ISSN No. 0976-5697

Volume 3, No. 3, May-June 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

On the Design of Classification System for Handwritten Devnagari Numeral with Image as an Input: A Neural Network Approach

Padma R. Bagde*
Departments. of Electronics and Tele-Communication
Sipna's College of Engineering and Technology.
Amravati, (M.S), India
padmabagde@gmail.com

S.A.Ladhake
[2]Sipna's College of Engineering and Technology.
Amravati, (M.S),
India
sladhake@yahoo.co.in

Krushna D. Chinchkhede
[3]Dept. of Applied Electronics,
SGB, Amravati University, Amravati (M.S), India,
krish.chinchkhede@gmail.com

Abstract: This paper demonstrates the use of single hidden layer MLP NN as a classifier for handwritten Marathi Numerals of Devnagari script. In present study, a MLP NN is designed with log sigmoid activation function for both hidden and output layer with neurons in hidden layer varied from 16 to 128 in steps of 16, constitutes 8 configurations of MLP NN trained three times each with memory efficient and fast Scaled Conjugate Gradient (SCG) algorithm. An image (64x64) of handwritten digits act as an input to the network, the training is controlled by early stopping criteria so that optimal network is derived. The intended network is analysed on various performance metric such as mse, best linear fit, correlation coefficient and misclassification rate. The scruples analysis of the result on different data partitions such as training, validation and testing provides best network to be further analysed. Further it is shown that the average classification accuracy for the best network is 97.37%, 89.49%, 90.38% and 95.86% on training, validation, testing and overall dataset respectively. On the basis of confusion matrix, results are elaborated with % misclassification for each output class distributed uniformly within dataset of 4465 samples. Network complexity in terms of weights and bias is 459994 connections from input to output.

Keywords: Handwritten Numerals recognition, MLP, Scaled Conjugate Gradient (SCG) algorithm, best regression fit, Confusion Matrix, log-sigmoid.

I. INTRODUCTION

Pattern recognition is formally defined as the process whereby received patterns are assigned to one of a prescribed number of classes (categories). The goal of pattern-recognition is to build machines, called, classifiers, that will automatically assign measurements to classes. A natural way to make class assignment is to define the decision surface. The decision surface is not trivially determined for many real-world problems. The central problem in pattern recognition is to define the shape and placement of the boundary so that the class-assignment errors are minimized. In classification problem, the task is to assign new inputs to one of a number of discrete classes or categories. Here, the functions that we seek to approximate are the probabilities of membership of the different classes expressed as functions of the input variables.

In classification, we accept a priori that different input data may be generated by different mechanisms and the goal is to separate the data as well as possible into classes. Here, the input data is assumed to be multi-class, and the purpose is to separate them into classes as accurately as possible. The desired response is a set of arbitrary labels (a different integer is normally assigned to each of the classes), so every element of a class will share the same label. Here, the functions that we seek to approximate are the probabilities of membership of the different classes expressed as functions of the input variables. Class assignments are mutually exclusive, so a classifier needs a nonlinear mechanism such as an all-or-nothing switch. A neural network performs pattern recognition by first undergoing a training session, during which the network is repeatedly presented a set of input patterns along with the category to which each particular pattern belongs. Later, a new pattern is presented to the network that has not been seen before, but which belongs to the same population of patterns used to train the network. The network is able to identify the class of that particular pattern because of the information it has extracted from the training data. Pattern recognition performed by a neural network is statistical in nature, with the patterns being represented by points in a multidimensional decision space. The decision space is divided into regions, each one of which is associated with a class.

The decision boundaries are determined by the training process. The construction of theses boundaries is made statistical by the inherent variability that exists within and between classes. It is very difficult to know which training algorithm will be the fastest for a given problem. It depends on many factors, including the complexity of the problem, the number of data points in the training set, the number of weights and biases in the network, the error goal, and whether the network is being used for pattern recognition (discriminate analysis) or function approximation

(regression). Recognition of handwritten numerals is being as a serious and important task in the field of document-image analysis. Another application could be seen in postal services to sort the incoming mail based on recognized handwritten postal zip codes. Banks are looking for automated systems capable of reading and recognizing handwritten checks and/or receipt amounts.

The challenges in handwritten numeral recognition arise not only from the different ways in which a single digit can be written, but also from the varying requirements imposed by the specific applications. The importance of utilizing multiagents in recognizing touching in handwritten numeral strings was addressed in [1]. STEPNET which is neural network classifiers with single layer training was used on data base comprising 8700 isolated digits, and a zip code data base from the U.S. Postal Service comprising 9000 segmented digits [2]. The modelling with manifolds of digitized images of handwritten digits allows a priori information about the structure of the manifolds to be combined with empirical data. Accurate modelling of the manifolds allows digits to be discriminated using the relative probability densities under the alternative models [3].

There are large number of techniques proposed for handwritten numeral recognition that vary in their complexity and efficiency, namely recognizing Handwritten Digits Using Hierarchical Products of Experts, in which main result is that it is possible to achieve discriminative performance comparable with state-of-the-art methods using a system in which nearly all of the work is done by separate density models of each digit class [4]. Representation and Recognition of Handwritten Digits Using Deformable Templates in these two characters are matched by deforming the contour of one, to fit the edge strengths of the other, and a dissimilarity measure is derived from the amount of deformation needed, the goodness of fit of the edges, and the interior overlap between the deformed shapes [5].

Handwritten Digit Recognition by Adaptive-Subspace Self-Organizing Map (ASSOM) is a recent development in self-organizing map (SOM) computation [6]. Automatic Feature Generation for Handwritten Digit Recognition different evaluation measures orthogonality and information, are used to guide the search for features. These features are used in a back propagation trained neural network [7], also provided an exhaustive survey of the feature extraction methods for offline recognition of isolated numerals/characters. The activation function plays an important role in governing the convergence of a network.

The commonly-used activation functions such as sigmoid. Hyperbolic tangent and sigmoid packet functions possess the monotonously increasing characteristics [8], comparable performance results were shown for multi-layer feed forward networks with monotonic and periodical activation functions when applied to handwritten digit recognition. Feature extraction is one of the fundamental problems of character recognition. The performance of character recognition system is largely depending on proper feature extraction and correct classifier selection, Multilayer perceptron was also used for classification for handwritten numeral along with different features[9].Slow Feature Analysis (SFA) is an unsupervised algorithm by extracting the slowly varying features from time series and has been used to pattern recognition successfully[10] , Method for calculating first-order derivative based feature saliency information in a trained neural network and its application to handwritten digit recognition[11]. Another different method proposed by researchers, in which pre-processing takes into account the discriminative properties of Laguerre moments and proposes a novel method to extracts optimal object features [12]. A rapid feature extraction method and Celled Projection (CP) that compute the projection of each section formed through partitioning an image [13].Segmentation it is also a essential part of image processing, a method of recognizing handwritten digits by fitting generative models that are built from deformable B-splines with Gaussian "ink generators" spaced along the length of the spline, can perform recognition driven segmentation [14] [15] [16] [17].

In this paper, a design of classifier for classification of handwritten Marathi Numerals has been investigated using MLP neural network trained with Scaled Conjugate Gradient algorithm on the local database. Neural network designed, for varying number of neurons in hidden and output layer. The paper is organized as follows. First the optimal MLP NN architecture for classification of handwritten Devnagari numeral is discussed. Later, the experimentation with log sigmoid activation functions is carried out with different trials. The best MLP NN based classifier is derived from various trials for the multiclass classification task. Finally the result analysis is carried out on various performance measures such as MSE, R-value (correlation coefficients), and percent classification accuracy for MLP NN with various trials. A 64-bit version of MATLAB® with Neural Network Toolbox running on desktop PC (Intel® Core™2 Duo CPU E8400 @3.00 GHz with 4GB of RAM) is specifically used for obtaining results.

II. MLP NEURAL NETWORK ARCHITECTURE

A. Design of a MLP NN based Classifier:

The multilayer feed-forward neural network can be used for both function fitting and pattern recognition problems. With the addition of a tapped delay line, it can also be used for prediction problems [18][19]. Feed-forward networks often have one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons. Multiple layers of neurons with nonlinear transfer functions allow the network to learn nonlinear relationships between input and output vectors. The linear output layer is most often used for function fitting (or nonlinear regression) problems. On the other hand, if we want to constrain the outputs of a network (such as between 0 and 1), then the output layer should use a sigmoid transfer function. This is the case when the network is used for pattern recognition problems (in which a decision is being made by the network). The configuration of the MLP NN is determined by the number of hidden layers, number of the neurons in each of the hidden layers, as well as the type of the activation functions used for the neurons.

It has been established that an MLP NN that has only one hidden layer, with a sufficient number of neurons, acts as universal approximators of non-linear mappings [20]. Experimentally, it can be verified that the addition of extra hidden layer can enhance the discriminating ability of the NN model. However, it does so at the cost of the added computational complexity. In practice, it is very difficult to determine a sufficient number of neurons necessary to achieve the desired degree of approximation accuracy. To determine the weight values, one must have a set of examples of how the outputs should relate to the inputs. The task of determining the weights from these examples is called training or learning. That is, the weights are estimated from the examples in such a way that the network, according to some metric, models the true relationship as accurately as possible.

a. Training the MLP NN with Scaled Conjugate Gradient (SCG) Method:

The supervised mode of training process requires a set of examples of proper network behavior, network inputs p and target outputs y .The process of training a neural network involves tuning the values of the weights and biases of the network to optimize network performance. The performance function F for feed forward networks is mean square error (mse), the average squared error between the network outputs a and the target outputs y is defined as follows

$$F = mse = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - a_i)^2 \qquad 1$$

There are two different ways in which training can be implemented: incremental mode and batch mode. In incremental mode, the gradient is computed and the weights are updated after each input is applied to the network. In batch mode, all the inputs in the training set are applied to the network before the weights are updated. The batch training is significantly faster and produces smaller errors than incremental training. For training multilayer feed forward networks, any standard numerical optimization algorithm [21] [22] can be used to optimize the performance function, but there are a few key ones that have shown excellent performance for neural network training. These optimization methods use either the gradient of the network performance with respect to the network weights, or the Jacobian of the network errors with respect to the weights. The gradient and the Jacobian are calculated using a technique called the back propagation algorithm, which involves performing computations backward through the network. The back propagation computation is derived using the chain rule of calculus [30]. Let us consider the simplest optimization algorithm gradient descent. It updates the network weights and biases in the direction in which the performance function decreases most rapidly, the negative of the gradient. Iteration of this algorithm can be written as

$$w_{k+1} = w_k - \alpha_k g_k \qquad 2$$

Where w_k a vector of current weights and biases, g_k is the current gradient, and α_k is the learning rate. This equation is iterated until the network converges such that mean square error in equation 1 is below certain threshold.

The fastest training algorithm is generally Levenberg-Marquardt algorithm [23].The quasi-Newton method is also quite fast. But both of these methods tend to be less efficient for large networks (with thousands of weights), since they require more memory and more computation time for these cases. Also, Levenberg-Marquardt performs better on function fitting (nonlinear regression) problems than on pattern recognition problems. When training the pattern recognition networks, Scaled Conjugate Gradient (SCG) and Resilient Back propagation (RB) are good choices. Their memory requirements are relatively small, and yet they are much faster than standard gradient descent algorithms.

Scaled Conjugate Gradient (SCG) [24] is a supervised learning algorithm for feed forward neural networks, and is a member of the class of conjugate gradient methods. SCG is second order method means that, it makes use of the second derivatives of the goal function, while first-order techniques like standard back propagation only use the first derivatives. A second order technique generally finds a better way to a (local) minimum than a first order technique, but at a higher computational cost. Like standard back propagation, CGMs iteratively try to get closer to the minimum. But while standard back propagation always proceeds down the gradient of the error function, a conjugate gradient method will proceed in a direction which is conjugate to the directions of the previous steps. Thus the minimization performed in one step is not partially undone by the next, as it is the case with standard back propagation and other gradient descent methods. SCG has been shown to be considerably faster than standard back propagation and then other CGMs [24].

B. Performance Measures:

a. Mean Square Error (MSE):

The process of training a neural network involves tuning the values of the weights and biases of the network to optimize network performance, as defined by the network performance function specified in equation 1. It measures the network's performance according to the mean of squared errors.

b. Regression Analysis:

The performance of a trained network can be measured to some extent by the errors on the training, validation, and test sets, but it is often useful to investigate the network response in more detail. One option is to perform a regression analysis between the network response a and the corresponding targets y . In regression analysis, the equation of 'best linear fit' is estimated in least squares sense

Best Linear Fit:

$$a = my + c \qquad 3$$

where m and c corresponds to the slope and the y-intercept of the best linear regression relating targets (y) to network outputs (a). If there were a perfect fit (outputs exactly equal to targets), the slope would be 1, and the y-intercept would be 0.

c. Correlation Coefficient (R-value):

The mean square error (MSE) can be used to determine how well the network output fits the desired output (targets), but it doesn't necessarily reflect whether the two sets of data move in the same direction. The correlation coefficient (R-value) between the network outputs and targets is a measure of how well the variation in the output is explained by the targets. If this number is equal to 1, then there is perfect correlation between targets and outputs. R-value close to 1 indicates a good fit. By definition, the correlation coefficient between a network output a and a desired output or target y is:

$$R = \frac{\frac{\sum_i (a_i - \bar{a})(y_i - \bar{y})}{N}}{\sqrt{\frac{\sum_i (a_i - \bar{a})^2}{N}} \sqrt{\frac{\sum_i (y_i - \bar{y})^2}{N}}} \qquad 4$$

Where $\bar{a} = \frac{1}{N}\sum_{i=1}^{N} a_i$ and $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$ The correlation coefficient is confined to the range $[1,-1]$. When $r = 1$ there is a perfect positive linear correlation between a and y that is, they co-vary, which means that they vary by the same amount. When $r = -1$, there is a perfectly linear negative correlation between a and y, that is, they vary in opposite ways. When $r = 0$ there is no correlation between a and y, i.e. the variables are called uncorrelated. Intermediate values describe partial correlations.

d. Confusion Matrices:

A confusion matrix is a simple methodology for displaying the classification results of a network. The confusion matrix is defined by labeling the desired classification on the rows and the predicted classifications on the columns. For each exemplar, a 1 is added to the cell entry defined by (desired classification, predicted classification). Since we want the predicted classification to be the same as the desired classification, the ideal situation is to have all the exemplars end up on the diagonal cells of the matrix (the diagonal that connects the upper-left corner to the lower right). Thus for 100% classification, confusion matrix is $[M \times M]$ diagonal matrix with M output classes.

III. MLP NN AS A CLASSIFIER

A. Dataset Used in the Experimental Setup:

The handwritten numeral dataset is derived from images obtained by scanning the handwritten Numerals. Each individual subject was asked to write a set of 10 numerals on A4 sized paper with blue or black pen 10 to 12 times. Each page digitized into RGB bitmap image using Astra 3400 scanner with the resolution of 400 dpi. Each scanned image is further processed to extract individual Numerals and stored in local database. Thus the dataset formed, consist of 4500 images of Numerals of variable size distributed uniformly among 10 classes, where each numeral (class) is repeated 450 times. Figure1 shows the typical digit set with 10 classes. The raw data (4465 out of 4500 images were selected, one sample of digit "Eight" and 34 samples of digit "Nine" are discarded) is stored as a multidimensional cell array in secondary storage, act as a local database.

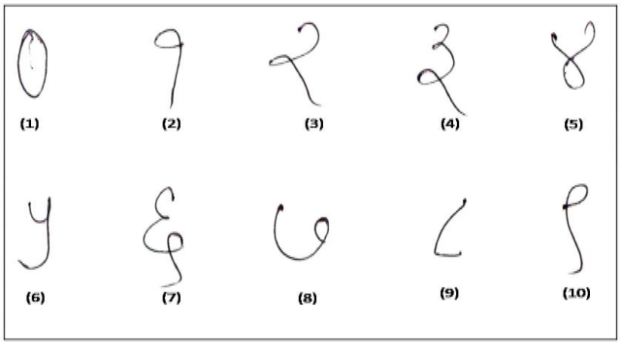


Figure: 1 The typical handwritten digit set showing 10 classes.

B. Pre-processing:

Morphology is a broad set of image processing operations that process images based on shapes. Morphological operations apply a structuring element to an input image, creating an output image of the same size. In a morphological operation, the value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its neighbors'. A set of pixels in a binary image that form a connected group is called an object or a connected component [25]. By choosing the size and shape of the neighborhood, one can construct a morphological operation that is sensitive to specific shapes in the input image. All 4465 images are converted into binary images; from binary image all connected components (objects) that have fewer than pixels are removed by morphological operation. Extra columns and rows filled with zeros are removed by automated process. Finally all images are resized to 64x64 pixels each are now ready for further processing Thus for each 64x64 image, we have array of 4096 features (observations). The complete database of 4465 images is represented as matrix of 4096x4465 constitutes an input to the neural network. The target is represented as matrix of 10x4465 constitutes the desired output of neural network. Figure 2 and 3 shows the block diagram of proposed system used during training and testing phase.

C. Dataset Partitions:

In order improve the generalization during training of multilayer networks; the general practice is to first divide the data into three subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases. The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error normally decreases during the initial phase of training, as does the training set error. However, when the network begins to over-fit the data, the error on the validation set typically begins to rise.

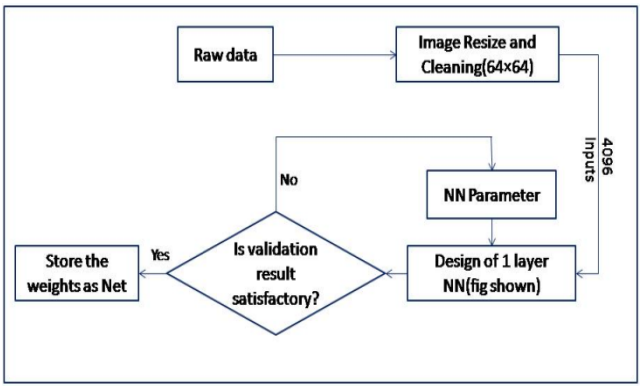


Figure: 2 Training Phase of handwritten digit classifier

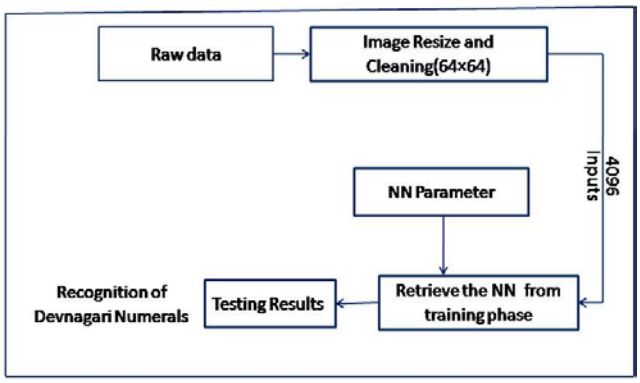


Figure: 3 Testing Phase of handwritten digit classifier

The network weights and biases are saved at the minimum of the validation set error. The test set error is not used during training, but it is used to compare different models. It is also useful to plot the test set error during the training process. If the error on the test set reaches a minimum at a significantly different iteration number than the validation set error, this might indicate a poor division of the data set. It is generally difficult to incorporate prior knowledge into a neural network; therefore the network can only be as accurate as the data that are used to train the network. It is important that the data cover the range of inputs for which the network will be used. Multilayer networks can be trained to generalize well within the range of inputs for which they have been trained. However, they do not have the ability to accurately extrapolate beyond this range, so it is important that the training data span the full range of the input space. Hence dataset (4096x4465) is divided randomly into three subsets; Table 1 shows the data partition scheme for three different trials T1 to T3.

Table: 1Data Partition Scheme (Input: Output)

	Training Set	Validation set	Testing set
Trial 1:T1	4096x3571:10x3571	4096x447:10x447	4096x447:10x447
Trial 2:T2	4096x3571:10x3571	4096x447:10x447	4096x447:10x447
Trial 3:T3	4096x3571:10x3571	4096x447:10x447	4096x447:10x447

The training set is 80%, validation and test sets are 10% each randomly selected from 4465 exemplars, so as to increase the robustness of proposed MLP neural network. Table II shows various parameters for MLP NN classifier used in the simulations.

Table: 2 Various parameters for MLP NN classifier learning with SCG algorithm for Log-sigmoid Transfer Function

Sr. No.	Parameter	Typical Range
1	Number of hidden layer	One
2	Number of neurons in hidden layer	16:16:128
3	Transfer function of hidden layer	Log-sigmoid transfer function
4	Transfer function in output layer	Log-sigmoid transfer function
5	Maximum Epochs	1000
6	Minimum Gradient	1.00E-06
7	Maximum Validation Checks	101
8	Learning Rule	Scaled conjugate gradient backpropagation
9	Sigma*	5.00E-05
10	Lambda**	5.00E-07

* Determine change in weight for second derivative approximation

** Parameter for regulating the indefiniteness of the Hessian

IV. EXPERIMENTAL DETERMINATION OF NN

A. MLP NN With Log Sigmoid Transfer Functions:

MLP NN architecture as mentioned in figure. 4 shows log sigmoid neurons used in both hidden and output layer. As apparent from the network, it has 4096 inputs connected to predefined neurons in hidden layer and 10 neurons in the output layer. Neurons in hidden layer are varied from 16 to 128 in step of 16.

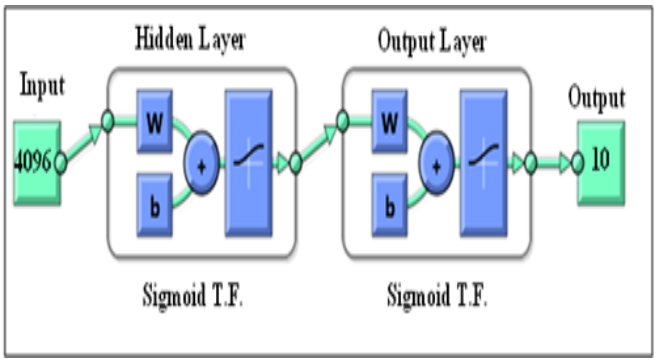


Figure: 4 MLP Neural Network (log-sigmoid transfer. function in hidden and output layer).

The error on the validation set is monitored during the training process. When the validation error increases for a specified number of iterations, the training is stopped, and the weights and biases at the minimum of the validation error are saved as a best network. The above mentioned NN network is trained for three different trials with random initialization each time. The process of training is closely monitored over crucial performance measures in order to judge the optimality of the classifier. Figure 4 and Figure 5 depicts the performance measure (*mse*) and classification

accuracy respectively for trial T1, similar results are obtained for trials T2 and T3.

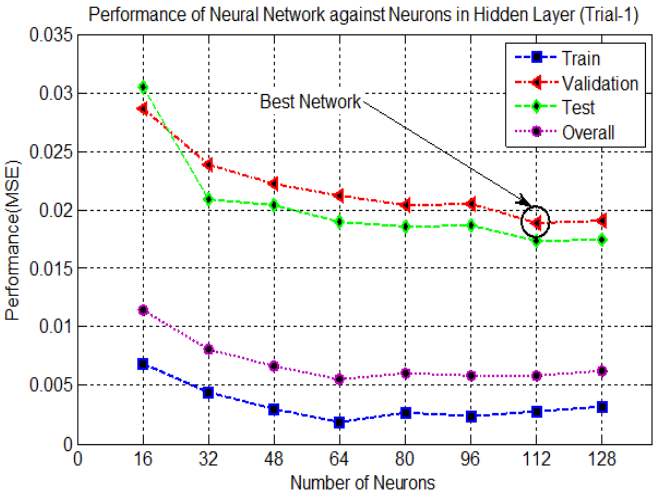


Figure: 5 Performances of NN vs. number of neurons during trial T1.

Table III depicts the performance measure (mse) for the NN on account of various trials T1 to T3 for 1000 epochs. From table 3, it is evident that none of the trial last up to 1000 epochs because of early stopping criterion. The *best* network is obtained during trial T1 with 112 neurons in hidden layer is depicted in table III. The performance metric of *best* network with 112 neurons in hidden layer is shown in figure 6; the *best* network on account of minimum validation error of 0.018837 is discovered at epoch 105.

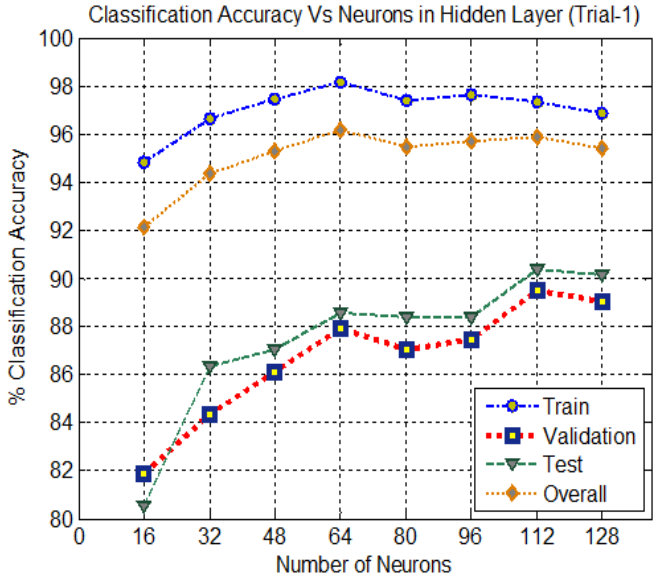


Figure: 6 % classification accuracy vs. number of neurons during trial T1.

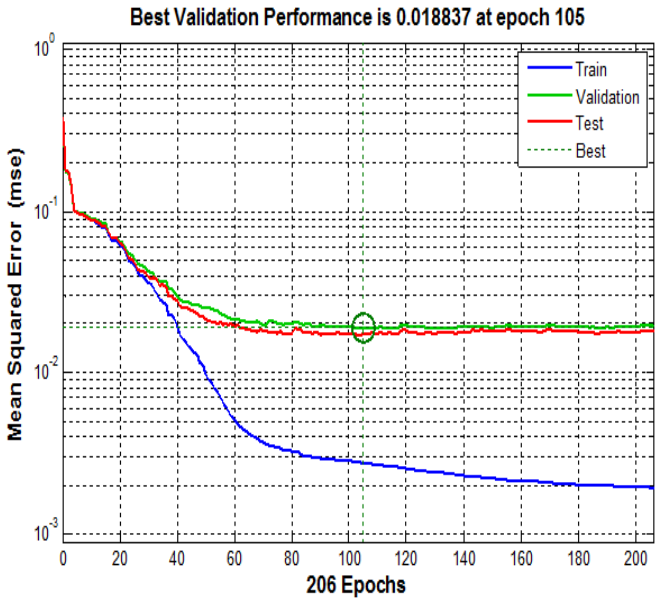


Figure: 7 Performance of network with 112 neuron

Training continued further until epoch 206, where it is stopped due to early stopping criterion on validation stop. The complete training state of the network is shown in figure 7 conforms the logic behind the validation stop, since the error on validation is increasing from epoch 105 onward while error on training is decreasing continuously indicating the over-fitting of dataset. Figure 7 also shows the graph of gradient versus epoch during the training of *best* network, which is also the criterion for early stopping.

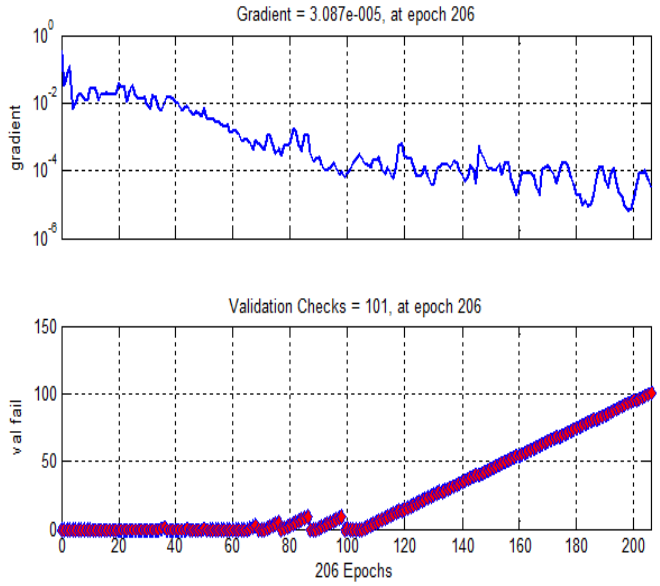
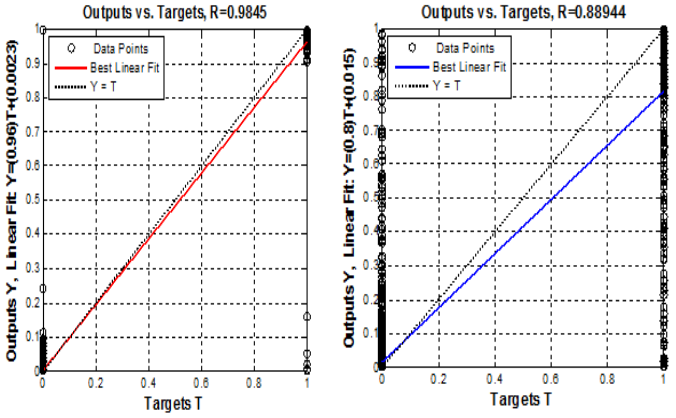


Figure: 8 Training state of network with 112 neurons

Table: 3 Overall performance of the classifier for different trials (columns header 1, 2, 3, 4 and 5 represents slope, y-intercept, r-value, mse and % overall accuracy on training, validation, testing and overall datasets respectively

TN	NN	Epochs		Train					Validation					Test					Total				
		Best	Total	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
T1	16	91	192	0.889	0.0113	0.962	0.0069	94.85	0.709	0.0256	0.826	0.0286	81.88	0.689	0.0260	0.814	0.0305	80.54	0.851	0.0142	0.935	0.0114	92.12
T2	16	59	160	0.858	0.0149	0.954	0.0084	94.57	0.685	0.0307	0.820	0.0295	79.42	0.693	0.0272	0.835	0.0273	81.88	0.824	0.0177	0.929	0.0124	91.78
T3	16	83	184	0.911	0.0076	0.968	0.0057	95.41	0.726	0.0238	0.834	0.0276	84.12	0.722	0.0216	0.838	0.0268	82.77	0.873	0.0106	0.943	0.0100	93.01
T1	32	55	156	0.928	0.0070	0.975	0.0045	96.64	0.757	0.0196	0.858	0.0239	84.34	0.769	0.0197	0.876	0.0209	86.35	0.895	0.0095	0.954	0.0081	94.38
T2	32	90	191	0.947	0.0041	0.982	0.0033	97.00	0.752	0.0182	0.858	0.0239	85.01	0.769	0.0180	0.869	0.0221	86.13	0.910	0.0069	0.959	0.0072	94.71
T3	32	54	155	0.912	0.0081	0.973	0.0049	96.25	0.741	0.0209	0.860	0.0234	85.91	0.747	0.0193	0.870	0.0220	87.02	0.878	0.0105	0.952	0.0085	94.29
T1	48	115	216	0.952	0.0038	0.983	0.0030	97.48	0.762	0.0154	0.868	0.0222	86.13	0.777	0.0148	0.880	0.0204	87.02	0.915	0.0061	0.962	0.0067	95.30
T2	48	105	191	0.969	0.0012	0.986	0.0024	97.59	0.789	0.0136	0.876	0.0210	86.35	0.793	0.0121	0.886	0.0194	88.59	0.933	0.0035	0.966	0.0060	95.57
T3	48	86	187	0.945	0.0038	0.980	0.0036	96.67	0.764	0.0163	0.867	0.0224	85.68	0.765	0.0156	0.875	0.0212	87.02	0.909	0.0062	0.959	0.0072	94.60
T1	64	158	191	0.978	0.0006	0.990	0.0019	98.15	0.793	0.0123	0.875	0.0212	87.92	0.811	0.0119	0.889	0.0190	88.59	0.943	0.0029	0.969	0.0055	96.17
T2	64	118	219	0.966	0.0017	0.987	0.0024	97.65	0.783	0.0134	0.882	0.0200	88.14	0.788	0.0136	0.886	0.0193	89.04	0.930	0.0041	0.967	0.0058	95.83
T3	64	162	195	0.977	0.0016	0.992	0.0014	98.66	0.787	0.0136	0.877	0.0209	87.47	0.789	0.0125	0.889	0.0190	88.81	0.939	0.0039	0.971	0.0051	96.55
T1	80	164	265	0.964	0.0016	0.985	0.0027	97.40	0.793	0.0138	0.880	0.0204	87.02	0.797	0.0125	0.891	0.0185	88.37	0.931	0.0040	0.966	0.0060	95.45
T2	80	108	209	0.962	0.0017	0.984	0.0028	97.26	0.777	0.0130	0.877	0.0209	85.91	0.784	0.0113	0.895	0.0181	89.49	0.925	0.0038	0.965	0.0062	95.34
T3	80	109	210	0.965	0.0019	0.987	0.0024	97.70	0.784	0.0149	0.875	0.0212	87.25	0.796	0.0126	0.895	0.0180	89.49	0.930	0.0043	0.967	0.0059	95.83
T1	96	172	255	0.974	0.0004	0.987	0.0024	97.62	0.801	0.0112	0.880	0.0205	87.47	0.806	0.0109	0.891	0.0186	88.37	0.940	0.0025	0.967	0.0058	95.68
T2	96	245	343	0.977	0.0006	0.989	0.0020	98.04	0.812	0.0100	0.891	0.0187	88.37	0.800	0.0092	0.892	0.0184	89.26	0.942	0.0024	0.970	0.0053	96.19
T3	96	425	425	0.982	0.0002	0.990	0.0017	98.29	0.808	0.0098	0.885	0.0196	88.81	0.819	0.0089	0.898	0.0175	89.71	0.948	0.0020	0.971	0.0051	96.48
T1	**112**	**105**	**206**	**0.963**	**0.0023**	**0.985**	**0.0028**	**97.37**	**0.802**	**0.0149**	**0.889**	**0.0188**	**89.49**	**0.808**	**0.0144**	**0.899**	**0.0173**	**90.38**	**0.932**	**0.0048**	**0.967**	**0.0058**	**95.88**
T2	112	223	223	0.979	0.0002	0.989	0.0019	98.07	0.802	0.0100	0.881	0.0204	89.26	0.796	0.0093	0.888	0.0191	89.71	0.943	0.0021	0.969	0.0055	96.35
T3	112	311	314	0.983	0.0003	0.991	0.0016	98.43	0.799	0.0099	0.884	0.0199	87.47	0.804	0.0094	0.893	0.0184	88.37	0.946	0.0022	0.971	0.0051	96.33
T1	128	150	251	0.960	0.0017	0.982	0.0032	96.86	0.795	0.0130	0.888	0.0191	89.04	0.809	0.0131	0.898	0.0175	90.16	0.929	0.0039	0.965	0.0062	95.41
T2	128	202	252	0.972	0.0006	0.986	0.0025	97.54	0.797	0.0101	0.884	0.0198	89.26	0.813	0.0109	0.896	0.0178	89.49	0.939	0.0025	0.968	0.0057	95.90
T3	128	211	249	0.975	0.0004	0.987	0.0023	97.76	0.801	0.0113	0.878	0.0208	88.81	0.805	0.0086	0.896	0.0179	89.93	0.941	0.0023	0.968	0.0057	96.08

The regression analysis of 'best linear fit' is estimated in least squares sense, which provides slope and y-intercept for 'best linear fit' along with correlation coefficient (R-value) are also depicted in table III for each trial. Figure 8 shows regression line 'best linear fit' for training, validation, testing and overall data set during trial T1 with 112 neurons in hidden layer. If we see the regression line corresponding to training set indicates that the neural network has been generalized well on training dataset with exception of few observations



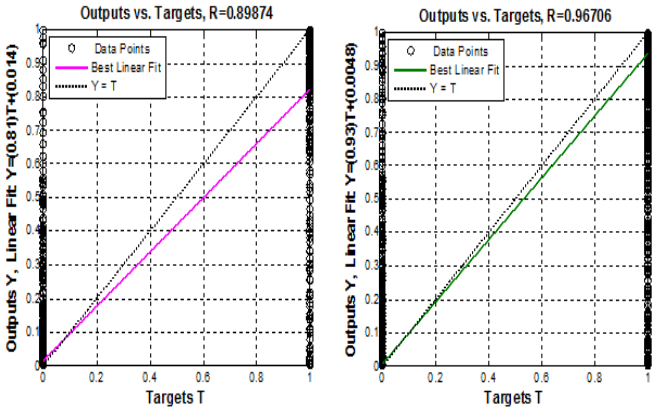


Figure 9 Regression fit (best linear fit) for (a) training, (b) validation, (c) testing and (d) Total dataset

The slope of best fit is 0.963 along with correlation coefficient (R-value) of 0.985 confirms the optimal training. Similar analysis on validation and test dataset is carried out indicates slope value of 0.802 and 0.808 and correlation coefficient of 0.88944 and 0.89874, these values are some extent not close to unity indicates the network in-ability to generalize on unseen data. This provides an intuition to modify the configuration of network or reduce the dimensionality of data.

V. RESULT ANALYSIS AND DISCUSSION

As discussed in previous section, the best network is fixed for 112 neurons in hidden layer. The best network is retrieved as a structure from local storage, it's complexity in terms of network weight is 459994 connections from input to output.

$$\text{Network complexity} = \left(No.Inputs \times No.Neurons + No.Neurons \times No.Classes\right) + b \times \left(No.Neurons + No.Classes\right)$$

Where b is bias which is treated as single node ($b = 1$).

$$\text{Network complexity} = \left(4096 \times 112 + 112 \times 10\right) + b \times \left(112 + 10\right) = 459994$$

As a classifier the final output is always percentage error on misclassification on individual output classes (digits), the confusion matrix as mentioned in Figure 9 portray the % misclassification rate.

	0	1	2	3	4	5	6	7	8	9
0	364	0	0	0	0	2	1	0	1	1
1	0	338	1	11	0	1	5	1	1	0
2	0	0	363	1	0	0	3	0	0	0
3	0	0	0	350	0	0	0	0	0	0
4	0	0	0	1	354	0	1	0	0	0
5	0	0	4	1	0	328	18	2	2	2
6	1	0	1	0	2	0	348	7	0	0
7	0	0	0	1	0	0	0	359	1	1
8	0	1	0	0	0	0	0	0	354	3
9	0	0	2	0	3	0	6	1	4	319

	0	1	2	3	4	5	6	7	8	9
0	35	0	0	0	0	0	0	0	0	0
1	0	38	1	1	0	1	2	0	0	0
2	0	0	36	4	0	0	2	0	0	0
3	0	0	2	44	1	0	0	0	0	0
4	0	0	2	0	48	0	2	0	0	0
5	0	0	0	0	2	48	4	0	0	0
6	0	0	0	0	1	1	37	4	0	1
7	1	1	0	0	0	0	1	37	0	0
8	0	0	0	0	0	0	0	1	40	5
9	0	0	0	3	1	0	2	0	1	37

	0	1	2	3	4	5	6	7	8	9
0	46	0	0	0	0	0	0	0	0	0
1	0	38	4	3	0	3	1	0	0	0
2	0	0	40	0	1	0	0	0	0	0
3	0	0	2	45	2	0	3	1	0	0
4	0	1	0	0	40	1	0	0	0	0
5	0	0	2	2	1	32	1	1	0	0
6	0	0	0	0	0	1	44	2	0	0
7	0	0	0	0	0	0	2	45	1	0
8	0	0	0	1	0	0	0	1	42	1
9	1	0	1	0	0	0	3	0	0	32

	0	1	2	3	4	5	6	7	8	9
0	445	0	0	0	0	2	1	0	1	1
1	0	414	6	15	0	5	8	1	1	0
2	0	0	439	5	1	0	5	0	0	0
3	0	0	4	439	3	0	3	1	0	0
4	0	1	2	1	442	1	3	0	0	0
5	0	0	6	3	3	408	23	3	2	2
6	1	0	1	0	3	2	429	13	0	1
7	1	1	0	1	0	0	3	441	2	1
8	0	1	0	1	0	0	0	2	436	9
9	1	0	3	3	4	0	11	1	5	388

Figure 10 Confusion Matrix for (a) training, (b) validation, (c) testing and (d) Total dataset

Finally, the consolidate result of misclassification is calculated from confusion matrix, Table IV given below shows % misclassification among the individual classes.

Table: 4 Overall performance of the classifier with 112 neurons

Class ID	Training Set				Validation Set				Testing Set				Total DataSet			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
0	369	364	5	1.36	35	35	0	0	46	46	0	0	450	445	5	1.111
1	358	338	20	5.59	43	38	5	11.6	**49**	**38**	**11**	**22.4**	450	414	36	8
2	367	363	4	1.09	42	36	6	14.3	41	40	1	2.44	450	439	11	2.444
3	350	350	0	0	47	44	3	6.38	53	45	8	15.1	450	439	11	2.444
4	356	354	2	0.56	52	48	4	7.69	42	40	2	4.76	450	442	8	1.778
5	357	328	29	8.12	54	48	6	11.1	39	32	7	17.9	450	408	42	9.333
6	359	348	11	3.06	44	37	7	15.9	47	44	3	6.38	450	429	21	4.667
7	362	359	3	0.83	40	37	3	7.5	48	45	3	6.25	450	441	9	2
8	358	354	4	1.12	46	40	6	13	45	42	3	6.67	449	436	13	2.895
9	335	319	16	4.78	44	37	7	15.9	37	32	5	13.5	416	388	28	6.731
Total	3571	3477	94	2.63	447	400	47	10.5	447	404	43	9.62	4465	4281	184	4.121

A : Total No. of instances B:Total No. of instances classified as a designated class
C: Total No. of instances misclassified D: % misclassification rate

Table IV indicates misclassification rate of 22.4% for digit "1" in test set.

VI. CONCLUSION

As a classifier, the best one hidden layer MLP NN with log-log activation function for hidden and output layer investigated gives an impression to perform reasonably. When it is evaluated on the training instances, it works as an almost good classifier with error rate of 2.63% on training. Here, the regression fit is found to be 0.985. On validation dataset, the error rate was significantly higher that is 10.5%, the regression fit is found to be 0.889. On test dataset, the error rate was that is 9.62%, the regression (best linear) fit is found to be 0.898. Thus it results in overall average classification accuracy of 95.88%. Overall single hidden layer MLP NN with log-log activation function demonstrate its acceptable discriminating ability in separating data possibly into 10 classes at the cost of increase computational complexity of 459994 connections from input to output. This provides an intuition to modify the configuration of network or reduce the dimensionality of data.

VII. REFFERENCES

[1] Reda Alhajj and Ashraf Elnagar, "Multiagents to Separating Handwritten Connected Digits" IEEE transactions on systems, man, and cybernetics—part a: systems and humans, vol. 35, no. 5, september 2005

[2] Stefan Knerr, L Con Personnaz, and G Crard Dreyfus, "Handwritten Digit Recognition by Neural Networks with Single-Layer Training" IEEE transactions on neural networks, vol. 3, no.6 November 1992.

[3] Geoffrey E. Hinton, Peter Dayan, and Michael Revow , "Modeling the Manifolds of Images of Handwritten Digits" IEEE transactions on neural networks, vol. 8, no. 1. january 1997

[4] Guy Mayraz and Geoffrey E. Hinton, "Recognizing Handwritten Digits Using Hierarchical Products of Experts" IEEE transactions on pattern analysis and machine intelligence, vol. 24, no. 2, February 2002

[5] Anil K. Jain, Douglas Zongker, "Representation and Recognition of Handwritten Digits Using Deformable Templates" IEEE transactions on pattern analysis and machine intelligence, vol. 19, no. 12, December 1997

[6] Bailing Zhang, Minyue Fu, Hong Yan, and Marwan A. Jabri, "Handwritten Digit Recognition by Adaptive-Subspace Self-Organizing Map (ASSOM)" IEEE transactions on neural networks, vol. 10, no. 4, july 1999

[7] Anil K. Jain, Douglas Zongker, "Representation and Recognition of Handwritten Digits Using Deformable Templates" IEEE transactions on pattern analysis and machine intelligence, vol. 19, no. 12, December 1997

[8] Paul D. Gader, ,and Mohamed Ali Khabou, , "Automatic Feature Generation for Handwritten Digit Recognition'' IEEE transactions on pattern analysis and machine intelligence, vol 18, no 12, december1996

[9] Sungwon Park, Yunho Yang, and Gyeonghwan Kim, Seon-Hwa Jeong, "A two-stage approach for segmentation and recognition of handwritten digit strings collected from mail pieces" Proceedings of the 17th International Conference on Pattern Recognition (ICPR'2004)

[10] Jung H. Kim1, Byungwoon Ham1, Jui K. Chenl and Sung-Kwon Park2, "The Learning of Multi-output Binary Neural Networks for Handwritten Digit Recognition" Proceedings of 1993 International Joint Conference on Neural Networks

[11] Y. Le Cun,L. D. Jackel,B. Boser,J. S. Denker,H. P. Graf,I. Guyon,D. Henderson,R. E. Howard,W. Hub bard, "Handwritten Digit Recognition: Applications of Neural Network Chips and Automatic Learning" November 1989 - IEEE Communications Magazine,

[12] Yaping Huang, Jiali Zhao, Mei Tian, Qi Zou and Siwei Luo "Slow feature discriminant analysis and its application on handwritten digit recognition" Proceedings of International Joint Conference on Neural Networks, Atlanta, Georgia, USA, June 14-19, 2009

[13] A.D. Parkins and A.K. Nandi, "Method for calculating first-order derivative based feature saliency information in a trained neural network and its application to handwritten digit recognition" IEE Proc.-Vis. Image Signal Process., Vol. 152, No. 2, April 2005

[14] Xu Ye, Zhang Wei, "On a Clustering Method for Handwritten Digit Recognition" Third International Conference on Intelligent Networks and Intelligent Systems 2010

[15] M. Zahid Hossain1,M. Ashraful Amin1,2, Hong Yan, "Rapid feature extraction for bangle handwritten digit recognition" Proceedings of the 2011 International Conference on Machine Learning and Cybernetics, Guilin, 10-13 July, 2011

[16] Michael Revow, Christopher K.I. Williams, and Geoffrey E. Hinton "Using generative model for handwritten digit recognition" IEEE transactions on pattern analysis and machine intelligence, vol. 18, no. 6, june 1996

[17] Zhongkang Lu, Zheru Chi, Wan-Chi Siu "Extraction and Optimization of B-Spline PBD Templates for Recognition of Connected Handwritten Digit Strings" IEEE transactions on pattern analysis and machine intelligence, ,vol. 24, no. 1, january2002

[18] Duda, R. O., and Hart, P. E., "Pattern Classification and Scene Analysis," New York: John Wiley (1973)

[19] Haykin S., Neural Networks: A Comprehensive Foundation, McMillan, New York (1994).

[20] Hornik K. M. Stinchcombe M., and White H. "Multilayer Feedforward Networks Are Universal Approximators," Neural Networks, 2(5), pp. 359-66 (1989).

[21] Fletcher R., Practical Methods of Optimization, 2nd edition, New York: John Wiley (1987).

[22] Powell M. J. D., "Radial basis functions for multivariable interpolation: A review," IMA Conference on Algorithms for the Approximation of Functions and Data, pp. 143-167, RMCS, Shrivenham, England (1985).

[23] M. Hagan, H. Demuth, and M, Beale, Neural Network Design, Thomson Learning, India, (2003).

[24] Martin Fodslette Møller: A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks 6(4): 525-533 (1993)

[25] R. Gonzalez, and R. Woods, Digital Image Processing, Second edition, Pearson Education, India, (2006).

