to DL started more recently as the development of DL started taking pace with the availability of enough computing power and large amount of data. So the introduction of unsupervised and semi supervised DL based approaches to solve ASR dates only a few years back. These approaches work well with very zero or very less amount

of transcribed data. Since in this age of internet, very large amount of unlabelled data can be found on the web, low resource languages like Nepali can benefit a lot from the use of unsupervised or semi supervised techniques for ASR.

In the case of Nepali language, due to the lack of transcribed data, we have not seen very remarkable results in speech processing tasks like speech recognition on adapting the same supervised techniques that have worked far better for English and western languages with huge amount of transcribed data. On the other hand, Nepali language has quite a complex morphology, difficult to represent and analyze even for humans, let alone computers. So another advantage of using unsupervised and semi supervised techniques could be the freedom that these techniques add to the models in learning the intermediate representations of speech.

Recently developed unsupervised and semi supervised speech processing techniques have proven to show far better results with very less amount of transcribed data in English and other languages as well. As we have been stressing to it, the adaptation of these techniques to Nepali is much easier with the context of data collection. So, the use of unsupervised approaches can lead to new developments in the field of AI and speech understanding for the Nepali language. While it is usually very difficult to collect transcribed text for Nepali speech, unlabelled, raw audio samples are easily available all over the Internet as well as various other sources like radio and television archives. This is one of the main reasons to follow an unsupervised approach for learning Nepali speech representations. To be specific, unsupervised and semi supervised techniques have shown some excellent results in the task of speech recognition in other languages recently.

1.1. Project Background

Speech recognition can be applied to a wide range of applications. The transcriptions generated by ASR models can be used in many interesting applications with the use of NLP techniques. This project focuses on the use of Nepali speech transcriptions of audio and video contents from ASR for content classification. We generally see videos and audios in the Internet and social media being classified based on their titles. Most of the time the classification tags for the videos/audios are given manually. However, best way to tag any sort of audios or videos would probably be to use the audio content it contains, not the title or any other attributes associated with it. Thus, in our project we have tried to build a web application that can generate transcriptions for video and the categories that best match the video based on the audio content. We have also tried to build an interface that can be used as a audio/video content portal with transcriptions of audio/videos displayed according to their classification tags and visualization of statistics of those content.

1.2. Motivation

Compared to other languages like English, there hasn't been much research and development in Nepali speech and language systems. At present, Nepali language comes as a low resource language due to the lack of efforts and contribution in collection of data and other resources for Nepali language.
For these reasons, unsupervised and semi supervised methodologies that require very less data can be really beneficial for Nepali and deserves some research focus and attention. This would help us build better ASR systems and incorporate Nepali Language Understanding in modern intelligent systems.

1.3. Objectives

The main objectives of the project can be summarized in the following points:

- To experiment with unsupervised/semi-supervised techniques of speech representation learning in Nepali language.
- To use and evaluate the unsupervised representations obtained by pretraining the unsupervised learning models in the downstream task of ASR.
- To explore NLP techniques for Nepali content classification.
- To develop an interactive web interface to demonstrate the components above as a complete pipeline.

1.4. Problem Statement

ASR is an interesting area of research in the application of ML these days. It is due to the fact that it has applications in different areas like speech-to-text conversion. With the development of new technologies and growing amount of data, different ML and DL models like RNN and Long Short-Term Memory (LSTM) have come as frameworks for developing ASR applications. These frameworks mostly provide a way to develop models whose parameters can be learned by providing sufficient amount of labelled data. These models worked well with languages like English, with a lot of researchers and contributors working on it for a long time to make large amount of labelled data available. For low resource languages like Nepali, it has always been a challenge since there hasn't been enough work to collect labelled

data for these data hungry models. On the other hand, the use of semi-supervised methods allow learning representations from unlabelled data which can then be trained to map speech to corresponding transcripts using very less amount of labelled data. This provides a more feasible method, in terms of availability of data to develop ASR system for low resource languages like Nepali. We also aim to use the output of ASR model to classify the audio into a set of suitable categories. This facilitates classifying a video/audio based on it's actual content rather than other less informative attributes like title.

1.5. Scope of the project

The main idea of the project is to train an unsupervised model to learn representations of Nepali speech and using those representations to obtain audio transcripts. Finally we'll use NLP techniques to classify the content into appropriate categories.

The final product of this project shall be a web interface which can give transcriptions and classification tags corresponding to a given audio and also a portal with audio/video content statistics. We do not aim to develop a product ready for release but one to demonstrate the prospects of using unsupervised learning for extracting speech representations for the Nepali language to aid downstream tasks like speech recognition which is usually constrained by unavailability of labelled data resources in supervised learning. This project intends to contribute for development in the field of Nepali speech representations and recognition, overcoming resource limitations in terms of data.

1.6. Organization of the Report

The report has been organized in following chapters:

- Chapter 1: It includes the introduction and background of our project, our motivation and objectives in doing the project and the scope it covers.
- Chapter 2: It includes review of existing literature and related works that have notable contribution to the problem we are solving.
- Chapter 3: It includes some theoretical background and links to other resources to help readers understand our work.
- Chapter 4: It includes the design specifications and feasibility assessment our our system.

- Chapter 5: It includes the methodology, details about application development and ML model development, experimental setups we followed to develop the project.
- Chapter 6: It includes the results we obtained by our experiment and corresponding analysis.
- Chapter 7: It includes the example outputs of our web application
- Chapter 8: It includes discussions regarding our views on potential causes and sources of errors in the project.
- Chapter 9: It includes conclusion of the entire project.
- Chapter 10: It includes the limitations in our project and some ideas for future enhancements.

2. LITERATURE REVIEW

2.1. Automatic Speech Recognition (ASR)

The solution to the problem of ASR seems to have started from the time of Graham Bell when he tried to convert sound waves into electric impulses. With time, various probabilistic models like Gaussian Mixture Model (GMM) in [1, 2] and HMM in [3, 4] started being used for ASR. Traditional ML algorithms required speech features to be hand crafted using techniques like Mel Frequency Cepstral Co-efficients (MFCC) in [5], Linear Predictive Coding (LPC) in [6], etc. With the advent of DL, fully connected feed forward neural network, Convolutional Neural Networks (CNN) started being used as deep feature extractors for ASR systems as in [7]. More recently, more powerful models based on recurrent networks like RNN[8] were developed. For training of these systems, the difference in input and output alignment requires an optimal algorithm that would correctly incorporate all alignments corresponding to a target output. For the purpose, Connectionist Temporal Classification (CTC)[9], which can automatically map input and output alignments during training using dynamic programming algorithms was developed. Incorporating attention mechanism to the RNN encoder-decoder showed great results in phoneme recognition.[10]

Following above ideas and techniques proven for English and other languages, there have been some efforts to develop automatic speech recognition system for Nepali as well. [11] developed a model that worked by maintaining a database to store patterns of amplitude of speech at a particular frequency for a particular user and using it to identify syllable and letter of user's speech at a certain frequency . [12] used HMM, creating Markov chains of training words as sequences of features obtained by MFCC to perform word level recognition of Nepali speech . [13] were one of the first to use DL for the Nepali ASR. They used Bidirectional RNN with LSTM cells and CTC training for character level speech recognition and then [14] used a similar CNN, Gated Recurrent Unit (GRU) and CTC based model. But due to the lack of huge amount of labelled dataset, these methods have not given very impressive results for Nepali Language.

While most of the approaches mentioned above are supervised ML algorithms, that require large amount of labelled dataset, in order to address the problem of unavailability of huge amount of labelled data sets especially in low resource languages, unsupervised approaches to learn speech representations started being developed. Inspired by the idea of developing fixed length representations for words in a text to use in NLP techniques in [15], Audio2Vec[16] proposed a way to learn audio embedding with the help of sequence-to-sequence

auto-encoder. The idea was further extended to use Generative Adversarial Network (GAN) for phoneme recognition in [17] and speech recognition in [17] tasks. Wav2vec[18] has two CNNs stacked on top of each other, to learn representations through unsupervised pretraining before finetuning the model for a downstream task paved pathways for other works on wav2vec series. Quantization of those learned representation to discrete representations using gumbel softmax or k-means clustering provided even better results in [19]. Wav2vec2.0[20] showed that learning powerful representations from speech audio alone followed by finetuning on transcribed speech can outperform other methods. It extended the preceding ideas by [18] and [20] to develop a model with a CNN feature extractor that outputs latent speech representations, stacked with a transformer encoder to output contextualized speech representations. The model was trained using a contrastive task of identifying quantized speech representations corresponding to latent speech representations with masked input.

Sharing of low level speech representations across multiple languages can help low resource languages use models pre-trained on other similar language as well as multilingual corpus to learn cross-lingual representations. [21] show that such representations have shown very effective results in the downstream tasks like speech recognition. This idea has also inspired Cross Lingual Speech Representations for Indic Languages (CLSRIL-23)[22], which is a model pretrained on top of wav2vec2[20], using 23 Indic languages, including Nepali. The use cross-lingual representations from such a model can thus be extremely useful for low resource languages like Nepali as it is closely related with many other Indic languages.

2.2. Text Classification

A state-of-the-art approach to classify speech into a set of categories involves getting transcriptions from the speech and using a set of NLP techniques on the obtained transcriptions. In order to assign tags to British Broadcasting Corporation (BBC)'s radio programmer's audio, [23] used the HUB4 acoustic model and a language model extracted from the Gigaword corpus5 to get the transcripts from the speech audio and later candidate terms were mapped to a set of identifiers on the basis of Term Frequency - Inverse Document Frequency (TF-IDF) score. The use of LSTM [24], an RNN model has been widely used for NLP applications like text classification. Recently, The introduction and development of transformer based models has introduced many powerful models for NLP tasks. Bidirectional Encoder Representations from Transformers (BERT) [25] is a transformer [26] model which has been massively adopted as a base model to get the representations of the input text which is then applied to downstream tasks. Multilingual Representations for Indian Languages (MuRIL) [27] is a model based on BERT architecture which has been trained on large corpus of Indic languages and has been used for language related downstream tasks in Indic languages. [28] used the

text representations from MuRIL model for offensive language identification in Dravidian languages .

3. THEORY

3.1. Machine Learning (ML)

The term machine learning was first coined by Arthur Samuel in 1959 describing two procedures which learnt to play a better game of checkers [29]. While the definition of the term itself has undergone slight changes, machine learning today is the science of getting computers to perform a task without explicitly programming them to do so. Since it's inception, ML has widely been regarded as a sub-field of AI. Today, some assert that ML is a separate field which overlaps with AI because it borrows models from statistics and probability theory.

While ML itself remains a broad spectrum, the core algorithms are usually fed with passive observations. ML algorithms rummage these passive observations to search for patterns in similar data. The vast sphere of Machine learning can be categorized, depending upon the type of data used, upon the feature extraction process and also on the statistical models used. DL(3.2) is a sub-field of machine learning where the features need not be handcrafted. While deep learning is a subset of machine learning, ML algorithms sometimes refer to the subset other than deep learning.

3.2. Deep Learning (DL)

DL is the subset of machine learning that uses multiple layers to extract higher-level and complex features from the raw input. The traditional machine learning algorithms require hand engineering to extract features from the data. However, deep learning uses a combination of various techniques to automatically learn features and complex patterns from the data. DL has been massively applied in the areas related to image processing, computer vision, natural language processing, speech processing, etc.

Neural networks are the fundamental building blocks of deep learning which are composed of layers of neurons each of which learns a certain feature from the data. Each neuron is associated with a series of weights attached to it. As shown in figure 3.1, a neuron computes the weighted sum of inputs and applies a non-linear activation function to it.

There are wide varieties of neural networks used in real world applications. In this section, we'll explain the two most widely used classes of neural networks, ANN and CNN.

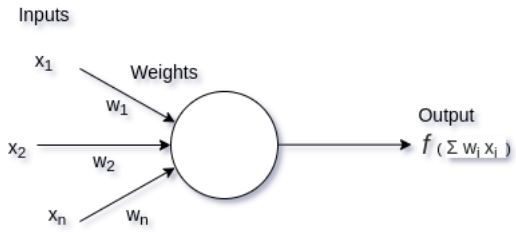

Figure 3.1: A neuron

3.2.1. Artificial Neural Networks (ANN)

Artificial Neural Networks (ANN)s are also called feed-forward neural networks which consists of multiple hidden layers in between the input and the output layer. These hidden layers learn extensive features from the raw input supplied to the network. Basically, each layer in ANN learns certain weights via the method of backpropagation.

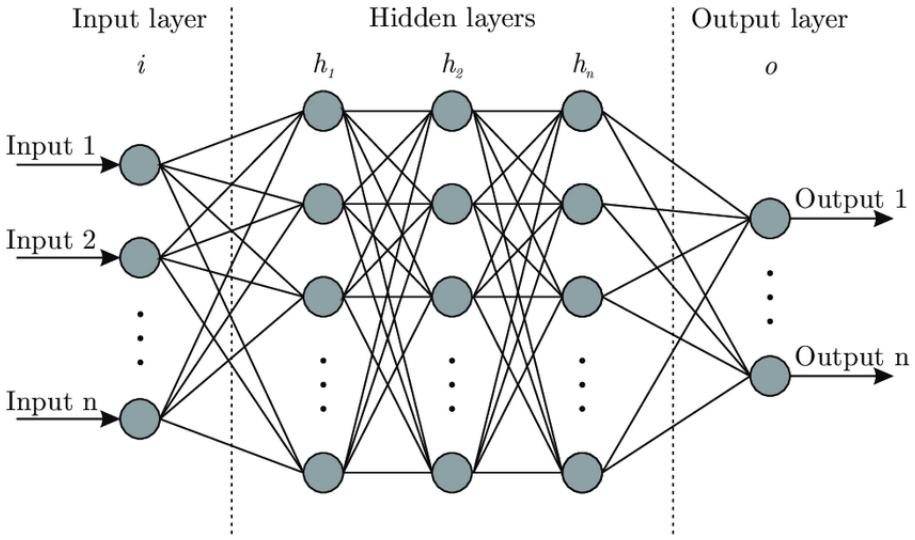

Figure 3.2: Architecture of Artificial neural networks

3.2.2. Convolutional Neural Networks (CNN)

CNNs is a class of neural networks applied mostly for the tasks of images and video processing. The building blocks of CNNs are filters which are used to extract features from the input using the convolution operation.

A typical CNN Architecture consists of a combination of these layers:

3.2.2.1 Convolutional Layer A filter matrix is convolved with input to detect certain patterns. For an image, the beginning convolution layers are responsible for capturing the low-Level features such as edges, color, gradient orientation, etc. As we proceed to the further layers, the architecture adapts to the high-level features as well, giving us a network that has a wholesome understanding of the input.

3.2.2.2 Pooling Layer The pooling layer is responsible for reducing the spatial size of the convolved feature. Max Pooling returns the maximum value from the portion of the input covered by the kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the input covered by the Kernel.

3.2.2.3 Fully Connected Layer Adding a Fully-Connected layer is usually a cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

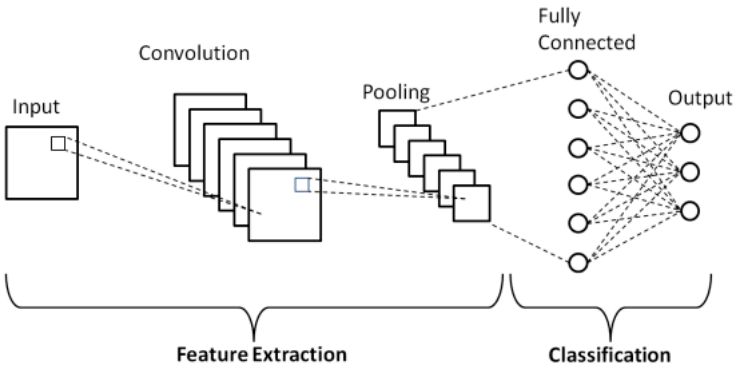

Figure 3.3: Architecture of Convolution neural network

Source: https://www.researchgate.net/figure/
Schematic-diagram-of-a-basic-convolutional-neural-network-CNN-architecture-26_fig1_336805909

3.3. Recurrent Neural Networks (RNN)

In ML, Representation Learning refers to the process of learning representations of input data in the process of transforming it to the output or by extracting some latent features from it. There are different ways in which representations are learnt in different types of ML Algorithms.

Probabilistic Models like Bayesian Classifiers learn representations that describe distribution

of the features followed by the input, that can be used for prediction tasks. On the other hand, in Deep Neural Networks, representations are obtained as a result of a series of linear and non-linear transformations the input data undergoes, through different layers in a hierarchical pattern with the goal of yielding representations that can be useful in some downstream task like classification, prediction, etc. Thus in DL, the depths in the network is one of the agent that aids in learning representations, be it supervised algorithms or unsupervised.

Through all these years, supervised learning has been the most successful approach for training deep neural networks. In the domain of Computer Vision, huge amount of labelled dataset such as the ImageNet, has helped training of deep neural network models and learn representations that not only solve the primary task of say image classification in this case, but also to be used as a good starting point for other tasks like object detection, facial recognition, etc. This concept of transfer learning has been used successfully in many different domains, other than Computer Vision as well.

So far, we have been only looked at how deep neural networks can learn generalize representations by solving supervised tasks. In such methods, we rely on having a lot of annotated data to learn the representations for at least the primary task.

However, supervised learning might not always be the most appropriate technique to learn representations and share them for multiple tasks. Unsupervised representation learning attempts to learn very useful features from unlabeled data that can be used to improve the performance of many downstream tasks and reduce the necessity of labelled data. Autoencoders used in Computer Vision, models like Generative Pretrained Transformer (Generative Pretrained Transformer (GPT)) and BERT [25] in NLP leverage unsupervised representation learning to tackle learning problem in respective domains. BERT is pre-trained on self-supervised learning task that does not require manually annotated data and its parameters can be finetuned to tackle downstream language tasks. In very recent years, there has been a significant role of such unsupervised representation learning in the domain of Speech as well. With the advent of such approaches in [18] and its successors, who follow architecture and training analogous to BERT in NLP, Speech related tasks for Low Resource Languages have been much easier to solve in terms of labeled data collection, with great results as well. For more detailed overview, see [30].

3.4. Transformer model

Before transformers, the dominant sequence transduction models were based on complex recurrent and CNN(3.2.2) that employed the encoder-decoder architecture. Those models

which fared better also employed an encoder-decoder architecture connected through attention mechanism. But the symbol positions of the input and output sequences is typically taken into account while factoring computation in recurrent models. They align the positions to steps which in turn precludes parallelization within training examples. Transformers deliberately abstain from recurrence and base solely on attention mechanisms to extract and use information from arbitrarily large contexts without passing it through intermediate recurrent connections.[26].

3.4.1. Self-attention

Attention-based approach compares an item of interest to a collection of other items in a way that reveals their relevance in the current context. The mechanism of performing a set of comparisons made to other elements within the same sequence is called self-attention. The attention process is better understood through, *query*, *key* and *value* abstractions.

- When the input embedding is a *query*, it serves as the current focus of attention as it is compared to all other preceding inputs.
- When the input embedding is a *key*, it is compared to the current focus of attention as a preceding input.
- When the input embedding plays its role as a *value*, it is used to compute the output for the current focus of attention.

The weight matrices W^Q, W^K and W^V are used to project each input vector x_i into a representation of its role as query, key, or value. To take advantage of parallelization, input embeddings of N tokens of input sequence are packed into a single matrix $X \in R^{N \times d}$. X is then multiplied by the key, query, and value matrices producing matrices $Q \in R^{Nd}$, $K \in R^{Nd}$, and $V \in R^{Nd}$ containing all the key, query, and value vectors.

$$Q = XW^Q; K = XW^K; V = XW^V \tag{3.1}$$

The self-attention scores can be calculated as the dot product between query vector and the preceding element's key vector in the form of requisite matrix. Since, the dot product may be arbitrarily large, the result of the dot product is divided by the square root of dimensionality of query and key vector, d_k.

$$SelfAttention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.2}$$

3.4.2. Layer normalization

The residual connection from the input to the encoder block and the self attention score is fed to the normalization layer. Layer norm is a slight variation to the z-score, applied to a single hidden layer. The mean μ and standard deviation σ is calculated over the vector to be normalized. For a hidden layer with dimensionality d_h,

$$\mu = \frac{1}{d_h}\sum_{i=1}^{d_h} x_i \tag{3.3}$$

$$\sigma = \sqrt{\frac{1}{d_h}\sum_{i=1}^{d_h}(x_i-\mu)^2} \tag{3.4}$$

On calculation of these values, the vector components are normalized.

$$\hat{x} = \frac{(x-\mu)}{\sigma} \tag{3.5}$$

The final equation constitute of two learnable parameters, gain(γ) and offset(β).

$$LayerNorm = \gamma\hat{x} + \beta \tag{3.6}$$

3.5. Connectionist Temporal Classification (CTC)

Connectionist Temporal Classification(CTC) is a classic methodology for sequence tasks like handwriting and speech recognition. CTC comes in handy for tasks where the alignment between input and output is not known. Let us suppose, for a speech recognition task there are input sequences $X = [x_1, x_2, ..., x_T]$ which represents audio and output sequences $Y = [y_1, y_2, ..., y_U]$ representing transcripts corresponding to the audio.

It's complicated to use normal supervised algorithms for this task because of the following

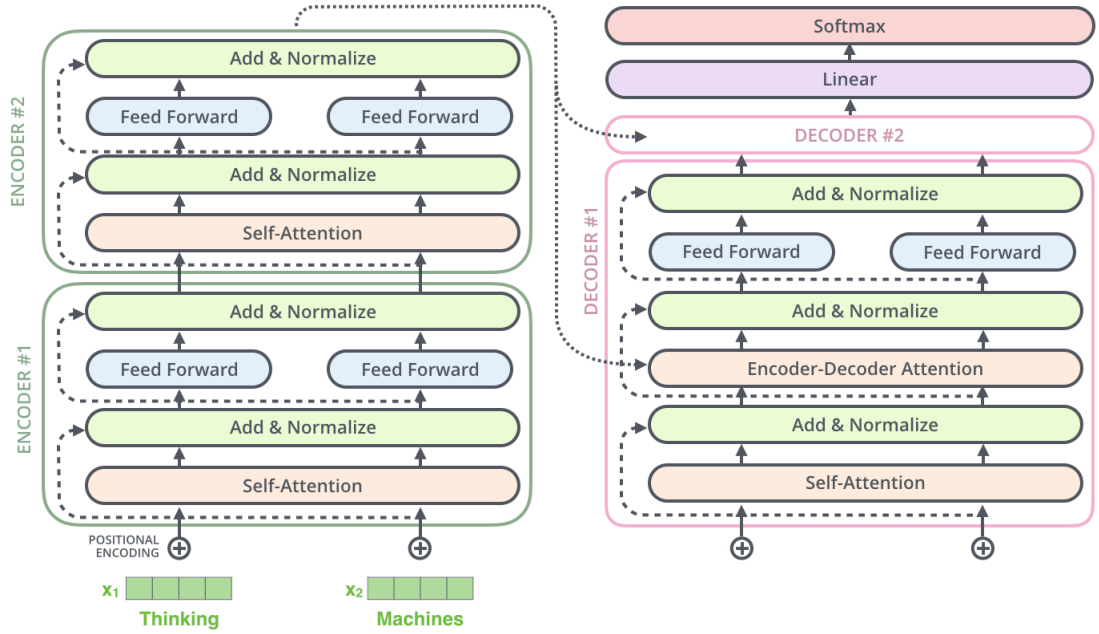

Figure 3.4: Simplified visualization of transformers with two stacked encoder and decoder

Source: https://jalammar.github.io/images/t/transformer_resideual_layer_norm_3.png

reasons:

1. X and Y can vary in length.
2. The ratio of lengths of X and Y vary from data to data.
3. Due to the reasons above, it's difficult to align an element of X to an element of Y.

CTC overcomes all the above challenges as it gives a probability distribution for all possible alignments for a given output Y. CTC works by outputting a single character for every frame of the input resulting in an output of length equal to the length of the input. It later uses a collapsing function to shorten the output length resulting in a shorter sequence.

The process of mapping a sequence of audio into text can be represented by the following diagram:

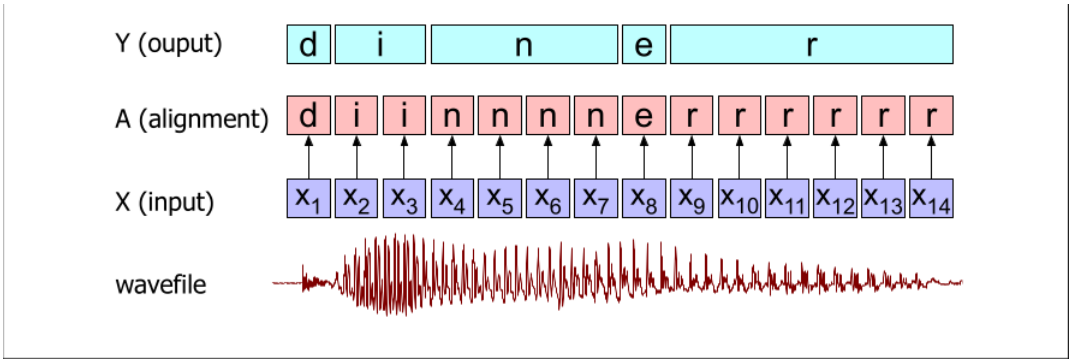

Figure 3.5: Visual representation of CTC algorithm

Source: https://web.stanford.edu/~jurafsky/slp3/26.pdf

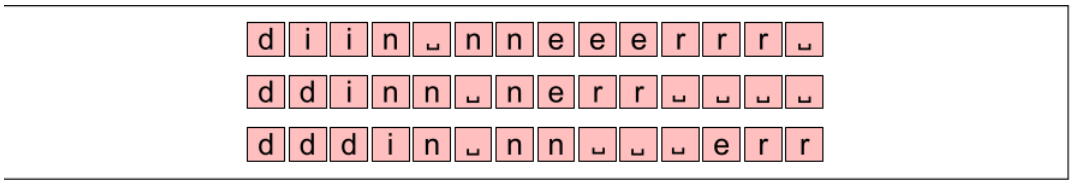

Figure 3.6: Valid alignments for the word "dinner"

Source: https://web.stanford.edu/~jurafsky/slp3/26.pdf

3.5.1. Alignment

As shown in the above diagram, an alignment is an intermediary step in CTC which represents the sequence of letters corresponding to each input frame. As shown above, the alignments have an additional token ε. This is a special token for a blank introduced by CTC when we don't want to include anything in the transcript for that corresponding input frame. ε is generally used in two cases:

- To represent the silence in the input audio frame.
- To represent a gap between two output labels with same value. Without ε, the word dinner would correspond to diner in the above example.

An important property of alignment is that an output element can have many alignments but not vice-versa. Fig. 3.6 shows some other valid alignments for the word "dinner".

3.5.2. Decoding

A typical output of a network trained using any CTC algorithm looks like the figure 3.7.

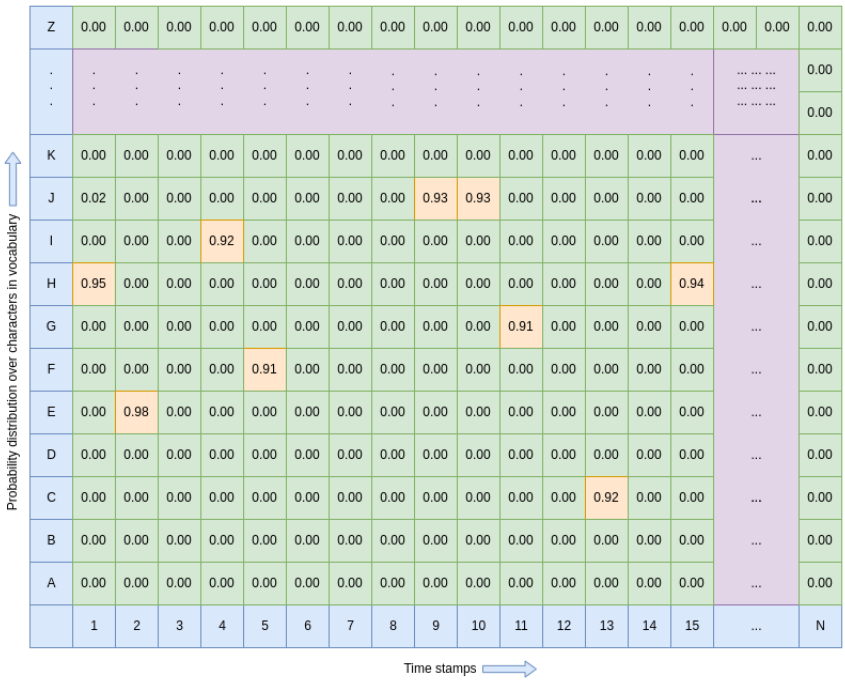

Figure 3.7: An output of a CTC Network

Here, we can see that for each time step in the output, the network gives probability distribution over all the characters in the vocabulary. Hence this output represents only an alignment with repeated characters and blank tokens included, and thus needs to be collapsed in order to give the final output.

But there can be various approaches to decoding the output to produce appropriate result as it is clear that multiple alignments can collapse to the same output. Before we see how a final output is assigned probability scores, let us first see how a particular alignment is assigned a probability score.

For an alignment $A = \{a_1, a_2, a_3, ..., a_n\}$ corresponding to an input X, CTC assumes the output at all time steps, t is independent of any other time step. So, probability of alignment A is thus given as:

$$P_{CTC}(A|X) = \prod_{t=1}^{T} p(a_t|X) \tag{3.7}$$

With the greedy approach, we can thus find the best alignment by choosing character with maximum probability at each time step and then collapse to a single output.

But with this approach, we may not obtain the best output possible as the most likely align-

ment, A need not correspond to the most likely output Y. For example: let the alignments [a, a, ε] and [a, a, a] individually have lower probability than [b, b, b]. But the sum of their probabilities is actually greater than that of [b, b, b]. The greedy approach of decoding will decode to Y=b as the most likely hypothesis, while it should have chosen Y= [a]. The algorithm needs to consider the fact that [a, a, a] and [a, a, ε] collapse to the same output.

So the most probable output Y should be the one with the highest sum of probabilities corresponding to all the alignments A that can map to Y.

$$Y^* = argmax_Y P_{CTC}(Y|X) \tag{3.8}$$

Naively summing over all the alignments though is very expensive. So this sum is computed by using a version of Viterbi beam search that stores in the beam the high-probability alignments that map to the same output and sums those.

In speech recognition and some other problems, including a language model over the outputs significantly improves accuracy. So we can include the language model as a factor in the inference problem. For that, the score given to an output also incorporates language model score (LM(Y)) whenever a word is inserted. The score is given as:

$$Y^* = argmax_Y P(Y|X) * LM(Y) * \alpha \tag{3.9}$$

Here, α is the Language Model (LM) score which is a hyperparameter.

3.6. Natural Language Processing (NLP)

Natural language processing is the branch of AI that enables computers understand and interpret human language. NLP aims to connect the areas of linguistics, computer science and AI together. The techniques of NLP allow computers to process language in the form of text or speech, understand the semantics of language and perform further analysis.

Some of the common and widely used tasks of NLP are language translation, named entity recognition, sentiment analysis, natural language generation, text summarization, text classification,etc.

3.6.1. Text classification

Text classification is an NLP technique that aims to assign a set of categories or tags to a set of texts. Assigning categories to documents like news articles, books, web pages etc are real world applications which are difficult and time consuming to perform manually. A text classifier takes input, a sentence or a group of sentences and automatically assigns a suitable category or a set of relevant tags to it. There are generally three broader approaches to text classification which are explained follows.

3.6.1.1 Rule-based classifier Rule-based approach aims to classify text to a certain category using a set of manually designed linguistic rules.

As an example, if we want to classify news articles into two groups: Sports and Economy, we need to define a certain set of rules beforehand. In this case, we can define lists of words that define each category such as “cricket”, “football”, “Ronaldo” for Sports and “stock-market”, “trade”, “monetization” for Economy. Now, whenever we want to classify a new text, we need to count the number of words in the text belonging to each category and assign the text to the category with the maximum score.

A disadvantage of this approach is that it’s extremely difficult and impractical to create a list of words that can generalize to a large set of sentences belonging to that particular category.

3.6.1.2 Machine Learning based classifier Machine learning can be used to classify a text into a certain category which requires a labelled dataset in order to build a classification model. A certain number of data points pre-labelled into suitable categories must be trained using an appropriate machine learning algorithm which can later be used to classify new sentences.

The first step in building a machine learning based text classifier is to extract features from the input texts and represent them in the form of vectors. A classic algorithm used for feature extraction in texts is TF-IDF which is a statistical measure to identify the importance of each word in a set of documents.

Finally, we can use one of the ML algorithms like Naive Bayes, Support vector machines (Support vector machines (SVM)), etc. as per our requirement.

3.6.1.3 Deep Learning based classifier

Different versions of neural networks are widely used for NLP and eventually for text classification. The advantage of using deep learning is that neural networks automatically learn the features from the data so it eliminates the necessity of hand-engineering the features as in machine learning. RNNs are widely used for sequence data and have been applied to various tasks of Natural language processing. LSTM, a type of is are very popular for NLP tasks like text classification . LSTM has multiple hidden layers and it performs well in memorizing the important relevant terms in the input text thus can be very useful for text classification.

An alternative and a better approach over RNNs is the use of transformer models for text classification. The parallelization ability in transformers make it extremely fast and effective over LSTMs. Besides, transformers use self-attention mechanism to better understand the semantics of the input text, so they produce good results in text classification. We can start with transformer models like BERT [25], initialize classification layers on top of it as per our requirement and finetune the model on our dataset. This simple step can be very effective for the purpose of text classification as well as other NLP tasks.

3.7. Language Model (LM)

Language models assign probability to a sequence of words. The assignment of these probabilities may be to a word or an entire sentence. Simpler n-gram assume that the probability of an occurrence of a word is based on the previous *n* words. As a language model is trained on larger texts, the number of unique words increase, and with it, the number of possible sequence. This means that a distributive, non-linear approach to language models is better suited for large corpus which gave rise to feed-forward and recurrent neural networks being used for this purpose. Language models are used for a variety of tasks ranging from semantic analysis to paraphrasing. On speech recognition systems, where noisy ambiguous input is a norm rather than an exception, it helps identify correct words through probabilities with context. For a speech recognition system, it is very difficult to distinguish तीनवटा घर छ मेरो and तीनवटा घर चमेरो . Language models can be useful in this case because quite clearly, the former has a higher probability of occurrence than the latter.

4. SYSTEM DESIGN

4.1. Requirement Specification

4.1.1. Functional requirements

Functional requirements are the functions that must be present in the designed system. It can be viewed from system side itself as well as from the user's side.

System side

1. The system should be able to take audio as input and provide corresponding text as output.
2. The system should be able to take audio as input and provide suitable categories to which it belongs as output.
3. The system should be able to take audio/video as input in multiple audio formats.
4. The system should be able to take Uniform Resource Locator (URL) of audio as input.

User's side

1. User should be able to get corresponding transcriptions of given audio.
2. User should be able to get the categories to which given audio belong.
3. User should be able to access the system from different regions provided that connection exists.

4.1.2. Non-functional requirements

Non-functional requirements are the requirements the specify the quality requirements of the system. Non-functional requirements of the system can be viewed under following topics.

1. **Performance**: It defines how fast the system responds to the user's actions.

2. **Availability**: The system should be available to users when they require it.
3. **Fault tolerant**: The system should be fault tolerant and should be prepared for crash recovery.
4. **Scalability**: With increasing users and data, system should have capacity to scale as needed.
5. **User friendly**: The system should be easy to use and interface should be as simple as possible.

4.2. Hardware and Software Requirements

4.2.1. Hardware Requirements

- **Server**: A server is required for training models for ASR and text classification. Central Processing Unit (CPU) with 16 GB ram and Graphics Processing Unit (GPU) of 8 GB or more is good enough for training.
- **Personal Computer (PC)s**: The system needs PC with web browser to run web interface. The PCs should have capabilities to ssh to remote servers.
- **Audio recorder**: Audio recorder can be used to collect audios which can be required for testing purposes.

4.2.2. Software Requirements

- **Operating System**: The models as well as the application will need an Ubuntu OS or any other Linux distribution.
- **Database System**: Any Structured Query Language (SQL) database to store audio data but the database can be implemented well with suitable Not only Structured Query Language (NoSQL) database to make the system capable of handling storage of large amount of data if needed.
- **Web Browser**: The application will be supported by any web browser.
- **Other utilities**: Programming language compilers and interpreters, git, suitable Integrated Development Environment (IDE) for editing, ssh or similar remote connection protocol.

4.2.3. Other Requirements

- Internet connection

4.3. Feasibility Assessment

Our project is concerned with building the system for ASR and audio classification for Nepali language, thus includes multiple subsystems. Feasibility is one of the factor that needs to be studied before starting on working any projects. It helps to shape the work flow and provides proper direction to achieve goals of the project. We assessed different types of feasibility before deciding to work on the project on the basis of output obtained from feasibility study. Various fields of feasibility assessment that we conducted are described below.

4.3.1. Technical Feasibility

Technical feasibility helps to know whether the system proposed can be designed and completed from technical point of view. Proper study of technical feasibility helps to know about the best selection that can be done for the project.

The technological components that are required for the development and operation of system were found be be manageable and easily available. The tools and frameworks for the development of system we proposed are open source components which makes the project technically feasible and even offers multiple choice for development as planned.

4.3.2. Operational Feasibility

Operational feasibility helps to know about the operating environment and it's resource requirements. It is concerned with whether or not the project will be operable after its completion.

The system designed should be used to obtain transcriptions and classify the audio based on transcription into one or more categories. Since the website will be running on the web it is necessary to handle multiple users request simultaneously. Due to the limitation in hardware we can adjust it to serve one request at a time by maintaining queue. It can be scaled accordingly if required.

4.3.3. Economic Feasibility

Economic feasibility helps to know whether the cost required to build the system is feasible or not. Unnecessary costs should be avoided and optimal cost should be allocated for the system design and development.

The system requires hardware for training a model. After training, model is loaded into the system and the system is hosted in a server. Both seem economically feasible to us. Though not the best, we have those resources to keep us going.

4.3.4. Legal Feasibility

Legal feasibility deals with whether the system lies within the legal boundary defined by the governing bodies. It must consider and follow things such as copyright policies and license requirements.

The components(data and models) used in system are either developed by ourselves or they are open sourced components with license that allows to use their work giving proper attribution.

4.3.5. Scheduling Feasibility

Scheduling feasibility helps to know whether the time period available is enough to complete the system.

The scheduling of system design and development was designed before starting the project to complete the project within the total available time.

4.4. Unified Modelling Language (UML) Diagrams

4.4.1. Use case diagram

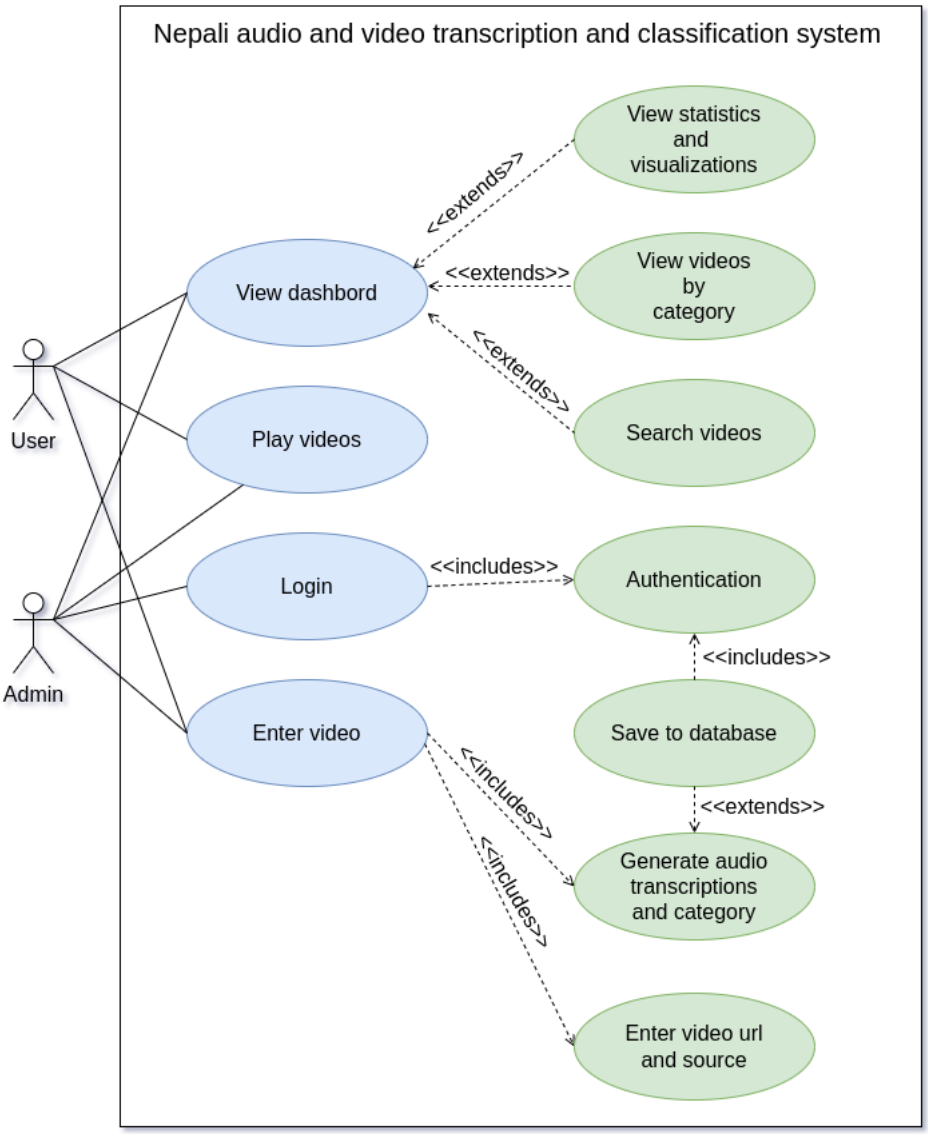

Figure 4.1: Use case diagram

4.4.2. Sequence diagram

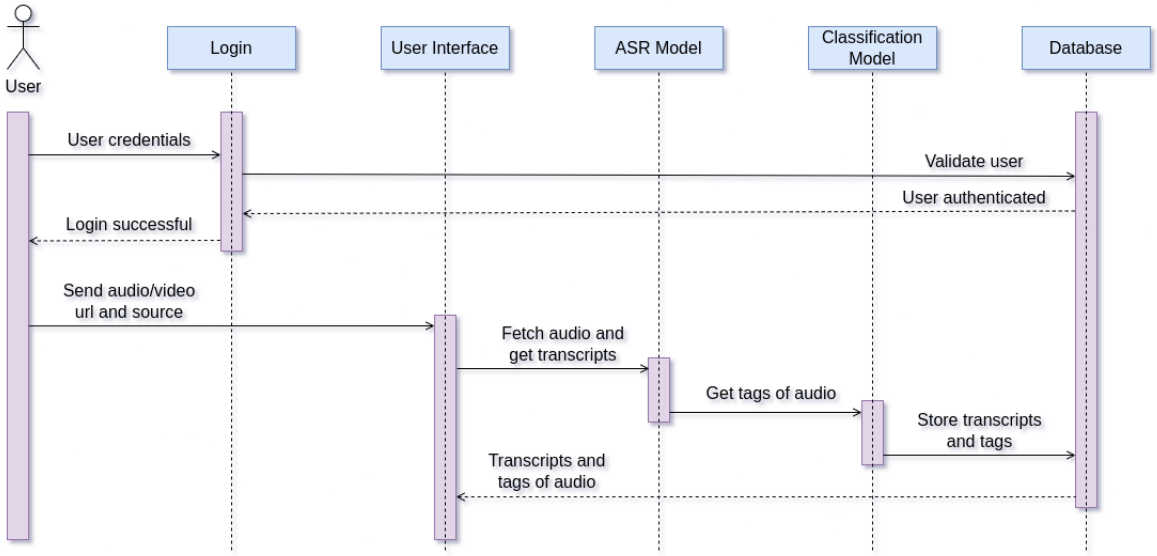

Figure 4.2: Sequence diagram

4.4.3. Activity diagram

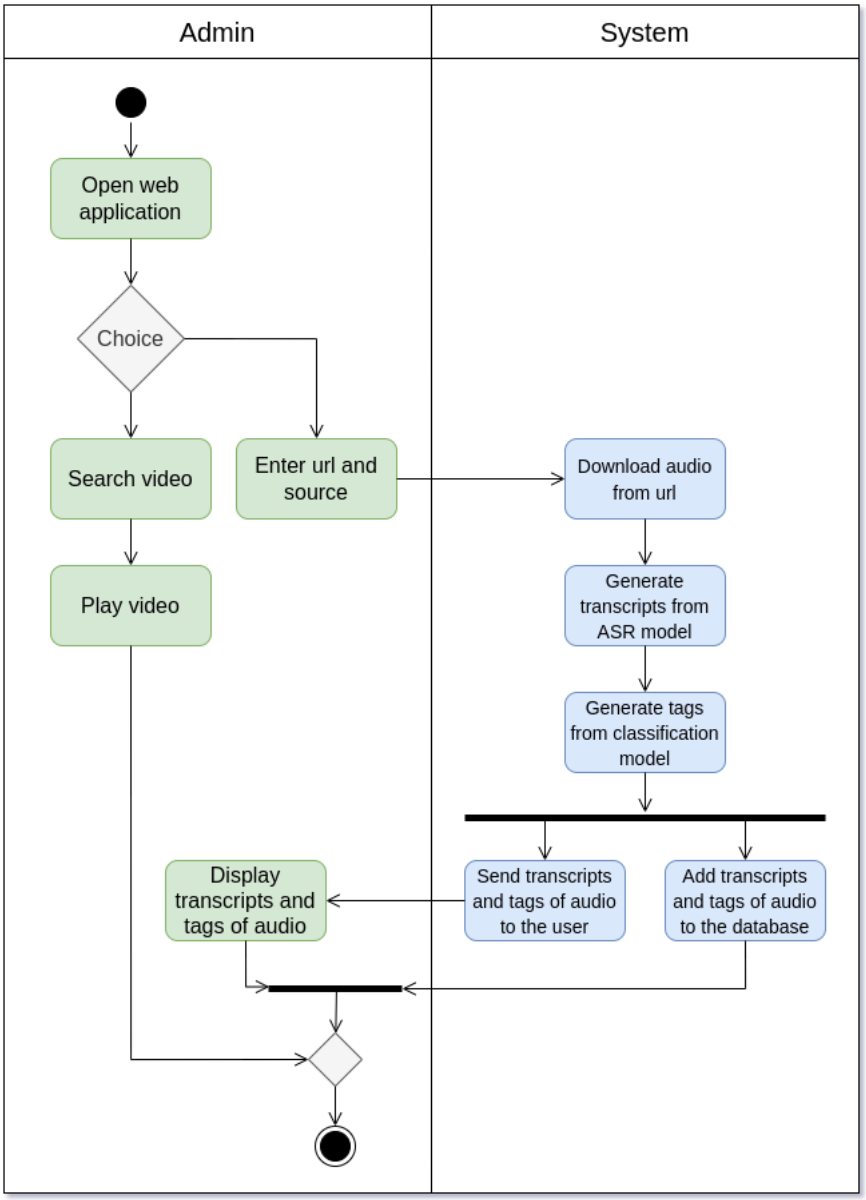

Figure 4.3: Activity diagram

5. METHODOLOGY

5.1. Nepali Speech Transcript Generation and Classification System

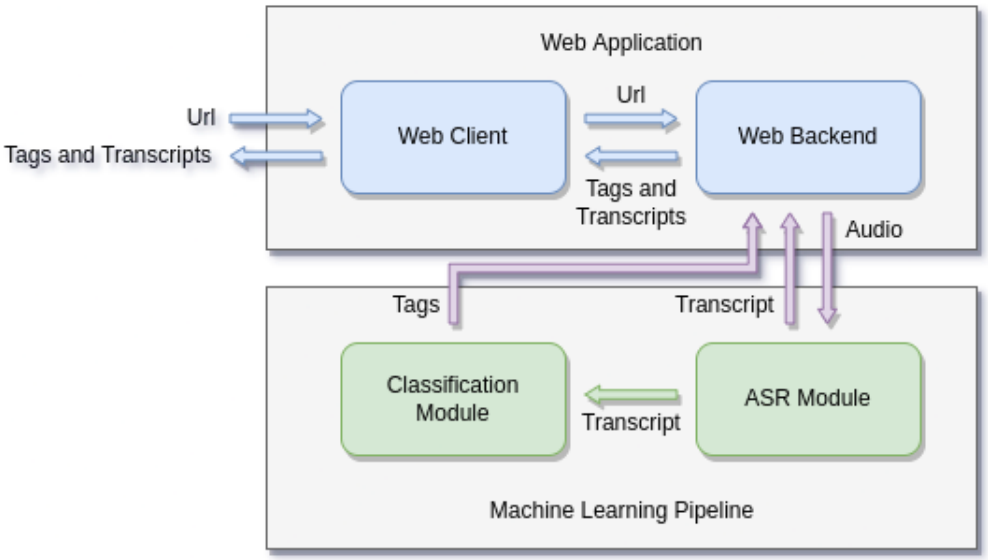

Figure 5.1: Overall system block diagram

The overall system can be broadly classified into two components: the Web Application and the Machine Learning Pipeline. A user sends an audio URL with its source via the web interface and after subsequent processing in the backend, returns the transcriptions and tags corresponding to the content of the audio. The web interface also offers a dashboard, where users can go through the news audios with corresponding transcripts and tags and view the statistics related to keywords and topics that are trending, based on the content uploaded by the admin. The interface also offers searching based on contents of audio.

The web client sends user's requests to the web backend. The backend then queries the ASR module for audio transcripts and then the classification module for tags with the obtained transcripts. The classification module returns the corresponding classification tags. The transcripts and the tags are returned to the web client for display. They are also stored in the database, if the logged in user is has admin privileges.

5.1.1. Web Application

The overall Application can be explained in terms of the components below:

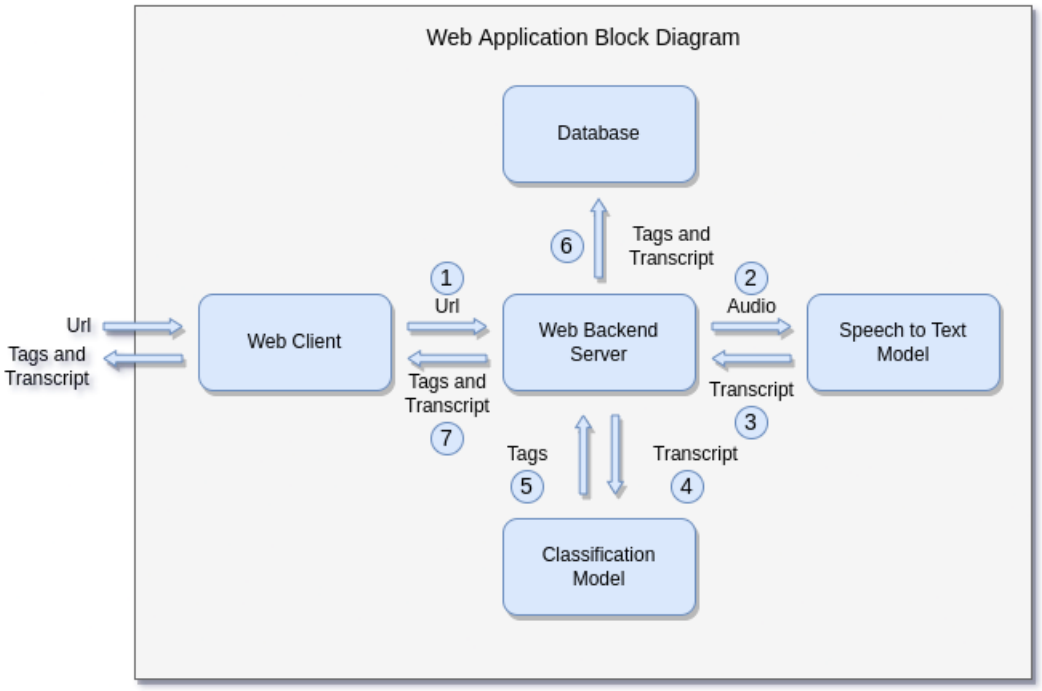

Figure 5.2: Web Application Block Diagram

The front end web client has been developed using ReactJavaScript (JS). It has two interfaces, one to send a request with URL to the backend that returns the transcripts and tags of the audio corresponding to the URL as response and also the other metadata related to the audio. The other interface displays all the audios and corresponding tags and transcripts from the database.

The web server has been developed using FastApi. It accepts HyperText Transfer Protocol (HTTP) POST request from the client with URL and source of the audio, queries ASR model for transcripts and requests classification model for classification tags corresponding the transcripts. It also saves these information in the database, as well as sends them as response to the client.

An ASR model trained in wav2vec2 framework comes next. A script that is called from back end takes audio path as an argument and queries this model. It sends Nepali transcriptions of the audio as a response.

A MuRIL based classification model has been hosted in docker container using tensorflow serving. It exposes Representational State Transfer (REST) Application Programming Interface (API) to query the model. The backend sends a POST request with a text and receives a response with corresponding tags.

Sqlite database has been used to store information about audio data. Each data point currently has following information when being saved to the database: Id, Title, URL, Tag, Transcript, Date.

5.1.2. Machine Learning Pipeline

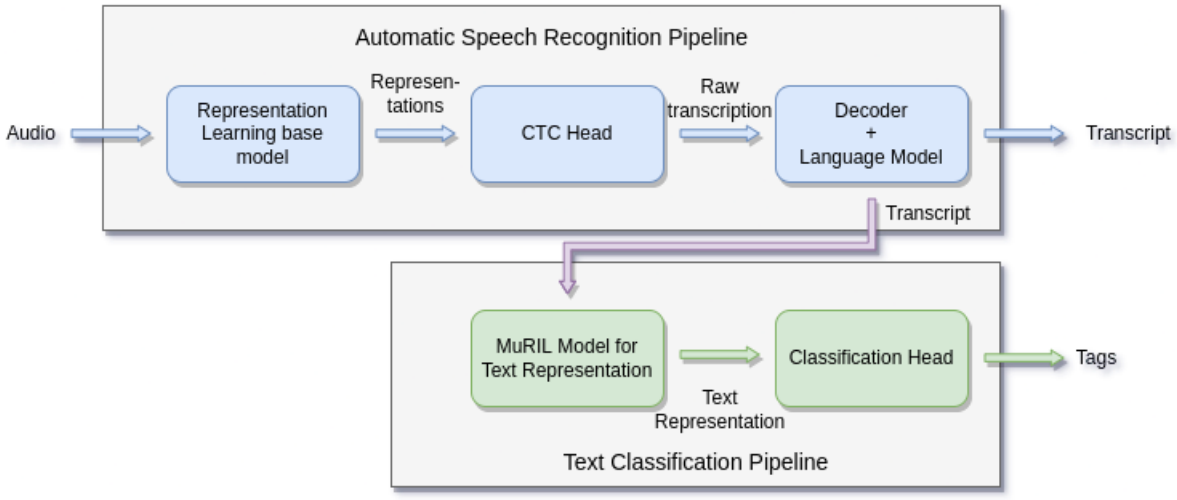

Figure 5.3: Machine Learning Pipeline

The machine Learning Pipeline consists of two major components: ASR Pipeline and the Text Classification Pipeline.

The ASR pipeline consists of a model trained with a projection head on top of a wav2vec2.0 based pretrained model using CTC training. The model maps each step in the down-sampled version of the input to a character in the vocabulary. The output thus obtained is passed to the Inference block to obtain appropriate transcripts.

Text Classification Pipeline consists of a model trained with a classification head on the top of the MuRIL Model. The MuRIL model gives text representations that are classified by the classification head to give the output as one or more suitable categories.

The models used in the ML Pipeline have been explained in detail in the next section.

The detail about each of the steps comprising the ML pipeline have been explained from section 5.3 to 5.9

5.2. Machine Learning Models

5.2.1. Model for Speech Representations: wav2vec 2.0

The wav2vec 2.0 framework proposes learning contextualized representations together with an inventory of quantized speech representations, substantially bettering results. Preprocessed audio is taken as input and converted to latent speech representations, which are then used to build contextualized representations. Furthermore, the latent speech representations are converted to quantized speech representations to represent targets for the self supervised task. Wav2vec 2.0 has two configurations: BASE and LARGE. We utilize the BASE configuration. The wav2vec 2.0 model is divided into following architectural blocks.

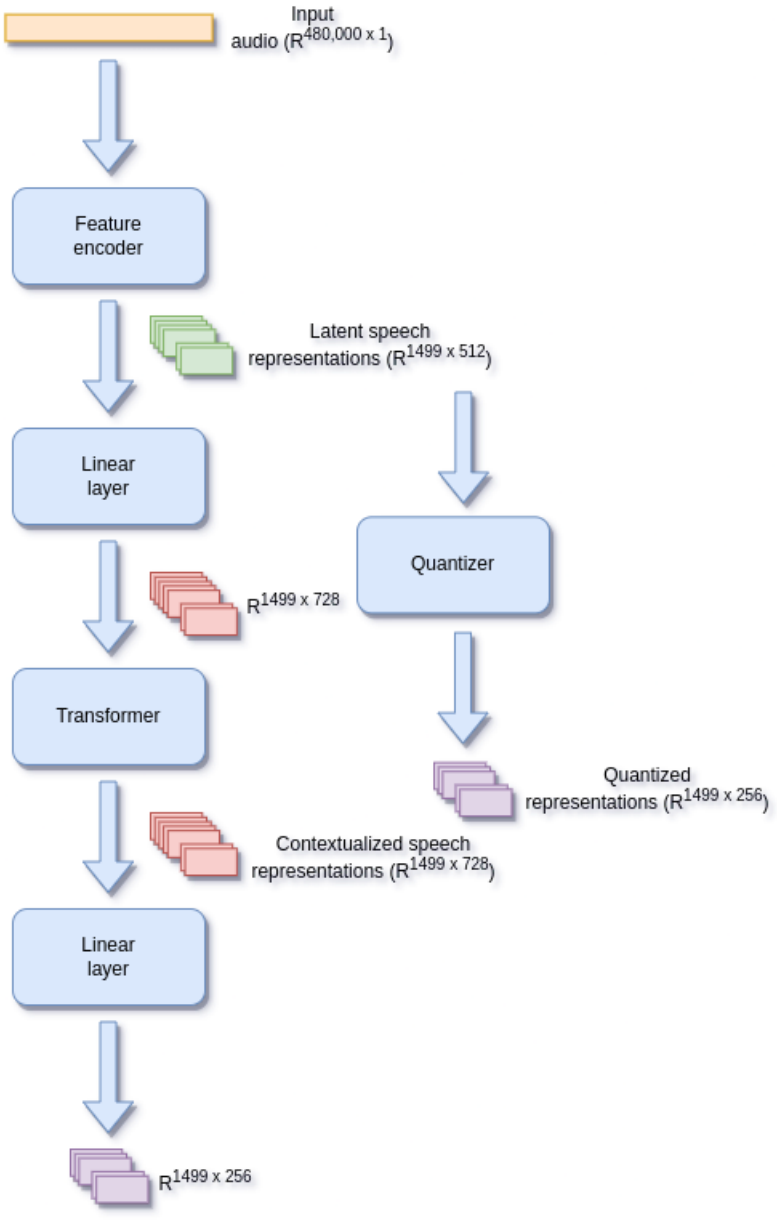

Figure 5.4: Wav2vec 2.0 architecture. The figure shows how a sample audio of 30 seconds with sample rate 16000 Kilo Hertz (KHz) is passed onto various layers of the model and outputs representations of 256 dimensions for each timestep.

5.2.1.1 Feature encoder The feature encoder consists seven blocks of 1 Dimensional (1D) convolutional layer with each block consisting of 512 channels. The convolution block is followed by layer normalization and Gaussian Error Linear Unit (GELU) activation function. The strides of the first convolutional block is 5 while that of the remaining block is 2. Likewise the kernel width for the seven blocks can be represented as (10,3,3,3,3,2,2). Each latent speech representation output by the feature encoder encompasses 25 millisecond (ms) of input signal, with a stride of 20 ms between each sample, i.e the input speech representations are down-sampled such that each 25 ms of the input represents one unit in latent space, with stride of 20 ms between the starting point of each sample in input space. The output of the feature encoder is dropped out at a rate of 0.1.

5.2.1.2 Transformer The wav2vec2.0 base configuration consists of 12 transformer blocks with a model dimension of 768 and 8 attention heads. The encoder layer dropout is kept at 0.05. The architecture utilizes convolutional layer instead of fixed positional embedding which acts as relative positional embedding.

5.2.1.3 Quantization module Quantization module performs discretization of feature encoder's output to a finite set of speech representations. The quantized representations are chosen using product quantization, which involves choosing codewords from multiple codebooks and concatenating them.

The BASE configuration is trained on 960 hours of Librispeech ASR corpus. The dataset is derived from audiobooks that are part of the LibriVox project and consists of 1000 hours of speech sampled at 16 KHz. The training split is partitioned into 100, 360 and 500 hours of data. The first two sets is on average of higher recording quality with accents closer to US English through a simple automatic test. The speakers in the corpus were ranked, with the lower WER speakers designated as "clean" and the higher WER speakers designated as "other". From the "clean" pool, 20 male and 20 female speakers were drawn at random and assigned to a development set. The same was repeated to form a test set. For each dev or test set speaker, approximately eight minutes of speech are used, for total of approximately 5 hours and 20 minutes each[31].

5.2.2. Model for Speech Representations: CLSRIL-23

Multilingual pretrained model is present only for wav2vec 2.0 Large architecture. Indic languages contribute a very small portion to the 53 languages on which the cross lingual rep-

resentations are learnt in the multilingual pretrained model [21]. Meanwhile, only monolingual model trained on English language is present for the wav2vec 2.0 Base architecture [20]. CLSRIL-23 is a self supervised based audio pretrained model which learns cross lingual speech representations from raw audio across 23 Indic languages. It is built on top of wav2vec 2.0 base architecture on 10000 hours of audio [22]. The 10000 hours include about 30 hours of Nepali data from All India Radio. A detail on the distribution has been given in Appendix 10.2

5.2.3. Model for Text Representations: MuRIL

In order to classify the text obtained from the ASR pipeline, we use MuRIL [27] which uses a BERT Large architecture [25] pretrained on 17 languages including English. The model makes use of transformer's encoder part with attention mechanism to learn contextual relations between words in a given text. The BERT large architecture consists of 24 encoder blocks and a total of 340 Million (M) parameters. MuRIL is pretrained on 17 languages, one of them being Nepali language. MuRIL is pretrained using two language modeling objectives: Masked Language Modelling (MLM) and Transistional Language Modelling (TLM). In both of these cases certain words in the input are masked before sending to the model and the model is trained with the objective of minimizing the loss while predicting these mask tokens.

In our case, we use the pretrained MuRIL model and finetuned it for our task of text classification. For this purpose, we add a dropout layer with dropout value 0.6 and a final dense layer with sigmoid activation of dimension 5, corresponding to the number of classes for classification.

The detailed architecture of the model we used is illustrated in figure 5.5

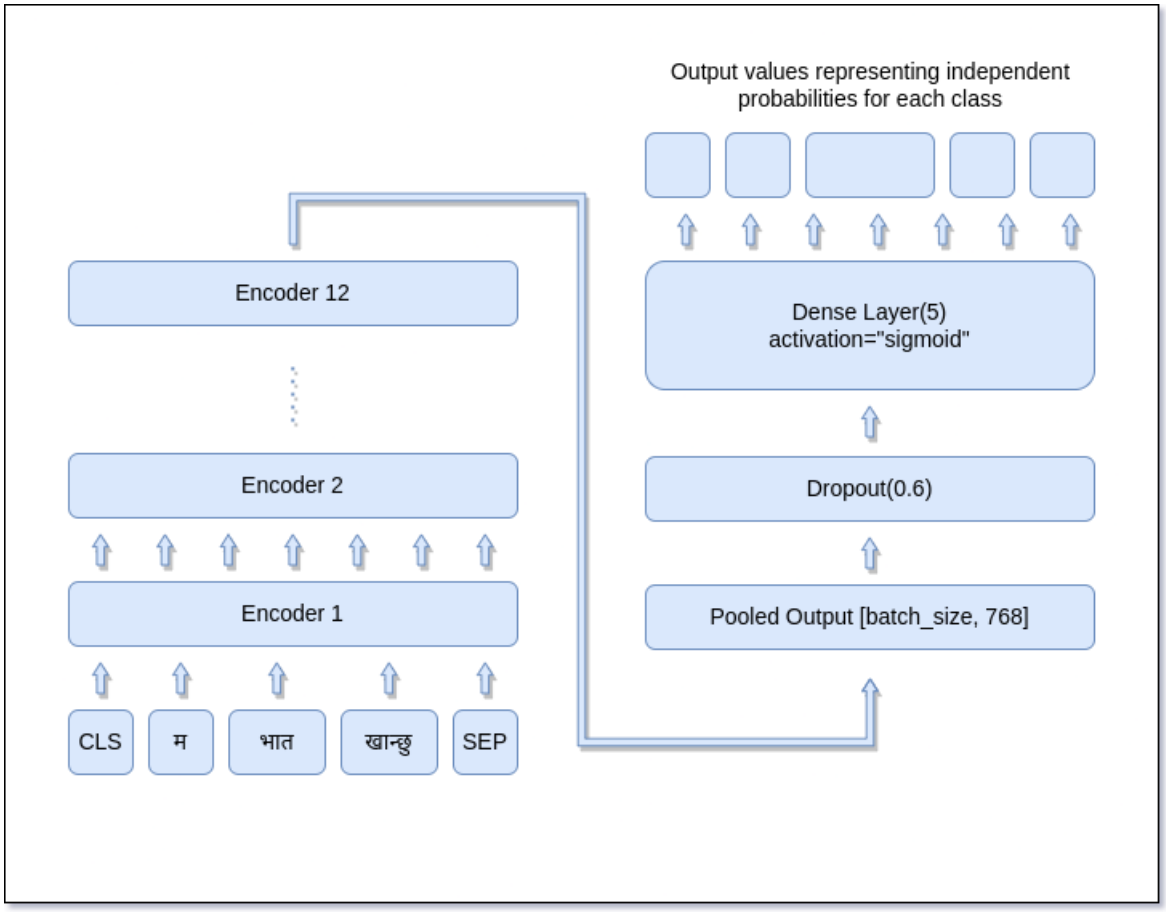

Figure 5.5: Model architecture for classification built on top of MuRIL

short For an input sequence, we obtain a pooled output of 768 dimension from the MuRIL model which gives the contextual representations for that particular input sequence. Then, this output is passed to the classification layer which finally outputs a 5 dimensional vector giving the prediction for each of our classes.

5.3. Dataset acquisition

5.3.1. Speech Data with transcriptions from OpenSLR dataset

We used open source Nepali dataset with audios and corresponding transcripts from OpenSLR, a site devoted to hosting speech and language resources. The transcribed audio consists of 157,905 utterances from 527 speakers amounting to 165 hours[32]. Audios in this dataset are already in the required format. So, no preprocessing was needed for the audio of this dataset.

5.3.2. OpenSLR female dataset

This dataset contains high quality transcribed audio data for Nepali[33]. A catalogue showing information about this recorded audio is shown below.

Gender	Female
Speakers	19
Audio samples	2064
Total duration	02:47:45

Table 5.1: Data distribution for OpenSLR female dataset

5.3.3. 074BCT Dataset

This dataset comprises labeled speech data of 21 individuals from 074 batch of Computer Engineering students from Pulchowk Campus, Institute of Engineering, Tribhuvan University, Nepal. The speakers comprise of 19 males and 2 females with each speaker recording audio for 30 different transcriptions amounting to a total of 630 labeled audio samples.

5.3.4. Text Classification Data

For this purpose we've used the publicly available Nagarik corpus [1]. However, for our purpose, we've used only 5 categories: "Sports", "National", "International", "Health" and "Economy". We've extracted individual sentences from the subset of news corpus for each of the five categories and used the corresponding sentence as a single data sample for our dataset. Following this process, we built a dataset containing a total of 22500 data with the following distribution.

Category	**Number of sentences**
Sports	4500
National	4500
International	4500
Health	4500
Economy	4500
Total	22500

Table 5.2: Distribution of sentences in different categories

[1] Source:https://github.com/ashmitbhattarai/Nepali-Language-Modeling-Using-LSTM/tree/master/Nepali_Corpus/Nagarik

5.4. Data Preprocessing

5.4.1. Audio Preprocessing

Before feeding the audio data to models for pretrainig, data should be preprocessed with following steps as per need:

1. Segment the audio into 30 seconds(maximum) each.
2. The background music should be then removed from the audio.
3. The silence contained in the audio, should be then removed by specifying a threshold decibel value.
4. The audio needs to be sampled at 16 KHz and any stereo stream should brought down to a mono channel.

5.4.2. Text Preprocessing

The data from OpenSLR used for finetuning contains text with punctuation marks and numbers. Since CTC algorithm maps time stamps that represent units of sound to characters, we can not use digits as characters in our vocabulary. Instead, we must first convert those numbers to words in our transcript. For this, we followed the following steps for the conversion and cleaning:

1. We first mapped all Nepali digits in our corpus to English and then made a list of all numbers in our corpus using regex.
2. After that, we used a script to get Nepali word corresponding to all those numbers and saved the mapping from Nepali numbers to words for every number in corpus in a key value pair data structure.
3. Then, we used regex to replace all the numbers to corresponding word mapping.
4. We also dropped all the punctuation marks from our text.

Below are some examples of how raw uncleaned texts were pre-processed to from clean texts with digits replaces by words and punctuation marks removed:

Raw text	Preprocessed text
११९५ ईश्वीमा त्यसले	एक हजार एक सय पन्चानब्बे ईश्वीमा त्यसले
युवतीका कुरा सुनेर आचार्य शङ्करले फेरि भने हे देवी!	युवतीका कुरा सुनेर आचार्य शङ्करले फेरि भने हे देवी

Table 5.3: Preprocessing of texts

5.5. Tools and technologies used

5.5.1. Automatic Speech Recognition (ASR)

- **fairseq**, a sequence-to-sequence learning toolkit for Torch tailored for Neural Machine Translation from Facebook AI Research.
- **Torch**, a scientific computation network framework and a machine learning library.
- **NCCL** backend for distributed NVIDIA GPU training.
- **Compute Unified Device Architecture (CUDA) toolkit** includes GPU accelerated libraries, debugging and optimization.
- **wav2letter** and **KenLM** for decoding and language model.

5.5.2. Classification

- **TensorFlow**, an end-to-end open source platform for machine learning.
- **CUDA toolkit**, includes GPU accelerated libraries, debugging and optimization.
- **Docker** for serving the model using REST APIs in an isolated container.

5.5.3. Web Application

- **React**, a front-end JavaScript library for building user interfaces based on User Interface (UI) components.
- **FastAPI**, a web framework for developing RESTful APIs in Python.
- **SQLite**, a lightweight database engine.

5.5.4. Miscellaneous

- **Python**, **Java** and **Shell scripts** for manipulating input in different stages of the ASR pipeline.

5.6. ASR: Pretraining self-supervised learning models for speech Representations

For pretraining using the wav2vec2.0 framework described in the 5.2.1, the raw audios (X) first need to be in the format specified in the 5.3. So audio data, which are of 30 seconds at maximum in length and 16 KHz frequency with mono channel are organized in batches after splitting into test, validation and training sets and fed to the feature encoder with 7 convolution blocks, which outputs latent representations of the given data, Z, such that X $\rightarrow$ Z. Those latent speech representations, Z are then quantized to give quantized representation, Q using G codebooks with V entries in each.

The latent space representations, Z are also fed to transformer, by masking certain portion of input using mask tokens. This masking is governed by two hyperparameters p and L. All the time steps are randomly sampled without replacement with probability p to obtain the starting indices and then L consecutive time steps from those sampled index are masked, allowing masked spans to overlap. The transformer then outputs the contextualized speech representations, C. Hence, the same input X, is transformed in two different ways, one to give quantized representatiosn Q and the other to give contextualized speech representations C. This flow of input through the model has been demonstrated in figure 2 5.2.1, with an audio input of 30 sec as example.

The contrastive task the transformer performs during the training is to identify true quantized latent speech representation $q_t \varepsilon$ Q, given the context network output $c_t \varepsilon$ C centered over masked time step t. For the purpose, K+1 quantized candidate representations, containing K distractors uniformly sampled from other masked time stamps in the utterance and one correct sample qt are used. Here, K is a tunable hyper-parameter. The loss function corresponding to this process has been explained below:

5.6.1. Loss Function

During pre-training, representations of speech is learnt by solving a contrastive task as explained above. With that, another loss called Diversity Loss is also added to ensure that a large portion of the codebook is used by the model while training. Similarly, third loss called

Features Penalty is also used to stabilize the model. So, the objective function is the weighted sum of these three losses, given by:

$$L_{total} = L_{contrastive} + \alpha L_{diversity} + \gamma L_{features\_pen} \tag{5.1}$$

where α and γ are tunable hyperparameters.

Contrastive Loss (Loss_0) The contrastive task in the pretraining involves identification of true quantized latent speech representation q_t, given the context network output c_t centered over masked time step t, given K+1 quantized candidate representations, with K distractor samples and one correct sample q_t. So, the contrastive loss is given by:

$$L_{contrastive} = -log \frac{e^{\texttt{cosine\_similarity}(q_t, c_t)/k}}{\sum_{q_i \in Q} e^{\texttt{cosine\_similarity}(q_i, c_t)/k}} \tag{5.2}$$

Diversity Loss (Loss_1) As explained above, the contrastive loss uses codebook to sample both positive and negative examples. Diversity Loss encourages equal use of all codebook entries in that process. If G codebooks, with V entries each have been used, diversity loss is the negative of entropy calculated over all codebook entries.

$$L_{diversity} = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} p_{g,v} log p_{g,v} \tag{5.3}$$

Features Penalty Loss (Loss_2) It is the mean of square of the features extracted from CNN, i.e. the latent speech representations, Z. This parameter is mostly to stabilize the model to prevent it from crashing late in training. It has very little effect on final accuracy and contribution to the total loss function.

$$L_{features\_pen} = Mean(Square(Z)) \tag{5.4}$$

5.7. ASR: Finetuning

The representations learned from the pretraining phase are then used in the downstream task of automatic speech recognition of Nepali language.

The labelled data for the task is first split into train, validation and test set. After cleaning the

data as explained in 5.4.2, since the CTC algorithm models the output sequence character-wise, it is required to convert the transcriptions of the dataset into individual character/tokens. So, the entire transcriptions was converted into individual characters. For that, we first converted the sentences into a file with individual words, separated by a special token "|" to denote space, which is used later on for building Language Model for inference 5.8.

As an example the Nepali sentence "यसमा समावेश गरिएको थियो" would be converted into "य स म ा | स म ा व े श | ग र ि ए क ो | थ ि य ो |". Next we build a vocabulary representing the number of occurrences of each individual tokens in our dataset. We observed that there were a total of 67 classes of characters in our dataset including the separator token "|". The five most frequent characters in our dictionary are as follows:

Character	Total occurrences
|	474849
ा	271135
्	216832
र	185596
न	153826

Table 5.4: Top 5 characters and their occurrences in our dataset

The fine-tuning is carried out by adding projection layer on top of the context network for mapping the transformer outputs into 67 classes representing the number of characters in the vocabulary. This maps the representations from transformer into 67 dimensions for each timestamp corresponding to the contextualized speech representations, c_t. The resulting output represents the probability distribution for each character in our vocabulary. Then, the model is trained to optimize the CTC loss.

5.7.1. Loss Function

To train a CTC-based ASR system, we use CTC loss function that is given by the sum of the negative log-likelihoods of the correct output Y for each input X over the dataset D.

$$L_{CTC} = \sum_{(X,Y)\varepsilon D} -logP_{CTC}(Y|X) \tag{5.5}$$

To compute this loss for a single pair of output Y given input X, we need the probability $P_{CTC}(Y|X)$. As we saw in 3.5, to compute the probability of a given output Y we need to sum over all the alignments that can collapse to output Y. So, that probability is given by:

$$P_{CTC}(Y|X) = \sum_{A\varepsilon A_{X,Y}} {}^{T}_{t=1} p_t(a_t|X) \tag{5.6}$$

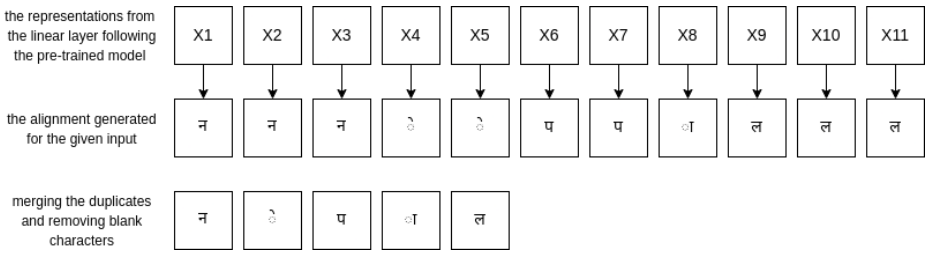

Figure 5.6: Steps of OpenSLR algorithm

Here $A_{X,Y}$ indicates the set of all the valid alignments for X that collapse to Y and a_t represents probability of an alignment A at time step t. Using the trained model, given input sequence

$$X = [x_1, x_2, \ldots, x_T]$$

can thus be mapped to an output sequences

$$Y = [y_1, y_2, \ldots, y_U]$$

i.e. the characters for the transcript of the audio.

Figure 5.6 shows how the text is generated by a pass through the model by generating alignments in the intermediary step.

5.8. ASR: Inference

The OpenSLR trained speech recognition model thus maps vector corresponding each timestep from the context network to a character in our vocabulary, i.e. each timestep in the output sequence to a (1, 67) vector, 67 being the number of characters in our vocabulary. The output of the network looks like Figure 3.7.

So, the output can be visualized as a 2d matrix, with probability distribution over all the characters in the vocabulary, for each time step t along the X-axis.

As explained in 3.5.2, greedily decoding the output doesn't consider the fact that a single output can be obtained from multiple alignments. So we have used modified beam search algorithm which overcomes the limitations of greedy search. A n-gram word level language model has also been used along with beam search. So, the score given to an output also incorporates LM score whenever a word is inserted in the decoded output with α score which is a hyperparameter, representing the weight given to Language Model score. The LM has been built using KenLM. The decoding has been implemented using python bindings from

flashlight library.

5.9. Text classification

The Nepali text obtained from the ASR model is then passed onto the classification pipeline built on top of MuRIL model. The detailed architecture of the classification model is explained in

5.9.1. Objective Function

The problem of text classification has been solved by training with an objective of minimizing the categorical cross-entropy loss function which corresponds to the following equation.

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_c log(\hat{y_c}) \tag{5.7}$$

where:
N = number of data samples,
C = number of classes,
y_c = True output label for class c,
$\hat{y_c}$ = Predicted probabilities for class c

However, in our implementation we use a loss function called sparse categorical cross-entropy which is a computationally efficient version of categorical cross-entropy due to the fact that it uses a direct class value as input instead of one-hot encoding over the classes. For example: For 4 classes problem, if the true output for a data is class 2, categorical cross-entropy takes the input as [0,0,1,0] while the sparse categorical cross-entropy takes the input directly as the class value, in this case 2.

5.10. Evaluation Metrics

5.10.1. Pretraining

The objective function and evaluation metrics used during pretraining explained in 5.6 are described below:

Accuracy Accuracy in the context of pretraining refers to the percentage of total quantized latent speech representation for a masked time step that the model is able to identify correctly.

$$Accuracy = \frac{Total\ True\ Positives}{Total\ Samples\ Observed} \tag{5.8}$$

Accuracy during pretraining and performance on downstream task are not always correlated. So, evaluation of model and stopping decisions cannot be solely made on the basis of accuracy only. So other metrics are also recommended to be used which have been described in the sections that follow.

Code Perplexity It is a measure of randomness while choosing codes from codebook. It is the exponentiation of negative entropy calculated over hard probabilities(one hot vectors) of codebook indices. It gives what percentage of the codebook is used at the instant by the model. Too low value of this would mean the model is focusing on few codes only from the codebook. Too high value indicates instability. To evaluate if a model is performing well, it should increase and reach upto 100-500 and then saturate.

Prob Perplexity It is a measure of randomness while choosing codes from codebook, similar to code perplexity. Only difference being, it is the exponentiation of negative entropy calculated over softmaxed average probability of codebook indices.

5.10.2. Finetuning

The objective function and evaluation metrics used during finetuning for the downstream task of speech recognition explained in 5.7 are described below:

Word Error Rate It is the standard evaluation metric for ASR systems. It is the minimum edit distance between two strings, one being the test string and the other being the correct string, which is the minimum number of substitutions, insertions, and deletions of words necessary to map the test string to the correct one. The Word Error Rate (WER) is given by:

$$100 * \frac{Num(Insertions) + Num(Substitutions) + Num(Deletions)}{Total\ no.\ of\ words\ in\ the\ correct\ string} \tag{5.9}$$

Character/ Utterance Error Rate It is the minimum edit distance between two words, one being the test word and the other being the correct, which is the minimum number of

substitutions, insertions, and deletions of characters necessary to map the test word to the correct one. The Character Error Rate (CER) is given by:

$$100 * \frac{Num(Insertions) + Num(Substitutions) + Num(Deletions)}{Total\ no.\ of\ characters\ in\ the\ correct\ word} \tag{5.10}$$

5.10.3. Text Classification

Accuracy Accuracy is simply the percentage of correct predictions the model makes. It describes how the model performs on all the classes. In our case of text classification, it indicates the fraction of the total number of data samples the classification model makes correct prediction.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ data\ samples\ in\ the\ set} \tag{5.11}$$

Precision and Recall Precision indicates what proportions of positive predictions are actually correct. Recall on the other hand indicates what portions of actual positives was identified correctly for the model. Often, accuracy is not able to truly evaluate a model's performance as it gives a measure over the entire classes. In cases of imbalanced dataset, accuracy isn't a good metric of evaluation. In such cases, Precision and recall are needed as it gives the performance evaluation for each of the classes in our dataset.

$$Precision = \frac{TP}{TP+FP}, Recall = \frac{TP}{TP+FN} \tag{5.12}$$

where:
TP = True positives
FP = False positives
TN = True negatives
FN = False negatives

For example: In our case of text classification, for a category "Sports", a precision of 0.8 indicates that only 80% of the predictions that are classified "Sports" are actually so. Similarly, a recall of 0.8 indicates that only 80% of the text that are actually "Sports" are classified as "Sports" by the model.

F1-score F1-score is the harmonic mean of precision and recall. It aims to combine precision and recall into a single metric.

$$F1score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.13}$$

5.11. Experimental Setup

5.11.1. Dataset

For ASR, we used the OpenSLR Nepali dataset(5.3.1) for both, pretraining and finetuning. The dataset was first split into train, validation and test sets in the ratio of 96:2:2. For pretraining, we used the train and validation sets only, because there is no meaning to test just the pretrained model, not finetuned for any task. Whereas for finetuning, we used all the data, i.e. the train and validation set during training and the test set to evaluate the performance of the trained model. Besides, OpenSLR female dataset(5.3.2), 074BCT dataset(5.3.3) are also used for testing purpose to evaluate the performance of ASR model.

For text classification, we use the dataset as stated in 5.3.4. The dataset consist of 22500 sentences belonging to a total of 5 categories which is split into 70:15:15 as train:valid:test set.

5.11.2. Training

Pretraining For pretraining, we have used the wav2vec2.0 framework. While training, we have trained two different models with all the other configurations the same but with two different model checkpoints to start from: one starting from the wav2vec2.0(5.2.1) and the other from the CLSRIL-23(5.2.2). While training, Adam optimizer with adam_betas (0.9, 0.98) with adam_eps 1e-06 is used. Initial learning rate of 0.0005 is used, which is set to increase linearly to reach 0.005 in 32,000 warmup updates, with weight decay of 0.01 using polynomial decay after that point. For losses described in 5.6.1, the weight factors α=0.1 and γ=10 are used. The default setting of G=2 and V=320 is used for quantization module(5.2.1.3). Masking probability p=0.065 and masking length M=10 is used.

For pretraining we use 2 GPUs, NVIDIA 1080 and NVIDIA 1070Ti with distributed world size of 2. Training was continued until the code_perplexity was seen to saturate, without dropping accuracy much compared to the best accuracy. This resulted in training for 30,000

updates. Training took about 161 hours.

Finetuning Finetuning of the pretrained models was done by adding a projection head on top as described in 5.7. The weights of feature encoder (CNN blocks) trained during pretraining were frozen. Learning rate of 0.00003, adam optimizer with adam_betas (0.9, 0.98) and adam_eps of 1e-08 was used. Masking probability p=0.65 is used for masking of feature encoder outputs as described in SpecAugment[34]. Finetuning of the model is done till validation WER and CER continues to decrease.

For the finetuning we used 2 GPUs, same as for pretraining. Finetuning was done for more than 100,000 steps till no improvement was seen in WER and CER, which took about 154 hours.

Text classification Text classification model was trained using the data described in 5.3.4. The dropout value for the dropout layers is set as 0.6 . The training is done in batch sizes of 16 with an objective to minimize sparse categorical cross-entropy. The optimizer used in this case is AdamW. AdamW [35] is a slight modification of Adam optimizer which implements weight decay differently. The classification model is trained for 10 epochs with a learning rate of $3e^{-5}$.

6. RESULTS

6.1. Automatic Speech Recognition

Through our experiments, we tried to analyze how the idea of learning speech representations through unsupervised pretraining in order to improve the performance on downstream tasks (ASR in our case) works for Nepali Language. We also wanted to see how low resource languages like Nepali can benefit from sharing of low level speech representations across different languages.

Besides, we have also tried to compare the improvement in performance on downstream task of ASR while using a model pretrained on multilingual corpus of Indic Language, CLSRIL-23(5.2.2) versus using the a pretrained on English language only, wav2vec2.0(5.2.1). This is to verify if low level speech representations across languages of similar origin and locality are more similar, giving a better performance.

We have also conducted small experiments to see if adding unsupervised pretraining with Nepali data to the pretrained models before finetuning for the downstream task has any impact in performance. If so, it can indicate some space for further improvement of Nepali Speech Recognition by pretraining on enough unlabeled Nepali data that would be fairly easy to collect compared to transcribed data. We have also tried to see how the incorporation of LM in decoding transcripts during inference improves the results.

With the objectives mentioned above, we have trained four different models in total. We had two pretrained models, CLSRIL-23(5.2.2) and wav2vec2.0(5.2.1). We developed additional two pretrained models by adding unsupervised pretraining on each of these models using OpenSLR Nepali dataset(5.3.1). So this gave us four model checkpoints to start finetuning for ASR.

We continued the training until validation probability perplexity of the models became stable, without significant drop in accuracy. But we can not draw conclusions about these models until we evaluate them on a downstream task. So in the sections that follow, we have analyzed these models based on their performance in the task of ASR. We have included the training logs for pretraining in Appendix-B(10.2).

For finetuning, we trained the four different models with all the other configurations the same but with four different model checkpoints to start from as explained above. We trained all four models until the validation WER started showing no further improvement. The training

results of these models based on their performance on validation set are summarized in the table and graphs below:

Models	Train	
	WER	CER
W2V_FT	12.41	2.38
C23_FT	11.2	2.18
W2V_PF	11.39	2.26
C23_PF	10.54	2.12

Table 6.1: Training Summary for different models. W2V_FT is the model we finetuned from wav2vec2.0's checkpoints, C23_FT is the model we finetuned from CLSRIL-23's checkpoints. W2V_PF is the model we finetuned after adding pretraining to wav2vec2.0's checkpoints. And finally C23_PF is the model we finetuned by adding pretraining to CLSRIL-23's checkpoint.

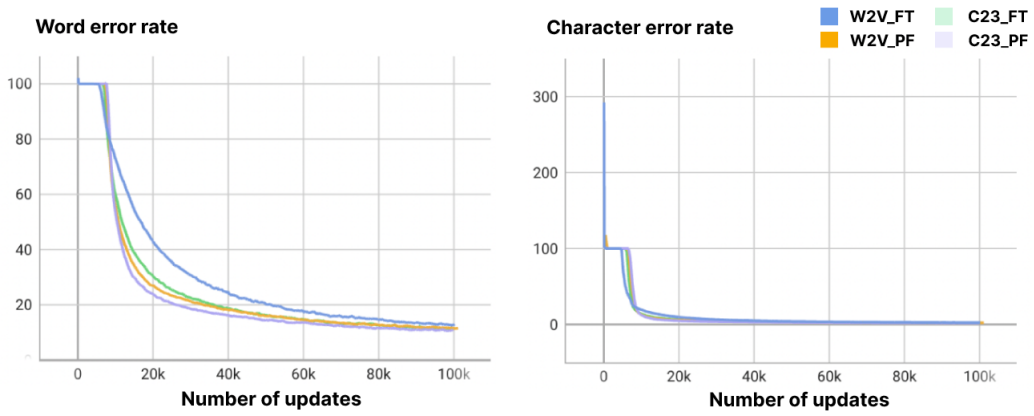

Figure 6.1: Comparison of finetuning metrics in 5.10.2 of W2V_FT, C23_FT, W2V_PF and C23_PF in the validation set

The models developed with pretraining before finetuning show slight improvements in validation WER and CER. Although, the final obtained WER are not much different from each other, the nature of decrease in WER is different. Models with added pretraining got to good WER in lesser number of epochs. This is probably because during pretraining the models already capture a lot of representation from training data so learning during finetuning becomes faster. We need to compare the models based on their performance in test sets too see if this has led to better generalization capabilities or the model has just overfit on train and validation set, giving lesser validation WER. It is an understood phenomenon in ML that models tend to perform well in the test split from the same distribution as train set than a different test set. So to assess the models better, we have used two additional test sets also.

So in overall, we evaluated all four models with three different test sets:

1. The test split of the OpenSLR dataset(5.3.1) as explained in (5.11.1)
2. OpenSLR female Dataset(5.3.2)
3. 074BCT Dataset(5.3.3)

During Inference, we experimented with multiple n-gram models using the model that performed best without LM. All n-gram models showed similar performance. The 5-gram model showed slightly better results among the n-gram models(10.2), so we chose 5-gram LM to proceed with.

The WER and CER of all four models in each of the datasets have been summarized in table below:

Model	LM	Test Dataset					
		OpenSLR test		OpenSLR female		074BCT	
		WER	CER	WER	CER	WER	CER
W2V_FT	No	14.0	2.81	29.6	6.19	49.97	11.79
	5-gram A	2.33	0.78	26.92	7.38	43.98	12.43
	5-gram B	21.58	4.34	29.03	7.24	43.04	10.78
	5-gram C	2.40	0.75	23.07	**5.99**	42.37	10.79
C23_FT	No	12.13	2.47	28.94	6.17	49.95	11.37
	5-gram A	**2.14**	0.67	26.53	7.95	42.64	11.83
	5-gram B	20.95	4.37	28.39	7.81	42.24	**10.11**
	5-gram C	2.16	**0.65**	**23.05**	6.54	**40.98**	10.16
W2V_PF	No	12.14	2.48	34.70	9.37	52.56	14.54
	5-gram A	2.56	0.72	29.00	10.58	45.66	15.20
	5-gram B	21.11	4.10	31.69	10.54	44.00	13.09
	5-gram C	2.63	0.75	26.03	9.45	43.52	13.34
C23_PF	No	11.81	2.28	32.51	7.56	53.77	14.51
	5-gram A	2.24	0.68	28.01	9.03	47.43	15.71
	5-gram B	20.78	3.99	30.02	9.04	44.86	13.46
	5-gram C	2.52	0.72	24.42	7.80	44.91	13.76

Table 6.2: Result of different models on test sets. The best result for each of the dataset has been highlighted in bold. W2V_FT is the model we finetuned from wav2vec2.0's checkpoints, C23_FT is the model we finetuned from CLSRIL-23's checkpoints. W2V_PF is the model we finetuned after adding pretraining to wav2vec2.0's checkpoints. And finally C23_PF is the model we finetuned by adding pretraining to CLSRIL-23's checkpoint. Among the 5-gram models, A is made from our train set transcriptions of OpenSLR, B is made from 86k sentences from Nagarik corpus[3]and C is made from the combination of the two.

Form the table above, C23_FT has the best overall result in CER and WER, with C23_FT and W2V_FT having very similar scores. C23_PF and W2V_PF have better overall results among all the models in the test split of OpenSLR dataset in some cases when comparing models without LM or with LM made of data that doesn't have OpenSLR dataset. But they have worse results in other two test sets in almost every case. Even in the test split of OpenSLR dataset, when comparing models by using LMs with OpenSLR dataset, C23_FT and W2V_FT have better overall results. So C23_PF and W2V_PF have performed better in the test split of OpenSLR dataset only in some cases when models are compared using

[3]https://github.com/ashmitbhattarai/Nepali-Language-Modeling-Using-LSTM/tree/master/Nepali_Corpus/Nagarik

LM with no OpenSLR dataset, i.e when the other two models (C23_FT and W2V_FT) have not seen the data from distribution of OpenSLR's datset at all. These are the evidences that C23_PF and W2V_PF has overfit the distribution of OpenSLR dataset and have not learnt general representations better. In other two test data sets, C23_FT and W2V_FT have performed better than C23_PF and W2V_PF. Again, C23_FT has slightly better overall performance than W2V_FT indicating that representations learnt during pretraining in Indic languages are indeed closer to Nepali than those learnt in model pretrained in English language only. But both the models have very similar score, indicating that even the pretraining on Indic Languages that CLSRIL-23 has undergone has not brought very significant improvement over wav2vec2 for the case of Nepali language.

Another important aspect to analyse from the results is the impact of using LM to decode the output. In all three test sets and all of the four models, decoding with LM has improved the average WER and CER compared to decoding without LM, indicating the impact of LM improving the quality of transcripts. Only in the OpenSLR test split, LM A has better results than other LMs. This is because the test set and the LM corpus for A come from same data distribution, despite being different splits. For other test sets, LM B with additional tests other than that in A, has shown the best performance. The choice among LMs has the least effect in 074BCT dataset. This is probably the result of the fact that some audios in the dataset are very noisy and unclear for even us to interpret.

6.1.1. Analysis of some transcripts

The crowd sourced data set[32] and high quality transcribed female data set[33] from OpenSLR have often used different transcriptions for each speaker. To conduct an empirical analysis, we create a new data set with 5 speakers, each contributing five audio samples of the same transcription. The speakers consist of 3 males and 2 females. The analysis is conducted for the best performing ASR model using different LMs.

Original	Speaker	No LM	LM C
अर्का पूर्व कप्तान शाक्य भने उपाधि जित्दैमा मक्ख परेर बस्न नहुने बताउँछन्	1	अर्का पूर्व कप्तान शाक्य भने उपाधि जित्दैमा मक्क परेर बस्न नहुने बताउँछन् कूर्	अर्का पूर्व कप्तान शाक्य भने उपाधि जित्दैमा मक्ख परेर बस्न नहुने बताउँछन्
	2	अर्का पूर्व कप्तान श्राख्य भनी उपाधि जितैमा मक्ख परेर बस्न नहुने बताउँछन् कू	अर्का पूर्व कप्तान शाक्य भनी उपाधि जितमा मक्ख परेर बस्न नहुने बताउँछन्
यसका लागि पानी उमाल्नुपर्छ छान्नुपर्छ वा अल्ट्राभोइलेट वा रिवर्स ओस्मोसिस फिल्टरले सफा गर्नुपर्छ	1	यसका लागि पानी उमाल्नुपर्छ छान्नुपर्छ वा अल्ट्राभोइलेट वा रिभर्स अस्मोसिस फिल्टरले सफा गर्नुपर्छ रा	यसका लागि पानी उमाल्नु पर्छ छाड्नुपर्छ वा अल्ट्रावायलेट वा रिभर्स अस्मि फिल्टरले सफा गर्नुपर्छ र
	2	यसका लागि पानी उमाल्नुपर्छ छाड्नुपर्छ वा अल्ट्रा भोइलेट वा रिबर्स उस्मसिसपिल्ट रले सफा गर्नुपर्छ	यसका लागि पानी उमाल्नु पर्छ छाड्नुपर्छ वा अल्ट्रा भाइले वा रिभर्स कसमस सेल्टा ले सफा गर्नुपर्छ

Table 6.3: Speakerwise performance on each sentence using no LM and combined 5-gram LM. [5]

Upon analysis of result transcripts, it was seen that in many cases, the use of LM has improved the quality of output transcripts. But there are cases where where LM has incorrectly changed the transcripts. The two sentences in the table above represent both the cases in output.

In first example, the use of LM seems to have improved the quality of transcripts for every speaker. We also observed that every word in the first sentence is present at least once in the corpus the LM is made of. While in the second example, *angantuk*(derived) words for Nepali language are present. These words like अल्ट्राभोइलेट and रिवर्स ओस्मोसिस are not very common and don't even have consistent pronunciation in Nepali. LM hasn't been able to correct errors in these words and phrases. In fact, for रिवर्स ओस्मोसिस, LM has changed almost every occurrence to रिभर्स अस्मि. It was seen that अल्ट्राभोइलेट and रिवर्स ओस्मोसिस were not present in the corpus of LM while रिभर्स अस्मि was.

These representative examples we have chosen show that although LM has improved the performance of our model, which is even evident from the scores in table 6.1, there are some

[5]This result is performed on top of C23_FT model. The words in the original text that are not present in the LM are highlighted in orange whereas the mistaken words are highlighted in magenta. Speaker 1 is male and Speaker 2 is female.

trade-offs. LM has wrongly corrected some transcripts that were actually produced as right without an LM. Since we have used a simple n-gram LM, it doesn't do very well with texts not seen while building it. That is why there has been a lot of mistakes in sentences when LM has to deal with new and different kind of words.

6.2. Text Classification

The text classification model for a total of four categories was trained for 10 epochs. The accuracies for the classification model are listed in table 6.4. As discussed in 5.10.3, since accuracy isn't able to evaluate the model's performance completely, we've used precision, recall and f1-score to analyze each of the categories which is listed in table 6.5. It is seen that the model performs more or less similar for all the categories. The F1-score is slightly less for the category "National" due to the diverse types of news that can be grouped under this category. The confusion matrix in figure 6.3 helps us to evaluate the model even better. From the confusion matrix, it can be seen that a total of 44 sentences which were actually "National" were classified as "Economy". In many cases, the news of these two categories are similar and can belong to both of the categories. Also, the training set also might have been inaccurate and disputed in certain cases for these categories. These are the probable reasons why our model confuses in certain occasions between these two categories. The accuracy changes and pattern of loss for training and validation set are depicted in figure 6.2 and Appendix-B(figure 10.8) respectively. Detailed logs about training for text classification are shown in Appendix-B.

Accuracies	Value
Training accuracy	99.05%
Validation accuracy	92.68%
Test accuracy	92.06%

Table 6.4: Accuracy in text classification

-	Precision	Recall	F-1 score	Support
Sports	0.98	0.95	0.96	665
Economy	0.88	0.91	0.90	673
International	0.96	0.94	0.95	656
National	0.88	0.89	0.88	691
Health	0.94	0.94	0.94	690

Table 6.5: Other metrics for text classification

Table 6.7 shows some of the outputs for a set of input texts. The probability score represents the independent probabilities for each of the 5 classes in the order of "Sports", "Economy" , "International", "National" and "Health". As the probability values are independent, we can obtain more than one categories for a particular input text. The threshold probability we have

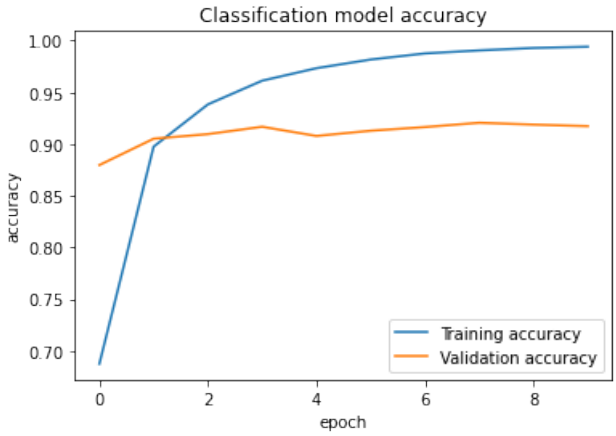

Figure 6.2: Model accuracy vs epoch comparison

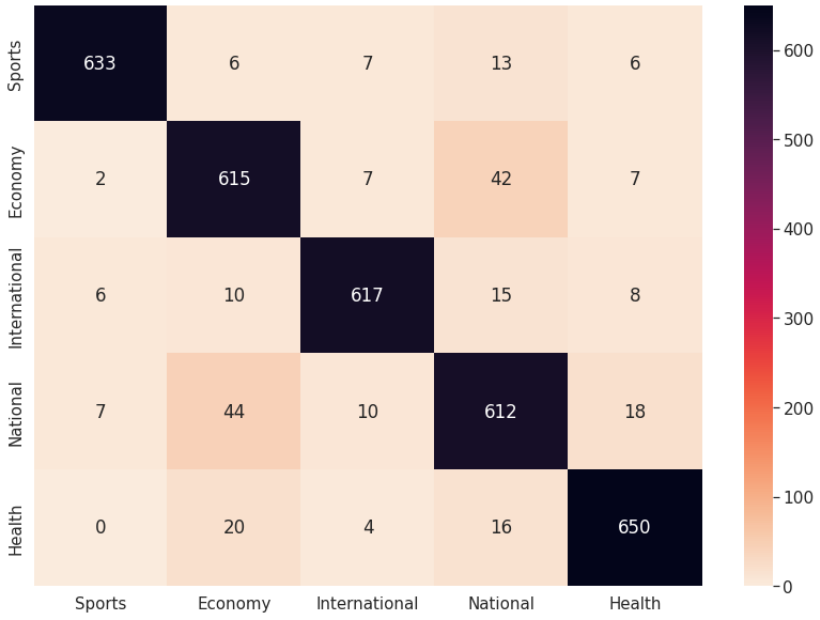

Figure 6.3: Confusion matrix for test set of classification model

set for our application is 0.7. Any category with a probability value greater than or equal to 0.7 is assigned to that particular input text.

Speaker	No LM	A	B	C
Male	24.71	39.08	25.86	26.43
Female	31.03	37.06	25.86	24.99

Table 6.6: Results of using different LMs on top of best performing model in the same transcription.

Input text	**Independent probabilities**	**Predicted categories**
अलमलमा काँग्रेस : नेतृत्व तालमेलको निर्देशन पठाउँछ, कार्यकर्ता एक्लै लड्छौं भन्छन्	[0.1949873, 0.27747023, 0.05792335, 0.9989458, 0.12741017]	National
रेमिट्यान्स तथा पर्यटक नआएपछि विदेशी मुद्रा आर्जनमा भएको कमिका कारण तिर्नु पर्ने ऋण भन्दा पनि कम वैदेशिक मुद्रा सञ्चितिले अत्यावश्यक वस्तु औषधि, इन्धन, खाद्यान्न लगायतको आयात करिब बन्द छ ।	[0.12890464, 0.9991125, 0.10256141, 0.25475365, 0.09709752]	Economy
यो खेलमा पोर्चुगिस ब्रुनो फर्नान्डेजले ह्याट्रिक गरे भने फ्रान्सका पल पोग्बाले ४ ओटा असिस्ट गर्न सफल भए	[0.9993343, 0.17365691, 0.15864772, 0.15841866, 0.08346665]	Sports
नेपालमा विद्युतीय सवारी साधनहरुको अहिलेको अवस्था अन्त्यन्तै नाजुक छ ।	[0.03934222, 0.99360144, 0.14609486, 0.7764218, 0.16220596]	Economy, National
रुसले युक्रेनमा आक्रमण गरेपछि अमेरिका र पश्चिमी देशहरूले रुसमाथि लगाएका कयौं प्रतिबन्ध लगाएका कारण रुसले पनि अमेरिकामाथि प्रतिबन्ध लगाएको बीबीसीले जनाएको छ ।	[0.19101802, 0.14724657, 0.99947286, 0.10264364, 0.14979446]	International
बेसारको प्रयोगबाट मुटुसम्बन्धी समस्या, पाचन प्रणालीको समस्या, डायबिटिज, डिप्रेसन, अल्जाइमर जस्ता रोगबाट बच्न सकिने तथ्य सार्वजनिक भएका छन् ।	[0.09029514, 0.36342692, 0.04512852, 0.28204438, 0.99888384]	Health
भारतीय प्रधानमन्त्री नरेन्द्र मोदीले कोरोना भाइरस(कोभिड–१९) को प्रकोपबाट डराउनुभन्दा एकजुट भएर सामना गर्नुपर्ने बताएका छन् ।	[0.10470441 0.05300167, 0.73453945, 0.08375183, 0.9987021]	International, Health

Table 6.7: Examples of text inputs with their predicted categories and probability values for each category

Table 6.7 shows some of the outputs for a set of input texts. The probability score represents the independent probabilities for each of the 5 classes in the order of “Sports”, “Economy” , “International”, “National” and “Health”. As the probability values are independent, we can obtain more than one categories for a particular input text. The threshold probability we have set for our application is 0.7. Any category with a probability value greater than or equal to 0.7 is assigned to that particular input text.

7. WEB APPLICATION OUTPUT

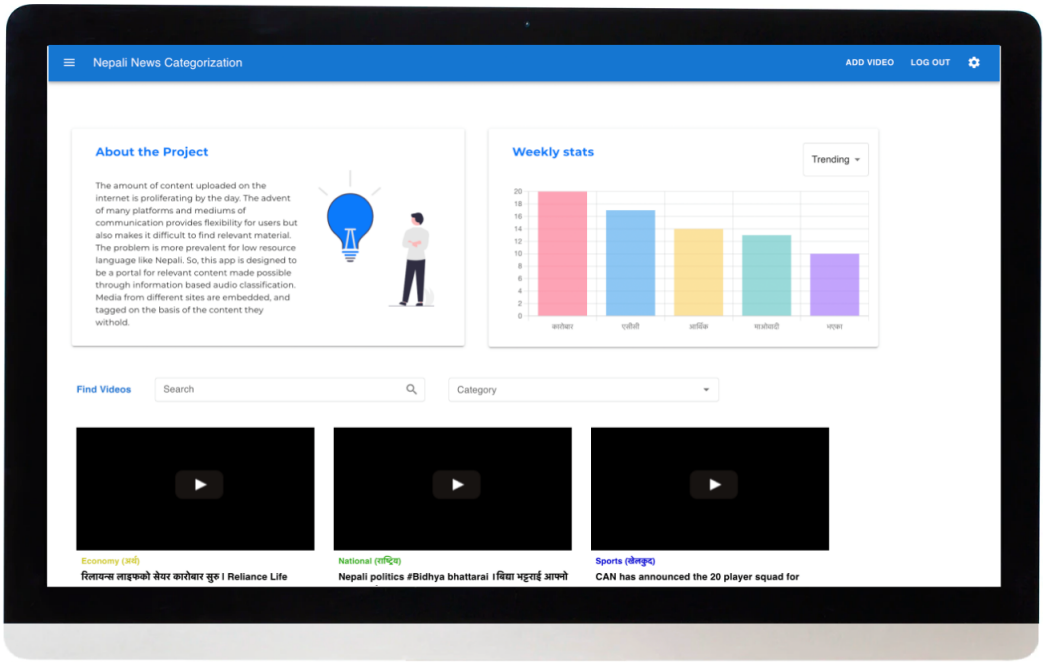

Figure 7.1: Web application dashboard

Figure 7.2: A video and it's transcriptions obtained from ASR model

8. DISCUSSION

With our ASR models, we obtained promising results for Nepali language compared to other models trained in supervised manner. But there were some constraints that limited our work. In case of semi-supervised approaches like we've used, large unlabelled datasets can make model learn low level representation very well. But it was difficult for us to get enough unlabelled data to pretrain our model with. Pretraining with enough unlabelled data would help us leverage greater benefits of using semi-supervised training techniques. Our results indicate that most of the pretrained models we started with gave almost similar results. To start with, CLSRIL-23 did not show very significant improvement compared to wav2vec2.0. It could possibly be because Nepali data is a very small part of its training data distribution(10.2). It is also not sure if the languages used in that distribution are close enough to Nepali language, despite being Indic. Further, the pretraining we added to models before finetuning gave us similar or slightly worse performance in test sets. This is because we did not have enough data for pretraining and also because we used the same data for pretraining as we used for finetuning to see if during pretraining, model can focus to capture some extra and gerenalizable details from the data. But it seems that this resulted in model trying more to memorize than the model learning any special and general features. Besides, due to limitations in hardware resources to train such huge models, we could not tune and retrain models to play around with hyperparameters. We mostly adopted the parameters recommended in the paper. We also noticed in many cases, the errors in transcripts were because of some typical features of grammar of Nepali language like *matras* and other symbols. Also, words derived from other languages don't have consistent pronunciation and translations in Nepali. We also noticed inconsistencies in use of spaces and *dhikas* in transcripts, especially in *naamyogi* words in Nepali. Such constraints of the language and inconsistencies in target transcripts has also resulted in large error rate on generated transcripts that are far more understandable to us upon reading than the scores indicate.

Similarly, in the case of audio classification by transcripts, we've obtained decent results but in certain cases the model assigns incorrect categories to audios. The reason behind this is the lack of effective training set. We've used individual sentences as a single data point in training set, however, a single sentence might not be able to represent a particular category that well. At present, our classification model supports only 5 categories. Even though we wanted to add more categories, we were constrained by the limited availability of labelled Nepali news data and the difficulty of manually labelling Nepali news to create a data corpus within a short time period.

9. CONCLUSION

In this project, we built a system of ASR for Nepali language followed by the classification of generated transcripts. For ASR, we collected data from various sources and preprocessed them using techniques like noise removal, change of sampling rate and number of channels. Different open source usable models and datasets were explored in the course of developing the system. Two pretrained models, wav2vec2.0 (pretrained previously in English only) and CLSRIL-23 (pretrained previously on 23 Indic languages) were used as base models for ASR. Unsupervised pretraining was added in both the models to develop two additional base models for finetuning. To understand the contribution of pretraining in learning low level representations better, we finetuned both the models once on the original versions without additional pretraining and then again with added pretraining. We used CTC training for finetuning these models and modified beam search for decoding. In order to improve the performance of ASR model, we also built different language models with different datasets. We evaluated the performance of these different models on three different test sets, each with three different language models and once without a language model. The use of Language Model gave better results for the models overall. We noticed that addition of pretraining by us did not bring any improvement in performance, probably due to limited amount of data. On the other hand, CLSRIL-23 without additional pretraining by us performed slightly better than wav2vec2.0 without additional pretraining by us, indicating that representations learnt during pretraining in Indic languages are indeed closer to Nepali than those learnt in model pretrained in English language only. But the difference was not very significant showing that there is more room for improvement to obtain a pretrained model, that captures low level representations corresponding to Nepali language better. The evaluation of model's performance was done using multiple test datasets which comprised of data from different distributions and speakers. Among the models used, the best results were seen in the model obtained by finetuning of CLSRIL-23(5.2.2), a model pretrained in 23 different Indic languages.

The output of ASR was further used for classification purpose, for which a classification model was built. For audio classification, a classification model was developed on top of MuRIL, an open source model. The audio samples provided to ASR yielded corresponding transcriptions which were then classified using the classification model. The trained model was evaluated using test dataset that consisted of sentences from different sources. The results showed a decent performance on classification task.

Finally, the overall system was presented using a web application.

10. LIMITATIONS AND FUTURE ENHANCEMENTS

10.1. Limitations

Despite our collective efforts, our final system is bounded by many limitations. Throughout the project development cycle, we had to work with limited computing resources. The models we used: wav2vec2, CLSRIL-23, MuRIL are large and complex models which require excessive computing resources if we are to train with huge dataset. In order to try learning better representations of audio, we had to perform pretraining of wav2vec2 and CLSRIL-23 with large amount of data but that requires multiple GPUs and huge computational power. Due to this reason and unavailability of usable and large dataset, we were bound to use a small dataset for pretraining which has limited the quality of our ASR model.

Our ASR model has many limitations regarding the quality of transcriptions it generates. The language models we built has certainly improved the quality of transcriptions. However, as we've built the language models from a small corpus, the transcriptions are often incorrect for less frequently used complex Nepali words and words primarily belonging to other languages.

The ASR model currently takes a significant time period for inference so there is a need for performance optimization. The ASR model cannot be used for generating transcripts of real-time audio/video.

Taking about the audio classification, the classification model is currently limited to 5 categories. Any audio whose content represents some other category shall get labelled as "Others". However, it is seen that in certain cases, such audios get labelled as one of the 5 categories which is a blunder by the classification model. This is a major limitation that needs to be worked upon.

10.2. Future Enhancements

Keeping in mind the above mentioned limitations, we aim to enhance our system in the following ways.

1. We can obtain better audio representations by pre-training self-supervised models like wav2vec2 and CLSRIL-23 with larger number of data. In the near future, we plan to collect raw Nepali audios of over 500 hours to pretrain those models in order to improve the quality of audio representations.

2. The language model can be improved by using huge Nepali text corpus of better quality and high variability.

3. The quality of transcriptions can be improved by implementing rules of Nepali grammar manually for certain cases where the ASR model usually fails.

4. In the near future we aim to make the ASR model applicable for real time audio and videos. This would require methods like building a lightweight model suitable for production environments, minimizing the time required to load the model by building a serving model, etc.

5. The classification model can be improved to support more than just five categories. A better approach is to allow admin to add new categories and certain data associated with that category, using the web application.

References

[1] Douglas Alan Reynolds. “A Gaussian mixture modeling approach to text-independent speaker identification”. PhD thesis. Georgia Institute of Technology, 1992.

[2] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. “Speaker verification using adapted Gaussian mixture models”. In: *Digital signal processing* 10.1-3 (2000), pp. 19–41.

[3] Biing Hwang Juang and Laurence R Rabiner. “Hidden Markov models for speech recognition”. In: *Technometrics* 33.3 (1991), pp. 251–272.

[4] Mark Gales and Steve Young. *The application of hidden Markov models in speech recognition*. Now Publishers Inc, 2008.

[5] Gin-Der Wu and Ying Lei. “A Register Array Based Low Power FFT Processor for Speech Recognition.” In: *J. Inf. Sci. Eng.* 24 (May 2008), pp. 981–991.

[6] Wiqas Ghai and Navdeep Singh. “Literature review on automatic speech recognition”. In: *International Journal of Computer Applications* 41.8 (2012).

[7] Tara N. Sainath et al. “Deep Convolutional Neural Networks for Large-scale Speech Tasks”. In: *Neural networks : the official journal of the International Neural Network Society* 64 (2015), pp. 39–48.

[8] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. “Speech Recognition with Deep Recurrent Neural Networks”. In: *arXiv e-prints*, arXiv:1303.5778 (Mar. 2013), arXiv:1303.5778. arXiv: 1303.5778 [cs.NE].

[9] Alex Graves et al. “Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks”. In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML ’06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 369–376. ISBN: 1595933832. DOI: 10.1145/1143844.1143891. URL: https://doi.org/10.1145/1143844.1143891.

[10] Jan Chorowski et al. *End-to-end Continuous Speech Recognition using Attention-based Recurrent NN: First Results*. 2014. DOI: 10.48550/ARXIV.1412.1602. URL: https://arxiv.org/abs/1412.1602.

[11] Chetan Prajapati et al. “Nepali Speech Recognition”. In: *Kathmandu. DOECE, IOE* (2008).

[12] Manish K Ssarma et al. “HMM based isolated word nepali speech recognition”. In: *2017 International Conference on Machine Learning and Cybernetics (ICMLC)*. Vol. 1. IEEE. 2017, pp. 71–76.

[13] Paribesh Regmi, Arjun Dahal, and Basanta Joshi. “Nepali speech recognition using rnn-ctc model”. In: *International Journal of Computer Applications* 178.31 (2019), pp. 1–6.

[14] Bharat Bhatta, Basanta Joshi, and Ram Krishna Maharjhan. “Nepali Speech Recognition Using CNN, GRU and CTC”. In: *Proceedings of the 32nd Conference on Computational Linguistics and Speech Processing ({ROCLING} 2020)*. 2020, pp. 238–246.

[15] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and their Compositionality”. In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[16] Yu-An Chung et al. “Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder”. In: *arXiv preprint arXiv:1603.00982* (2016).

[17] Da-Rong Liu et al. “Completely unsupervised phoneme recognition by adversarially learning mapping relationships from audio embeddings”. In: *arXiv preprint arXiv:1804.00316* (2018).

[18] Steffen Schneider et al. “wav2vec: Unsupervised pre-training for speech recognition”. In: *arXiv preprint arXiv:1904.05862* (2019).

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

[20] Alexei Baevski et al. “wav2vec 2.0: A framework for self-supervised learning of speech representations”. In: *arXiv preprint arXiv:2006.11477* (2020).

[21] Alexis Conneau et al. “Unsupervised cross-lingual representation learning for speech recognition”. In: *arXiv preprint arXiv:2006.13979* (2020).

[22] Anirudh Gupta et al. “CLSRIL-23: cross lingual speech representations for indic languages”. In: *arXiv preprint arXiv:2107.07402* (2021).

[23] Yves Raimond et al. “Automated semantic tagging of speech audio”. In: *Proceedings of the 21st International Conference on World Wide Web*. 2012, pp. 405–408.

[24] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[25] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[26] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https : / / proceedings . neurips . cc / paper / 2017 / file / 3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[27] Simran Khanuja et al. “Muril: Multilingual representations for indian languages”. In: *arXiv preprint arXiv:2103.10730* (2021).

[28] Bhargav Dave, Shripad Bhat, and Prasenjit Majumder. “IRNLP_DAIICT@ DravidianLangTech-EACL2021: offensive language identification in Dravidian languages using TF-IDF char n-grams and MuRIL”. In: *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*. 2021, pp. 266–269.

[29] Arthur L. Samuel. “Some studies in machine learning using the game of Checkers”. In: *IBM JOURNAL OF RESEARCH AND DEVELOPMENT* (1959), pp. 71–105.

[30] Thalles Santos Silva. “A Few Words on Representation Learning”. In: *https://sthalles.github.io* (2021). URL: https : / / sthalles . github . io / a - few - words - on-representation-learning/.

[31] Vassil Panayotov et al. “Librispeech: an asr corpus based on public domain audio books”. In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2015, pp. 5206–5210.

[32] Oddur Kjartansson et al. “Crowd-sourced speech corpora for javanese, sundanese, sinhala, nepali, and bangladeshi bengali”. In: (2018).

[33] Keshan Sodimana et al. “A Step-by-Step Process for Building TTS Voices Using Open Source Data and Framework for Bangla, Javanese, Khmer, Nepali, Sinhala, and Sundanese”. In: *Proc. The 6th Intl. Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU)*. Gurugram, India, Aug. 2018, pp. 66–70. URL: http://dx.doi.org/10.21437/SLTU.2018-14.

[34] Daniel S Park et al. “Specaugment: A simple data augmentation method for automatic speech recognition”. In: *arXiv preprint arXiv:1904.08779* (2019).

[35] Ilya Loshchilov and Frank Hutter. “Fixing weight decay regularization in adam”. In: (2018).

APPENDIX-A

CLSRIL-23 Data Distribution

The language distribution on train and valid sets in CLSRIL-23 is shown below.

Language	Time in hours	
	Train	**Valid**
Assamese	254.9	1.2
Bengali	331.3	1.7
Bodo	26.9	0.13
Dogri	17.1	0.07
English	819.7	3.7
Gujarati	336.7	1.7
Hindi	4563.7	23.52
Kannada	451.8	2.14
Maithili	113.8	0.6
Konkani	36.8	0.19
Malayalam	297.7	1.56
Manipuri	171.9	0.9
Marathi	458.2	2.2
Nepali	31.6	0.18
Odia	131.4	0.63
Punjabi	486.05	2.53
Santali	6.56	0.03
Tamil	542.6	2.67
Telugu	302.78	1.41
Urdu	259.68	1.20
Kashmiri	67.8	0.37
Sanskrit	58.8	0.30
Total	**9767.77**	**48.93**

Table 10.1: CLSRIL-23 data distribution

Application for Supervised data Collection

With an aim of building a speech data corpus for Nepali language, we have built a web interface. The web interface displays a certain sentence to the user and the user can record the audio for the corresponding sentence. In order to facilitate audio classification, we've added options so that the user can select categories that best represent that particular sentence. After the user submits it, the audio as well as the categories representing that audio gets recorded in our database. Figure 10.1 shows the web interface for this purpose.

This facilitates building a data corpus for Nepali speech which helps to record large amount of audios as well as categories for each audio.

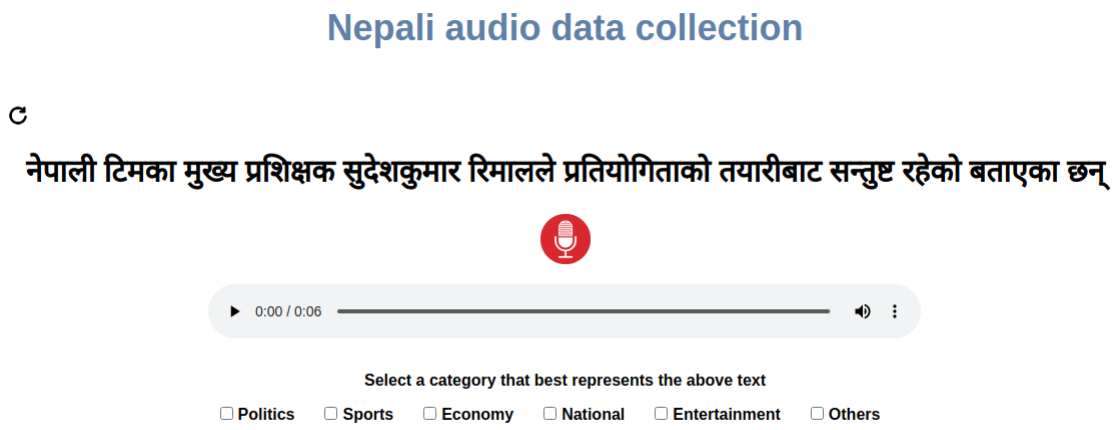

Figure 10.1: Application for supervised data collection of Nepali speech

APPENDIX-B

Pretraining Logs

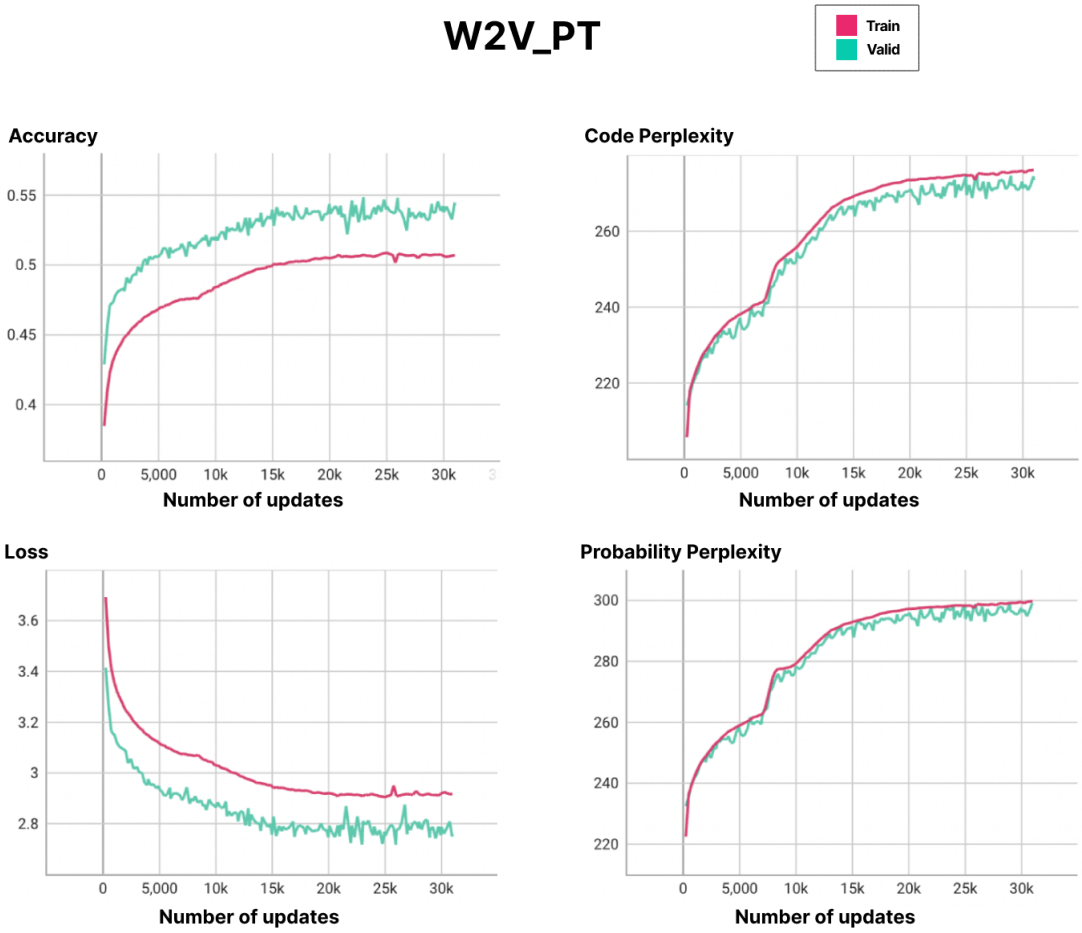

Figure 10.2: Comparison of train and valid statistics for W2V_PT

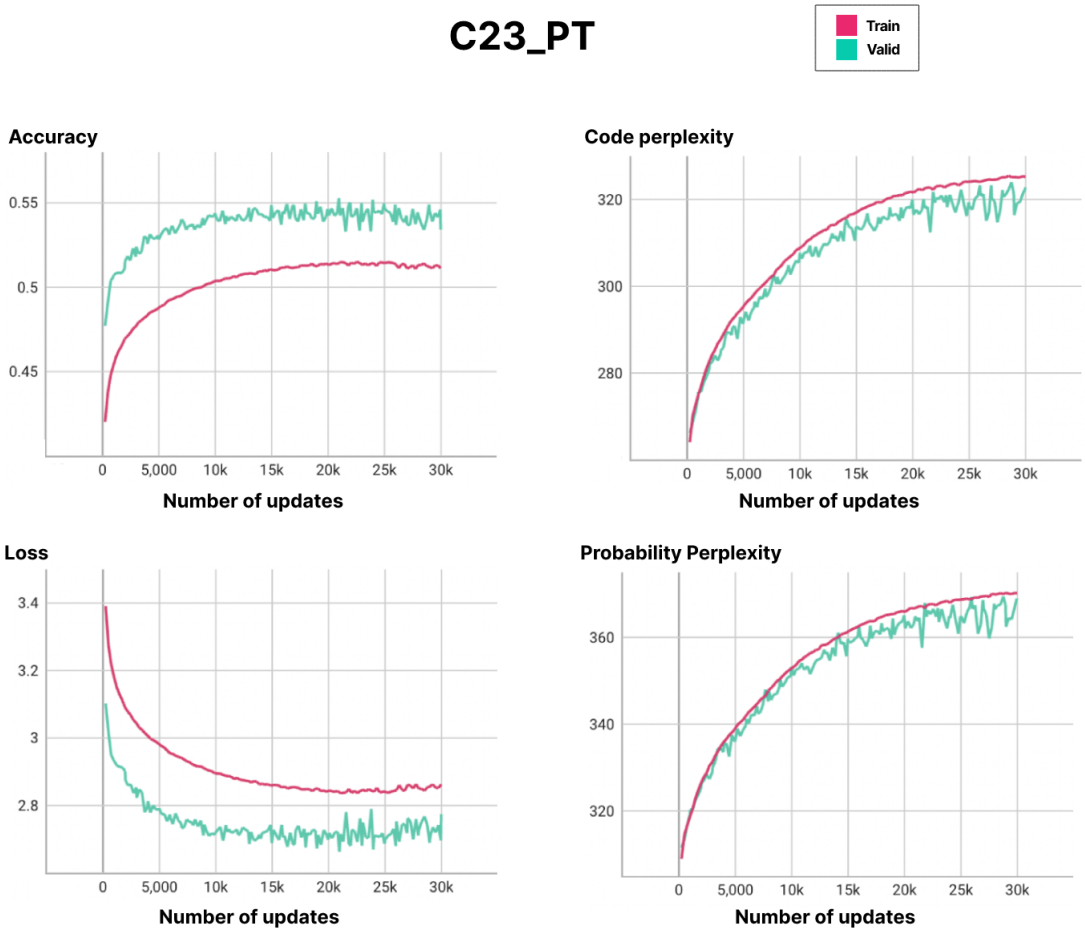

Figure 10.3: Comparison of train and valid statistics for C23_PT

Finetuning Logs

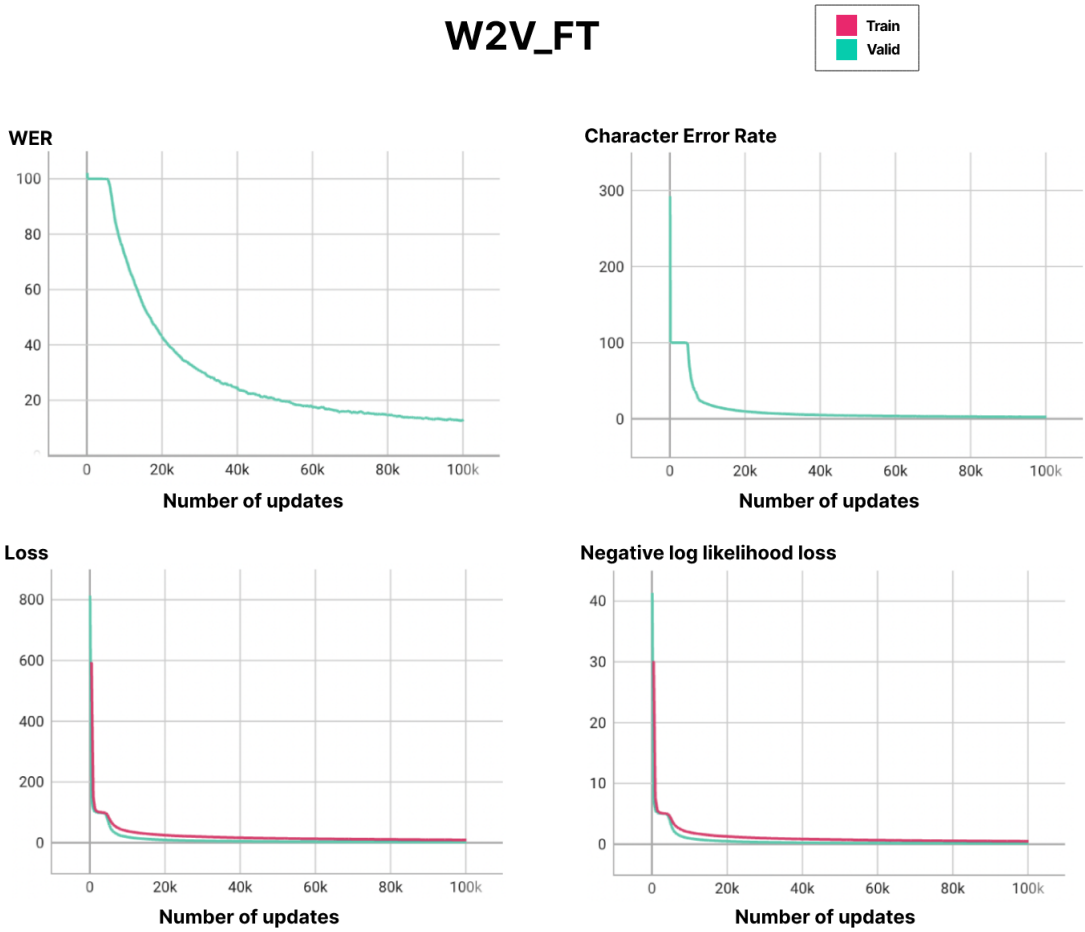

Figure 10.4: Comparison of train and valid statistics for W2V_FT

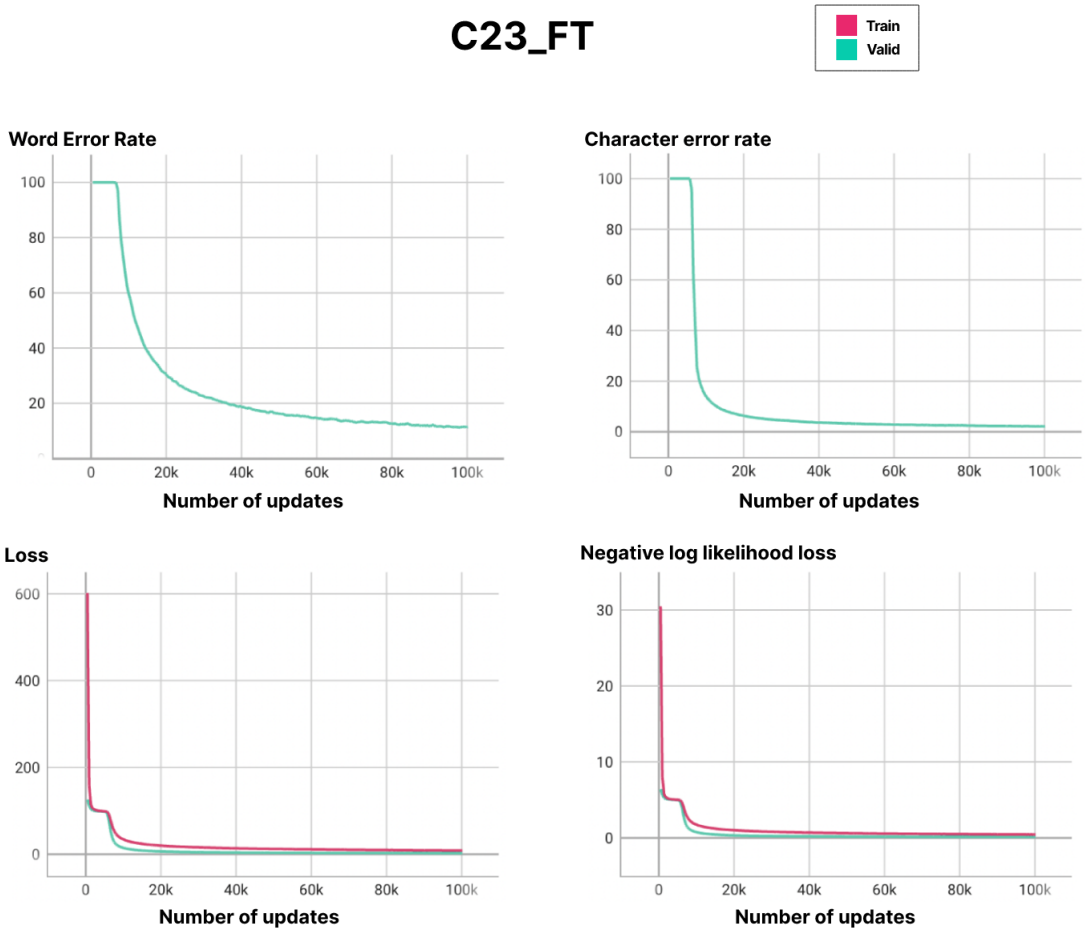

Figure 10.5: Comparison of train and valid statistics for C23_PT

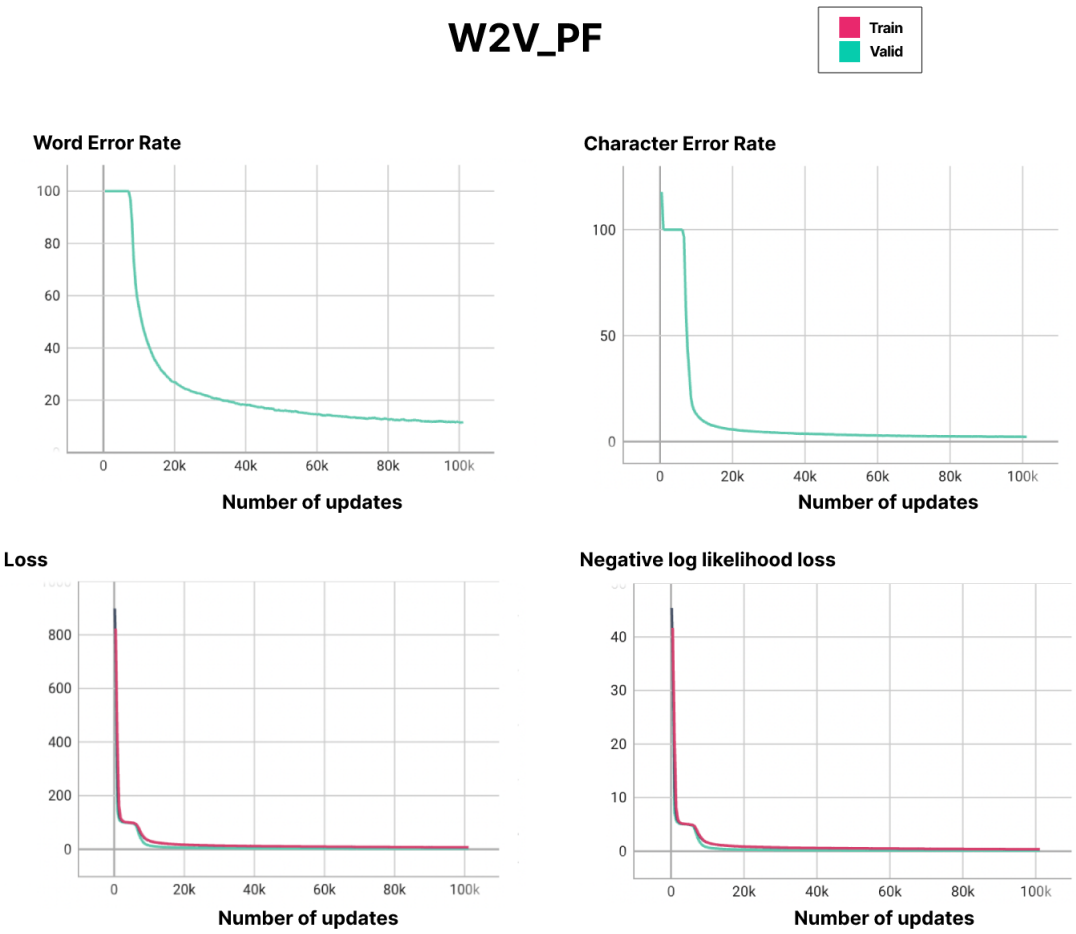

Figure 10.6: Comparison of train and valid statistics for W2V_PF

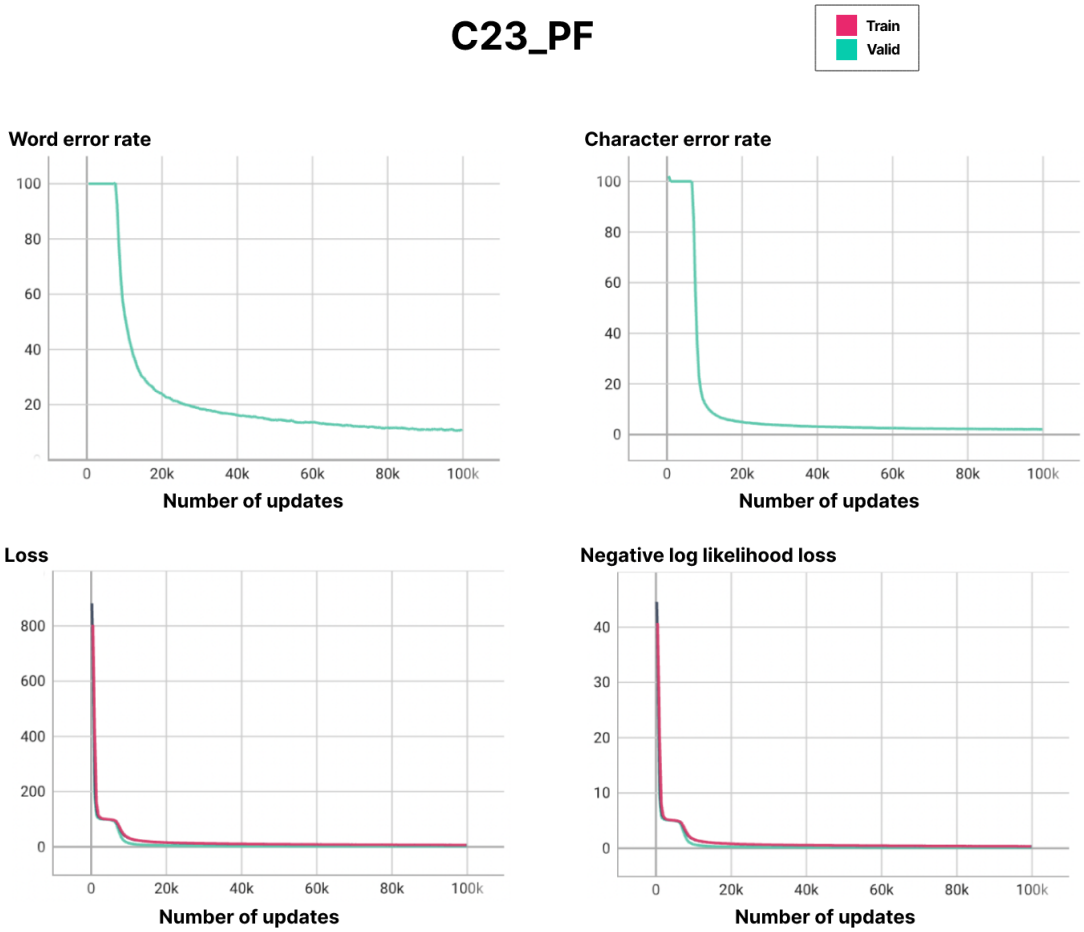

Figure 10.7: Comparison of train and valid statistics for C23_PF

Text classification Logs

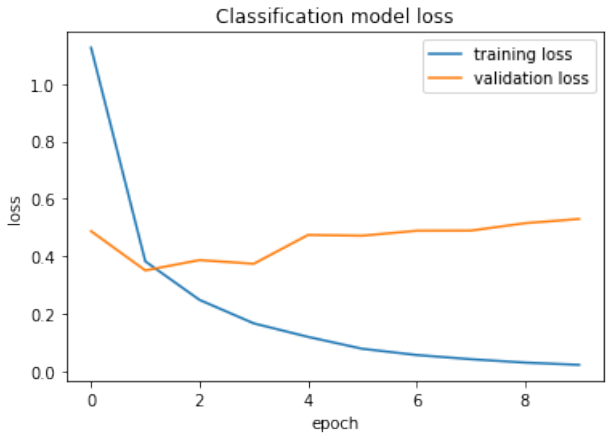

Figure 10.8: Cross-entropy loss vs epoch comparison for text classification

APPENDIX-C

Variation of WER and CER in different language models

The variation of WER and CER when different language models were used as shown in the table below.

Model	LM	Test Dataset					
		OpenSLR test		OpenSLR female		074BCT	
		WER	CER	WER	CER	WER	CER
C23_FT	3-gram	2.36	0.69	**23.02**	**6.54**	**40.98**	10.17
	4-gram	2.23	0.67	23.04	6.55	**40.98**	**10.16**
	5-gram	**2.16**	**0.65**	23.05	**6.54**	**40.98**	**10.16**
	6-gram	**2.16**	**0.65**	**23.02**	**6.54**	41.01	10.17

Table 10.2: WER and CER variation with LM

Original	Speaker	No LM	Combined 5-gram model
यसका लागि पानी उमाल्नुपर्छ छान्नुपर्छ वा अल्ट्राभोइलेट वा रिवर्स ओस्मोसिस फिल्टरले सफा गर्नुपर्छ	Speaker 1	यसका लागि पानी उमाल्नुपर्छ छान्नुपर्छ वा अर्ड्राहोइलेट वा रिवर्स अस्मोसिस फिल्टरले सफा गर्नुपर्छ	यसका लागि पानी उमाल्नु पर्छ छाड्नुपर्छ वा अल्ट्रावायलेट वा रिभर्स अस्मि फिल्टरले सफा गर्नुपर्छ
	Speaker 2	यसका लागि पनि उमाल्नुपर्छ छाड्नुपर्छ वा अल्ट्राभइलेट वा रिभर्स असमसिस फिल्टरले सफा गर्नुपर्छ	यसका लागि पनि उमाल्नु पर्छ छाड्नुपर्छ वा अल्ट्रावायलेट वा रिभर्स अस्मि फिल्टरले सफा गर्नुपर्छ
	Speaker 3	यसका लागि पानी उमाल्नुपर्छ छान्नुपर्छ वा अल्ट्राभयलेट वा रिवर्ज अस्मोसिस फेल्टरले सफा गर्नुपर्छ	यसका लागि पानी उमाल्नु पर्छ छाड्नुपर्छ वा अल्ट्रावायलेट वा रिभर्स अस्मि फिल्टरले सफा गर्नुपर्छ
	Speaker 4	यसका लागि पानी उमाल्नुपर्छ छान्नुपर्छ वा अल्ट्राभोइलेट वा रिभर्स अस्मोसिस फिल्टरले सफा गर्नुपर्छ राा	यसका लागि पानी उमाल्नु पर्छ छाड्नुपर्छ वा अल्ट्रावायलेट वा रिभर्स अस्मि फिल्टरले सफा गर्नुपर्छ र
	Speaker 5	यसका लागि पानी उमाल्नुपर्छ छाड्नुपर्छ वा अल्ट्रा भोइलेट वा रिबर्स उस्मसिसपिल्ट रले सफा गर्नुपर्छ	यसका लागि पानी उमाल्नु पर्छ छाड्नुपर्छ वा अल्ट्रा भाइले वा रिभर्स कसमस सेल्टा ले सफा गर्नुपर्छ
अर्का पूर्व कप्तान शाक्य भने उपाधि जित्दैमा मक्ख परेर बस्न नहुने बताउँछन्	Speaker 1	अर्का पूर्व कप्तान शाक्य भने उपाधि जित्दैमा मख्ख पर र बस्न नहुने बताउँछन्	अर्का पूर्व कप्तान शाक्य भने उपाधि जित्दैमा मख्ख परेर बस्न नहुने बताउँछन्
	Speaker 2	अर्का पूर्व कप्तान शाक्य भने उपाधि जित्दैमा मक्क परेर बस्न नहुने बताउँछन् कूर्	अर्का पूर्व कप्तान शाक्य भने उपाधि जित्दैमा मक्ख परेर बस्न नहुने बताउँछन्
	Speaker 3	अर्का पूर्व कप्तान श्राख्य भनी उपाधि जितैमा मक्ख परेर बस्न नहुने बताउँछन् कू	अर्का पूर्व कप्तान शाक्य भनी उपाधि जितमा मक्ख परेर बस्न नहुने बताउँछन्
	Speaker 4	अर्का पूर्व कप्तान शाक्य भने उपाधि चित्दैमा मक्ख परेर बस्न नहुने बताउँछन् पू	अर्का पूर्व कप्तान शाक्य भने उपाधि जित्दैमा मख्ख परेर बस्न नहुने बताउँछन्
	Speaker 5	अर्घा पूर्वकप्तान शाक्य भने उपाधि जित्दैमा मक्ख परेर बस्न नहुने बताउँछन्	अर्का पूर्व कप्तान शाक्य भने उपाधि जित्दैमा मक्ख परेर बस्न नहुने बताउँछन्

Table 6.8: Speakerwise performance on each sentence using no LM and combined 5-gram LM on top of C23_FT model. The words in the original text that are not present in the LM are highlighted in orange whereas the mistaken words are highlighted in magenta.