Chapter 3

Speech Recognition using features based on Discrete Cosine Transform

In chapter 2, we discussed the preliminaries and methodologies used in automatic speech recognition. This includes details of feature extraction and classification techniques. This chapter focuses on the elementary work for the recognition of digits spoken in the Gujarati language using three different techniques. For this purpose, we have used the classical discrete cosine transform-based feature extraction technique, the mel-frequency ceptral coefficient (MFCC). For classification purposes, we have used three different models. The first is based on finding a dynamic time warping distance, and the other two are machine learning models, namely multilayered perceptron and radial basis function networks. The phase-wise detailed description of work and discussion of performance results are explained in the subsequent sections.

3.1 Recording

The recording is done using the usual microphone on a laptop or mobile device. It can also be done with the use of an external microphone, which normally provides better speech with less noise. The recording is done in a normal room environment with modest noise. But if the real-life application requires recognising a speech in a noisy environment, then recording may be done in the noisy environment. So the recording environment is vital to the type of application perceived.

During recording, there are two important factors to be chosen, namely the sampling rate and the number of channels. Sampling rate is related to the discretisation of continuous speech. It is defined as the number of samples per second in a recording. The usual recording sampling rate for music is around 48,000 samples per second. For speech, the

common sampling rates of the recording are 8000 or 16,000 samples per second. If the sampling rate is high, more data needs to be processed while training the model. This affects the training time and accuracy of the model. Hence, the ideal sampling rate to be considered for speech recognition is 8000-16,000 samples per second. As mentioned previously, another important term is the number of channels in the recording. In stereo recording, the number of channels is two, and in mono recording, there is a single channel. Stereo channels are normally required for songs and music. For speech, mono recording is sufficient. Mono channel means there is a single array of samples. While in stereo recording, there are two arrays of samples. These two arrays refer to the left and right speakers of instruments like headphones.

Speech recordings can be stored on a computer or mobile device in various formats, like *.wav, *.mp3, *.aac, *.wma, and *.flac. But the most commonly used audio format is a *.wav file. The value of samples during discretisation of continuous speech can be either a float or an unsigned integer, leading to different types of file extensions. The *.wav format has further encodings like 16-bit signed, 24-bit signed, 32-bit signed, 8-bit unsigned, 32-bit float, and 64-bit float. These encodings correspond to the data type of the numbers that are forming the digital speech signal. More commonly, 16-bit signed encoding is used.

The recordings are done in any of two ways. In the first one, each word spoken by a speaker is recorded as a single file. On the other hand, the sentences spoken by the speaker are recorded as a single file. That is, as per the requirements of the applications, recordings can be done word-wise or sentence-wise. Once the recording is done, the output is an array of numbers. The number of elements in this array will depend on the sampling rate of the recording. The values in the array will depend on the format and encoding of the stored recording. So if the 3 seconds of speech are recorded using a mono channel with a sampling rate of 8000 samples per second and in 16-bit signed integers in *.wav format, then we will get a single-dimensional array with 24,000 values in the range of -32,768 to 32,767. This value can be normalised between -1 and 1. The next important step in the speech recognition system is removing noise.

For the models discussed in this chapter, to construct the vocabulary, we selected 10 words from the Gujarati language: digits one to ten. The recording is done using the open-source software named Audacity in a normal laboratory environment. The recording is done at a sampling rate of 16,000 Hz with a single mono channel. During recording, each speaker utters digits one to ten in the Gujarati language with a slight pause between two digits. This recording is stored as an audio file with the extension *.wav and signed 16-bit PCM (pulse-code modulation) encoding. The plot of one such recording is shown in the Figure 3.1. We have recorded the vocabulary of digits, one to ten, spoken by 10

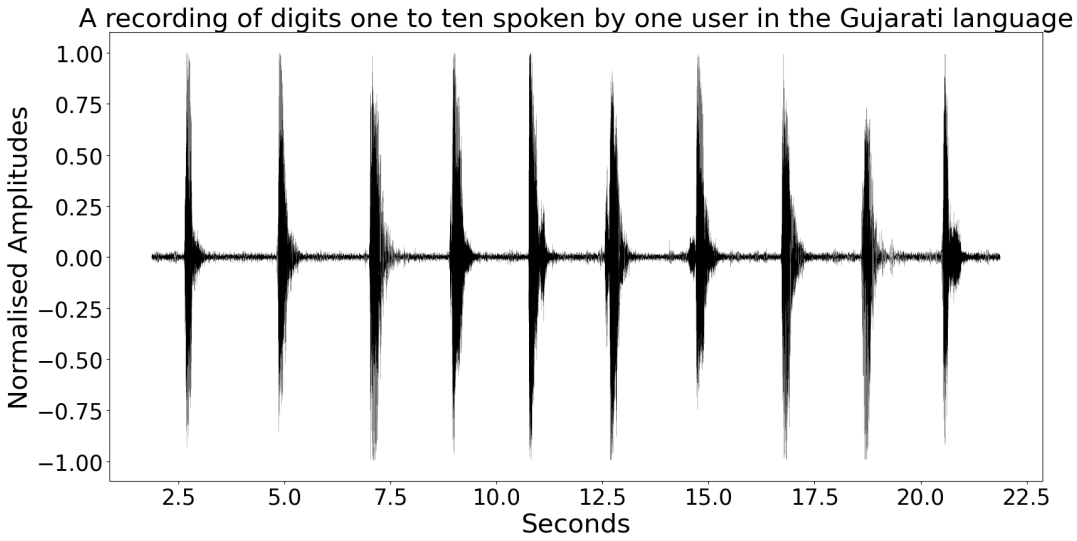

Figure 3.1: A recording of digits one to ten spoken by one speaker in the Gujarati language

speakers in the Gujarati language.

3.2 Noise Removal

In the recording with the usual microphone, even if the recording is done in a quiet environment, it is possible that the recording has some background noise. Some of this background noise can be due to a fan, air conditioner, speaker's breath, or noise from the recording device like a laptop or computer. It is important to remove these noises from the speech signal, as in the next steps, the features are to be extracted from these recordings. The advantage of removing noise from the recording is that the features extracted from it will be purely based on the speech data.

Noise removal can be done automatically or manually. First, the utmost care has to be taken during the recording. Secondly, if the recording still has inherent noise, the open-source software Audacity can be used for noise removal. This reduces background noises like hum, whistle, whine, buzz, "hiss," fan noise, and FM carrier noise. It filters the noise using the noise reduction algorithm. This algorithm is based on Fourier analysis, which finds the spectrum of pure tones that generate the background noise in the quiet sound. This algorithm finds the frequency spectrum of each short segment of audio. Any pure tones that are not sufficiently louder than their average levels of background noise are reduced in volume. Thus, the noise can be easily removed after the recording. However, if the application demands the system recognise the speech in an environment with noise, then this step of noise removal can be skipped.

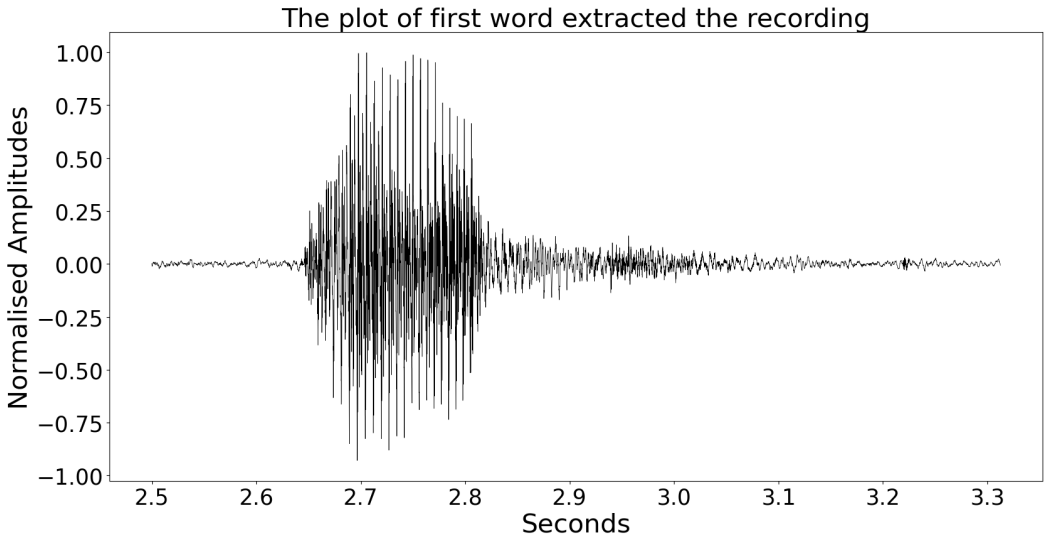

Figure 3.2: The plot of first word extracted from the recording shown in the Figure 3.1

3.3 Extracting Words

Figure 3.1 shows the speech signal for the digits one to ten spoken in Gujarati by one speaker. We can observe that the signal is a combination of high-amplitude phases along with very small-amplitude ones in between. These high amplitude phases corresponds to the spoken digits and the small amplitude phases corresponds to the pauses in between these spoken digits.

In the initial work, we extracted the words manually from the speech using the Audacity software. This is done by cutting the high-amplitude wave parts, each corresponding to a digit. From the single recording, such as shown in the Figure 3.1, we crop it to signals corresponding to digits and store them as a separate *.wav files. The plot of one such cropped word is shown in the Figure 3.2. Thus, the database consists of 10 files, each containing spoken digits by single speakers. In total, 10 (digits) $\times$ 10 (speakers) = 100 files. These are further used for feature extractions.

Figures 3.2 and 3.3 show the first word (the digit Ek in Gujarati) spoken by two different speakers. Here, we can observe that even though the word is the same, there are variations in the amplitudes representing the utterance of this word by two different speakers. In the next section, we describe the feature extraction process.

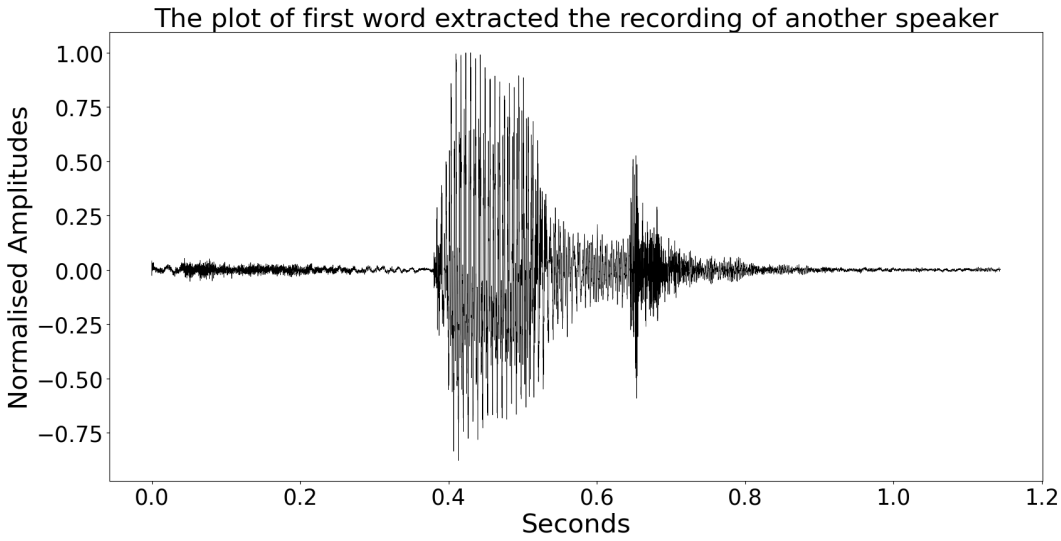

Figure 3.3: The plot of first word extracted from the recording of another speaker

3.4 Feature Extraction

For the models in this chapter, we have extracted features from the speech using the mel-frequency cepstral coefficient method. First of all, the pre-emphasising of the recorded speech signal $x(t)$ is done using equation (3.1).

$$s(t) = x(t) - 0.95x(t) \tag{3.1}$$

Next, the pre-emphasised signal is divided into frames with 256 samples and overlapping 100 samples with the adjacent frame. These frames are represented by equation (3.2). Here, index i represents the frame number. The total number of frames depends on the length of the signal.

$$s_i(t), \qquad 0 \leqslant t \leqslant 255 \tag{3.2}$$

Then, the hamming window, defined by equation (3.3), is applied to all the frames. This gives equation (3.4).

$$w(t) = 0.54 - 0.46\cos\left(\frac{2\pi t}{255}\right), \qquad 0 \leqslant t \leqslant 255 \tag{3.3}$$

$$s_i(t)w(t), \qquad \forall i \tag{3.4}$$

Further, we take the discrete Fourier transform of the equation (3.4). This is done using the equation (3.5). The power spectrum of equation (3.5) is given by equation (3.6).

$$\mathrm{DFT}_i(f) = \sum_{t=0}^{255} s_i(t)w(t)e^{\frac{-2f\pi jt}{256}}, \qquad 0 \leqslant f \leqslant 255 \tag{3.5}$$

$$\mathrm{PS}_i(f) = \frac{1}{256}|\mathrm{DFT}_i(f)|^2, \qquad 0 \leqslant f \leqslant 127 \tag{3.6}$$

Next, the frequency scale is converted to a mel-scale using the equation (3.7). Here, f is the input frequency and M is the output mel.

$$\mathrm{Mel}(f) = 1127 \ln\left(1 + \frac{f}{700}\right) \tag{3.7}$$

On this mel-scale, we define 20 triangular-shaped filters defined by equation (3.8).

$$\mathrm{Filter}_f(k) = \begin{cases} \frac{k-\mathrm{Mel}(f-1)}{\mathrm{Mel}(f)-\mathrm{Mel}(f-1)}, & \mathrm{Mel}(f-1) \leqslant k \leqslant \mathrm{Mel}(f) \\ \frac{\mathrm{Mel}(f+1)-k}{\mathrm{Mel}(f+1)-\mathrm{Mel}(f)}, & \mathrm{Mel}(f) \leqslant k \leqslant \mathrm{Mel}(f+1) \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

Then, we apply these mel-scaled filters to the power spectrum, which gives $\hat{S}_k$, the energy output of k^{th} filter.

$$\hat{S}_k = \mathrm{PS}_i(f) \cdot \mathrm{Filter}_f(k) \tag{3.9}$$

Finally, the mel-frequency cepstral coefficients are determined by taking the discrete cosine transform of the log-energy output of k^{th} filter. This is determined by equation (3.10). For each frame, the first nine coefficients are included in the feature vector.

$$\hat{c}_n = \sum_{k=1}^{20} \ln\left(\hat{S}_k\right) \cos\left[n\left(k - \frac{1}{2}\right)\frac{\pi}{K}\right] \tag{3.10}$$

Various parameters for the pre-processing and feature extraction processes are summarised below:

- Number of speaker: 10
- Vocabulary: Digits 1-10 spoken in Gujarati
- Database: 100 *.wav files
- Feature extraction technique: MFCC
- Pre-emphasising: $\alpha = 0.95$
- Frame-width: $N = 256$ samples
- Overlapping frames: $M = 100$ samples

- Hamming window: $\beta = 0.46$
- Number of filters: 20
- Number of coefficient per frame: 9

Before using the extracted features for classification of spoken digits, it is important to make their lengths equal. This is required for some machine learning models. So, next section explains a procedure to do this.

3.5 Making the Lengths of Feature vector equal

The length of the feature vector for different words is not equal. Because it depends on how fast or slow a speaker is speaking. To train machine learning algorithms like multilayered perceptron or radial basis function networks using these feature vectors, one more preprocessing technique is required that makes the lengths of the feature vectors equal.

The lengths of two arrays can be made equal using various methods. In signal processing, several padding techniques are used to pad values on one or more boundaries of a signal. Some common techniques are zero padding, reflect padding, replicate (or edge) padding and circular padding. In zero padding, the required number of zeros are placed at one or both boundaries of the signal. Reflect padding is achieved by mirroring the signal at the boundaries. Replicate (or edge) padding involves replicating the extreme boundary values on both sides. Circular padding repeats the signal values in such a way that a circular pattern, of signal values, is maintained.

We have used zero padding for models in this chapter, because this method will not introduce any additional noise in the existing speech signal. Using this method, we append zeroes at the end of the array to make its length equal to the longest feature vector. Suppose the longest feature vector is of length 1000 and another feature vector is of length 929. In this case, we have to add 71 zeroes to the feature vector, which has a length of 929. This method is computationally inexpensive. But it can lead to a large number of zeroes in the data.

In the subsequent sections, the classification of feature vectors using various models, is explained.

	Speaker S2										
Speaker S1	**Digits**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	1	**2200**	4391	3083	4411	3082	3674	4453	3921	4293	2901
	2	3984	**2191**	2891	4489	3006	4025	3818	3538	2994	2453
	3	2645	4457	**1503**	5172	4044	3553	4551	4292	3321	3926
	4	6399	6857	3869	**3714**	4267	4294	**3128**	4301	6524	6391
	5	3356	4559	3084	3798	**2178**	3874	4227	**3065**	4185	2481
	6	3775	7193	3313	5305	5903	**3082**	5146	5925	4585	6579
	7	6932	6618	4736	4171	5277	5250	3520	4881	7302	7493
	8	4450	5364	2153	4908	4053	4111	4300	3313	4134	4038
	9	2900	5234	2748	7225	5942	4995	5975	5440	**2302**	4630
	10	3338	3100	3629	4838	2774	4569	4867	3650	4054	**1402**

Figure 3.4: DTW distances between two speakers using MFCC

3.6 Model 1: Dynamic Time Warping

As mentioned before, we observe that the speech signals for the same word, spoken by two different speakers, are entirely different. Hence, the length of the speech signals differs. Therefore, the corresponding feature vector lengths are also different. The idea of this model is to determine how close these two utterances are based on the features obtained from them. In the chapter on preliminaries, we learned how dynamic time warping can be used to find the distance between two unequal length vectors. The similarity computation formula, given in the equation (3.11), is used for the two feature vectors of unequal lengths, as

$$D(x_i, y_j) = |x_i - y_j| + \min\{D(x_i, y_{j-1}), D(x_{i-1}, y_{j-1}), D(x_{i-1}, y_j)\} \tag{3.11}$$

Here, i and j represent indices for lengths of signals x and y respectively. The features of all the words by one speaker compared with those of the other speakers are shown in the Figure 3.4. Each cell is the dynamic time warping distance, found using equation (3.11), between the feature vectors corresponding to words spoken by a speaker S1 and the feature vectors corresponding to words spoken by another speaker S2.

The column under the title '1' represents the comparison of the digit '1' spoken by speaker S2 with all the digits of speaker S1. The highlighted cells represent the smallest dynamic time warping distance in each column. We can observe that, in the first column, the minimum distance is in the first cell only. This means that even if the utterance of the same word by two different speakers may not match, their features show the correlation.

Overall, Figure 3.4 concludes that 8 out of 10 digits spoken by the selected speakers match according to minimum distance. This process is repeated for all speakers, keeping one speaker's spoken word features as a template. Similarly, tables are prepared for comparison of the feature vectors of words of one speaker with all other speakers. Overall, 84.36 % matching is achieved considering data corresponding to all the speakers, as in [75]. However, this accuracy increases to 95.56 % if the outlier data is removed from the analysis.

3.7 Model 2: Multilayered Perceptrons

In the next approach, to achieve our objective of classifying words spoken by various speakers, we used multilayered perceptrons, which are efficient classifiers. Here, we used feature vectors, obtained from the digits 1-10 spoken by speakers, as inputs to a multilayered perceptron and trained it to classify digits. We had 10 digits 1-10 spoken by 10 speakers as row data.

The feature vectors that are obtained from speech may not be equal in length because they represent different digits spoken by different speakers. Some words are longer as compared to others. Since some speakers speak faster while others speak slower, This makes all the feature vectors of different lengths. To use them as an input to the multilayered perceptron, it is necessary to make the length of each vector equal. This is achieved by zero padding at the end of the input vectors that are smaller than the vector with the maximum length.

We observe that, out of all the hundred feature vectors, the maximum length of the feature vectors is 1251. So, we made all the other feature vectors equal to length 1251 by appending zeroes to the smaller ones.

A multilayered perceptron with architecture $N^2_{1251,50,1}$ is constructed as shown in the Figure 3.5. The input is $\vec{x} = (x_1, x_2, \ldots, x_{1251})$. The neurons in the hidden layers are $\vec{z} = (z_1, z_2, \ldots, z_{50})$. The output is y. It is the multilayered feed-forward architecture of an artificial neural network. This architecture has 1251 input neurons, representing feature vectors, and 1 output neuron, representing digits 1-10. After trying out different numbers of neurons, 50 neurons in the hidden layer gave the best result.

In this multilayered perceptron model, we use the sigmoid activation function for the hidden layer and the output layer, given by equation (3.12).

$$f(\vec{w} \cdot \vec{x}) = \frac{1}{1 + e^{-\vec{w} \cdot \vec{x}}} \tag{3.12}$$

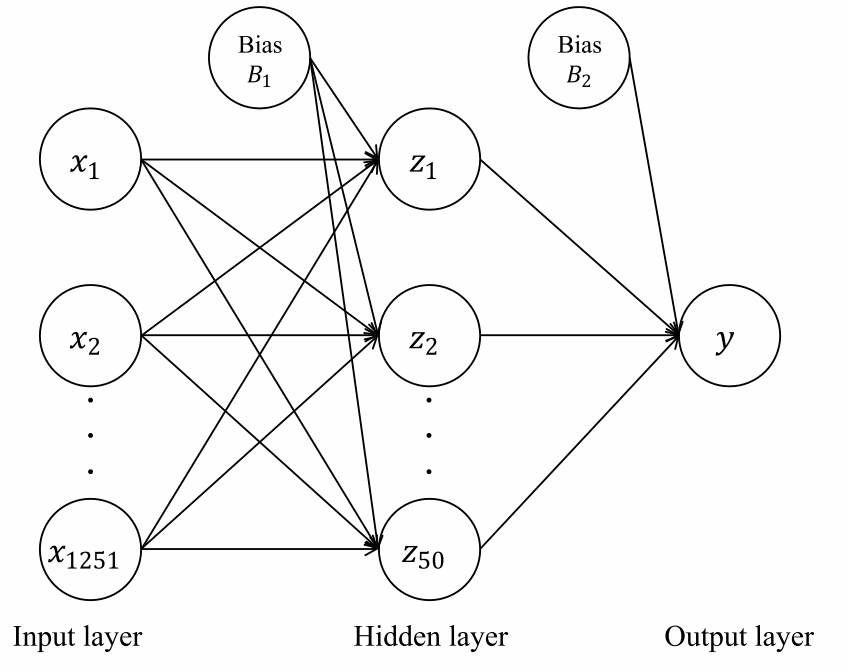

Figure 3.5: Architecture of the multilayered perceptron used in the model

We use 80% of the data for training and preserve the remaining 20% of the data for testing. Squared error loss, defined by equation (3.13), is used as a loss function.

$$E = \frac{1}{2}\sum_{i=1}^{N}(d_i - y_i)^2 \tag{3.13}$$

An error back-propagation algorithm is used to train the weights of this network using the equations (3.14) and (3.15), for hidden and output neurons, respectively.

$$v_j^{(k+1)} = v_j^{(k)} + \eta_o\,(d-y)\,f_o'\left(\sum_{j=0}^{50} v_j^{(k)} z_j\right) z_j \tag{3.14}$$

$$w_{ji}^{(k+1)} = w_{ji}^{(k)} + \eta_h\,(d-y)\,f_o'\left(\sum_{j=0}^{50} v_j z_j\right)\sum_{j=0}^{50} v_j f_h'\left(\sum_{i=0}^{1251} w_{ji}^{(k)} x_i\right) x_i \tag{3.15}$$

For the training dataset, 100% accuracy is obtained. The model is then tested with the remaining 20% of the data. Out of all the 20 test words, 15 are recognised correctly, giving an accuracy of 75%. This model, after classifying a test digit, gives the output in Gujarati language digit as well as the corresponding image in signed language, as shown in the Figure 3.6, refer [76]. Such representation is useful for people with disabilities or people who know only Gujarati.

There are many possible reasons for the lower accuracy of this model. One of the reasons is the size of the dataset. Another reason may be due to the overfitting of the model.

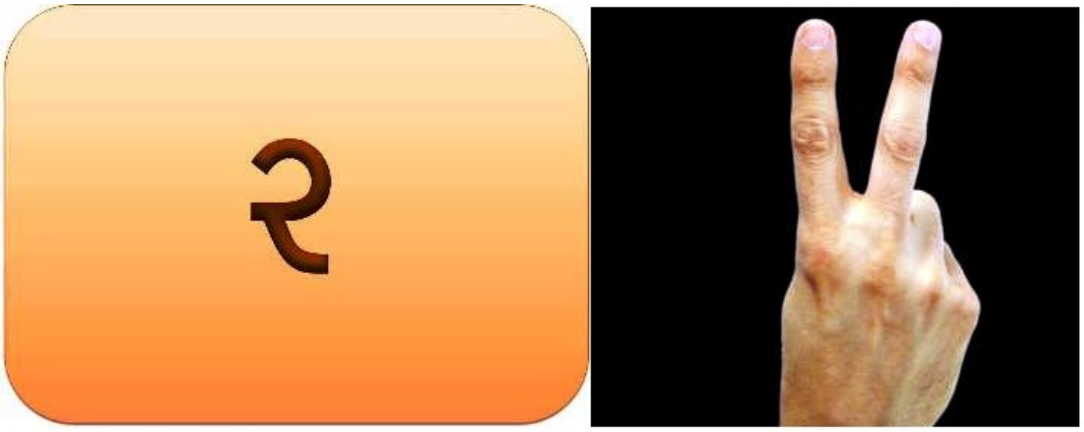

Figure 3.6: Output of the recognised digit 2

3.8 Model 3: Radial Basis Function Network

The next model, for speech recognition of Gujarati words, uses a radial basis function network approach. It is one type of multilayered feed-forward artificial neural network used for classification problems. The main reason for using a radial basis function network is to improve accuracy for speech recognition and to achieve faster convergence for training data.

For this model, similar to the previous work, first of all, the lengths of all the feature vectors are made equal to the largest length, 1251. This is done by inserting additional zeros in place of missing values. By this way, we get 100 feature vectors of the same length (1251), representing the digits 1 to 10 spoken by ten speakers in Gujarati. These 100 feature vectors are used for training the radial basis function network.

The radial basis function network with architecture $N^2_{1251,100,4}$ is constructed as shown in the Figure 3.7. There are 1251 neurons in the input layer given by $\vec{x} = (x_1, x_2, \ldots, x_{1251})$, 100 neurons in the hidden layer $(J_1, J_2, \ldots, J_{100})$ and 4 neurons in the output layer $\vec{y} = (y_1, y_2, y_3, y_4)$. The Gaussian radial basis function, given by equation (3.16), is used as an activation function in the 100 neurons of the hidden layer.

$$\phi(r) = e^{\frac{-r^2}{2\sigma^2}} \tag{3.16}$$

The 4 neurons in the output layer suffice for the classification of 10 digits, as they together represent the binary equivalent of digits 1 to 10. The digit 1 is represented by (0,0,0,1), the digit 2 is represented by (0,0,1,0), and so on. Finally, digit 10 is represented by (1,0,1,0). The binary representation of digits is helpful to reduce the number of output neurons. This optimises the number of neurons in the output layer.

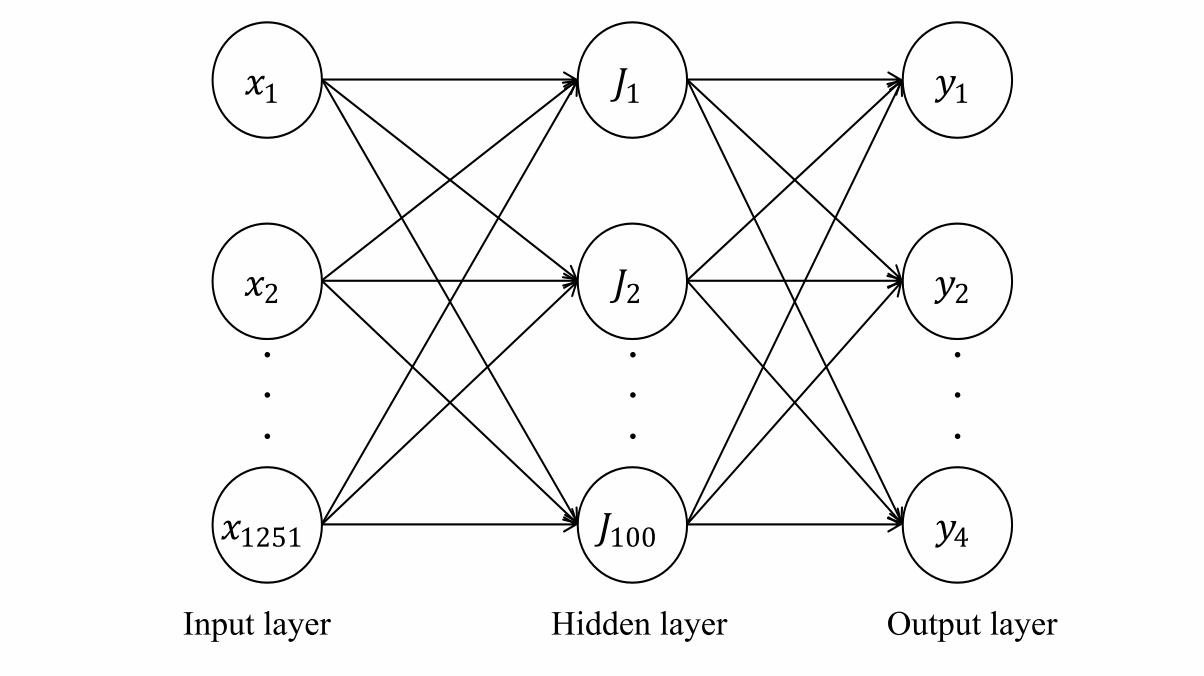

Figure 3.7: Architecture of the radial basis function network used in the model

To train the radial basis function network, it is important to normalise the input data. So, the data is normalised using the equation (3.17), for the i^{th} feature in the sample.

$$z_i = \frac{x_i - \bar{x_i}}{s_i}, \tag{3.17}$$

Here, x_i is the i^{th} feature in the input data, $\bar{x_i}$ is the mean and s_i is the standard deviation of the i^{th} feature.

To start training, the centres and weights of the network are chosen randomly, initially. During the training, they are updated using equations (3.18) and (3.19).

$$w_{mi}^{k+1} = w_{mi}^{k} + \frac{2\eta_1}{100} \sum_{n=1}^{100} \sum_{i=1}^{4} e_{ni} \sum_{m=1}^{100} \phi\left(||\vec{x_n} - \vec{c_m}||\right) \tag{3.18}$$

$$\vec{c_m}^{\,k+1} = \vec{c_m}^{\,k} - \frac{2\eta_2}{100} \sum_{n=1}^{100} \sum_{i=1}^{4} e_{ni} \sum_{m=1}^{100} w_{mi} \phi'\left(\left\|\vec{x}_n - \vec{c_m}^{\,k}\right\|\right) \frac{\vec{x}_n - \vec{c_m}^{\,k}}{||\vec{x}_n - \vec{c_m}^{\,k}||} \tag{3.19}$$

During the training, the radial basis function network is able to successfully classify all the 100 feature vectors given for training.

For testing, the new recording of 10 digits by an unknown speaker is considered. The network is able to classify nine features out of 10, giving an accuracy of 90%. The time required for training is just 8 seconds, which is significantly less as compared to the multilayered perceptrons refer [77]. The output for the recognised digit is the image of

the corresponding digit in Gujarati as well as its equivalent sign language form, as shown in Figure 3.6.

3.9 Accuracy Comparison for Isolated Words based on MFCC

The accuracies obtained for these approaches are summarised in the Table 3.1. We conclude that the radial basis function network algorithm yields the highest accuracy for the recognition of isolated words using MFCC.

Classification algorithm	**Accuracy**
Dynamic time warping	84.36%
Multilayered perceptron	75.00%
Radial basis function network	90.00%

Table 3.1: Accuracies obtained for Speech recognition of isolated words using MFCC

We achieved relatively good accuracy 90% in the models based on a radial basis function network. The dynamic time warping gave a moderate 84.36% accuracy. Due to overfitting and the size of the data, the accuracy of the multilayered perceptron is relatively low, i.e., 75%.

3.10 Conclusion

In this chapter, we discussed three elementary models of speech recognition. The objectives of these models are to demonstrate various methods for classification and determine which one works best. In all three models, we considered speech recognition of isolated words in the form of digits 1 to 10 spoken in the Gujarati language by 10 different speakers. For each model, the feature extraction is done using the mel-frequency cepstral coefficients. For the classification, we have used three approaches: dynamic time warping, multilayered perceptrons, and radial basis function networks.

In the next chapter, we discuss about the similar models done with the different feature extraction technique and show that such a feature extraction changes the accuracy. We will also consider recognition of continuous sentences in the next chapter.