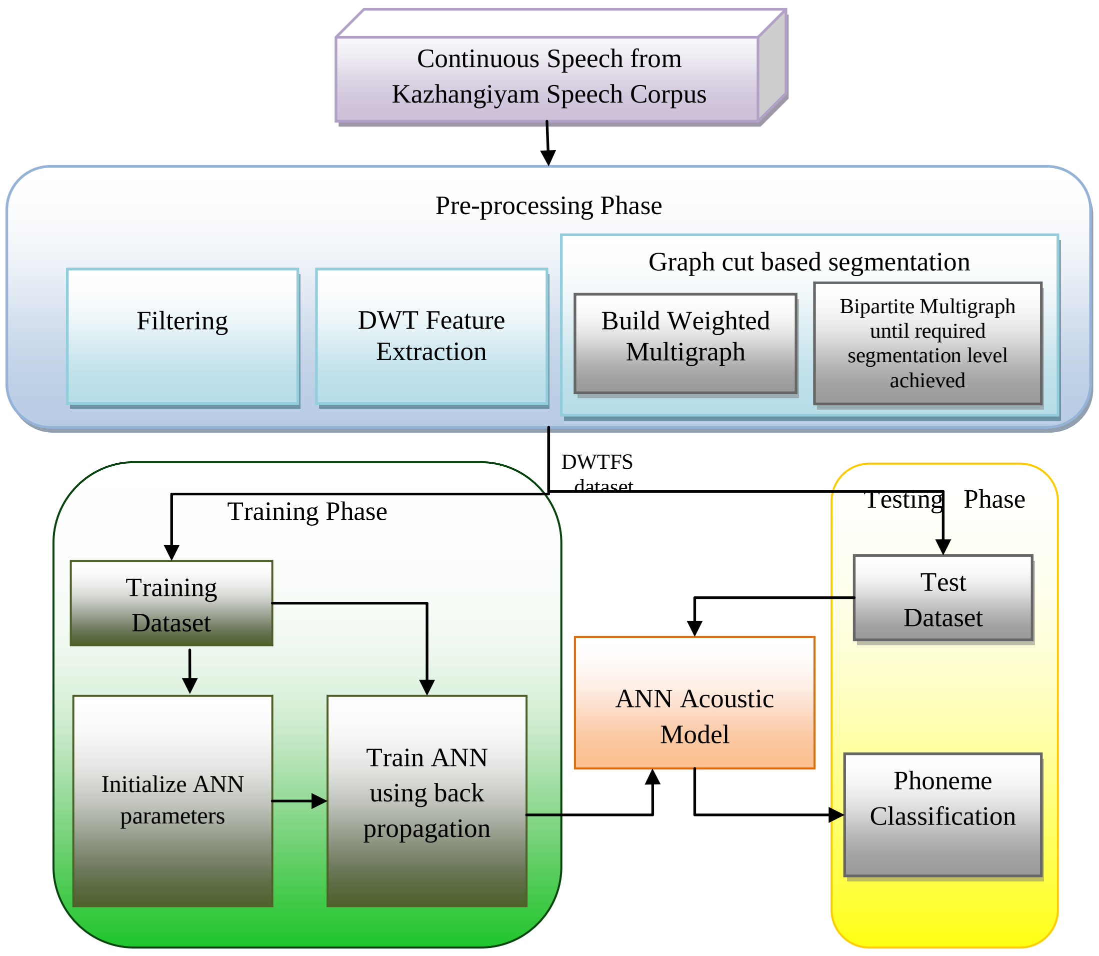
4. ARTIFICIAL NEURAL NETWORKS AND ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM FOR PHONEME RECOGNITION

This chapter discusses the pilot study undergone during the initial stage of the research on building a phoneme recognition model. The study has been conducted by building two models; one using ANN and the other using Adaptive Neuro-Fuzzy Inference System (ANFIS) to classify the phonemes in the dataset. The performance of the acoustic models discussed in this chapter has been analyzed based on the various standard measures in the literature namely precision, recall, F-measure and classification accuracy on the broad classes of phonemes namely, vowels, nasals, stops and others and are detailed in the experiments and results section. The experiments show the performance of ANN phoneme recognition model outperforming to the performance of ANFIS phoneme recognition model.

4.1 TAMIL PHONEME RECOGNITION MODEL USING ANN

The phases involved in building ANN phoneme recognition model is shown in Fig. 4.1. The dataset developed as described in chapter 3 is used to build the ANN phoneme recognition model. The class of ANN, namely the Multilalyer Perceptron (MLP) is used here. First the infrastructural parameters like number of layers and number of neurons in each layer are defined. The training dataset is passed batch-wise to this simple ANN with one hidden layer where the input feature vectors of the phonemes are fed to the input layer of the ANN and propagated to the output layer through the neurons activated in the hidden layer. The supervised training is performed by evaluating the error observed at the output layer and propagating it back through the layers of the network from the output layer to the input layer. The learning process is repeated for the required number of epochs until the network converges to the expected level of accuracy. For the training phase of the model, the model parameters namely connection weights and bias to each neuron are initialized. The test dataset is then fed to the model to evaluate the performance of the model built. The complete process involved in model building is portrayed in the following algorithm.



Continuous Speech from Kazhangiyam Speech Corpus
Pre-processing Phase
Filtering
DWT Feature Extraction
Graph cut based segmentation
Build Weighted Multigraph
Bipartite Multigraph until required segmentation level achieved
DWTFS dataset
Training Phase
Training Dataset
Initialize ANN parameters
Train ANN using back propagation
ANN Acoustic Model
Testing Phase
Test Dataset
Phoneme Classification

Fig. 4.1 Architecture of ANN Based Phoneme Recognition Model

Algorithm 4.1 Building ANN based phoneme recognition model using back propagation algorithm

Step 1: Define hyperparameters namely, the number of hidden layers *m-1*, number of neurons in each layer of ANN r_i, *for* $i=0,1,\ldots.m$, the learning rate γ, number of epochs and batch size.

Step 2: Randomly initialize the connection weights w_{ji}^{k}, for all connections from *j* node in layer *k-1* to *i* nodes in layer *k* and biases of all nodes represented as b_i^k for i^{th} node in layer *k*.

Step 3: Repeat steps 3 to 7 for the given number of epochs

Step 4: Repeat steps 4 to 7 for each batch of the training dataset.

Step 5: Perform the forward pass by supplying the data points in the current batch to the input layer until it reaches the output layer

a. Initialize input layer l_0 by assigning $o_i^0 = x_i$, where o_i^0 is the output of layer l_0 and x_i, the input vector.

b. For all hidden layers in sequence from $l_1\ to\ l_{m-1}$

   Do for k from 1 to m-1

   i. Calculate product sums of input to neuron,

$$h_i^k = b_i^k + \sum_{j=1}^{r_{k-1}} w_{ji}^k o_j^{k-1}, for\ i = 1, \ldots . r_k$$

   ii. Calculate $o_i^k = g(h_i^k), for\ i = 1, \ldots . r_k$

c. Evaluate the output at the output layer l_m

   i. Calculate $\mathrm{h_i^m = b_i^m + \sum_{j=1}^{r_{m-1}} w_{jl}^m o_j^{m-1}, for\ i = 1, \ldots . r_k}$

   ii. Calculate $\mathrm{o_l^m = g(h_l^m)}$

Step 6: Evaluate the mean squared error at the output layer as follows,

$E(X) = \frac{1}{2}\sum_{i=1}^{N}(o_i - y_i)^2$, where o_i, is actual output vector and y_i, is expected output vector for the given input vector, x_i.

Step 7: Back propagate the change in error from the output layer to the input layer to fine tune the weight and bias parameters of the network.

a. Calculate change in weights as follows,

$$\Delta w_{ij}^k = -\gamma \frac{\partial E(X)}{\partial w_{ij}^k}$$

b. Calculate change in biases as follows,

$$\Delta b_i^k = -\gamma \frac{\partial E(X)}{\partial b_i^k}$$

c. Update all weight and biases as follows,

$$w_{ij}^k = w_{ij}^k + \Delta w_{ij}^k \text{ and } b_i^k = b_i^k + \Delta b_i^k$$

Thus to build an ANN based acoustic model, the hyperparameters like the number of layers, number of neurons in each layer, the activation function used are decided and the respective parameters of ANN like bias and connection weights are initialized. The model is then pretrained using contrastive divergence technique and then trained using backpropagation which propagates the error observed in the output layer.

Experiment and Results

In this experiment, a DWTFS dataset of Kazhangiyam corpus discussed in section 3.4 is used for building a phoneme classification model. The DWTFS dataset used here is a collection of data points, where each data point represents the DWT features of segmented phonemes of the speech corpus. Each phoneme is defined as a data point with 90 DWT features. From a total of 667260 phoneme samples defined in the dataset excluding the data points falling into 'sil' class, the remaining 635030 data points are split into training set and testing test with 70:30 proportion contributing 444521 instances in training dataset and 190509 instances in the test dataset.

The experiment is conducted in MATLAB R2013a. The hyperparameters for ANN to build the phoneme recognition model are set as follows. The number of neurons in input layer is set as 90, number of neurons in output layer as 38, learning rate or step ratio as 0.01 and number of epochs as 500. The activation function of neurons is defined as a sigmoid function. The experiment is conducted by varying the number of hidden neurons as 70, 150, 200 and 250. The accuracy of the model when trained with datasets by varying the number of instances as 31200, 62400, 124800, 249600 and 444521 are evaluated which is portrayed in Table IX and Fig 4.2. The ANN model is perceived to improve its accuracy with the increase in the number of neurons in the hidden layer but tends to start decreasing at some point when exceeding beyond 200 for this problem. Further investigation on the performance of ANN continued with 200 hidden neurons.

Table IX Phoneme Recognition Rate (%) of ANN Model by Varying the Size of Hidden Layer

Training Dataset Size	Number of Neurons in Hidden Layer			
	70	**150**	**200**	**250**
31200	50.4	50.2	51.3	50.8
62400	53.2	54.1	54.8	53.5
124800	57.8	58	58.1	57.8
249600	59.5	60.2	60.4	58.2
444521	63.9	64.2	64.5	64.1

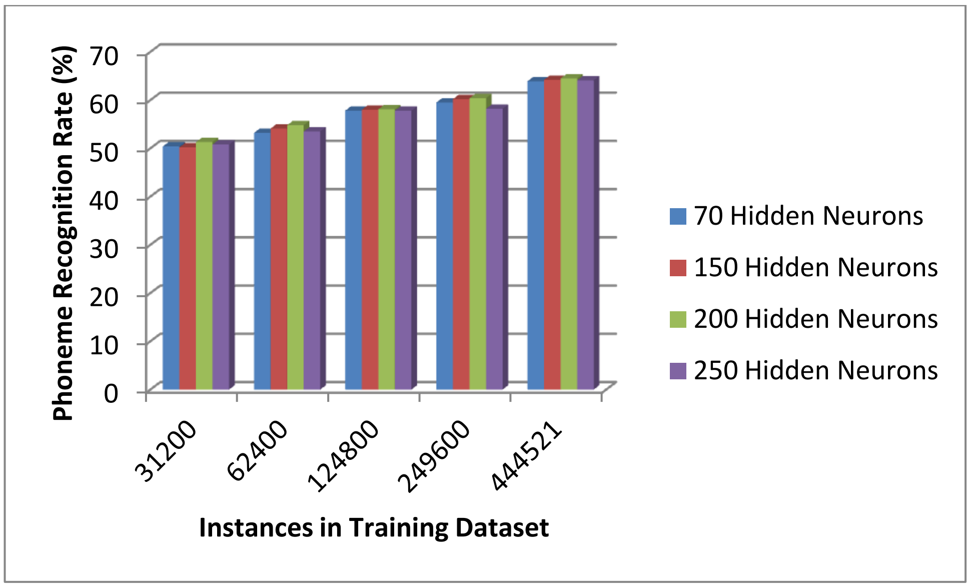


Fig. 4.2 Phoneme Recognition Rate Observed for ANN Model with Varied Number of Neurons in the Hidden Layer

The results were observed and recorded in terms of precision, recall and F-measure and are manifested in Table X. The model turned out with an increasing precision, recall and F-measure values for datasets with increasing samples. The maximum precision, recall and F-measure observed are 0.62, 0.59 and 0.6 respectively for the training dataset of size 444521. The minimal increase observed for the last case as shown in Fig 4.3. implies some sort of stabilization reached by the model. The higher recognition rate of 78% is observed for vowels in ANN with an average recognition rate of 64.5%. The recognition rate of the categories nasals, stops and others are noted to be 69%, 51% and 60% respectively for ANN model. For the model built using the training dataset of size 444521, the PERwas evaluated as 35.5%.

Table X Precision, Recall and F-measure of ANN Model by Varying the Training Set Size

Training Dataset Size	Average Precision	Average Recall	F-measure
31200	0.49	0.48	0.48
62400	0.51	0.49	0.5
124800	0.56	0.54	0.55
249600	0.6	0.57	0.58
444521	0.62	0.59	0.60

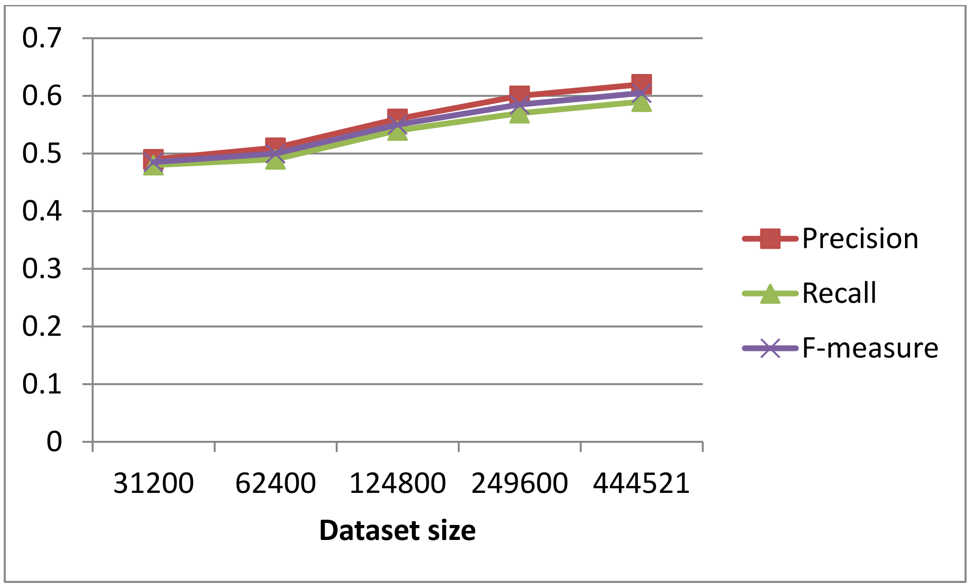


Fig. 4.3 Comparison of Performance metrics with Varying the Dataset Size

Findings

The experiments conducted in this work to build ANN based phoneme recognition model exposed certain facts. As speech is a highly variable data, it requires a considerable number of hidden neurons to capture the patterns of phoneme spoken by different people at different instances. Too less number of neurons in the hidden layer lack the capability in capturing the variability details and too high counts of neurons in hidden layer lacks with the level of generalization, as the networks does not provide probable situation to learn due the availability of more number of neurons to capture varied collection of information.

4.2 TAMIL PHONEME RECOGNITION MODEL USING ANFIS

In the second case, an ANFIS based acoustic model has been built in order to explore the performance of ANFIS on Tamil phoneme recognition problem. Before a formal description on the methodology of ANFIS based phoneme recognition model, Linear Discriminant Analysis – a dimensionality reduction technique, architecture of ANFIS and its learning procedure are elaborated here.

Linear Discriminant Analysis (LDA) is a dimensionality reduction technique and is supervised. Its objective is to project the input feature space into linear subspace in the direction that maximizes the separation between the elements of various classes that helps to avoid overfitting. It also gives an additional benefit of reducing the computational cost as a result of reducing the input dimension. The steps involved in LDA algorithm is summarized below.

Algorithm 4.2 LDA

Step 1: For each of *m* different classes in the dataset *D*, evaluate the *d*-dimensional mean vectors $\mu_1, \mu_2, \ldots\ldots \mu_m$ for the given input matrix *X* of order $n \times d$.

Step 2: Compute the scatter matrices.

Step 3: Compute within class scatter matrix $S_W = \sum_{i=1}^{m} S_i$, where $S_i = \sum_{x \in D_i} (x - \mu_i)(x - \mu i)T$ with $Di \subset D$ holding the samples of class *i*.

Step 4: Compute between class scatter matrix $S_B = \sum_{i=1}^{m} N_i(\mu_i - \mu)(\mu_i - \mu)^T$, where μ is the overall mean, μ_i and N_i are mean and number of samples of each respective class.

Step 5: Solve using general eigen value formulate as $S_W^{-1} S_B e = \lambda e$, to evaluate the eigenvectors e=$(e_1, e_2, \ldots\ldots e_d)$ and their corresponding eigenvalues $\lambda = (\lambda_1, \lambda_2, \ldots\ldots \lambda_d)$ respectively.

Step 6: Arrange the eigen values in decreasing order and select *k* eigenvectors corresponding to the first *k* eigenvalues to form a $d \times k$ order matrix *W*.

Step 7: The transformed feature space is evaluated using $Y = X \times W$

The LDA algorithm assumes the data in the attributes under consideration to be in Gaussian distribution of same variance. With this assumption, it calculates the mean of each attribute and scatter matrices as given in step 1 and 2. The within class scatter matrix defines the separability or variance between different classes and is given by the mean differences of the classes. The between class scatter matrix defines the distance of each instance to the mean of its class. The scatter matrices obtained are used to formulate an eigen value problem, from whose resulting set of eigen vectors and values, few eigen vectors with greater eigen values are selected to form a transformation matrix. The transformation matrix is then applied on the input matrix to build the transformed feature space with fewer dimensions.

The output of LDA is given as input to adaptive neuro-fuzzy inference system to build the acoustic model. It is a combination of adaptive neural networks and fuzzy logic which takes the advantage of soft computing techniques of neural networks and the ability of fuzzy logic to transform human knowledge into quantitative rules [80]. The type-3 ANFIS with an adaptive network is a feed forward network that has the capability to learn itself. The type-3 ANFIS [81] is a five layer feed forward network whose structure with two inputs, one output and nine rules is shown in the Fig. 4.4, where the square node denote the adaptive nodes, that are capable of learning and the circle nodes are fixed nodes. The rules of the network are Takagi and Sugeno type, which takes the form "*If Input1=x_1 and Input2=x_2 Then Output is y=ax_1+bx_2+c*". The consequent part of the rules' generated are built as a linear combination of the inputs and a constant term. The weighted average of the outputs from each rule is given as the final output of the network. The type-3 ANFIS architecture can be explained as follows:

Layer 1: This layer is a fuzzification layer, where the inputs are fuzzified to represent the linguistic terms of the system. The nodes of this layer are adaptive and are represented by some continuous piecewise differentiable functions like trapezoidal, triangular or bell-shaped functions. Each node of this layer is defined with a membership function and is represented as follows,

$$O_i^1 = \mu A_i(x) \tag{4.1}$$

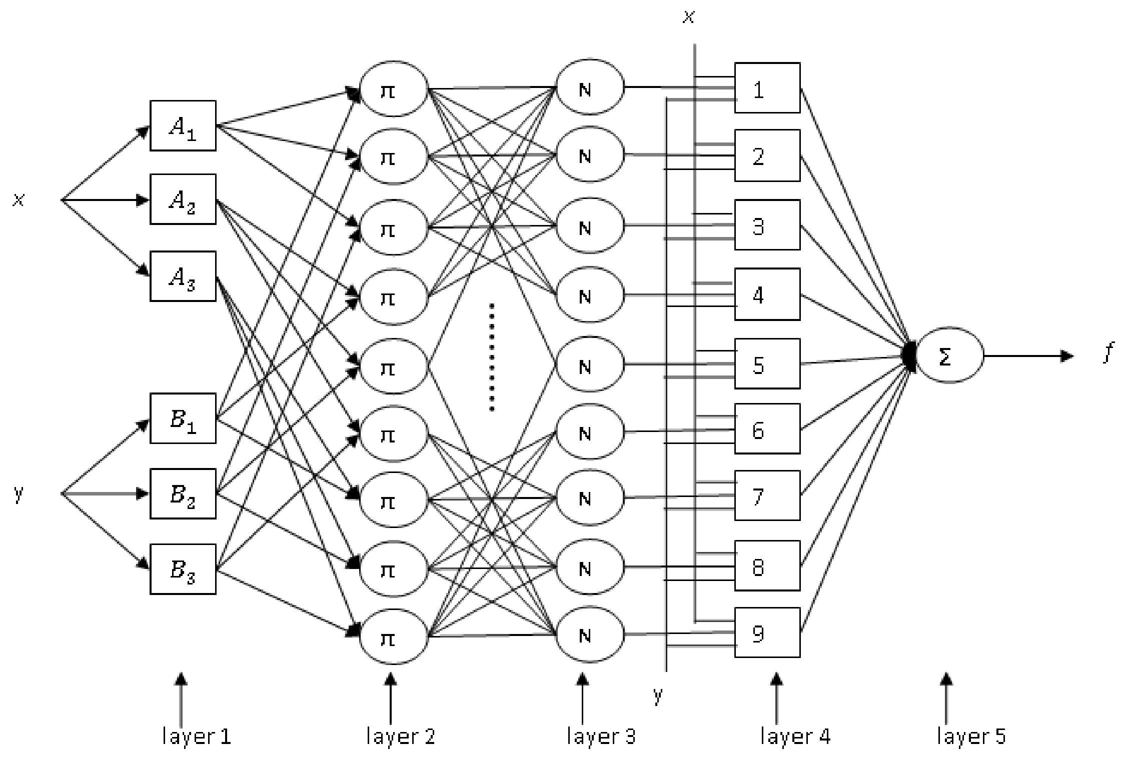


Fig. 4.4 Architecture of Type-3 ANFIS

where x is the input for the node i, and A_i is a function representing the linguistic term associated with the node. For a bell-shaped membership function, the membership values usually lies between 0 and 1 and is given as follows,

$$\mu A_i = \frac{1}{1+\left[\left(\frac{x-c_i}{a_i}\right)^2\right]^{b_i}} \tag{4.2}$$

where the parameter defining the bell-shape a_i, b_i and c_i forms the parameter set. These parameters are thereby referred as premises parameters, as they help defining the premises part of the rule set.

Layer 2: Each node in this layer is a circle node whose function is to produce the product of all incoming signals from the previous layer. The output of the nodes in this layer indicates the firing strength of the rule, which is represented as,

$$w_i = \mu A_i(x) \times \mu B_i(y) \tag{4.3}$$

With $i=1,2$ for two input network, $\mu A_i(x)$ and $\mu B_i(y)$ denoting the membership function for the respective inputs.

Layer 3: This layer represented with nodes labelled N, gives the ratio of the firing strength of the i^{th} rule to the sum of firing strengths of all rules. For the given nine rule network, this ratio is calculated as follows,

$$\overline{w_i} = \frac{w_i}{\sum_{i=1}^{9} w_i} \tag{4.4}$$

Layer 4: Each node in this layer is represented as a combination of the output of layer 3 and linear function of inputs and given by

$$O_i^4 = \overline{w_i}(p_i x + q_i y + r_i) \tag{4.5}$$

where $(\overline{w_i})$ is the output of previous layer and p_i, q_i and r_i are the consequent parameters, the parameters of layer 4.

Layer 5: This layer is the output layer, which provides the summation of all incoming signals and is given by,

$$O_1^5 = \sum_i O_i^4 \tag{4.6}$$

With this structure, the ANFIS model follows a hybrid learning rule, which employs two passes, the forward pass and the backward pass. In the forward pass, the network learns from the functional signals that moves forward through the layers of the network. In the backward pass, the least square error is calculated and is back propagated to each layer through gradient descent.

During the training process, in the backward pass, the error observed in the output layer is back propagated to the previous layers to fine tune the premise parameters $\{a_i, b_i, c_i\}$ that defines the shape of the membership functions [82]. The error at the output layer is calculated as a squared error follows:

$$E_p = \sum_{k=1}^{N_L} (d_k - X_{k,p}^L)^2 \tag{4.7}$$

where d_k is the k^{th} component of the desired output, vector $X_{k,p}^L$ is the k^{th} component of the actual output of ANFIS in forward pass for p^{th} input data vector, $N_L$denotes the number of nodes in layer L. The error starts propagating backward through each and every node of layer 4 of the network as a derivative of square error which is given by,

$$\varepsilon_{L,j} = \frac{\partial E_p}{\partial_{j,p}^L} = -2(d_j - X_{j,p}^L) \tag{4.8}$$

$\forall\, j$ node in layer L. Further, the internal nodes lying in the other layers *(l)* of ANFIS are updated by using the chain rule as follows,

$$\frac{\partial E_p}{\partial X_{l,i}} = \sum_{u=1}^{N_{l+1}} \frac{\partial E_p}{\partial X_{u,p}^{l+1}} \frac{\partial X_{u,p}^{l+1}}{\partial X_{l,i}} \tag{4.9}$$

With the internal node updated with the error as in the above equation, any parameter y of some set of nodes, *S* in the networks can be updated as follows,

$$\frac{\partial E_p}{\partial y} = \sum_{z \in S} \frac{\partial E_p}{\partial z} \frac{\partial z}{\partial y} \tag{4.10}$$

Thus the overall error of parameter y is given as a summation of error contributed by each input data and the gradient descent for the parameter y is given as

$$\Delta y = -\gamma \frac{\partial E}{\partial y} \tag{4.11}$$

with γ being the learning rate.

The methodology of building ANFIS based acoustic model is shown in Fig. 4.5. Initially LDA is applied on the dataset to reduce the dimensionality of the feature space by linearly transforming the features from the original space into a new space with lower dimensionality. The transformed feature space inputs are given to the ANFIS training module where each input variable is fuzzified with three linguistic terms namely low, medium and high and represented by Gaussian functions. The layer 1 of ANFIS model has nodes representing the linguistic units of features in the new space with each linguistic term represented as Gaussian function. The model is then trained with the training dataset using two passes, where the ANFIS learns by propagating the input till the output layer and fine tunes the parameters in the backward pass using gradient based back-propagation technique to propel the error observed in the output layer as discussed earlier in this section. This learning procedure of forward and backward pass is repeated to few hundred epochs to train the network. The training phase is followed by a testing phase with the test dataset to evaluate the performance of the ANFIS phoneme recognition model. The experimental results of training the ANFIS for kazhangiyam phoneme dataset are discussed in the following section.

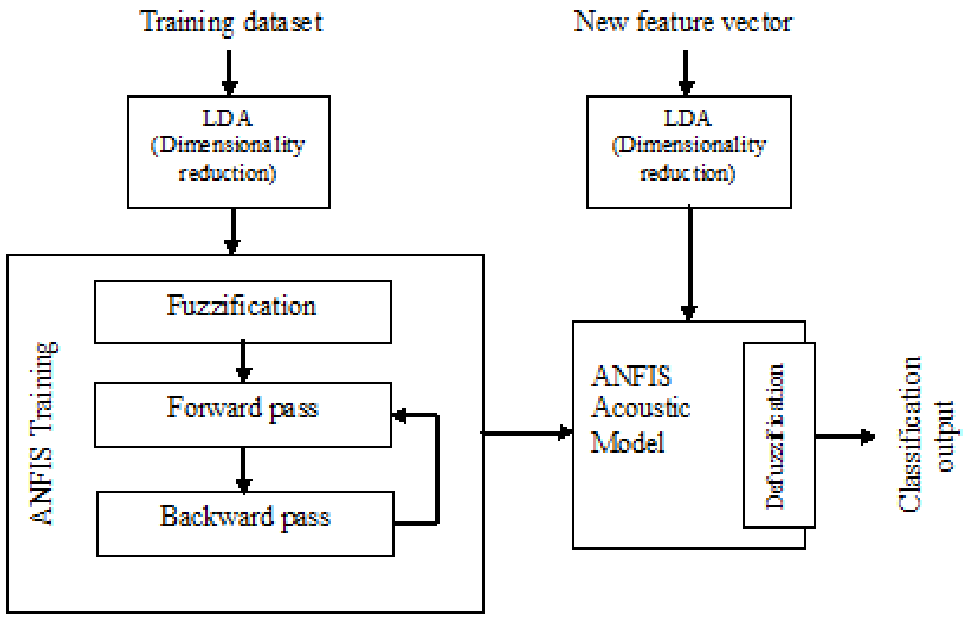


Fig. 4.5 Architecture of ANFIS Based Phoneme Recognition Model

Experiment and Results

ANFIS algorithm is implemented using the DWTFS dataset in kazhangiyam speech corpus for building a Tamil phoneme classification model. The dataset is divided into the training set and testing set where a subset with maximum of 31200 data points spanning all the 39 classes of the dataset with 800 samples in each class has been used for training ANFIS based phoneme recognition model. The model is tested using the test dataset a subset of with maximum of 15600 data points contributed with 400 samples from each class. LDA has been applied on the DWTFS dataset to perform dimensionality reduction. Finally the nine dimensional input vectors are passed to the ANFIS, where each input propagates through one of three nodes dedicated to the respective linguistic terms defined for that input. Thus layer 1 of this ANFIS is designed with 27 nodes, where each set of three nodes defining the three linguistics terms of a given input value. With 27 nodes defined in layer 1, layer 2, 3 and 4 follows with 27 nodes each. With this infrastructure defined, the ANFIS model for Tamil phoneme recognition is trained with 500 epochs. The model built so, is tested further for its efficiency with the test dataset.

The precision, recall and F-measure observed in ANFIS models by increasing the number of samples in training sets are portrayed in Table XI. It is observed that the efficiency of the ANFIS acoustic model improves with the number of samples in the training dataset, but it is observed that the rate of improvement in precision, recall and F-measure is minimal when compared to the rate of increase in number of samples. The average precision, recall and F-measure observed for the training dataset of size 31200 is 0.61, 0.56 and 0.5839 respectively.

Table XI Average Precision, Recall and F-measure Observed for ANFIS Model Built for Phoneme Dataset of Various Sizes

Training Dataset Size	**Average Precision**	**Average Recall**	**F-measure**
7800	**0.34**	**0.45**	**0.3873**
15600	**0.38**	**0.51**	**0.4355**
23400	**0.58**	**0.54**	**0.5592**
31200	**0.61**	**0.56**	**0.5839**

From Table XII, the number of rules generated by the ANFIS model and the execution time is observed to be increasing at greater rates with increase in the number of samples. The number of rules generated is 121, 384, 665 and 956, where as the execution time taken by ANFIS to build the model from the training dataset is observed as 153, 198, 285 and 360 minutes respectively for a dataset of size 7800, 15600, 23400 and 31200 samples when executed with a system with i3 processor. It can also be seen that the precision, recall and F- measures increase rate is diminishing with the greater overheads of knowledge base size i.e. number of rules and execution time for ANFIS model. The greater increase in the number of rules generated in ANFIS as shown in Fig. 4.6 is due to the increase in dataset size by adding more number of samples from different speakers. This shows the high variability in the features representing the phonetic units of the language, reflecting the characteristics of multiple speakers and the effect of co-occurring phonemes because the numbers of samples in the datasets are increased by including samples from more speakers. The time taken is also observed to be high for training ANFIS even for a smaller dataset of size 7800. The study also shows that the increase in the number of samples does not reflect the ANFIS model efficiency much as expected in terms of accuracy rather it increases the time overhead in training and classifying the data points.

Table XII Number of Rules Generated and Time Taken by ANFIS for Various Sizes of Training Dataset

Number of Samples in Training dataset	Number of rules generated	Execution time (in minutes)
7800	121	153
15600	384	198
23400	665	285
31200	956	360

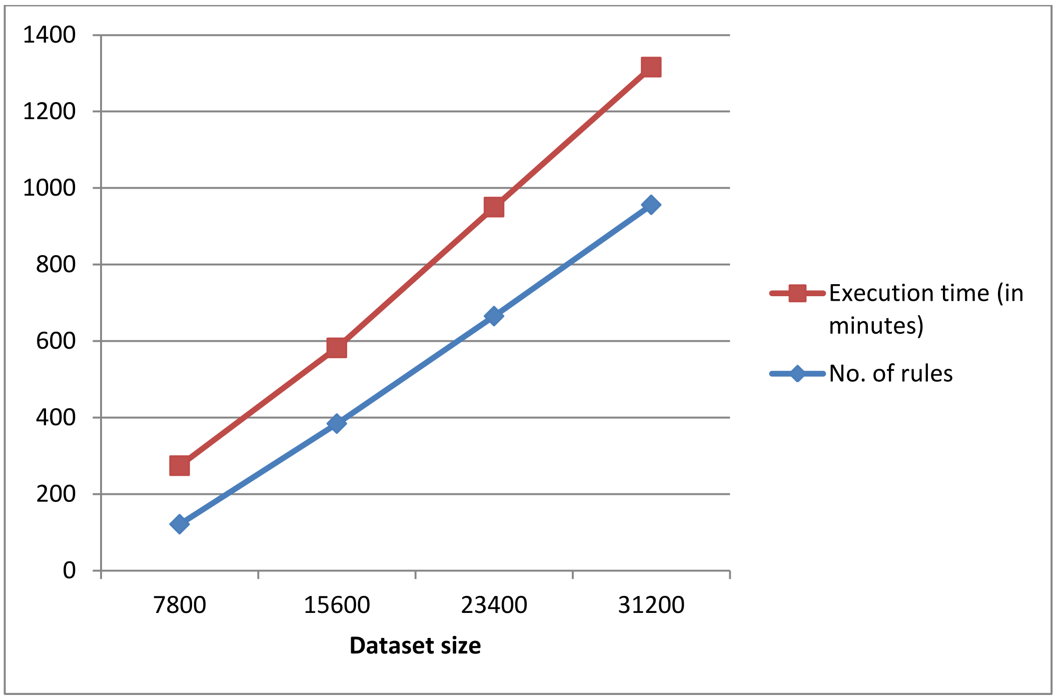


Fig. 4.6 Comparison of Number of Rules Generated and Time Taken by ANFIS with Respect to Dataset Size

The recognition rates for the various categories vowels, nasals, stops and others are observed as 74%, 67%, 48% and 59% respectively with an average accuracy of 62% for the ANFIS model trained with dataset of size 31200. The phoneme error rate of the ANFIS model was evaluated as 38%. The recognition rates for various categories show that the ANFIS model is capable of recognizing the vowels with greater accuracy when compared to the other type of phonemes in Tamil. During the experiments the phonetic units that span for longer duration are observed to be classified correctly when compared to the phonetic units with shorter duration, thus reflecting greater accuracy for vowels compared to the other categories. The results show decent and comparable recognition accuracy to the state of art models in the literature [14, 83].

Performance Comparison of ANN and ANFIS Phoneme Recognition Model

The results of ANN based phoneme recognition model is compared with the results of ANFIS model with various metrics such as overall precision, recall, F-measure and accuracy in Table XIII. The overall average precision observed for ANN and ANFIS is 0.62 and 0.61 respectively, where as the recall is observed as 0.59 and 0.56 respectively. The average F-measure that resulted for the models ANN and ANFIS is 0.6 and 0.5839 respectively. The best accuracy of the two models is observed as 64.5% for ANN when compared to ANFIS's accuracy of 62%.

Table XIII Performance Comparison of ANN and ANFIS Phoneme Recognition Models

Model	**Precision**	**Recall**	**F-measure**	**Accuracy (%)**
ANN	0.62	0.59	0.60	64.5
ANFIS	0.61	0.56	0.5839	62

From the comparative analysis of ANN and ANFIS models, it is proved that the efficiency to classify Tamil phonetic segments is better with ANN model when compared to ANFIS model. Generalizing the variability observed in the phonemes in a precise way is better implemented with ANN when compared to the model built using ANFIS. This is realized with the exponential growth of the knowledge base of ANFIS and in turn complicating the classification task. The recognition rate perceived with ANN is found to outperform ANFIS model for all considered phoneme categories and portrayed in the Fig. 4.7.

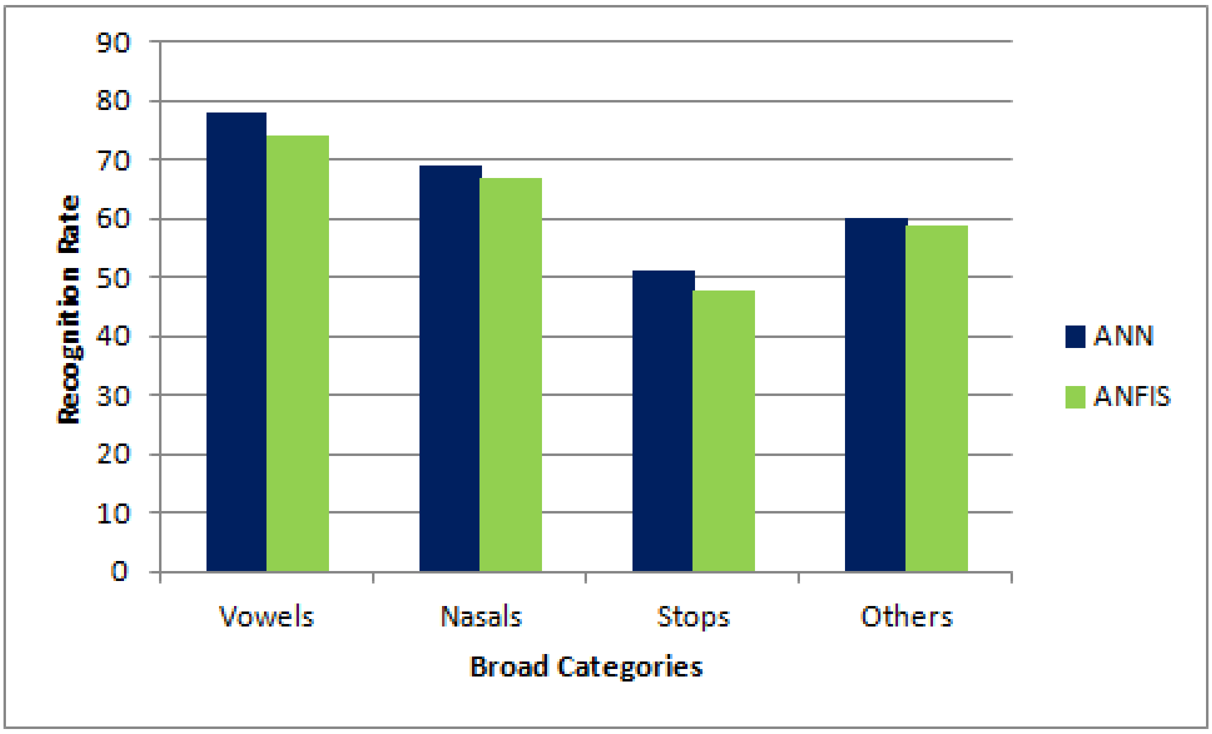


Fig. 4.7 Recognition Rate of ANN and ANFIS Models for Various Categories of Phonemes

Findings

The fact here is, the variability in the characteristics of speakers and co-occurrences of phonemes in continuous speech adds more complexity to the problem. The reflection of the complexity of the problem is seen in the complexity of the ANFIS model built. It is inferred with increase in knowledge base size proportionately to the increase in the training dataset size which is handled better with ANN model elucidated earlier in this chapter. Building an ANFIS model requires less number of instances to train the model which is a back track for the phoneme recognition problem inbuilt with high variability. A simple ANN Tamil acoustic model is observed to be more generalized when compared to ANFIS model, thus providing better classification and proved its efficiency in terms of various performance measures. ANN is found to be capable of capturing more complex and variable features representing the phonetic units of the Tamil language which is observed to lack in ANFIS because of the inability to generalize the phonetic classes.

SUMMARY

This chapter elucidated the process of building artificial neural network and adaptive neuro-fuzzy inference system models for Tamil phoneme classification problem. The methodology and detailed procedures used for training the models were discussed. Details on the experiments conducted and the results observed were explicated. The key findings on the research accomplished during the experiments were summarized. The results of the experiments conducted motivated to apply recent deep learning techniques for building more efficient model for phoneme classification. The model build so will be capable of handling the variability in the feature space and to reduce the time complexity incurred in model building process. The DBN, a deep learning architecture is chosen for further research and the respective models will be presented in the forth coming chapters.

Remarks

- The article titled ‘ANFIS for Tamil Phoneme Classification’ is published in International Journal of Engineering and Advanced Technology, Vol. 8, No. 6, pp. 2907-2914, 2019, ISSN: 2249-8958, Blue Eyes Intelligence Engineering and Science Publication. DOI: 10.35940/ijeat.F8804.088619 (Published-Scopus Indexed)