8. WEIGHTED MEAN SQUARE ERROR LOSS FUNCTION FOR DBN BASED PHONEME RECOGNITION

In the era of big data, a huge set of problems with imbalanced dataset are found. Imbalanced dataset is a dataset where, the number of instandces belonging to each class is highly skewed. The variability of the samples in each class – the class imbalance, strongly influences the accuracy of the model built to represent any problem. When a model of imbalanced dataset is observed, it can be noticed that the accuracy of the model is heavily influenced by the rare classes. Those rare classes with lesser number of samples, contributes more to misclassification error rates. This is due of the fact that most standard measures in the literature treat all classes equally.

Need for Weighted Mean Square Error (WMSE) Loss Function

Machine learning is a powerful learning mechanism which learns to self-build the model from the dataset available. But, one of the main challenges of these learning mechanisms is learning from rarities. Although a wide range of machine learning algorithms are in existence which are capable of self-learning from the datasets, comparably least work can be seen to address the problem of data imbalance. Most of the loss function that is used in the machine learning algorithms treat all classes in the dataset equally. But in most real-world datasets, the classes are imbalanced. Similarly, there are some phoneme classes in Tamil language which appear rare in speech. Thus, contributes fewer samples for those classes. Performing self-induced leaning requires a huge set of data, which will be lacking in rare cases. The multi-class classification problem discussed in this thesis to classify phonemes in continuous Tamil speech uses Kazhangiyam dataset DWTFS, which is highly imbalanced dataset. In our earlier work on phoneme recognition, we have built various DBN models trained using back-propagation to classify the phonemes of Tamil continuous speech. In all the previous training process the standard mean square error was used as loss function in the back-propagation training process.

There are various loss functions in the literature which are used to evaluate the neural networks. Some of them are mean square error, Classification Error Rate (CER), Validation Error, root mean square error, Median Absolute Percentage Error (MdAPE) and more [91,92]. But in case of imbalanced data the contribution of majority class highly influences on the loss function than the minority class. It leads to model accuracy driving towards the majority class, failing to capture the details of the minority class. Hence the weighted mean square error is proposed to handle the effect of imbalanced dataset in building classification models and

described in this chapter. The importance of using WMSE for training and pretraining DBNs are presented.

8.1 TRAINING AND BUILDING DBN MODELS USING PROPOSED WMSE

The problem is formulated to handle imbalanced dataset using a suitable loss function while pre-training and training the deep belief network. The proposed architecture of building CD-DBN with WMSE loss function for phoneme recognition is depicted in Fig. 8.1.

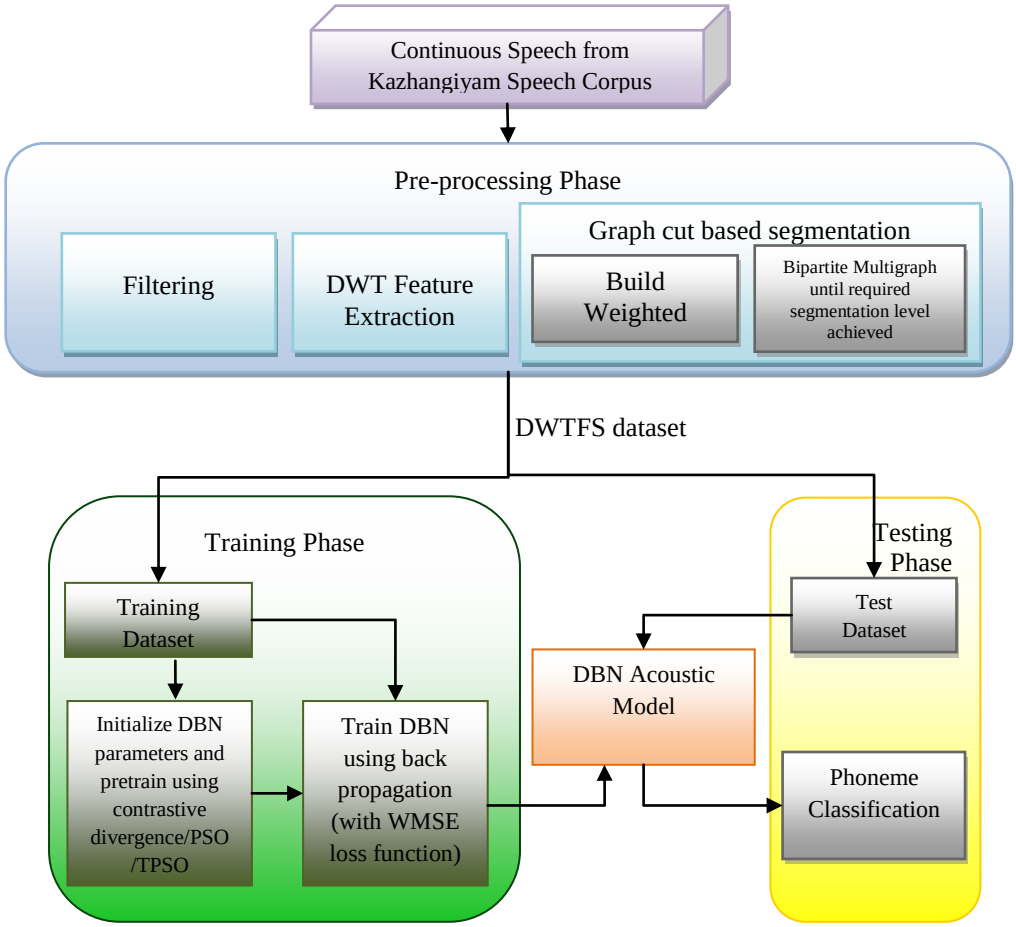

Fig. 8.1 Architecture of DBN Phoneme Recognition Models with WMSE Loss Function

The proposed WMSE loss function to handle imbalanced multi-class dataset follows the standard MSE loss function. This loss function is expressed in terms of a squared difference between the predicted output and the expected output and is given as follows,

$$L = \frac{1}{M}\sum_{i=1}^{M}\sum_{n}\frac{(y_n^{(i)}-o_n^{(i)})^2}{N} \tag{8.1}$$

where M is the number samples in the training dataset, $y_n^{(i)}$ and $o_n^{(i)}$, denotes the predicted and expected output, and N, denotes the number of classes.

The MSE loss function handles the error raised by the samples of all the classes equally. This happens to be acceptable for problems with balanced dataset. But when it comes to the imbalanced dataset, the model being built is already moulded towards the majority classes. The error caused by misclassification of a sample in majority classes is evaluated more accurately in MSE than for the one in minority class. But in the proposed WMSE function, the error raised by the members of different class is handled differently by a class influence term, β_n which is multiplied with the error of each member in n^{th} class. The WMSE loss is given as follows,

$$L = \frac{1}{I}\sum_{i=1}^{I}\sum_{n}\frac{(y_n^{(i)}-o_n^{(i)})^2}{N}\,\beta_n \tag{8.2}$$

where β_n is the ratio between the number of samples in the n^{th} class to the number of samples in the training dataset. The class influence factor β_n is given by

$$\beta_n = \frac{\mathbb{C}_n}{I} \tag{8.3}$$

where $\mathbb{C}_n$ is the number of instances in class *n* of the training dataset. This class relevance is greater for classes with more number of samples in the training dataset than classes with mere samples, thus the classes with greater influence value contributes more to error than the classes with lesser influence.

WMSE loss back-propagation

Supervised learning using back-propagation technique is used for training the DBN. The objective function of this back-propagation training is to minimize the WMSE loss function. Back-propagation technique evaluates the error of the model at the output layer and propagates its deviation from the expected output to the preceding layers in the form of gradients at each neuron and fine-tunes the DBN. The gradients are evaluated as the derivative of the WMSE loss function. The gradients at the output layer *L* is thus given by,

$$\delta_j^{(L)} = \beta_j\, y_j^{(L)}(1-y_j^{(L)})(y_j^{(L)}-o_j) \tag{8.4}$$

Further, the back-propagation of error for the neurons in hidden layers $l = L-1, L-2, \ldots 2$ is given as usual by,

$$\delta_j^{(l)} = y_j^{(l)}(1-y_j^{(l)})\sum_{q=1}^{m} w_{jq}^{(l+1)}\,\delta_q^{(l+1)} \tag{8.5}$$

where, $y_j^{(l)}$ denotes the j^{th} neuron in l^{th} layer, $w_{jq}^{(l(+1)}$ is the connection weight between j^{th} neuron in l^{th} layer and q^{th} neuron in $(l+1)^{th}$ layer. The weights and biases of the DBN layers are learnt with a learning rate γ, updated as before, and are given as follows,

$$\Delta w_{jq}^{(l)} = -\gamma y_j^{(l)} \delta_q^{(l+1)} \tag{8.6}$$

and

$$\Delta b_j^{(l)} = -\gamma \delta_q^{(l+1)} \tag{8.7}$$

The above proposed WMSE loss function is incorporated instead of MSE used with the CD-DBN, PSO-DBN and TPSO-DBN to pretrain and train the acoustic models namely, CD-DBN-WMSE, PSO-DBN-WMSE and TPSO-DBN-WMSE. The first model, CD-DBN-WMSE is pretrained layer by layer using the contrastive divergence technique, in which the output of each RBM acts as the input to the next RBM. The DBN is then fine-tuned by back-propagating the WMSE cost using back-propagation algorithm. In back propagation training, the WMSE cost δ^L is evaluated at the output layer L as given in equation 8.4 and is propagated back to the underlying layers by evaluated the error at each layer δ^l as given in equation 8.5, and updating the connections weights and bias of each layer *l*, by calculating the change in respective weights and biases using equation 8.6 and equation 8.7. The entire methodology is portrayed in algorithm 8.1 takes the training dataset as input and produces a efficient DBN model as output.

Algorithm 8.1 Training CD-DBN using WMSE back-propagation

Step 1: Define the number of layers *L*, number of neurons in each layer η_l of the DBN

Step 2: Initialize the DBN parameters biases, *b* and connection weights, *w* with random initial values

Step 3: Train DBN using contrastive divergence

Step 4: For i = 1 to maxIter

Step 5: Evaluate $\delta_j^{(L)} = \beta_j \, y_j^{(L)}(1 - y_j^{(L)})(y_j^{(L)} - o_j)$

Step 6: Evaluate the gradients of WMSE using equations 8.4 to 8.7 and backpropagate

Step 7: End for

The second model is built using PSO-DBN replaces the contrastive divergence based pre-training with PSO as discussed in chapter 6. The proposed algorithm for building PSO-DBN-WMSE that uses WMSE loss function while pre-training and training is built as follows. The

pre-training phase starts by initialising the position and velocity of the individuals in the population. In each generation of PSO, once the new position and velocity of the individuals are evaluated, the cost of the individuals in the population is estimated using WMSE to keep track of the individual's local best and the global best of the entire population. Following the pre-training phase, the training phase starts by transforming the global best of PSO to DBN and further fine-tunes using WMSE back-propagation.

Algorithm 8.2 Training PSO-DBN using WMSE back-propagation

Step 1: Define the number of layers *L*, number of neurons in each layer η_l of the DBN

Step 2: Define the population *P* of size *s* for PSO based pre-training phase

Step 3: Initialize the position of individuals $x_i(1)$ in the population to represent DBN parameters biases, *b* and connection weights, *w* with random initial values

Step 4: Define the velocity of individuals $v_i(1)$ as zero

Step 5: For *t = 1 to maxGen*

For each individual *i* in the population *P*

a. Calculate the new velocity $v_i(t+1) = \omega v_i(t) + c_1 r_1\big(p_i(t) - x_i(t)\big) + c_2 r_2\left(p_g(t) - x_i(t)\right)$

b. Calculate the new position $x_i(t+1) = x_i(t) + v_i(t+1)$

c. Calculate WMSE loss, $\psi_i = \frac{1}{M}\sum_{i=1}^{M}\sum_n \frac{\left(y_n^{(i)} - o_n^{(i)}\right)^2}{N}\beta_n$

d. Update personal best cost and position (p_i), if $\psi_i < bestcost(p_i)$

e. Update global best cost and position (p_g), if $\psi_i < bestcost(p_g)$

f. Update inertia (ω)

End For

Step 6: End For

Step 7: Build PSO-DBN using the global best individual $\left(p_g\right)$ of PSO

Step 8: For *i = 1 to maxIter*

Step 9: Calculate $\delta_j^{(L)} = \beta_j\, y_j^{(L)}\left(1 - y_j^{(L)}\right)\left(y_j^{(L)} - o_j\right)$

Step 10: Evaluate the gradients of WMSE using equations 8.4 to 8.7 and backpropagate

Step 11: End for

The third model is built using TPSO-DBN with WMSE cost in both pre-training and training phases. TPSO introduced in previous chapter that uses a temperature term to control the velocity of the individuals is used to pretrain the DBN which uses WMSE to evaluate the fitness of the particles in the population during each generation. The TPSO pretraining is then followed by WMSE loss function based back propagation training as elucidated earlier. The process to build TPSO-DBN-WMSE is given in algorithm 8.3. The methodology is similar to PSO-DBN except, a temperature term controls the velocity the individuals.

Algorithm 8.3 Training TPSO-DBN using WMSE back-propagation

Step 1: Define the number of layers *L*, number of neurons in each layer η_l of the DBN

Step 2: Define the population *P* of size *s* for TPSO based pre-training phase

Step 3: Define the temperature function *h*

Step 4: Initialize the position of individuals $x_i(1)$ in the population to represent DBN parameters biases, *b* and connection weights, *w* with random initial values

Step 5: Define the velocity of individuals $v_i(1)$ as zero

Step 6: For *t = 1 to maxGen*

For each individual *i* in the population *P*

a. Calculate the new velocity $\mathrm{v_i(t+1) = \omega v_i(t) + h(t)v_i(t) + c_1 r_1\left(p_i(t) - xit + c2r2pgt - xit\right.}$

b. Calculate the new position $\mathrm{x_i(t+1) = x_i(t) + v_i(t+1)}$

c. Calculate WMSE loss, $\psi_i = \frac{1}{M}\sum_{i=1}^{M}\sum_n \frac{\left(y_n^{(i)} - o_n^{(i)}\right)^2}{N}\beta_n$

d. Update personal best cost and position (p_i), if $\psi_i < \mathrm{bestcost}(p_i)$

e. Update global best cost and position (p_g), if $\psi_i < \mathrm{bestcost}(p_g)$

f. Update inertia (ω)

End For

Step 7: End For

Step 8: Build PSO-DBN using the global best individual (p_g) of PSO

Step 9: For *i = 1 to maxIter*

Step 10: Calculate the error at the output layer using WMSE

Step 11: Evaluate the gradients of WMSE using equations 8.4 to 8.7 and backpropagate

Step 12: End for

8.2 EXPERIMENTS AND RESULTS

The dataset DWTFS of Kazhangiyam speech corpus which is a labelled dataset that was used in various other models discussed in previous chapters is again used here to implement CD-DBN-WMSE, PSO-DBN- WMSE and TPSO-DBN-WMSE phoneme recognition models in Matlab. The influence of using a dynamic proposed WMSE loss function to handle imbalanced dataset is verified. While pretraining the DBNs using PSO or TPSO with WMSE loss function it is observed that while evolving over generations, WMSE yield overall minimal loss when compared to the models built by pretraining with the MSE loss function, even during the pretraining phase. The loss observed for MSE based pretraining is around 0.04 which is shown in Fig. 8.2, where as the loss observed for WMSE based pretraining is around 0.01 which is shown in Fig. 8.3. The improvement in the cost observed while building the PSO-DBN-WMSE or TPSO-DBN-WMSE confirms that the loss observed in pretraining with WMSE is highly influenced by the introduction of class influence factor.

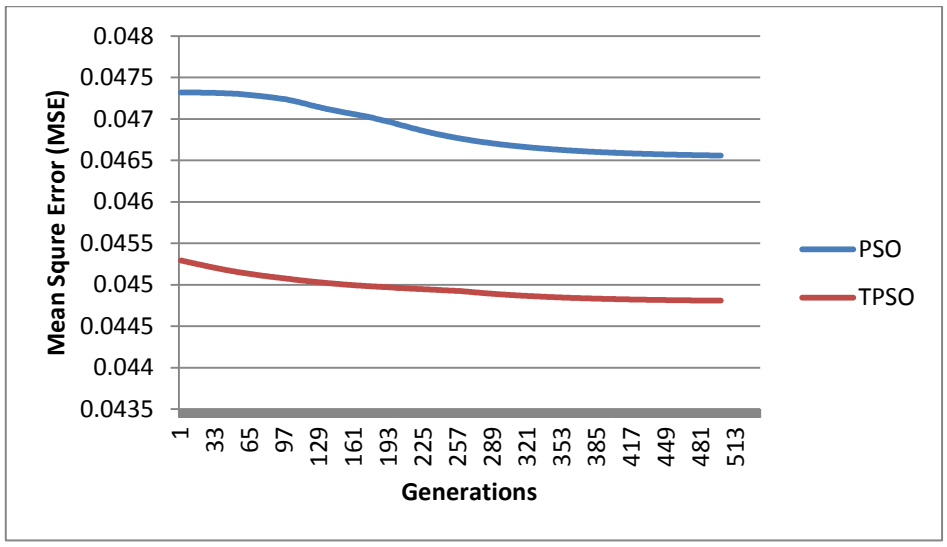

Fig. 8.2 Loss Acquired while Pretraining through PSO and TPSO with MSE

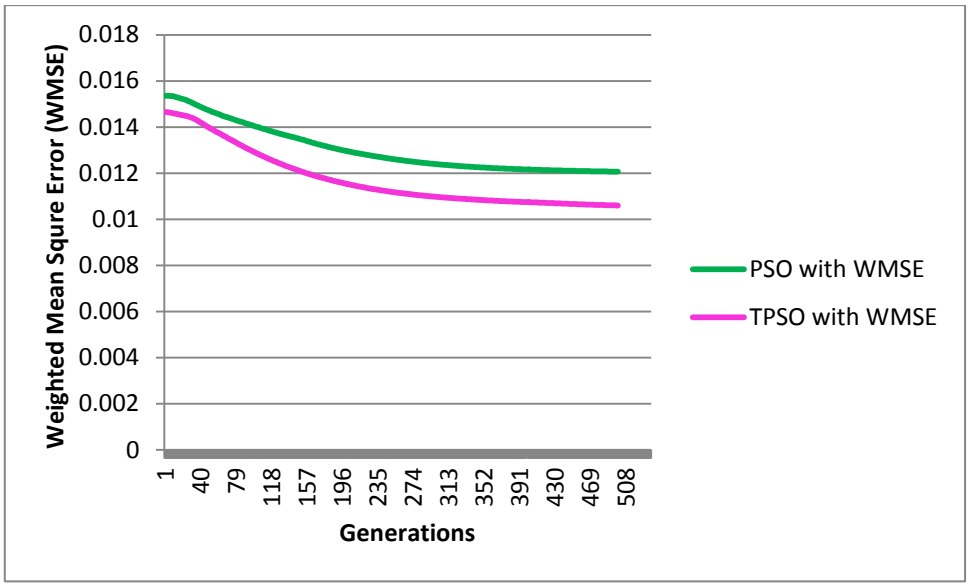

Fig. 8.4 Loss Acquired while Pretraining through PSO and TPSO with WMSE

The experimental results of various methods are analyzed using the standard root mean square error and phoneme error rate for the Kazhangiyam DWTFS dataset, and presented in Table XXVIII. It is observed that the RMSE of the models CD-DBN, PSO-DBN and TPSO-DBN trained using WMSE shows an improvement over the models trained using the MSE cost function. The RMSE values observed when building the DBN models with WMSE are 0.01438, 0.01361 and 0.00815 during training and 0.01488, 0.01496 and 0.0089 during testing for various models considered respectively. An average of 2% reduction in the phoneme error rate for all the DBN models is also noticed. The observed PERs are 11.72%, 10.83% and 7.26% for CD-DBN, PSO-DBN and TPSO-DBN with WMSE cost function and 14.62%, 12.2% and 10.5% for the respective models with MSE cost function for training data. Further, the lowest PER is observed for TPSO-DBN for the test data and is recorded as 8.31%.

Table XXVIII Performance Analysis of Various DBNs Built using MSE and WMSE Cost Function

Method	Training Phase		Testing Phase	
	RMSE	PER (%)	RMSE	PER (%)
CD-DBN-MSE	0.01523	14.62	0.01528	14.97
CD-DBN-WMSE	0.01438	11.72	0.01488	13.33
PSO-DBN-MSE	0.01549	12.2	0.01620	12.98
PSO-DBN-WMSE	0.01361	10.83	0.01496	10.45
TPSO-DBN-MSE	0.0094	10.5	0.0105	10.81
TPSO-DBN-WMSE	0.00815	7.26	0.0089	8.31

The DBN models are further analyzed against the performance measures like precision, recall, F-measure and Accuracy. The observed readings are recorded in the Table XXIX for the models experimented for the dataset DWTFS. The TPSO-DBN-WMSE version is observed to outperform the other models under examination in terms of precision, recall, F-measure and accuracy. The precision, recall and F-measure for TPSO-DBN-WMSE model are 0.8798, 0.8371 and 0.8573 respectively. The maximum F-measure is observed as 0.8573, 0.7681 and 0.4831 for TPSO-DBN-WMSE, PSO-DBN-WMSE and DBN-WMSE models respectively. The recognition accuracies of CD-DBN, PSO-DBN and TPSO-DBN when pertained and trained with WMSE is evaluated to 87.41%, 89.36% and 92.22% respectively which is observed to outperform the respective models built with MSE loss function. The increase in the recognition accuracy for the models is observed as 2.21%, 1.95% and 2.87% respectively.

Table XXIX Comparison of the Various DBN Models in terms of Precision, Recall and F-measure

Model	Precision	Recall	F-measure	Accuracy (%)
CD-DBN-MSE	0.2991	0.2533	0.2743	85.21
PSO-DBN-MSE	0.6877	0.6115	0.6473	87.41
TPSO-DBN-MSE	0.8079	0.7735	0.7903	89.35
CD-DBN-WMSE	0.5112	0.4572	0.4831	87.41
PSO-DBN-WMSE	0.7855	0.7542	0.7681	89.36
TPSO-DBN-WMSE	0.8798	0.8371	0.8573	92.22

The comparison of recognition accuracies of various models trained with MSE and WMSE is portrayed in Fig. 8.4. The accuracies of the WMSE version of CD-DBN, PSO-DBN and TPSO-DBN are found to tower above the respective MSE versions. This shows the influence of using WMSE cost function during DBN training phase. The average improvement in terms of accuracy while using WMSE cost function is found to be around 2%.

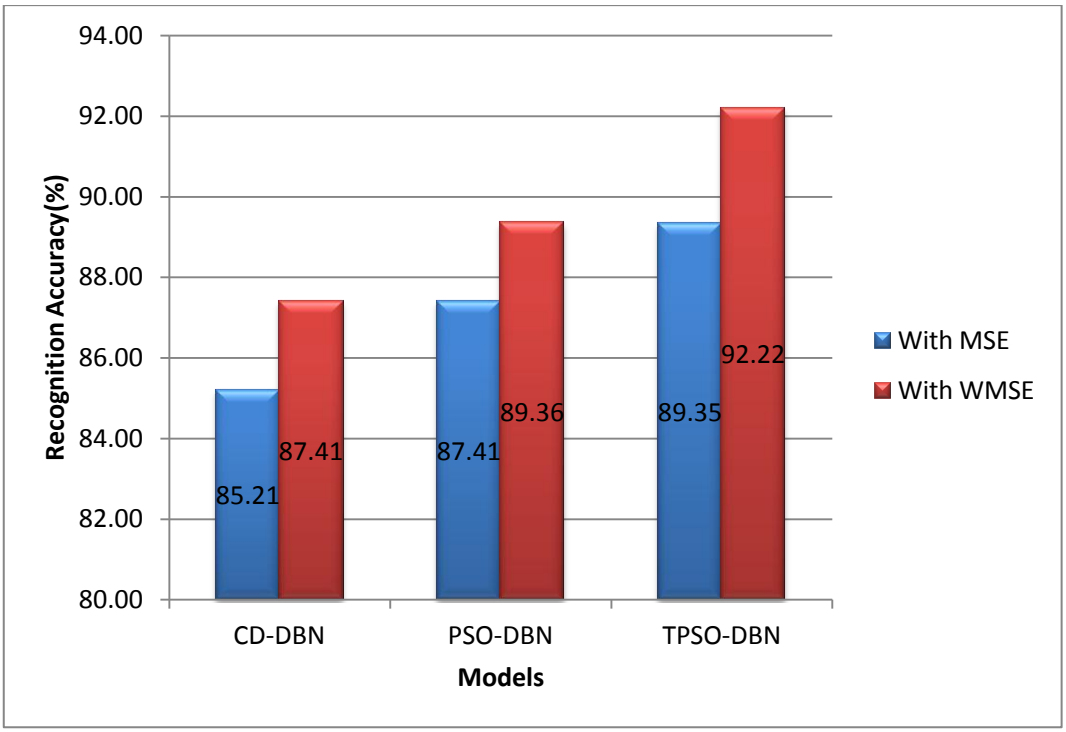

Fig. 8.4 Comparison of Recognition Accuracies of Various Models Built with MSE and WMSE Cost Functions

The results it shows that treating classes with varied number of samples has a greater influence in the model building process. It is inferred from the results that the proposed weighed mean square error as a loss or cost function during the pre-training and training phases has high influence in building better models than using a measure like mean square error that has been designed for dataset with balanced classes. The ROC curves for various minority classes /uu/, /au/, /sh/, /nj/, /zh/ and /j/ in the DWTFS dataset is demonstrated for various DBN models built using MSE or WMSE as loss functions in Fig. 8.5. The ROC curves of most of the models with WMSE as loss function is observed to contribute better AUC when compared to their counterparts with MSE loss function confirming the importance of the class influence term in the WMSE measure. The average AUC evaluated for CD-DBN-WMSE, PSO-DBN-WMSE and TPSO-EDN_WMSE models are 0.7301, 0.8747 and 0.9403 and are observed to be better than its MSE counterparts discussed in previous chapter. The highest average AUC is observed for TPSO-DBN-WMSE and the lowest average AUC observed for CD-DBN-WMSE amongst the three WMSE models experimented in this chapter.

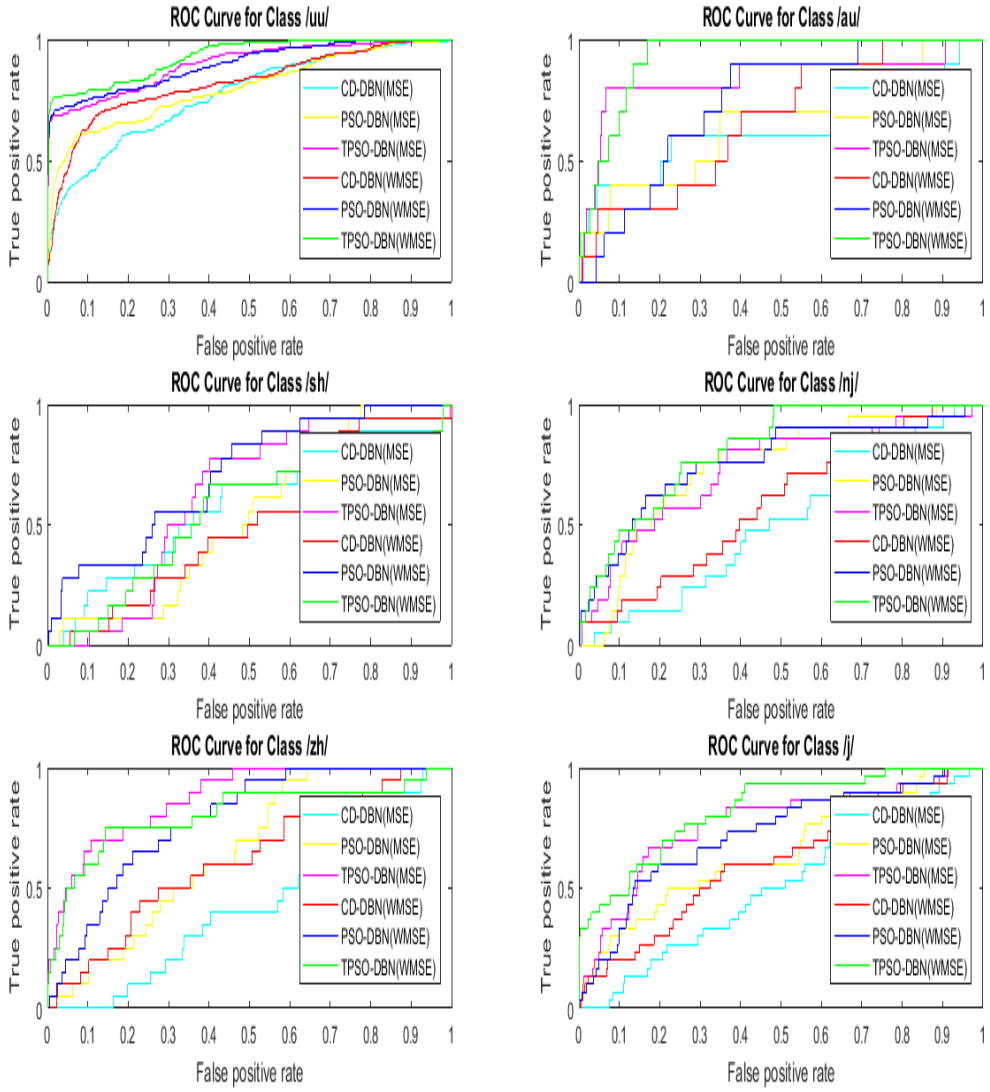

Fig. 8.5 ROC comparison of few minority classes /uu/, /au/, /sh/, /nj/, /zh/, and /j/ in DWTFS dataset for various DBN models

Findings

The use of proposed WMSE loss function which evaluates the loss based on the class influence factor reduces the overall loss as a reflection of minimizing the contribution of false positives to the overall error. This has been evinced with improved recognition accuracy in the models built with WMSE loss function. Applying the dynamic cost function WMSE helps to improve the self-learning capability of the DBNs, thus supporting to build more reliable models. This is evident from the ROCs observed for minority classes and overall AUC values.

SUMMARY

This chapter explained the proposed weighted mean square error loss function to handle the problem incurred with imbalanced data while training the model. The architectures to build CD-DBN, PSO-DBN and TPSO-DBN phoneme recognition models that use WMSE as the cost function were discussed. The results of the various models experimented with DWTFS dataset of Kazhangiyam corpus were analyzed and reported with tables and charts. Finally the findings of the proposed work were summarized.

Remarks

- The article titled 'Weighted Mean Square Error to Train Deep Belief Networks for Imbalanced Data' is published in International Journal of Simulation Systems, Science & Technology, Vol. 19, No.6, 2018, ISSN online: 1473-804x, ISSN print: 1473-8031. United Kingdom Simulation Society. DOI: 10.5013/IJSSST.a.19.06.14 (Published-Scopus Indexed)