USING RECONTRUCTED PHASE SPACES TO PERFORM AUTOMATED SPEECH RECOGNITION FOR HINDI LANGUAGE

Submitted in partial fulfilment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning

by

HITESH GOYAL (19BAI1129)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2023

USING RECONTRUCTED PHASE SPACES TO PERFORM AUTOMATED SPEECH RECOGNITION FOR HINDI LANGUAGE

Submitted in partial fulfilment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning

by

HITESH GOYAL (19BAI1129)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2023

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

DECLARATION

I hereby declare that the thesis entitled "USING RECONTRUCTED PHASE SPACES TO PERFORM AUTOMATED SPEECH RECOGNITION FOR HINDI LANGUAGE" submitted by me, for the award of the degree of Bachelor of Technology in Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning, Vellore Institute of Technology, Chennai, is a record of bonafide work carried out by me under the supervision of Guide Name

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai

Date: 24/04/2023

Signature of the Candidate

VIT®

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled **"USING RECONTRUCTED PHASE SPACES TO PERFORM AUTOMATED SPEECH RECOGNITION FOR HINDI LANGUAGE"** is prepared and submitted by **Hitesh Goyal (19BAI1129)** to Vellore Institute of Technology, Chennai, in partial fulfilment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning** programme is a bonafide record carried out under my guidance. The project fulfils the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Signature of the Guide:

Name: Dr. Nagreemulla Khan

Date: 25/4/23

Signature of the Examiner 1

Name: Dr. NT. Sureshkumar

Date: 26/04/23

M. Keerthivasan

Signature of the Examiner 2

Name: KEERTHIVASAN MADURAI -VERIZON

Date: 26/04/2023

Approved by the Head of Department
B. Tech. CSE with Specialization in Artificial Intelligence and Machine Learning

Name: Dr. Sweetlin Hemalatha C

Date: 24 – 04 – 2023

(Seal of SCOPE)

ABSTRACT

Automated speech recognition (ASR) is a challenging task, especially for low-resource languages like Hindi. ASR has evolved significantly over the years, with earlier approaches relying on Spectrograms, Mel Frequency Cepstral Coefficients (MFCCs), and other similar features classified using Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs). With advances in computational capacity, data availability, and research, these models have been replaced by more powerful models, including neural networks like artificial neural networks, reconstructed neural networks, and transformers.

Despite these advancements, there is still room for improvement in ASR models, particularly for Hindi language. In this paper, we propose a new approach to ASR using reconstructed phase spaces (RPS) to extract robust features from speech signals. Our approach leverages the non-linear dynamics of speech to capture the temporal structure of spoken words to improve ASR performance.

The Devanagari script used to write Hindi consists of sound units that are logically divided in terms of the source of articulation and the amount of affrication. Previous research has shown potential for RPS to provide information about articulation and affrication, making it a promising approach for training ASR models on Hindi.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr. A Nayeemulla Khan, Professor, SCOPE, Vellore Institute of Technology, Chennai, for his constant guidance, continual encouragement, understanding; more than all, he/she taught me patience in my endeavor. My association with him is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Deep Learning and Speech Processing.

It is with gratitude that I would like to extend thanks to our honorable Chancellor, Dr. G. Viswanathan, Vice Presidents, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan and Mr. G V Selvam, Assistant Vice-President, Ms. Kadhambari S. Viswanathan, Vice-Chancellor, Dr. Rambabu Kodali, Pro-Vice Chancellor, Dr. V. S. Kanchana Bhaaskaran and Additional Registrar, Dr. P.K. Manoharan for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to Dean, Dr. Ganesan R, Associate Dean Academics, Dr. Parvathi R and Associate Dean Research, Dr. Geetha S, SCOPE, Vellore Institute of Technology, Chennai, for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr. Sweetlin Hemalatha C, Head of the Department, Project Coordinators, Dr. Padmavathy T V, Dr. Abdul Quadir Md, and Dr. Priyadarshini R, B.Tech. Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning, SCOPE, Vellore Institute of Technology, Chennai, for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staff at Vellore Institute of Technology, Chennai, who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Place: Chennai

Date: **24/04/2023** **Hitesh Goyal**

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

RPS	Reconstructed Phase Space
ASR	Automated Speech Recognition
MI	Mutual Information
DL	Deep Learning
NN	Neural Network
ANN	Artificial Neural Network
kNN	k-Nearest Neighbors
SVM	Support Vector Machine
ML	Machine Learning
MFCC	Mel Frequency Cepstral Coefficient
HMM	Hidden Markov Model
GMM	Gaussian Mixture Model
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
CTC	Connectionist Temporal Classification
CVP	Consonant-Vowel Pair
ResCNN	Residual Convolutional Neural Network
BiLSTM	Bidirectional Long Short-Term Memory

Chapter 1

Introduction

1.1 INTRODUCTION

1.1.1 RECONSTRUCTED PHASE SPACES

Reconstructed Phase Spaces (RPS) is a nonlinear data analysis technique that relies on the Poincare recurrence theorem. This theorem establishes that certain dynamical systems will eventually return to a state either exactly the same as or arbitrarily close to their initial state after a finite period. An RPS plot can be created by plotting a system $f(x)$ at a time t with itself at another time $t+d$, where d is a delay value.

1.1.2 AUTOMATED SPEECH RECOGNITION

Automated Speech Recognition (ASR) is a speech processing technique used to identify spoken language in an audio recording and transcribe it into text format. Many ASR systems rely on two models: a speech model which extracts speech features, and a language model which processes these features into relevant words based on context and pronunciation patterns.

State-of-the-art ASR systems are now able to achieve high levels of accuracy and are used in various applications such as virtual assistants, transcription services, and language learning platforms.

1.1.3 DEEP LEARNING AND NEURAL NETWORKS

Deep Learning techniques and Neural Networks are currently the most widely used algorithms in many different fields. They are used for a wide range of tasks, such as speech processing, object detection, stock market prediction, and disease identification, among others. Deep Learning techniques have also shown great potential in natural language processing tasks such as sentiment analysis and language translation. We propose to use Neural Networks due to their highly versatile nature and ability to analyze images and sequential data.

1.2 OVERVIEW

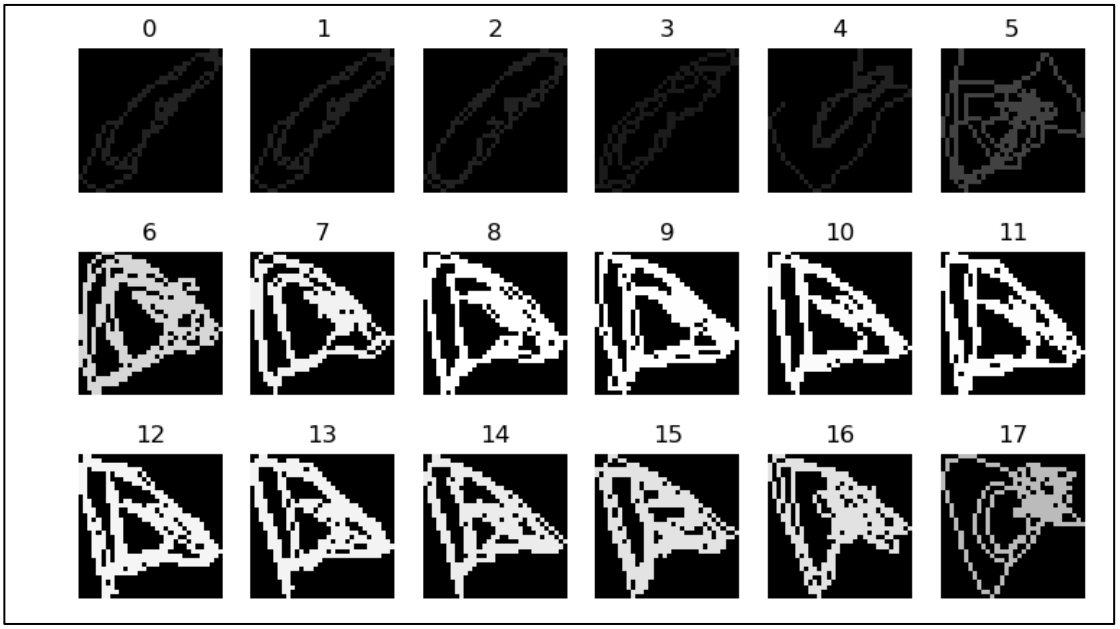
1.2.1 AN OVERVIEW OF RPS APPLICATIONS

Plots generated using RPS analysis are a powerful tool in the analysis of various systems, such as weather patterns, physiological signals, and financial data. Recently, RPS analysis has also been applied to speech processing, with the aim of identifying different units of speech such as phonemes or graphemes.

By choosing the optimal delay value, RPS plots generated can be informative in identifying speech units. These RPS portraits generate patterns that can form relationships with each other, which can be mapped onto the articulation and sound source.

Given below is a figure showing RPS visualization of the sound “Ba”.

Figure 1. Visualization of RPS portrait sequences



1.2.1 CHOICE OF OPTIMAL DELAY

In order to obtain the most informative RPS plot, the optimal delay value must be determined. Mutual Information (MI) can be used to identify the optimal delay value because it measures the dependency between a signal and its delayed version. A range of delay values are used to calculate MI and the first minima is selected as the most effective delay value. This process is repeated for several signals from the dataset, and the delay value with the most corresponding signals is selected.

1.2.2 CHOICE OF DIMENSIONS

The dimensionality of an RPS can also affect its informativeness. False Nearest Neighbors is a method that has been used in earlier studies to identify the best embedding dimension. However, for our study, we use a two-dimensional RPS plot as it can be used with NNs without more complexity or computational requirements.

1.2.3 DIRECT UTILIZATION OF RPS PORTRAITS

Previous studies have largely utilized derived features such as Maximum Lyapunov Exponents, Fractal dimensions, and Diagonal Features in the application of RPS in speech recognition. However, such indirect use of RPS portraits fails to provide a comprehensive understanding of speech units. Instead, our paper proposes to utilize the RPS portraits directly to identify speech units. By utilizing the actual RPS portraits, our method allows for a more accurate and thorough analysis of speech signals.

1.2.5 SMALLEST UNITS OF SPEECH

Phonemes, are studied under the fields of phonology and phonetics. Phonology deals with the distribution and use of phonemes in different languages, whereas phonetics takes a language-independent approach.

1.2.6 EARLY ASR

Earlier research in speech processing employed phonemes as units to perform classification and identification. This was achieved by using Mel Spectrograms and MFCCs, among other features, on Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs). Before being able to classify the sound units, these ASR systems heavily relied on alignment-based techniques. They used HMMs to identify Triphones, a sequence of three phonemes which were then classified in an unsupervised approach using GMMs.

1.2.7 CONTEMPORARY ASR SYSTEMS

Currently, advanced ASR systems still utilize Mel Spectrograms and MFCCs among other features for speech feature extraction. However, these systems now employ neural networks such as Convolutional Neural Networks (CNNs) to extract spatial

and temporal features, followed by Long Short-Term Memory (LSTM) or Transformers to extract sequential dependencies and language-related information.

These ASR systems are now trained using an alignment-free technique, employing the Connectionist Temporal Classification (CTC) loss function. The CTC loss function sums over all possible alignment probabilities. To avoid unnecessary computation, the loss computation takes into consideration sequential dependencies as a critical constraint. Since transcribed audio data is much more readily available compared to audio data with phoneme alignments, it is much easier to train ASR systems using the CTC loss function.

1.2.8 TRAINING ASR SYSTEMS

To train ASR systems, a large amount of speech data is needed. This data is typically transcribed using either manual or automatic transcription. The transcription data is then used to train the ASR system to recognize and classify spoken words.

1.3 CHALLENGES

1.3.1 CHALLENGES IN AUDIO PROCESSING

Audio data is saved as a simple array, but the amount of information one can and must extract from it is huge. This means that one must have a lot of knowledge about audio, speech, and signal processing to carry out a colossal task like ASR.

Audio from different sources may have different types of noises and a difference in microphone quality can hugely affect the audio data itself. Ensuring the usage of correct audio data and using appropriate techniques to normalize audio waves is very important.

Most audio processing systems are not perfect. This means small errors from the audio processing systems will take a toll on the overall research.

1.3.2 CHALLENGES IN RPS

RPS techniques have been used sparingly in the field of signal processing. They have been used even more sparingly in the field of speech processing. So much so that there are very limited studies which can be used as benchmarks or guiding systems to explore its full potential.

The lack of a decent code base calls for more custom code which needs to be made and used correctly to ensure any fruitful results. This can be very challenging especially considering the number of minor nuances one may come across.

A simple example of a nuance is selecting the appropriate delay value. When developing the code for delay value, we saw that most codes were not available in Python language and the ones that were available needed some refactoring to ensure correct usage.

RPS plots are made from plotting a signal with its delayed form. This means that we need a line-drawing algorithm to do so. The algorithms for line-drawing need a much heavier computation as compared to feature extraction techniques like MFCCs and Mel-Spectrograms which makes the process long and tedious.

In a relatively faster line-drawing algorithm the number of parameters available for the line-drawing are limited because of which the RPS plots tend to lose some information which may have been useful for prediction.

Vowels tend to have larger amplitudes as compared to consonants or noise. This means that the RPS plot for a consonant will have a smaller range as compared to a vowel. When we normalize the ranges completely, noise also gains as much importance in prediction as the consonants and vowels which can hinder with the models' learning.

Another challenge in RPS plots is the consideration of window size and stride length to generate the portrait sequence. If the window size and stride length are too small, the RPS portraits do not have enough of a structure to be identified as a particular sound unit. On the other hand, a larger window and stride may see a vowel and consonant in the same portrait because of which the consonant can become too small to give appropriate features.

1.3.3 CHALLENGES IN DEFINING THE ARCHITECTURE

Defining the architecture of neural network (NN) models for state-of-the-art speech recognition systems like Wave2Vec is a difficult task. These NN architectures are well-defined for Mel-Spectrogram and MFCC features, including the dimensions processed by NNs, the flow of data through the NN, and the final output.

However, creating a completely new architecture that ensures the proper processing of dimensions using CTC loss along with RPS is a major challenge. Even with the currently proposed architectures, it is unclear whether the method of processing dimensions or using data is optimal.

NN architectures require various parameters like activations, dropouts, normalizations, and regularizations for them to learn effectively. Other hyperparameters like the network size, learning rate schedule and optimizer also add to the complexity of the training.

1.3.4 CHALLENGES IN ASR

Automatic speech recognition (ASR) is an intricate task with various challenges. The identification of phonemes, the smallest unit of speech, is a fundamental challenge, and even in aligned data, errors can occur. Combining multiple phonemes and understanding transitions from one phoneme to the next add significant complexity to ASR. Variability can arise in phoneme pronunciation due to triphone sets, making phoneme identification difficult.

Word classification is another challenge, as the vocabulary of most languages is too extensive to create a direct classifier. It is more reasonable to break the word down into phonemes in audio and graphemes in text for ASR. However, the pronunciation of many words can differ slightly from their corresponding phoneme or have spelling differences.

Sentence classification in continuous speech is even more complex, as certain sounds can be over-emphasized, and speech units can be left out. Language models can aid in sentence level context identification, but ASR datasets vary in microphone quality, accents, and speaking speeds, adding further complexity to the task.

1.3.5 CHALLENGES IN DEFINING LANGUAGE

Redesigning the language structure in Hindi to solve the language barrier issue is a challenging task. Unlike English, which has 26 alphabets but inconsistent spelling and pronunciation, Hindi has consistent spoken and written forms, but complications arise from half letters, combining vowels, and word morphology.

1.3.6 COMPUTATION CHALLENGES IN RPS-BASED SPEECH RECOGNITION

Computation is a key challenge in the development of machine learning (ML) and deep learning (DL) projects, and it can have a significant impact on the performance of a model. In particular, the computation requirements for RPS-based speech recognition are substantial and can cause major bottlenecks in the training process. This is due to the lack of optimization for RPS, which requires a large amount of computation to generate RPS portraits for each audio file. Additionally, the need to continuously augment RPS features further complicates the process by requiring on-the-fly RPS generation, which adds to the training time.

Moreover, the memory and GPU capacity required for RPS-based speech recognition are substantial, and can make all the difference in quickly tuning a model for improvement. Each RPS portrait also requires significant memory to store, which multiplies the memory requirements for a single audio file. These computational challenges must be addressed to develop efficient and effective RPS-based speech recognition models.

1.4 PROJECT STATEMENT

1.4.1 PROBLEM DEFINITION

Automatic Speech Recognition (ASR) has been a prominent field of research for decades now. Despite the significant progress made in the field, achieving a high level of accuracy in ASR for all languages remains a challenging problem. Additionally, the lack of ASR systems trained on RPS features makes it difficult to determine the feasibility of using RPS for Hindi ASR. The objective of this project is to develop a deep learning-based ASR system for Hindi that utilizes RPS features.

1.4.2 IMPORTANCE OF THE PROJECT

This project is significant because it tackles a challenging problem in the field of ASR and CVP classification. The use of RPS features is a novel approach to Hindi ASR, and no published study has ever used RPS features as directly as this project.

Moreover, this project aims to contribute to the development of the Hindi language, which has a rich cultural heritage and a large population of speakers. The development of an accurate and efficient ASR system for Hindi can help in preserving

and promoting the language in various domains such as education, literature, and entertainment.

1.4.3 METHODOLOGY

The proposed methodology involves several stages, including data collection, preprocessing, feature extraction, modeling, and evaluation. The data collection involves gathering a large and diverse dataset of Hindi speech samples that include variations in speaker gender, age, and accent. The data is then preprocessed to remove noise, silence, and other artifacts.

The feature extraction stage involves using the RPS technique to extract features from the preprocessed audio signals. RPS features provide a more informative representation of speech signals, making them ideal for ASR systems. The RPS features are fed into a deep neural network (DNN) that is trained to transcribe the speech into text. The DNN model architecture is designed to optimize the performance of the ASR system.

In addition to the ASR system, the project aims to develop a CVP classification system using the same RPS features. The CVP classification system should be able to classify the speaker's gender and age with high accuracy. The CVP classification system uses a separate DNN model architecture that is trained on a subset of the data used for the ASR system. The CVP classification system can help in several applications such as call center management, security systems, and speech-based personal assistants.

The proposed methodology will be evaluated on several metrics such as accuracy and F1-score. The evaluation will be done on a separate test set, which is not used during the training process, to ensure the generalization ability of the system.

In conclusion, this project aims to develop an accurate and efficient ASR system for Hindi that utilizes RPS features. The system's ability to classify the speaker's gender and age adds value to the ASR system, making it applicable in several domains. The project's novelty lies in the use of RPS features, which have not been used in Hindi ASR systems to date. The proposed methodology is expected to achieve high accuracy and provide valuable insights for further research in the field of ASR and CVP classification.

1.5 OBJECTIVES

1.5.1 RPS-BASED VOWEL CLASSIFICATION

This research presents a simple Convolutional Neural Network (CNN) model that classifies RPS portraits of different vowels with high accuracy. The CNN model is used to demonstrate the potential of RPS in speech analysis. The CNN model's different layers can extract spatial information, which is then passed into a classifier that predicts one of the five vowel classes.

As vowels are more straightforward to distinguish and can be partially separated from the rest of the sound, creating a simple image classifier on the RPS portrait of a vowel is achievable.

1.5.2 CVP CLASSIFICATION WITH RPS

Classifying a Consonant-Vowel Pair (CVP) is more complex than vowel classification, as it requires analyzing more than one RPS portrait. We propose a CNN-LSTM based architecture that accurately identifies CVPs. The CNN module reduces the image dimensions while extracting important spatial features, and the LSTM model extracts all sequential dependencies from the sequence of features. Consonants can be composed of more than one phoneme, and accurate identification requires paying close attention to the transition between phonemes.

1.5.3 ASR USING RPS

Our proposed CNN-LSTM architecture transcribes continuous speech audio into text, aiming to identify speech units from continuous speech and accurately transcribe them into words and sentences. Despite several challenges and scope for improvement, the current results show promise in the use of RPS for ASR. To train a model on the Hindi language, we identified the various graphemes of the language in Hindi phonology and how they translate to the phonemes and phonetics. Doing so helped ensure that the model was being trained on appropriate features and that the model did not have any inconsistencies or redundancies. A rule-based decoder was also created to normalize the text before model training.

The primary objective of our research is to demonstrate the potential of RPS in speech analysis by using it for vowel and CVP classification and ASR. Our study's

uniqueness lies in the direct application of RPS features for ASR systems trained on the Hindi language. Our study's significance lies in the possibility of achieving high accuracy in speech analysis tasks, which is critical in developing more efficient and accurate speech technologies.

We believe that our proposed models can achieve better results by optimizing various hyperparameters and model architectures. This study can serve as a foundation for more advanced and diverse research in the field of speech analysis using RPS.

1.6 SCOPE OF PROJECT

1.6.1 ABOUT THE DATASET

In order to perform CVP classification, the first step involved isolating the CVP from the minor amount of noise using voice activity detection. The audio files were then normalized to ensure consistency across the dataset. Entire sequences of RPS portraits were generated for all the cleaned audio files and the exact middle RPS portrait was chosen based on the assumption that most clean audio files will have the vowel in this part. Although this method is not completely accurate, it was implemented to create a proof of concept for tentative vowel classification.

To perform CVP classification, entire RPS sequences had to be generated, which were generated on the fly by varying the window length and stride used for generating individual RPS portraits. One of the major challenges of preprocessing was the amount of time the dataset creation for vowels and RPS sequences took. Other challenges included reducing the RPS image size and cleaning the audio files as correctly as possible to ensure the data was reliable for training models.

1.6.2 MODEL ARCHITECTURE

The vowel classifier was trained on a 4-layer CNN where the first layer expanded the channels, and the remaining layers halved them at every layer. The CNN contained MaxPool for dimensionality reduction and had dropouts and batch normalization as well. All of this was followed by a 2-layer classifier for the 5-vowel classification.

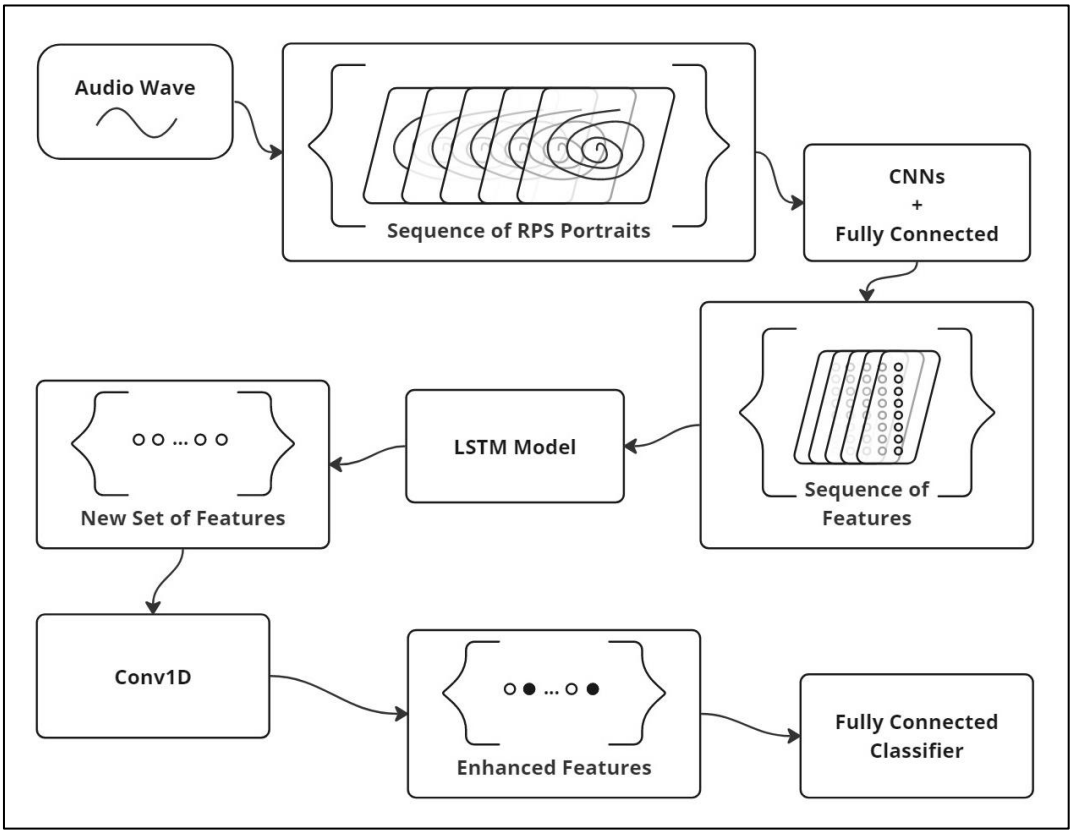
The CVP classifier used CNN layers for extracting spatial features, followed by dropouts and batch normalization. These CNNs were also implemented with a MaxPool to reduce the dimensions. All RPS portraits were passed through the same

CNN parallelly and then sequentially passed into a Bi-LSTM network. The final weights of the Bidirectional Long Short-Term Memory (BiLSTM) were then passed into a classifier for the 125 CVP classification.

Another CVP classifier was developed by combining RPS and MFCC using a Spatial Residual CNN (ResCNN) for RPS followed by 2 temporal ResCNNs, one for RPS and one for MFCC. Their final outputs were then concatenated and passed into the BiLSTM followed by the same classifier.

The RPS-MFCC combination was also implemented in the ASR model. However, the ASR model had a few differences. The classifier classified all LSTM outputs rather than just the last, the number of CNNs and LSTMs used was different, and the number of classes was different because ASR performed grapheme classification. The figure given below shows the overall training pipeline from data preprocessing to prediction.

Figure 2. General Architecture of Speech processing with RPS



The architecture shown above gives us the general information about speech tasks using RPS systems. It explains the flow of data through the model.

1.6.3 EVALUATION OF CLASSIFICATION MODELS

The evaluation of the classification models was done to measure the accuracy of the models. To ensure consistency in the dataset splits, a randomness seed was given for the train-validation-test splits, which were done at a ratio of 80-10-10. This helped in preventing bias towards any particular subset of the dataset.

For the vowel and CVP classification, the main metric used for evaluation was accuracy. The accuracy measures the percentage of correctly classified instances out of the total number of instances. The higher the accuracy, the better the model performance. The accuracy was used as the evaluation metric because the class distributions were almost even, making accuracy a simple and efficient metric.

After the training and tuning of the classification models, the final evaluation included a classification report f1 scores. The precision measures the percentage of correctly classified instances in a class out of the total number of instances predicted as belonging to that class. The recall measures the percentage of correctly classified instances in a class out of the total number of instances that belong to that class. The f1 score is the harmonic mean of precision and recall and gives an overall measure of the model performance. The confusion matrix was also generated to recognize what is being classified and/or misclassified.

The evaluation results were analyzed to identify the strengths and weaknesses of the classification models. The results were then used to fine-tune the models for better accuracy and to ensure that the models generalize well to unseen data. The classification models were evaluated multiple times to ensure that the results were reliable and consistent.

Chapter 2

Literature Review

2.1 INTRODUCTION

2.1.1 NEED FOR A LITERATURE REVIEW

The literature survey is an essential component of any research study. It is conducted to gain a comprehensive understanding of the existing research on the topic of interest, identify gaps in the existing knowledge, and build upon the existing work. In this section, we present a brief overview of the literature related to the current research topic.

The current research project aims to build upon the existing work by exploring the use of RPS features in combination with machine learning techniques for speech recognition in the Hindi language.

2.1.2 USAGE OF RPS

One of the important topics in the literature related to speech recognition is Reconstructed Phase Spaces (RPS). RPS is a technique that involves the reconstruction of phase spaces using the auto-mutual information and false nearest neighbors to identify the attractors in the speech signal. Extracting RPS features and combining them with existing frameworks has shown promising results in various speech recognition tasks.

2.1.3 MODELLING TECHNIQUES DISCOVERED

Modelling techniques play a vital role in speech recognition, and several techniques have been explored in the literature. One of the most widely used modelling techniques is Mel-Frequency Cepstral Coefficients (MFCC). MFCC is a technique that involves extracting the spectral envelope of the speech signal and then applying the Discrete Cosine Transform (DCT) to obtain a set of coefficients. HMMs and GMMs are commonly used with MFCC for speech recognition.

Machine learning techniques and deep learning techniques have also been extensively studied in the literature related to speech recognition. Support Vector Machines (SVMs), Neural Networks (NNs), and Convolutional Neural Networks (CNNs) are

some of the machine learning techniques that have been used for speech recognition. Deep learning techniques like Reconstructed Neural Networks (RNNs), Long Short-Term Memory (LSTM), and Gated Reconstructed Unit (GRU) have shown remarkable success in various speech recognition tasks.

2.1.4 SPEECH RECOGNITION TECHNIQUES DISCOVERED

Speech recognition has been studied in various forms, including Phoneme Recognition, Word Recognition, Automated Speech Recognition (ASR), Continuous Speech Recognition, and the use of Kaldi toolkit. Phoneme recognition involves identifying individual sounds in a speech signal, whereas word recognition involves identifying whole words. ASR involves recognizing speech in real-time and is a challenging task due to the variability in speech signals.

Continuous Speech Recognition involves recognizing speech that is continuous and has no pauses. Kaldi toolkit is a speech recognition toolkit that provides a complete pipeline for training and testing speech recognition systems.

2.2 LITERATURE REVIEW

2.2.1 DETERMINING EMBEDDING DIMENSION FOR PHASE-SPACE RECONSTRUCTION USING A GEOMETRICAL CONSTRUCTION

Kennel et al. (1992) [1] explains how to determine the best embedding dimension for phase space reconstruction. They propose the geometrical construction approach based on the False Nearest Neighbors algorithm to determine the minimum embedding dimension required to capture the dynamics of the underlying system. The authors show that this approach provides reliable results in determining the embedding dimension. One advantage of this approach is that it does not require any prior knowledge of the underlying system. A potential disadvantage is that it does not provide a method for determining the optimal delay.

2.2.2 SELECTION OF EMBEDDING DIMENSION AND DELAY TIME IN PHASE SPACE RECONSTRUCTION

Ma and Han (2006) [2] propose a method for determining the optimal delay for phase space reconstruction. They introduce a multiple auto-correlation using average displacement algorithm to determine the delay that optimizes the reconstruction of the

phase space. They compare their method with the False Nearest Neighbors algorithm and show that their method provides more accurate results. However, they do not provide a method for determining the optimal embedding dimension.

2.2.3 TIME SERIES CLASSIFICATION USING GAUSSIAN MIXTURE MODELS OF RECONSTRUCTED PHASE SPACES

Povinelli et al. (2004) [3] propose a method for classifying time series signals using reconstructed phase spaces and Gaussian mixture models (GMMs). They show that this method outperforms traditional classification methods such as Hidden Markov Models (HMMs) and k-Nearest Neighbors (kNN) in terms of classification accuracy. The advantage of this method is that it can be applied to a wide range of time series signals. However, this method does not use RPS directly for speech recognition.

2.2.4 SPEECH RECOGNITION USING RECONSTRUCTED PHASE SPACE FEATURES

Lindgren et al. (2003) [4] propose a method for enhancing speech recognition by using RPS features in conjunction with Mel Frequency Cepstral Coefficients (MFCCs). They show that their method outperforms traditional speech recognition methods such as HMMs and Artificial Neural Networks (ANNs) in terms of recognition accuracy. One advantage of this method is that it can be used to recognize continuous speech. However, this method does not use RPS to its full extent.

2.2.5 STATISTICAL MODELS OF RECONSTRUCTED PHASE SPACES FOR SIGNAL CLASSIFICATION

Povinelli et al. (2006) [5] present several approaches for signal classification using RPS features and statistical models such as Bayes Maximum Likelihood and ANN. They show that their methods outperform traditional classification methods such as kNN and Support Vector Machines (SVMs) in terms of classification accuracy. One advantage of this method is that it can be used to classify a wide range of signals. However, this method does not use RPS features directly.

2.2.6 CALCULATION OF AVERAGE MUTUAL INFORMATION (AMI) AND FALSE-NEAREST NEIGHBORS (FNN) FOR THE ESTIMATION OF EMBEDDING PARAMETERS OF MULTIDIMENSIONAL TIME SERIES IN MATLAB

Wallot and Mønster's (2018) [6] paper introduces the use of False Nearest Neighbors (FNN) and Average Mutual Information (AMI) for the estimation of the embedding parameters of multidimensional time series in MATLAB. The study aimed to identify the best delay and embedding dimension for speech recognition systems. The paper concludes that the proposed method is efficient in identifying the best delay and embedding dimension. However, it does not use RPS for speech recognition. The pros of this method include being a more efficient way to estimate the embedding parameters and providing a more accurate model for speech recognition. On the other hand, the cons of the method are the absence of the use of RPS, which could have provided a better performance.

2.2.7 PHONEME CLASSIFICATION IN RECONSTRUCTED PHASE SPACE WITH CONVOLUTIONAL NEURAL NETWORKS

Wesley, Khan, and Shahina's (2020) [7] paper introduces the use of Reconstructed Phase Space (RPS) with Convolutional Neural Networks (CNNs) for phoneme classification. The study aimed to show the effectiveness of RPS in classifying phonemes and its relation to acoustics. The paper explains how RPS relates to speech, and it is among the only papers to use RPS images directly. The pros of the method include its high accuracy, and the use of RPS provides a better performance than traditional approaches. However, the cons of the method are that it only uses RPS for English and on aligned data.

2.2.8 USING GAUSSIAN MIXTURES FOR HINDI SPEECH RECOGNITION SYSTEM

Aggarwal and Dave's (2011) [8] paper discusses a method to perform speech recognition for Hindi by using Gaussian Mixture Models (GMM). The study aimed to represent Hindi phonology by using this method. The paper uses Mel-Frequency Cepstral Coefficients (MFCC) and Hidden Markov Models (HMM) for classification. The pros of the method include representing Hindi phonology and providing a way to

perform speech recognition in Hindi. However, the cons of the method are that it uses older techniques for classification, which may not be as accurate as newer methods.

2.2.9 DISCRIMINATIVELY TRAINED CONTINUOUS HINDI SPEECH RECOGNITION USING INTEGRATED ACOUSTIC FEATURES AND RECONSTRUCTED NEURAL NETWORK LANGUAGE MODELING

Kumar and Aggarwal's (2021) [9] paper shows the usage of discriminative training and uses Reconstructed Neural Networks (RNNs) for language modeling for continuous Hindi speech recognition. The study aimed to provide a way for better language modeling in speech recognition systems. The paper uses MFCC, Gammatone Frequency Cepstral Coefficients (GFCC), and GMM for classification. The pros of the method include using deep learning for language models, which provides better performance. However, the cons of the method are that it does not use deep learning for classification of speech, which may result in lower accuracy.

2.2.10 DETECTION OF PHONEMIC ASPIRATION FOR SPOKEN HINDI PRONUNCIATION EVALUATION

Patil and Rao's (2016) [10] paper focuses on detecting mispronounced phonemes of non-native Hindi speakers. The paper employs Mel-Frequency Cepstral Coefficients (MFCC), Hidden Markov Model (HMM), and Gaussian Mixture Model (GMM) algorithms to determine the correct phone pronunciation. The authors use Hindi phonetic analysis alphabet and traditional classification techniques, such as HMM and GMM, for their analysis. The paper's pros include its emphasis on non-native speakers and its use of proven classification techniques. The cons of the paper include its limitation to Hindi phonetic analysis and its reliance on older classification methods.

2.2.11 STRUCTURAL ANALYSIS OF HINDI PHONETICS AND A METHOD FOR EXTRACTION OF PHONETICALLY RICH SENTENCES FROM A VERY LARGE HINDI TEXT CORPUS

Malviya, Mishra, and Tiwary (2016) [11] aim to extract phonetically rich sentences from a large Hindi text corpus by classifying Hindi alphabets with respect to phonetics. The paper uses Hindi Phonetically Rich/Balanced Sentence Extraction techniques and performs a statistical analysis of Hindi databases. While the authors'

paper aims for Automatic Speech Recognition (ASR), it only develops a database. The paper's pros include its detailed analysis of Hindi phonetics and its development of a balanced sentence database. The cons of the paper are its limited application to ASR and its failure to develop a complete ASR system.

2.2.12 AUTOMATIC SPEECH RECOGNITION WITH ARTICULATORY INFORMATION AND A UNIFIED DICTIONARY FOR HINDI, MARATHI, BENGALI AND ORIYA

Dash, Kim, Teplansky, and Wang (2018) [12] aim to improve Automatic Speech Recognition (ASR) by using articulatory information obtained from sensors along with speech. The authors use MFCC, HMM, Long Short-Term Memory (LSTM), and Deep Neural Networks (DNN) algorithms to predict speech. While the paper's usage of sensors improves the performance of ASR, it also necessitates the use of additional sensor data as features. The paper's pros include its innovative use of sensor data and its application to multiple languages. The cons of the paper include its reliance on sensor data and its need for additional data as features.

2.2.13 PHONETICALLY BALANCED CODE-MIXED SPEECH CORPUS FOR HINDI-ENGLISH AUTOMATIC SPEECH RECOGNITION

Pandey et al. (2018) [13] develop a phonetically balanced corpus of code-switched speech for Hindi and English, which can be used for Automatic Speech Recognition (ASR). The paper focuses on phone conversion from English formats to Hindi and does not discuss any specific algorithms. The paper's pros include its development of a phonetically balanced corpus and its potential use for ASR. The cons of the paper are its lack of information on specific algorithms and its limited discussion on the development of the corpus.

2.2.14 THE KALDI SPEECH RECOGNITION TOOLKIT

The paper by Povey et al. (2011) [14] presents the Kaldi toolkit, which is a powerful and flexible tool for performing Automated Speech Recognition (ASR). The paper highlights the toolkit's versatility, which allows it to be used with contemporary algorithms for ASR, making it a valuable asset for researchers and developers. However, the Kaldi toolkit primarily uses Hidden Markov Models (HMMs) and

Gaussian Mixture Models (GMMs), which are relatively old algorithms that may not be suitable for some modern applications.

The pros of the Kaldi toolkit include its versatility and flexibility, which make it an asset for researchers and developers. It is also relatively easy to use and can be used with contemporary algorithms. However, the toolkit's reliance on HMMs and GMMs can be a disadvantage for some modern applications, which require more advanced algorithms.

2.2.15 THE PYTORCH-KALDI SPEECH RECOGNITION TOOLKIT

The paper by Ravanelli et al. (2019) [15] introduces the Pytorch-Kaldi toolkit, which is a modified version of the Kaldi toolkit that uses Pytorch to incorporate deep learning techniques for ASR. The paper highlights the benefits of using Pytorch for ASR, such as improved accuracy and the ability to handle complex data types. However, the support for the Pytorch-Kaldi toolkit is limited to Linux operating systems.

The pros of the Pytorch-Kaldi toolkit include its ability to incorporate deep learning techniques for ASR, which can lead to improved accuracy and better handling of complex data types. The Pytorch-Kaldi toolkit also benefits from the Kaldi toolkit's versatility and flexibility. However, the support for the toolkit is limited to Linux operating systems, which may be a disadvantage for some users.

2.2.16 INDIAN LANGUAGE SPEECH SOUND LABEL SET (ILSL12)

The paper by Dr Samudravijaya et al. (2012) [16] presents a standard set of labels in the roman script for speech sounds which are commonly used in Indian Languages. They provide a label for similar sounds across various Indian languages providing a method of standardizing the graphemes and their respective phoneme mappings. This helps provide insights on transliteration and can help in solving the issue of redundant characters as well.

2.2.17 OTHER CONTEMPORARY APPROACHES

Automatic speech recognition (ASR) is a widely studied research field that has witnessed significant advances in recent years. One of the most important benchmarks in this field is the Word Error Rate (WER), which measures the

percentage of words that are incorrectly transcribed by an ASR system. The current state-of-the-art models and approaches have been studied by Kolobov et al. [17], who found that the best-performing model is Deep Speech FR (DSFR). Developed by Baidu Research, DSFR uses deep neural networks (DNNs) and has been shown to outperform other state-of-the-art speech recognition systems on benchmark datasets such as the Wall Street Journal and Switchboard corpora.

In addition to DSFR, there are other models that have shown promising results in ASR tasks. For example, DS-ES, Mozilla is specifically designed for Spanish and also uses DNNs for feature extraction and classification [18]. Another recent advancement that has created a new benchmark in ASR is Wav2Vec2, an end-to-end feature extraction technique that uses a contrastive self-supervised learning approach to learn speech representations from raw waveforms. This method has been employed in the development of multilingual ASR systems in languages such as French [19], Arabic [20], Turkish [21], and Spanish [22].

Overall, these recent advances in ASR have the potential to greatly improve speech recognition accuracy across a range of languages and contexts. However, it is important to continue exploring new models and approaches in order to further improve ASR performance and expand its applications.

2.3 RESEARCH GAPS

2.3.1 LIMITATIONS OF USING LINEAR TIME-SERIES SIGNAL FOR SPEECH RECOGNITION

Most speech recognition systems use linear time-series signals to represent speech data. Speech is a complex signal which means that traditional linear methods may not be able to capture all the variations present in speech. Nonlinear analysis methods can help to capture the underlying structure of speech data, which can lead to better speech recognition accuracy.

2.3.2 LACK OF EXPLORATION IN EXTRACTION TECHNIQUES FOR SPEECH RECOGNITION

Extraction techniques are used to extract relevant features from speech data, which are then used for speech recognition. Most research studies use extraction techniques such as MFCCs. However, there are many other extraction techniques available that

have not been explored extensively in speech recognition. These techniques may offer better accuracy than the traditional methods currently in use. Therefore, it is essential to investigate the use of various extraction techniques in speech recognition to determine their effectiveness.

2.3.3 RPS AS A STANDALONE SYSTEM FOR SPEECH RECOGNITION

Wesley et al. explored the use of RPS as a standalone system for speech recognition, achieving good accuracy for the TIMIT dataset. However, this approach was limited to English phonemes only. There is a need to investigate the use of RPS in other languages and to compare its performance with traditional speech recognition systems. This gap can be addressed by exploring the use of RPS for speech recognition in other languages and analyzing its effectiveness.

2.3.4 CLASSIFICATION OF HINDI ALPHABETS FOR SPEECH RECOGNITION

Malviya et al. suggested how the Hindi alphabet system separates its alphabets, which are closely associated with the way speech is produced in humans. Further research can be conducted to investigate the relationship between the classification of Hindi alphabets and their acoustic properties. This can help in the development of more accurate speech recognition systems for Hindi language.

2.3.5 INVESTIGATING THE USE OF HYBRID SYSTEMS FOR SPEECH RECOGNITION

Another research gap in speech recognition is the lack of investigation into the use of hybrid systems. A hybrid system can combine the strengths of different approaches to provide a more accurate speech recognition system. For example, a hybrid system can combine the strengths of traditional linear methods with the power of deep learning techniques to create a more accurate system. Investigating the use of hybrid systems for speech recognition can lead to better accuracy and more robust systems. My study aims to explore the use of a hybrid system that combines linear and nonlinear methods with deep learning techniques to improve speech recognition accuracy.

Chapter 3

Implementation Details

3.1 DATASET

3.1.1 CONSONANT-VOWEL DATASET

The consonant-vowel dataset used in this study is a subset of the dataset collected by Indian Institute of Technology, Madras, which consists of 29 consonants and 5 vowels. The subset used in this study comprises 25 consonants paired with 5 vowels, excluding alveolar consonants such as 't', 'th', 'd', and 'dh', which results in a total of 125 syllable-like CVPs. The dataset contains over 10 recordings of each of the 125 CVPs from 5 male speakers, resulting in approximately 10,000 audio files, some of which do not contain any CVPs.

3.1.3 VOWEL CLASSIFICATION DATASET

Since the dataset only includes 5 vowels, an imperfect system was created to test the viability of RPS features for vowel classification. A single RPS portrait was generated from the sequence of portraits by taking the median RPS portrait, assuming that the center portrait would represent a vowel. The classification dataset was generated for two resolutions - 64x64 and. The dataset was created beforehand and loaded into the code for model training, limiting the possibility of data augmentation and experimentation.

While these limitations could be resolved, the task of vowel classification was only an intermediate step to check the overall viability of RPS plots, the use of neural networks on them, and to create an architecture capable of extracting these spatial features. The primary goal was the implementation of the ASR system.

3.1.4 CVP CLASSIFICATION DATASET

An on-the-fly approach was used to create the CVP classification dataset. The audio signals were pre-loaded from a file and the RPS portrait sequence for each signal was created based on the index requested. The reason behind this approach was that the RPS portrait sequences are too large to be saved and loaded as a single file.

Additionally, the key parameters of the dataset, such as window and stride, need to be varied during training, making it less viable to create a single file. Furthermore, the dataset would be too large to store even in memory, making it impractical.

The CVP classification dataset includes two resolutions, 32x32 and 64x64, to explore the performance of the model.

3.1.5 ASR FOR HINDI USING COMMON VOICE DATASET

The Common Voice Corpus 11 for Hindi, an open-source project developed by Mozilla, was used in this study. It contains 13 hours of validated speech data in Hindi contributed by 333 unique speakers and is freely available for research purposes. The dataset is structured into three folders: train, test, and validated. The train folder contains most of the data, and multiple listeners validated the data to ensure its quality.

Our study aimed to perform ASR for Hindi using the Common Voice dataset and employed the CNN-LSTM architecture. We used the standard 80-10-10 split of the dataset for training, validation, and testing. Our findings demonstrate the effectiveness of using this dataset for ASR for Hindi.

In conclusion, the Common Voice dataset is a valuable resource for speech recognition researchers. Our study contributes to the growing body of research on ASR for Hindi using the Common Voice dataset.

3.1.6 BENCHMARK DATASETS FOR ASR FOR HINDI

The models that performed best for ASR on the Common Voice Hindi dataset were not trained on it. Instead, they were trained on a larger crowdsourced dataset named "Indic voice", available on Hugging Face.

This dataset includes over 95 hours of speech data in over 99,000 files, along with a validation set of over 5 hours of speech data in over 3,800 files. As many benchmark models have been trained on this data, it makes sense to use it as well.

Since the same sentences are repeated by multiple users, and the audio data is cleaner than Common Voice, this dataset is a better choice for training a model whose performance has not yet been tested.

Training on structured data, and a lot of it, has the potential to yield better results. Furthermore, with a larger dataset, the model can more effectively learn the transitions of speech features across a sentence, making it a promising approach.

3.2 PREPROCESSING

3.2.1 METADATA CREATION

The dataset contained approximately 10,000 signals distributed across 125 folders, each corresponding to a combination of 5 vowels and 25 consonants. To create the metadata, we extracted various features from each audio signal, including the specific syllable, consonant, and vowel. We also added features such as sound source and articulation source, with the latter derived from a dictionary mapping.

The aim of the metadata was to assist in both modelling tasks and the analysis of RPS portraits. Since RPS portraits have a relationship with articulation, the metadata could provide further insights and be useful for training speech recognition systems.

3.2.2 PREPROCESSING AUDIO SIGNALS

To clean and normalize the audio files, we utilized the Librosa speech processing library. The audio was resampled to a sample rate of 8000 Hz to standardize the dataset. The trim function in Librosa was used to ensure the audio files were neither too long nor too short. Padding was applied to all audio files on both ends to ensure that they all had exactly 8000 samples. Finally, the audio signals were normalized with the normalize utility in Librosa.

After the preprocessing steps, the dataset contained 9,889 audio signals for further processing. These steps are necessary to ensure that the audio signals are of a uniform format and quality, which is essential for training and evaluating machine learning models.

3.2.3 CREATING RPS FEATURES

To generate RPS portrait sequences for any audio dataset, a general function was created, allowing for the variation of several parameters, including RPS portrait resolution, window size, stride length, and delay value for the RPS embeddings. The first step in generating RPS image sequences was to create phase space embeddings,

which involve a delayed version of the original signal. For a signal S, read at time t, with a delay value d and a window size w, the embedding is defined as:

$RPS\_Embedding(S) = [S_{t \text{ to } end-d}, S_{t+d \text{ to } end}]$

These embeddings are then segmented by sliding a window over the embedding, resulting in a sequence of embeddings determined by the window size and stride length. This sequence is then passed into a plotting function, which generates individual RPS portraits for the required resolution. The RPS portraits are then appended into a new vector, creating the final RPS portrait sequence of the signal with all the defined parameters. RPS features were generated for all 9,889 audio files, varying the different parameters.

While window size and stride length can be varied across training for data augmentation and model improvement, other parameters, such as RPS portrait resolution, remain fixed for a specific model in training.

The delay feature can fundamentally change the shapes of RPS portraits, and while it can be changed within a training session, it must ultimately be made constant to ensure the model can learn shapes for a specific delay. Upon using the Mutual Information algorithm, we identified the best delay for ASR to be 6. So, we used the delay of 6 to train all our models.

3.2.4 PREPROCESSING RPS FOR ASR

In our initial experimentation, we encountered a problem where the ASR model was unable to learn from the standalone RPS portrait sequences. Upon further investigation, we discovered that the RPS portraits for noise, consonants, and vowels were all equally important, making it impossible for the model to identify which portraits to focus on or how to differentiate between noise and audio.

To address this issue, we modified the RPS features by multiplying the range of amplitudes in each window with the corresponding RPS portraits. This resulted in varying pixel values for all RPS portraits within a range. We then normalized all image pixel values by rescaling all ranges between 0 and 1, using the highest and lowest amplitude values to identify the range of amplitudes in each audio signal.

3.2.5 PREPROCESSING ASR TEXT

We identified all the various graphemes that could be directly converted to phonemes with guidance from the Indian Language Speech sound Label set (ILSL12). We then addressed redundant characters in terms of phonemes, such as vowels in Hindi, which can be written directly as vowel characters or as syllables in the form of a consonant-vowel combination called a "matra." To handle this, we added a rule that every vowel preceded by a space or another vowel would be treated as a complete vowel, and if it is preceded by a consonant or was the first letter, it would be a matra. These rules ensured that the final prediction had only the necessary characters and avoided redundancies and confusion that could have impacted the ASR model.

3.2.6 PREPROCESSING MFCC

To ensure compatibility between the RPS and MFCC features when passing them into the model and combining them, we had to ensure that certain aspects of the MFCCs and RPS portraits were the same. Specifically, the combination of RPS and MFCC features was dependent on both sequences having the same number of time steps. To ensure this, we extracted both sets of features from the same signal and provided the MFCC transform function with the same window and stride lengths as used for creating RPS portrait sequences. If their lengths varied, we padded the sequence lengths in the case of RPS or reduced them in the case of MFCC to match them. As the features were combined along the time dimension, this preprocessing was essential.

3.2.7 DATA AUGMENTATION IN RPS

As mentioned earlier, some parameters of RPS portraits can be varied within a single training setup to enhance the model’s learning capacity. By varying the window size, we change the number of lines which are drawn in a single RPS portrait. Too many lines may overlap to hide some features and too few lines may not be able to show the patterns which can help. By varying the stride, we vary the smoothness of RPS transitions across the sequence of portraits. A smaller value will give a smoother RPS sequence transition but will also increase the number of portraits and thus memory requirement.

3.3 ARCHITECTURE

3.3.1 32X32 CNN FOR VOWEL CLASSIFICATION

In this study, we propose a CNN architecture for vowel classification, which takes a single RPS portrait as input. The architecture consists of three convolutional layers and two fully connected layers. Each convolutional layer has filters of size (3, 3), a stride of 1, and a ReLU activation function, followed by a max-pooling layer of size (2, 2). Batch normalization and dropout are applied after each convolutional layer before the next layer.

The output of the last convolutional layer is flattened and fed into two fully connected layers. The first fully connected layer is activated with ReLU, and the second layer is followed by a Log SoftMax layer for classification. To better understand the data flow, we provide a table below that shows the layer-wise change in dimensions and number of features.

Table 1. CNN Architecture for Vowel Classification on 32X32 RPS Portrait

LAYER	INPUT DIMS	OUTPUT DIMS	FEATURES
Input	(32, 32)	(1, 32, 32)	1024
Conv_1, ReLU	(1, 32, 32)	(32, 32, 32)	32768
MaxPool, Batch Norm, Dropout	(32, 32, 32)	(32, 16, 16)	8192
Conv_2, ReLU	(32, 16, 16)	(64, 16, 16)	16384
MaxPool, Batch Norm, Dropout	(64, 16, 16)	(64, 8, 8)	4096
Conv_3, ReLU	(64, 8, 8)	(128, 8, 8)	8192
MaxPool, Batch Norm, Dropout	(128, 8, 8)	(128, 4, 4)	2048
Linear_1, ReLU	(8192)	(256)	64
Linear_2, Log SoftMax	(256)	(5)	5

This model provides an effective approach for vowel classification with high accuracy, which can be relevant in further speech processing.

3.3.2 64X64 CNN FOR VOWEL CLASSIFICATION

In this study, we propose a CNN architecture for vowel classification, which takes a single RPS portrait as input. The architecture consists of four convolutional layers and two fully connected layers. Each convolutional layer has filters of size (3, 3), a stride of 1, and a ReLU activation function, followed by a max-pooling layer of size (2, 2). Batch normalization and dropout are applied after each convolutional layer before the next layer.

The output of the last convolutional layer is flattened and fed into two fully connected layers. The first fully connected layer is activated with ReLU, and the second layer is followed by a Log SoftMax layer for classification.

Table 2. CNN Architecture for Vowel Classification on 64X64 RPS Portrait

LAYER	INPUT DIMS	OUTPUT DIMS	FEATURES
Input	(64, 64)	(1, 64, 64)	4096
Conv_1, ReLU	(1, 64, 64)	(32, 64, 64)	131072
MaxPool, Batch Norm, Dropout	(32, 64, 64)	(32, 32, 32)	32768
Conv_2, ReLU	(32, 32, 32)	(64, 32, 32)	65536
MaxPool, Batch Norm, Dropout	(64, 32, 32)	(64, 16, 16)	16384
Conv_3, ReLU	(64, 16, 16)	(128, 16, 16)	32768
MaxPool, Batch Norm, Dropout	(128, 16, 16)	(128, 8, 8)	8192
Conv_4, ReLU	(128, 8, 8)	(256, 8, 8)	16384
MaxPool, Batch Norm, Dropout	(256, 8, 8)	(256, 4, 4)	4096
Linear_1, ReLU	(4096)	(128)	256
Linear_2, Log SoftMax	(128)	(5)	5

This model provides an effective approach for vowel classification with high accuracy, which can be relevant in further speech processing.

3.3.3 CNN-LSTM FOR CVP CLASSIFICATION

In this section, we propose a CNN-LSTM architecture for CVP classification, which takes RPS portrait sequences as input. The architecture consists of CNNs for parallel processing of RPS portraits and an LSTM for sequential processing.

The CNN blocks used in the architecture are either normal CNNs or Residual CNN blocks. Each subsequent layer reduces the features of the image by half by either doubling the number of channels or using MaxPool with a stride length of 2. This process halves the image resolution and doubles the channels.

Based on the proposed architecture, one or more layers of CNN blocks are used to process all RPS portraits individually while retaining their sequence and order throughout the prediction process. This ensures the model can learn genuine features based on order rather than a randomized and mixed-up version.

The output of the CNNs is a multi-channel image, which is flattened and passed into a fully connected layer to reduce the dimensions of the features before passing them into the LSTM. The dimensions are kept in check to avoid excessive memory usage.

The BiLSTM block is used for sequential processing of the sequence of features, with the cell states of the previous feature vector passed into the next one. LSTMs allow passing sequences of variable length and extract sequential information while doing so. The BiLSTM block returns a new sequence of features as its final output, doubling the features in number because of the forward and backward sequence outputs, which are then passed into the classifier block.

After experimentation, we found that replacing one of the linear layers with a 1-dimensional convolutional (Conv1D) layer improves performance drastically. The classifier block consists of a Conv1D layer followed by an activation function and a fully connected layer, which is Log SoftMax activated for a final classification into 125 syllable-like, CVP classes.

We propose 2 models on 32X32 image sequences which take less memory and do not see a performance loss.

3.3.4 IMAGE SEQUENCE WITH A SIMPLE CNN-LSTM

We propose a simple CNN-LSTM model for CVP classification, inspired by vowel classifiers. The model consists of two CNN blocks that include a CNN layer, followed by a MaxPool, Batch Norm, GELU activation, and a Dropout in that order. The CNNs' output is then flattened and passed into a fully connected layer. The resulting feature vector sequence is passed into a BiLSTM layer that extracts the sequential information and doubles the number of features for each time step.

The last output of the BiLSTM is passed into the classifier for a final classification. The classifier contains a Conv1D layer that acts as a feature enhancer and improves the model's performance.

The table below shows the dimensions through the network:

Table 3. Simple CNN-LSTM architecture for CVP Classification

LAYER	INPUT DIMS	OUTPUT DIMS	FEATURES
Input	(t, 32, 32)	(t, 1, 32, 32)	t X 1024
Adding a channel dimension.			
Conv_Block_1	(t, 1, 32, 32)	(t, 8, 16, 16)	t X 2048
Conv_Block_2	(t, 8, 16, 16)	(t, 16, 8, 8)	t X 512
Flattening the CNN output.			
Linear_1, ReLU	(t, 512)	(t, 64)	t X 64
BiLSTM	(t, 64)	(t, 128)	t X 128
Selecting only the last output of BiLSTM and adding a channel dimension.			
Conv1D, GELU	(1, 1, 128)	(1, 1, 128)	128
Linear_2, Log SoftMax	(128)	(125)	125

Note that the number of features in this model may seem low, but the model performs well even with this feature size, demonstrating the viability of RPS portraits in explaining speech.

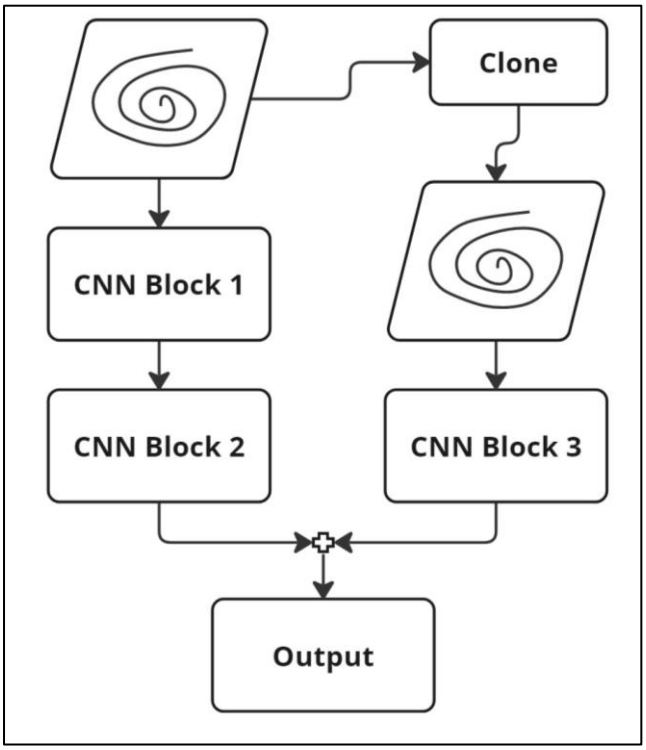
3.3.5 IMAGE SEQUENCE WITH A RESIDUAL CNN-LSTM

In this section, we propose a Residual CNN-LSTM model inspired by Wave2Vec for image sequence classification. The model is comprised of a single Residual CNN (ResCNN) block consisting of three CNN blocks. The first CNN block contains a CNN layer followed by MaxPool, Batch Norm, GELU activation, and Dropout in that order. The second CNN block only includes a CNN layer followed by Batch Norm, both using a kernel size of (3, 3).

The third layer, which resamples the residue, includes a convolutional layer with a (1, 1) kernel that maps from input channel size to output channel size, followed by MaxPool to obtain the exact number of dimensions. The residual image is cloned and processed through two CNN blocks. The first CNN block adjusts the dimensions and number of channels, while the second one extracts features. The third CNN block resamples the residual image to match the processed one, which is then added to the output of the second CNN block. The output then goes through an activation followed by Dropout.

The figure below shows one Residual CNN block from image input to combining the outputs.

Figure 3. Residual CNN Block



The remainder of the architecture is identical, with the exception of the number of features. The table below shows the dimensions through the network.

Table 4. Residual CNN-LSTM architecture for CVP Classification

LAYER	INPUT DIMS	OUTPUT DIMS	FEATURES
Input	(t, 32, 32)	(t, 1, 32, 32)	t X 1024
ResCNN_Block	(t, 1, 32, 32)	(t, 1, 16, 16)	t X 256
Linear_1, ReLU	(t, 256)	(t, 256)	t X 256
BiLSTM	(t, 256)	(t, 512)	t X 512
Selecting only the last output of BiLSTM and adding a channel dimension.			
Conv1D, GELU	(1, 1, 512)	(1, 1, 512)	512
Linear_2, Log SoftMax	(512)	(125)	125

It should be noted that this model utilizes a greater number of features yet still achieves comparable performance.

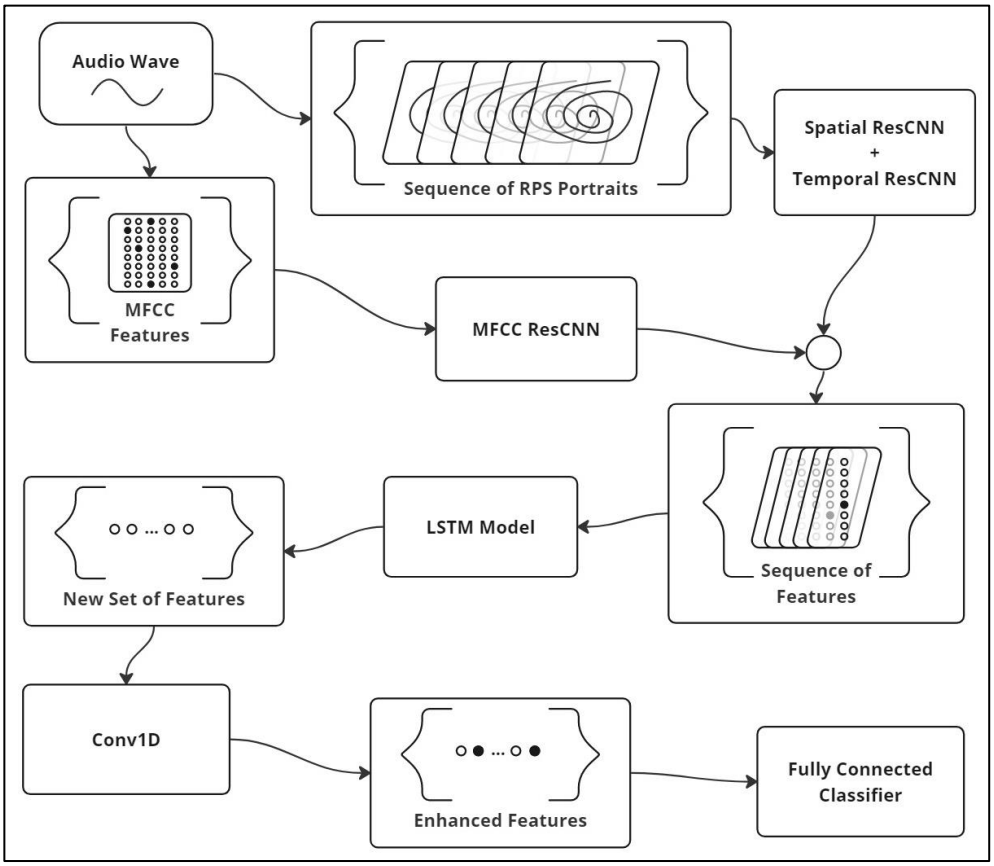
3.3.6 CNN-LSTM for CVP Classification with RPS and MFCC

In this section, we present our proposed architecture for CVP classification which combines the RPS and MFCC features. The architecture is only different until the features for MFCC and RPS are combined. The RPS features are first passed through a Spatial ResCNN which extracts features and reduces the resolution using a stride length of 2. The kernel size of the all ResCNNs is (3, 3) and it retains the number of channels.

The resulting feature vector of shape (time, channels, x, y) is then transformed by flattening the RPS dimension and passing the time dimension as an image feature with the shape (channels, time, x*y) through another ResCNN, which is a Temporal ResCNN that extracts temporal dependencies.

Similarly, the MFCC features are passed through a Temporal ResCNN with the only difference being in the number of features being processed, which can be called the MFCC ResCNN. The output of the MFCC ResCNN and the Temporal ResCNN are concatenated and reshaped for the rest of the architecture. The Figure 3 helps visualize the flow of data and feature combination from a high-level perspective.

Figure 4. Model architecture combining RPS and MFCC



The remainder of the architecture is identical, with the exception of the number of features. The table below shows the dimensions through the network:

Table 5. CNN-LSTM with MFCC and RPS features for CVP Classification

LAYER	INPUT DIMS	OUTPUT DIMS	FEATURES
Input	(t, 32, 32) RPS (t, 32) MFCC	(t, 1, 32, 32) RPS (1, t, 32) MFCC	t X 1024+32
Spatial_ResCNN	(t, 1, 32, 32)	(t, 1, 16, 16)	t X 256
Temporal_ResCNN	(1, t, 256)	(2, t, 256)	t X 512
MFCC_ResCNN	(1, t, 32)	(2, t, 32)	t X 64
Concatenating and reshaping the outputs.			
BiLSTM	(t, 576)	(t, 1152)	t X 1152
Selecting only the last output of BiLSTM and adding a channel dimension.			
Conv1D, GELU	(1, 1, 1152)	(1, 1, 1152)	1152
Linear, Log SoftMax	(1152)	(125)	125

The CNN-LSTM architecture with Spatial and Temporal CNNs combines RPS and MFCC features, making it more complex than the other models. However, it outperforms all other models, indicating the potential for combining different feature extraction techniques and improving speech processing. To the best of our knowledge, an approach like this has never been published before, making its scope even greater with the possibility of further experimentation.

3.3.6 CNN-LSTM for ASR with RPS and MFCC

The CNN-LSTM model for ASR uses the same set of layers as the previous architecture. The only difference we see is in the number of layers of some blocks. We use the same number of Spatial ResCNN and Temporal ResCNN layers for RPS. We use 2 MFCC ResCNN layers and use a BiLSTM with 3 layers followed by the same classifier of Conv1D and a fully connected layer with Log SoftMax.

A major difference in predictions is that we now classify all the features in the sequence. This is because our output is now not a character or sound unit but sequence of characters. Also, the number of characters is now 79 with 26 English alphabets, one space and 52 characters from Devanagari script.

Table 6. CNN-LSTM on MFCC and RPS features for ASR

LAYER	INPUT DIMS	OUTPUT DIMS	FEATURES
Input	(t, 32, 32) RPS (t, 32) MFCC	(t, 1, 32, 32) RPS (1, t, 32) MFCC	t X 1024+32
Reshaping to add channel dimension			
Spatial_ResCNN	(t, 1, 32, 32)	(t, 1, 16, 16)	t X 256
Temporal_ResCNN	(1, t, 256)	(2, t, 256)	t X 512
MFCC_ResCNNs	(1, t, 32)	(4, t, 32)	t X 128
Concatenating and reshaping the outputs.			
BiLSTM	(t, 640)	(t, 1280)	t X 1280
Selecting only the last output of BiLSTM and adding a channel dimension.			
Conv1D, GELU	(t, 1, 1280)	(t, 1, 1280)	1280
Linear, Log SoftMax	(1280)	(79)	79

3.4 TRAINING

This section describes the training procedure of the classifiers used in the study. The classifiers' outputs were Logarithmic probability distributions due to the Log SoftMax activation, and they were trained with Negative Log Likelihood (NLL) Loss. The optimizer used for training all models was AdamW or Adam with weight decay. The implementation of AdamW was chosen as it was said to help train the model faster and regularize better by managing the weight decays correctly. To deal with the high memory requirements of the RPS classifier, a gradient accumulation system was implemented. This system accumulated the losses of predicted batches until the desired batch size was reached before actually taking the optimizer step.

3.4.1 COMMON PARAMETERS IN TRAINING

The vowel and CVP classifiers were trained with different parameters to optimize their performance. However, certain parameters were kept the same across all classifiers. The models were trained using a fixed batch size, and the learning rate was reduced over multiple epochs from a high initial value to a lower value.

3.4.2 VOWEL CLASSIFICATION

The vowel classifiers were trained with a batch size of 32 and a learning rate of 0.001. The models were trained for 50 epochs in a single session. The classifier on 64X64 images stopped learning much after the 7th epoch, while the 32X32 images' classifier stopped learning much after the 11th epoch.

3.4.3 CVP CLASSIFICATION

The CVP classifiers were trained with variable batch sizes ranging from 128 to 32. The learning rate was also reduced over multiple epochs starting from 0.001 to 0.0001. The models were trained for a variable number of epochs ranging from 250 to 350 across multiple training sessions. All models were trained until they stopped learning much more. During the training, the window size and stride were reduced from 500 to 100. In the last few epochs, the sliding window was marginally increased to anywhere between 125 and 175 until the minor improvement stopped. A single training order cannot be determined for CVP classifiers because their learning varied largely on the starting weights of the untrained models.

3.4.4 ASR USING RPS AND MFCC

The ASR model was trained with a fixed batch size of 64, and the learning rate was reduced over multiple epochs from 0.001 to 0.0001. The model was trained for 400 to 500 epochs across multiple training sessions. It was trained until both validation and training metrics stopped improving. During the training, the window size and stride were reduced from 350 to 100. In the last few epochs, the sliding window was increased to anywhere between 100 and 200.

3.4.5 LOSS FUNCTIONS IN ASR

In speech recognition tasks, the input and output are sequential, and the length of the input and output sequences can vary. Therefore, a simple loss function like Negative Log-Likelihood (NLL) loss cannot be used to train the ASR model. Instead, we used the Connectionist Temporal Classification (CTC) loss function.

CTC loss is a popular loss function used in speech recognition that allows the model to learn from sequences of variable length. It works by mapping the variable-length input sequence to a fixed-length output sequence by collapsing repeated characters and removing blanks. The model then predicts a probability distribution over the possible output sequences.

The CTC loss function measures the difference between the predicted output sequence and the ground truth output sequence. It does this by summing over all possible alignments of the predicted and ground truth output sequences. The gradient of this loss is then backpropagated through the model to update the parameters.

3.5 EVALUATION

3.5.1 EVALUATING THE CLASSIFIERS

The datasets for both, were almost balanced, which ensured that accuracy was a suitable metric to evaluate model performance during training.

All the models were trained and tuned using the accuracy metric. The final performance of the classifiers was evaluated by measuring the per-class precision, recall, and F1 score. The per-class evaluation ensured that all classes were being predicted accurately and efficiently.

The precision score measures the ratio of correctly predicted positive samples to the total number of predicted positive samples. The recall score measures the ratio of correctly predicted positive samples to the total number of actual positive samples. The F1 score is the harmonic mean of precision and recall and provides a balanced evaluation metric.

The results of the evaluation indicated that all classes were predicted accurately, with high precision, recall, and F1 scores. These scores indicated that the classifiers performed well on both the vowel and syllable classification tasks.

3.5.2 EVALUATING THE ASR MODEL

The ASR model is evaluated based on its ability to accurately transcribe speech into text. The evaluation metrics used for ASR tasks are Word Error Rate (WER) and Character Error Rate (CER).

Word Error Rate (WER) is the percentage of words that are incorrectly transcribed by the ASR model. It is calculated as follows:

WER = (S + D + I) / N

Where S represents the number of substitutions (words that are incorrectly transcribed), D represents the number of deletions (words that are missed out), I represents the number of insertions (extra words that are not present in the original speech), and N represents the total number of words in the original speech.

Character Error Rate (CER) is the percentage of characters that are incorrectly transcribed by the ASR model. It is calculated as follows:

CER = (S + D + I) / N'

Where S, D, I, and N are the same as in the WER calculation, but N' represents the total number of characters in the original speech.

The WER and CER metrics are used to measure the accuracy of the ASR model. A lower WER or CER value indicates higher accuracy. In general, a WER or CER value of less than 10% is considered good, while a value of less than 5% is considered excellent. But, most benchmark models for ASR on the Common Voice Corpus 11 for Hindi show a high WER and CER to which our model is comparable.

3.6 RESULTS

3.6.1 VOWEL CLASSIFICATION

Table 7. Evaluation of vowel classifiers

MODEL	METRIC	TRAIN	VALIDATION	TEST
32X32 CNN	Accuracy	100%	90%	91%
	F1-score	1.00	0.90	0.91
64X64 CNN	Accuracy	100%	**91%**	**92%**
	F1-score	1.00	**0.91**	**0.92**

3.6.2 CVP CLASSIFICATION

Table 8. Evaluation of CVP classifiers

MODEL	METRIC	TRAIN	VALIDATION	TEST
CNN	Accuracy	99%	87%	86%
	F1-score	0.99	0.87	0.86
ResCNN	Accuracy	99%	84%	85%
	F1-score	0.99	0.83	0.84
MFCC	Accuracy	98%	**92%**	**92%**
	F1-score	0.98	**0.92**	**0.91**

3.6.2 ASR MODEL

Given below is the evaluation of the ASR model trained on RPS and MFCC features. Note that these results are of a model trained on Common Voice 11 and tested with an internal split.

Table 9. Evaluation of ASR model

METRIC	TRAIN	VALIDATION	TEST
WER	0.09%	78.82%	77.82%
CER	0.02%	38.63%	37.77%

Chapter 4

Conclusion and Future Work

4.1 CONCLUSION

4.1.1 VOWEL CLASSIFICATION

The vowel classification models using 32x32 and 64x64 CNNs performed very well on all sets, achieving 100% accuracy on the training set, and 90% and 91% accuracy respectively on the validation and test sets for the 32x32 CNN, and 91% and 92% accuracy respectively on the validation and test sets for the 64x64 CNN. Both models achieved high F1-scores, indicating a good balance between precision and recall. These results suggest that the CNN models are effective for vowel classification tasks.

4.1.2 CVP CLASSIFICATION

The CVP classification models also performed well, with the MFCC-based model achieving the highest accuracy on all sets, achieving 98% accuracy on the training set and 92% accuracy on the validation and test sets. The ResCNN-based model achieved slightly lower accuracy, but still performed well with 99% accuracy on the training set and 84% and 85% accuracy respectively on the validation and test sets. The CNN-based model achieved 99% accuracy on the training set, but lower accuracy on the validation and test sets with 87% and 86% accuracy respectively.

The F1-scores for all models were good, but slightly lower than the vowel classification models. These results indicate that both the MFCC and ResCNN-based models are effective for CVP classification, but the ResCNN-based model may require further optimization.

4.1.3 ASR

The ASR model achieved very low WER and CER on the training set, with a WER of 0.09% and a CER of 0.02%. However, the model performed significantly worse on the validation and test sets, achieving a WER of 78.82% and 77.82% respectively and a CER of 38.63% and 37.77% respectively. These results suggest that the model may have overfit to the training set, and further optimization may be necessary to improve its performance on unseen data.

4.1.4 CONCLUSION

Overall, these results provide insight into the effectiveness of different models for vowel and CVP classification tasks and ASR systems. The high accuracy and F1-scores achieved by the vowel classification models suggest that CNNs are effective for such tasks, while the good performance of the MFCC and ResCNN-based models for CVP classification tasks indicate that these models are well-suited for such tasks. However, the performance of the ASR model on unseen data highlights the need for further optimization and development in this area.

In conclusion, this research provides a good starting point for future work in the field of speech processing and classification. The findings suggest that CNNs are an effective model for vowel classification tasks, and MFCC and ResCNN-based models are effective for CVP classification tasks. These models can be further optimized to achieve even better results. The ASR model can also be further developed to improve its performance on unseen data. Further research in this area can lead to the development of more advanced and accurate speech processing and classification systems.

4.2 FUTURE WORK

The current study has demonstrated the effectiveness of various deep learning models for speech classification and ASR tasks. However, there is still much room for improvement and further research in these areas. In this section, we discuss some of the potential avenues for future work.

One potential direction for future research is to explore the effectiveness of different data augmentation techniques. Data augmentation can be used to artificially increase the size of the training set by creating additional samples from the original data. This can be especially useful when the original dataset is small or imbalanced. In the current study, we only used basic data augmentation techniques such as varying window size and stride length. However, there are many other techniques that could be explored, such as adding noise, changing the pitch or speed of the audio, or synthesizing new data using generative adversarial networks (GANs).

Another area for future work is to investigate the use of transfer learning for speech classification and ASR tasks. Transfer learning involves training a model on one dataset and then fine-tuning it on a different dataset. This can be useful when the target dataset is small, as it allows the model to leverage knowledge learned from the source dataset. In the current study, we only trained our models on the target dataset. However, transfer learning could potentially improve model performance, especially for the ASR task where there is a limited amount of labeled data available.

In addition, future research could explore the effectiveness of different architectures for speech classification and ASR tasks. In this study, we used basic CNN-LSTM based architectures for CVP classification and ASR. However, there are many other architectures that could be explored, such as attention-based models, transformer models. These architectures may be able to capture more complex patterns in the data and lead to improved performance.

Another potential direction for future research is to investigate the use of multimodal learning for speech classification and ASR tasks. Multimodal learning involves training models on multiple modalities, such as audio and video or audio and text. This can be useful when there is complementary information across modalities that can improve model performance. For example, for the ASR task, using both audio and text inputs may lead to improved performance, especially for challenging accents or speech disorders.

Finally, future research could investigate the use of explainable AI techniques for speech classification and ASR tasks. Explainable AI involves building models that are transparent and interpretable, so that humans can understand how the model is making decisions. In the current study, we did not focus on explainability, but it is an important consideration for real-world applications of speech classification and ASR.

In conclusion, this study has demonstrated the effectiveness of RPS features and their model architectures for speech classification and ASR tasks. However, there is still much room for improvement and further research in these areas. Future work could explore the use of different data augmentation techniques, transfer learning, alternative architectures, multimodal learning, and explainable AI. By continuing to innovate and refine these techniques, we can further advance the field of speech processing and enable new and exciting applications.

Appendix

APPENDIX 1 - IMPLEMENTATION CODE

1.1 CNN MODEL ARCHITECTURE FOR VOWEL CLASSIFICATION

```
class Net(nn.Module):
    def __init__(self, n_classes, resolution):
        super(Net, self).__init__()
        self.n_classes = n_classes
        self.min_channels = 32
        self.num_cnn_layers = 4
        self.resolution = resolution
        self.ann_input_size =
((self.resolution**2)*self.min_channels)//(2**(self.num_cnn_layer
s+1))
        self.ann_output_size = self.ann_input_size//4

        self.define_model()

    def define_model(self):
        self.layers = nn.ModuleList()

        # First convolutional layer
        self.layers.append(
            nn.Conv2d(1, self.min_channels, kernel_size=3,
padding=1))
        self.layers.append(nn.MaxPool2d(2))
        self.layers.append(nn.ReLU())
        self.layers.append(nn.BatchNorm2d(self.min_channels))
        self.layers.append(nn.Dropout2d(0.4))

        # Additional convolutional layers
        for i in range(1, self.num_cnn_layers):
            self.layers.append(nn.Conv2d(self.min_channels * (2
** (i-1)),
                                         self.min_channels * (2 ** i),
kernel_size=3, padding=1))
            self.layers.append(nn.ReLU()),
            self.layers.append(nn.MaxPool2d(2))
            self.layers.append(nn.BatchNorm2d(self.min_channels *
(2 ** i)))
            self.layers.append(nn.Dropout2d(0.4))

        self.fully_connected = nn.Sequential(
            nn.Linear(self.ann_input_size,
```

```
                        self.ann_output_size),
            nn.ReLU(),
            nn.Linear(self.ann_output_size, self.n_classes),
            nn.LogSoftmax(dim=1)
        )

    def forward(self, ip):

        x = ip.reshape((-1, 1, self.resolution, self.resolution))

        del ip
        torch.cuda.empty_cache()

        for layer in self.layers:
            x = layer(x)

        x = torch.flatten(x, 1)
        x = self.fully_connected(x)

        return x


model = Net(n_classes=n_classes, resolution=SIZE).cuda()


criterion = nn.NLLLoss()

epochs = 50
batch_size = 32
desired_batch_size = 32

lr=1e-3

optimizer = torch.optim.AdamW(model.parameters(), lr=lr)

grad_accumulation = desired_batch_size/batch_size
num_batches = len(train_data)//batch_size
```

1.2 CNN-LSTM MODEL ARCHITECTURE

```
class CNN(nn.Module):
    def __init__(self, in_channels, out_channels, kernel,
dropout, stride):
        super(CNN, self).__init__()
        self.in_channels = in_channels
        self.out_channels = out_channels
        self.kernel = kernel
        self.dropout = dropout
        self.stride = stride
        self.define_model()

    def define_model(self):
        self.layers = nn.Sequential(
            nn.Conv2d(self.in_channels,
                      self.out_channels,
                      stride=self.stride,
                      kernel_size=self.kernel,
                      padding=self.kernel//2),
            nn.MaxPool2d(2),
            nn.BatchNorm2d(self.out_channels),
            nn.GELU(),
            nn.Dropout2d(0.4)
        )

    def forward(self, ip):

        return self.layers(ip)

class CNNet(nn.Module):
    def __init__(self, resolution, min_channels, num_layers):
        super(CNNet, self).__init__()
        self.min_channels = min_channels
        self.resolution = resolution
        self.num_layers = num_layers
        self.define_model()

    def define_model(self):
        self.layers = nn.ModuleList()

        # First convolutional layer
        self.layers.append(
            CNN(
                in_channels=1,
```

```
                out_channels=self.min_channels,
                kernel=3,
                dropout=0.4,
                stride=1,
                ))

        # Additional convolutional layers
        for i in range(1, self.num_layers):
            self.layers.append(
                CNN(
                    in_channels=self.min_channels * (2 ** (i-1)),
                    out_channels=self.min_channels * (2 ** i),
                    kernel=3,
                    dropout=0.4,
                    stride=1))

    def forward(self, ip):

        x = ip.unsqueeze(1)
        del ip
        torch.cuda.empty_cache()

        for layer in self.layers:
            x = layer(x)
        return x

class BidirectionalLSTM(nn.Module):

    def __init__(self, rnn_dim, hidden_size, dropout,
batch_first):
        super(BidirectionalLSTM, self).__init__()

        self.BiLSTM = nn.LSTM(
            input_size=rnn_dim,
            hidden_size=hidden_size,
            num_layers=1,
            batch_first=batch_first,
            bidirectional=True)

        self.dropout = nn.Dropout(dropout)

    def forward(self, x):
        x, _ = self.BiLSTM(x)
        x = self.dropout(x)
        return x
```

```
class CNNLSTMNet(nn.Module):
    def __init__(self, n_classes, resolution):
        super(CNNLSTMNet, self).__init__()
        self.n_classes = n_classes
        self.min_channels = 8
        self.num_cnn_layers = 2
        self.num_lstm_layers = 1
        self.dropout = 0.4
        self.resolution = resolution
        self.ann_input_size = ((self.resolution**2) *

 self.min_channels)//(2**(self.num_cnn_layers+1))
        self.ann_output_size =
self.ann_input_size//self.min_channels
        self.define_model()

    def define_model(self):

        self.CNN = CNNet(
            resolution=self.resolution,
            min_channels=self.min_channels,
            num_layers=self.num_cnn_layers
        )

        self.fully_connected = nn.Sequential(
            nn.Linear(self.ann_input_size,
                      self.ann_output_size),
            nn.GELU(),
            nn.Dropout(self.dropout)
        )

        self.bilstm_layers = nn.Sequential(*[
            BidirectionalLSTM(
                rnn_dim=self.ann_output_size if i == 0 else
self.ann_output_size*2,
                hidden_size=self.ann_output_size,
                dropout=self.dropout,
                batch_first=False)
            for i in range(self.num_lstm_layers)
        ])

        self.classifier = nn.Sequential(
            nn.Conv1d(
                in_channels=1,
                out_channels=1,
                kernel_size=5,
```

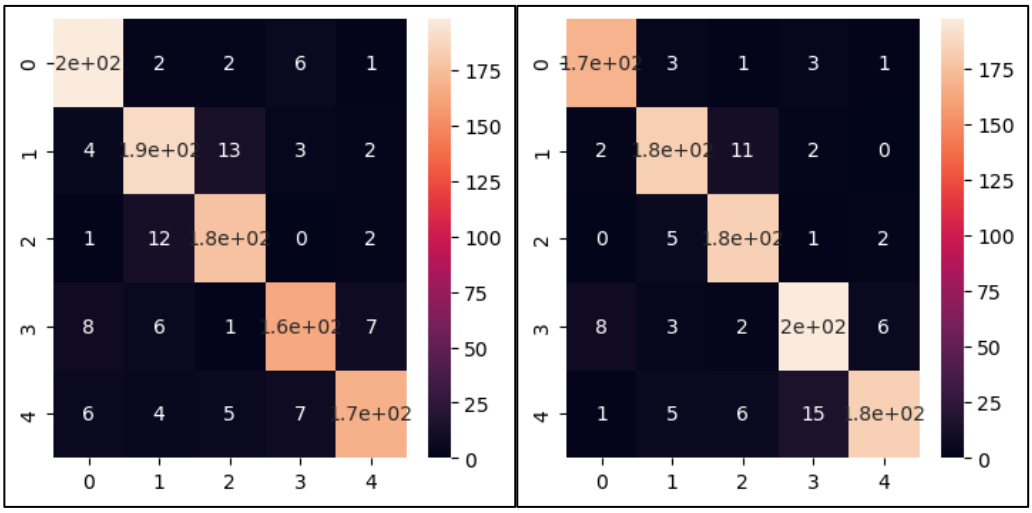
```
                stride=1,
                padding=2),
            nn.GELU(),
            nn.Linear(self.ann_output_size*2, self.n_classes),
            nn.LogSoftmax(dim=1)
        )

    def forward(self, images):

        # (time, x, y)
        # (time, channels, x, y) [time is passed as a batch]
        x = self.CNN(images)
        x = x.view(x.size(0), -1)  # (time, channels*x*y)
        # (time, features) [time is passed as a batch]
        x = self.fully_connected(x)
        # (time, batch, features) [batch size is always 1]
        x = x.view(x.size(0), 1, -1)
        # (time, batch, features*2) [time feature exists so batch
size is 1]
        x = self.bilstm_layers(x)
        # (time=last, batch=1, predictions) [time is passed as a
batch]
        x = self.classifier(x[-1])

        return x


epochs = 100
batch_size = 32
desired_batch_size = 32
grad_accumulation = desired_batch_size//batch_size

best_loss = 0.46
learning_rate = 1e-3

# create the dataloader
workers = 0
train = DataLoader( train_set,
                    batch_size=batch_size,
                    shuffle=True,
                    collate_fn=pad_collate,
                    pin_memory=True,
                    num_workers=workers)

val = DataLoader(   val_set,
                    batch_size=batch_size,
                    shuffle=True,
```

```
                    collate_fn=pad_collate,
                    pin_memory=True,
                    num_workers=workers)

test = DataLoader(  test_set,
                    batch_size=batch_size,
                    shuffle=True,
                    collate_fn=pad_collate,
                    pin_memory=True,
                    num_workers=workers)

# Initialise model
model = CNNLSTMNet(n_classes=n_classes, resolution=SIZE).cuda()

# Define loss function
criterion = torch.nn.NLLLoss()

# https://www.assemblyai.com/blog/end-to-end-speech-recognition-
pytorch/
# Define optimizer
optimizer = torch.optim.AdamW(model.parameters(),
lr=learning_rate)
```

APPENDIX 2 – OUTPUT SCREENSHOTS

2.1 CONFUSION MATRICES FOR VOWEL CLASSIFICATION

Figure 5. Confusion matrices of Vowel Classifier's Validation and Testing



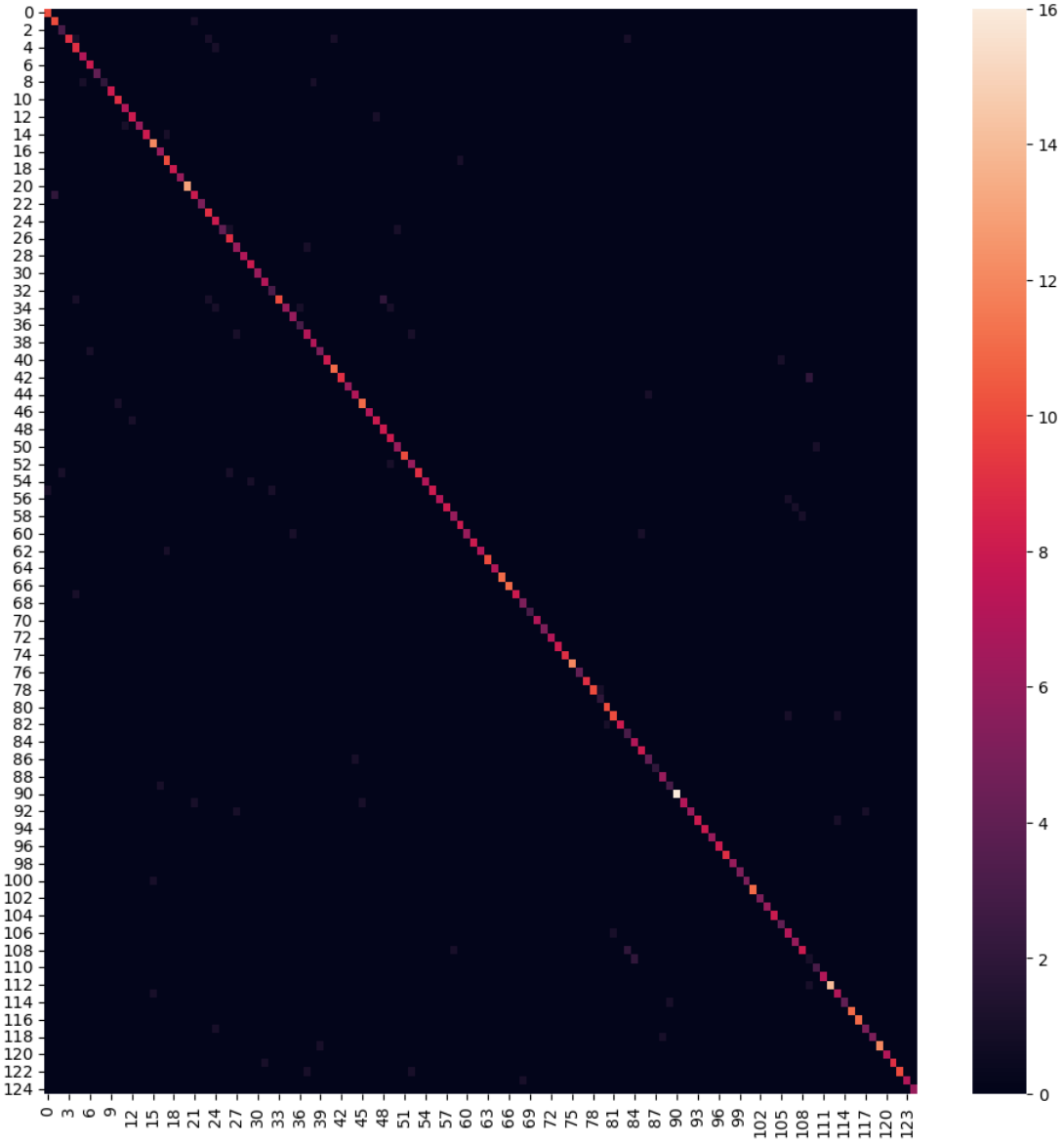
Validation Data Test Data

2.2 TEST CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.94	0.95	0.95	176
1	0.92	0.92	0.92	197
2	0.90	0.96	0.93	191
3	0.90	0.91	0.91	216
4	0.95	0.87	0.91	210
accuracy			0.92	990
macro avg	0.92	0.92	0.92	990
weighted avg	0.92	0.92	0.92	990

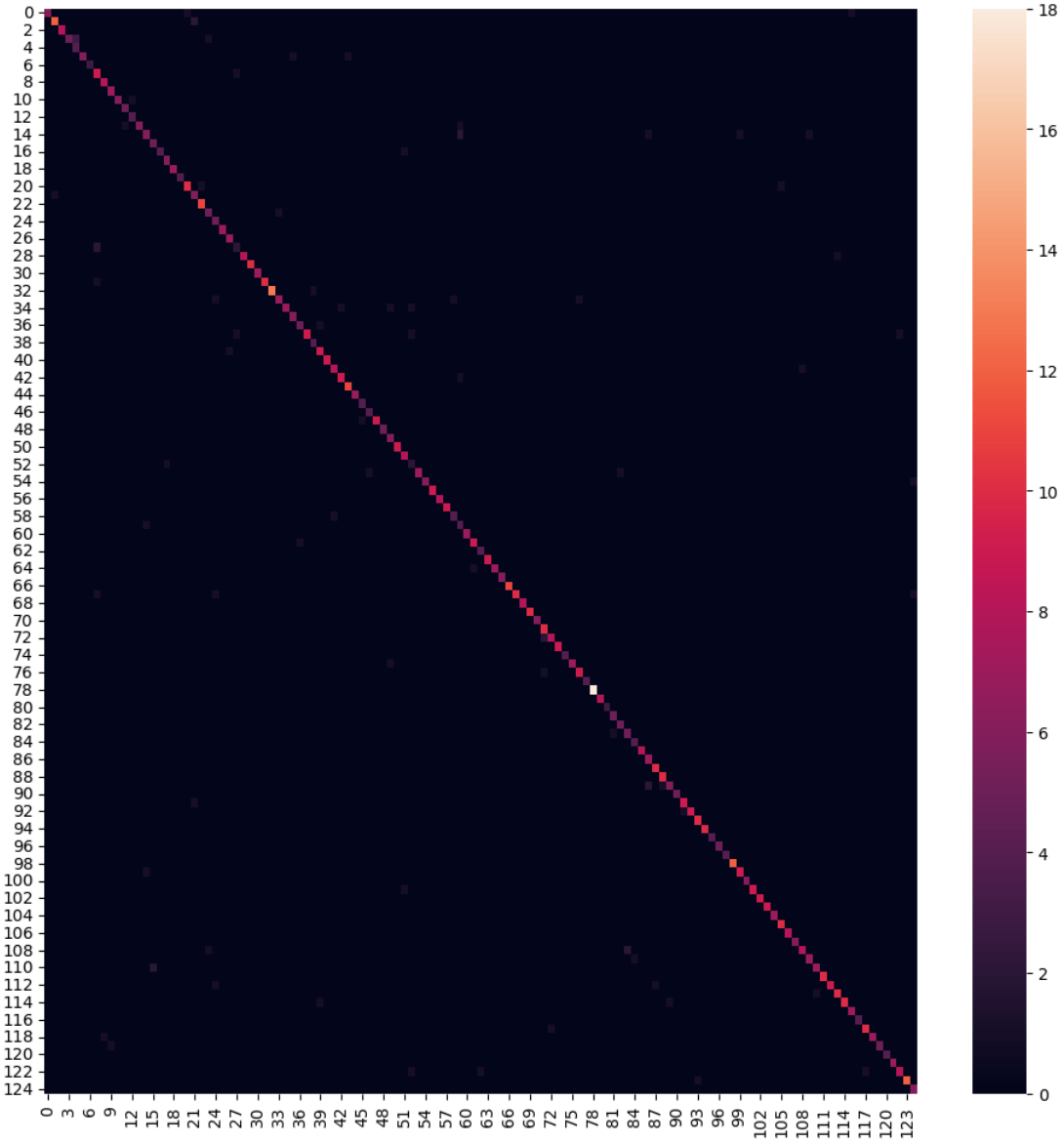
2.3 VALIDATION CONFUSION MATRIX FOR CVP CLASSIFICATION

Figure 6. Confusion matrix of CVP Classifier with RPS and MFCC (Validation)



2.4 TEST CONFUSION MATRIX FOR CVP CLASSIFICATION

Figure 7. Confusion matrix of CVP Classifier with RPS and MFCC (Testing)



REFERENCES

[1] Kennel, M. B., Brown, R., & Abarbanel, H. D. (1992). Determining embedding dimension for phase-space reconstruction using a geometrical construction. Physical review A, 45(6), 3403.

[2] Ma, H. G., & Han, C. Z. (2006). Selection of embedding dimension and delay time in phase space reconstruction. Frontiers of Electrical and Electronic Engineering in China, 1(1), 111-114.

[3] Povinelli, R. J., Johnson, M. T., Lindgren, A. C., & Ye, J. (2004). Time series classification using Gaussian mixture models of reconstructed phase spaces. IEEE Transactions on Knowledge and Data Engineering, 16(6), 779-783.

[4] Lindgren, A. C., Johnson, M. T., & Povinelli, R. J. (2003, April). Speech recognition using reconstructed phase space features. In 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP'03). (Vol. 1, pp. I-60). IEEE.

[5] Povinelli, R. J., Johnson, M. T., Lindgren, A. C., Roberts, F. M., & Ye, J. (2006). Statistical models of reconstructed phase spaces for signal classification. IEEE Transactions on Signal processing, 54(6), 2178-2186.

[6] Wallot, S., & Mønster, D. (2018). Calculation of average mutual information (AMI) and false-nearest neighbors (FNN) for the estimation of embedding parameters of multidimensional time series in MATLAB. Frontiers in psychology, 9, 1679.

[7] Wesley, R. J., Khan, A. N., & Shahina, A. (2020). Phoneme classification in reconstructed phase space with convolutional neural networks. Pattern Recognition Letters, 135, 299-306.

[8] Aggarwal, R. K., & Dave, M. (2011). Using Gaussian mixtures for Hindi speech recognition system. International Journal of Signal Processing, Image Processing and Pattern Recognition, 4(4), 157-170.

[9] Kumar, A., & Aggarwal, R. K. (2021). Discriminatively trained continuous Hindi speech recognition using integrated acoustic features and reconstructed neural network language modeling. Journal of Intelligent Systems, 30(1), 165-179.

[10] Patil, V. V., & Rao, P. (2016). Detection of phonemic aspiration for spoken Hindi pronunciation evaluation. Journal of Phonetics, 54, 202-221.

[11] Malviya, S., Mishra, R., & Tiwary, U. S. (2016, October). Structural analysis of Hindi phonetics and a method for extraction of phonetically rich sentences from a very large Hindi text corpus. In 2016 Conference of The Oriental Chapter of International Committee for Coordination and Standardization of Speech Databases and Assessment Techniques (O-COCOSDA) (pp. 188-193). IEEE.

[12] Dash, D., Kim, M. J., Teplansky, K., & Wang, J. (2018). Automatic Speech Recognition with Articulatory Information and a Unified Dictionary for Hindi, Marathi, Bengali and Oriya. In INTERSPEECH (pp. 1046-1050).

[13] Pandey, A., Srivastava, B. M. L., Kumar, R., Nellore, B. T., Teja, K. S., & Gangashetty, S. V. (2018, May). Phonetically balanced code-mixed speech corpus for Hindi-English automatic speech recognition. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).

[14] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., ... & Vesely, K. (2011). The Kaldi speech recognition toolkit. In IEEE 2011 workshop on automatic speech recognition and understanding (No. CONF). IEEE Signal Processing Society.

[15] Ravanelli, M., Parcollet, T., & Bengio, Y. (2019, May). The pytorch-kaldi speech recognition toolkit. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 6465-6469). IEEE.

[16] Indian Language TTS Consortium, & ASR Consortium. (2016). Indian Language Speech sound Label set (ILSL12).

[17] Kolobov, Rostislav & Okhapkina, Olga & Omelchishina, Olga & Platunov, Andrey & Bedyakin, Roman & Moshkin, Vyacheslav & Menshikov, Dmitry & Mikhaylovskiy, Nikolay. (2021). MediaSpeech: Multilanguage ASR Benchmark and Dataset.

[18] Pons, A., Serrà, J., Serra, X., & Rodríguez, A. (2021). End-to-end Speech Recognition in Spanish using Deep Learning. arXiv preprint arXiv:2102.02770.

[19] Baevski, A., & Auli, M. (2020). Wav2vec 2.0: A framework for self-supervised learning of speech representations. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (pp. 4433-4440).

[20] Kharroubi, H., Albadr, B., Habash, N., & Khalifa, S. (2021). Wav2Vec 2.0 for Arabic Speech Recognition: The Effect of Training Data and Model Size. arXiv preprint arXiv:2101.05058.

[21] Okutan, A. B., & Taneli, Y. (2021). Wav2Vec2 for Turkish Speech Recognition. arXiv preprint arXiv:2104.05739.

[22] Pons, A., Serrà, J., Serra, X., & Rodríguez, A. (2021). End-to-end Speech Recognition in Spanish using Deep Learning. arXiv preprint arXiv:2102.02770.