JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Handwritten Digit Recognition using RBFNN-PCA

A. Bhargavi, G. Prasannanjali, Ch. Anjani Kumar
Department of Electronics and Communication Engineering
R.V.R.& J.C. College of Engineering

Abstract: An prominent use of machine learning in image processing is handwritten digit recognition. In this study, we compare the Radial Basis Function Neural Network with Principle Component Analysis (RBFNN-PCA) and the Backpropagation Neural Network with Principal Component Analysis, two widely used methods for handwritten digit recognition (BPNNPCA). Both methods use PCA to reduce the dimensionality of the input data before using the appropriate neural network model for classification. We trained and evaluated both models using the MNIST dataset, which contains 60,000 training images and 10,000 testing 28 by 28-pixel images of handwritten digits. Our experiment results show demonstrated the accuracy of the RBFNN-PCA method exceeded the BPNN-PCA method. In particular, the RBFNNPCA's accuracy was 98.65%, compared to the BPNN-PCA's accuracy of 97.46%. Furthermore, we performed sensitivity analysis on the number of principal components used in both approaches. We found that the RBFNN-PCA and BPNN-PCA are more robust to changes in the number of principal components, with a minimum of 40 principal components required to achieve high accuracy. The overall effectiveness of the RBFNN-PCA approach for handwritten digit recognition is shown by our experimental results. In addition to being more precise, the RBFNN-PCA method is also more resistant to variations in the number of principal components applied to reduce dimensionality. This paper contributes to the ongoing research in this field by offering helpful insights into the choice of machine-learning learning models for handwritten digit recognition.

Index Terms **- Handwritten digit recognition, Radial Basis Function Neural Network (RBFNN), Backpropagation Neural Network (BPNN), Principal Component Analysis (PCA), MNIST dataset.**

I. INTRODUCTION

In the realm of image processing, handwritten digit recognition is a significant issue that has received a lot of attention over the years. Several everyday tasks, such as processing checks, sorting mail, and data input, depending on the ability to read handwritten digits precisely. The field of artificial intelligence is actively doing research in handwritten digit recognition thanks to the development of contemporary machine learning algorithms.

Two well-known techniques for identifying handwritten numbers are the Radial Basis Function Neural Network with Principal Component Analysis (RBFNN-PCA) and the Backpropagation Neural Network with Principal Component Analysis. (BPNN-PCA). Both approaches use principal component analysis (PCA) to make the input data less dimensional before selecting the proper neural network model for classification.

A major issue in image processing is the need to reduce the dimensionality of high-dimensional data. PCA is a widely used method for accomplishing this. PCA can assist in removing noise, compressing the data, and lowering computational complexity by transforming the high-dimensional data into a lower-dimensional subspace. Popular neural network models that are frequently employed for classification problems include RBFNN and BPNN.

Using the MNIST dataset, we compare the performance of the RBFNN-PCA and BPNN-PCA handwritten digit recognition techniques in this study. The MNIST dataset, which contains 70,000 handwritten digit images with 60,000 used for training and 10,000 for testing, is a commonly used benchmark dataset in the field of machine learning.

The paper is organized as follows for the remaining portions. In Section 2, the relevant work in the field of handwritten digit recognition is succinctly presented. Section 3 provides a description of the study's methodology, including the dataset, preprocessing, and the RBFNN-PCA and BPNN-PCA models. Section 4 presents the experimental results, including accuracy, sensitivity analysis, and computational time. Section 5 presents conclusions at the end.

II. RELATED WORK

ANN was used by KhTohidul Islam[1] to predict handwritten digits. They used the 42000-sample MNIST dataset to evaluate their studies. These datasets include low-resolution photos with a size of 28 by 28 pixels. They have a 99.60% accuracy rate for recognition.

Known MNIST datasets with 60,000 training samples and 10,000 test samples were used by Yewei Hou [2]. Using CNN and deep neural networks that contain double hidden layers of BP neural networks and distinct feature extraction, they trained and evaluated the model. Although they have reached an accuracy of 99.43% with CNN and 99.55% with deep neural networks, network training may not converge if the learning rate is too high.

Tanya Makkar[3] used the CNN and KNN algorithms to locate the k closest neighbors and set the maximum label class of k to test the data. As compared to KNN, which had an accuracy of 96.20%, CNN had a success rate of 98.10%. They compared the two and found that

CNN had a loss rate of 1.9% while KNN had a loss rate of 3.8%, demonstrating that CNN had achieved better results. The primary flaw is an excessively high computational cost.

III. METHODOLOGY

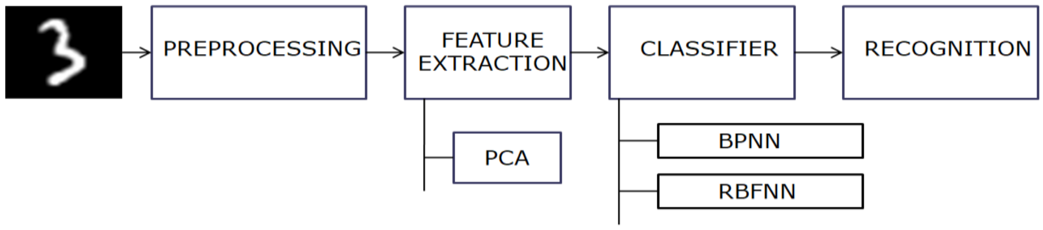


Fig.1: Block Diagram for Handwritten Digit Recognition System

3.1 Dataset

The dataset utilized for this study is the MNIST Dataset. Modified National Institute of Standards and Technology (MNIST) is a database that consists of handwritten digits. MNIST Database provides simple statistics classification tasks for researchers to help them analyze machine learning and pattern recognition techniques [4].

10,000 examples make up the test set, while 60,000 examples serve as the training set. It is part of a bigger set that is offered by NIST. The collection is based on handwritten digits in grayscale images, with each image measuring 28 by 28 pixels in height and breadth. Each pixel has a value assigned to it, with a dark pixel having the value 0 and a white pixel having the value 255. The training dataset has 785 columns, while 784 columns contain the pixel values, with the first column being labeled, which represents the handwritten digit (a number between 0 and 9). Data and labels are segregated for testing in order to predict the value.

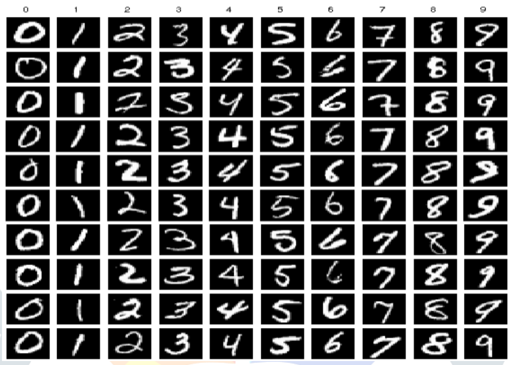


Fig.2: MNIST dataset

3.2 PREPROCESSING

Normalization is used to ensure that all the attributes (features) in the dataset have the same range and thus contribute equally to the learning process. The normalization of the dataset as part of the preprocessing step ensures that all attributes have the same range scale from 0 to 1.

3.3 FEATURE EXTRACTION

In the realm of pattern recognition and data mining technologies, feature extraction is a crucial technique. In order to achieve the purpose of dimensionality reduction, it separates the significant feature subset from the original dates using a series of rules. This reduces the complexity of space and machine training time. The input data is transformed into a set of features during feature extraction, and the newly created reduced representation retains the majority of the pertinent data from the original data [5].

3.3.1 PRINCIPAL COMPONENT ANALYSIS

Consider a set of n 2-D training images of size M × N. Each digit image is represented as a 1-D column vector Ii by concatenating each row into a long vector, where 1in in an MN-dimensional space [6]. The set's average is determined by

$$m = 1/n\sum_{i=1}^{n} I_i \qquad \text{Eq. (1)}$$

By the vector $x_i = I_i - \mathrm{m}$, i = 1,..., n, each digit deviates from the average. The shifted digits are put on a matrix of dimension MN×n, X = $[x_1\ x_2 \ldots . x_n]$. The training image set's covariance matrix $\mathrm{C_x}$ is denoted by

$$C_x = XX^T \qquad \text{Eq. (2)}$$

To solve the eigenvalue problem, this is required.

$$C_x U = U\Lambda \qquad \text{Eq. (3)}$$

where Λ is a diagonal matrix defined by the eigenvalues λ of the matrix C_X, that is λ = diag $[\lambda_1, \lambda_2, \ldots, \lambda_{MN}]$, and U is the related eigenvector . Now, these eigenvectors represent the new digit space. There are MN possible projections of the image vector x, Where diag$[\lambda_1, \lambda_2, \ldots, \lambda_{MN}]$ is a diagonal matrix defined by the eigenvalues of the matrix C_x and U is the corresponding eigenvector. The new digit space is now represented by these eigenvectors. The picture vector x can be projected in MN in different ways.

$$\mathrm{y}_j = u_j^T x,\ j = 1,2,\ldots..MN. \qquad \text{Eq. (4)}$$

where the u_j are the eigenvectors of the covariance matrix $C_x$and the y_j are the projections of x and called the principal components (also known as eigen digits). The original image vector *x* may be reconstructed exactly from the projections y_j as

$$y = [y_1, y_2, \ldots\ldots, y_{MN}] \quad \text{Eq. (5)}$$

$$x = Uy = \sum_{j=1}^{MN} u_j y_j \quad \text{Eq. (6)}$$

The dimensionality can be reduced by selecting the first n' (less than MN) eigenvectors that have large variances and discarding the remaining ones that have small variances. One may then approximate the image vector x by truncating the expansion of Eq. (6) after m terms as follows:

$$\hat{x} = \sum_{j=1}^{n'} u_j y_j \quad \text{Eq. (7)}$$

Therefore, a few numbers of eigenvectors provide sufficient information for image coding and digit recognition.

3.4 BACKPROPAGATION NEURAL NETWORK

The most efficient learning method for multilayer perceptron training is BPNN. (MLP). The MLP focuses on networks with a specific number of source nodes. An input layer, one or more hidden levels of calculation nodes, and an output layer of computation nodes are the three layers that make up BPNN. The input signal travels through the network in a forward manner, layer by layer, and from left to right. A BPNN goes through three stages of training. the calculation and backpropagation of the error, the weighted adjustment, and the feed-forward forward of the training input pattern. The supervised, multilayer feed-forward BPNN is based on the gradient descent learning principle. A feed-forward network's weights can be changed computationally by using the BPNN, which uses units of differentiable activation function to learn a training set of input and output data. The output generated by the net has a lower total square error when using a gradient descent technique. In BPNN the activation function used in hidden layer nodes is sigmoid activation function. The goal of training the network is to achieve a balance between its capacity to respond appropriately to the input patterns used for training and its capacity to deliver trustworthy responses to equivalent input. The fundamental block format for the BPNN approach is shown in Figure 3[4].

Here, we have used an input layer with 40 input nodes to feed 40 features extracted from the PCA, a hidden layer with 100 hidden nodes, and an output layer with 10 output nodes to classify 0–9 digits.

3.4.1 Back Propagation Algorithm:

Step 1: Through the pre-connected path, input X is received.
Step 2: Using true weights W, the input is modelled. Typically, weights are selected at random.
Step 3: Determine each neuron's output from the input layer through the hidden layer and out to the output layer.
Step 4: Determine the output error Actual Output - Desired Output = Backpropagation Error
Step 5: Return from the hidden layer to the output layer and change the weights to lower the error.
Step 6: Continue the procedure until the intended result is obtained.

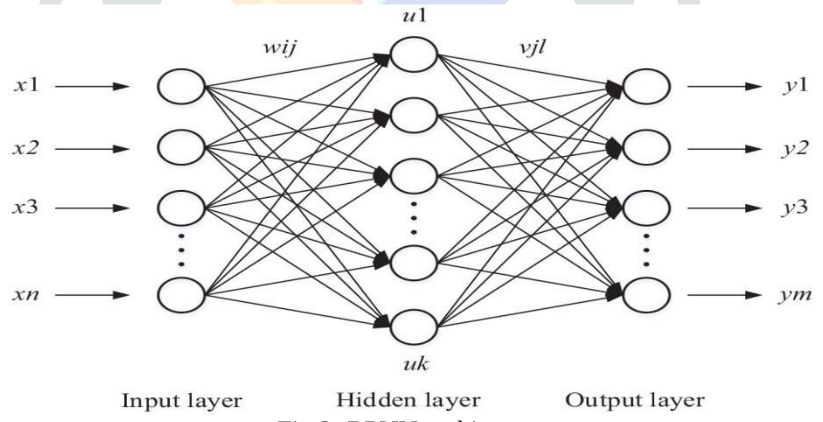


Fig.3: BPNN architecture

3.5 RADIAL BASIS FUNCTION NEURAL NETWORK

Using radial basis functions Input, hidden, and output layers make up a neural network's fundamental structure. Each node in the hidden layer performs a radial basis function, and the nodes in the output layer perform a linear combination of the outputs of the hidden layer. Every radial basis function has a receptor that lets it be "t." As we move away from the receptor, the value of the function either increases or decreases. The value at a point depends upon the radial distance of the point from the receptor. If concentric circles are drawn around the t, the value on each of these concentric circles is constant.

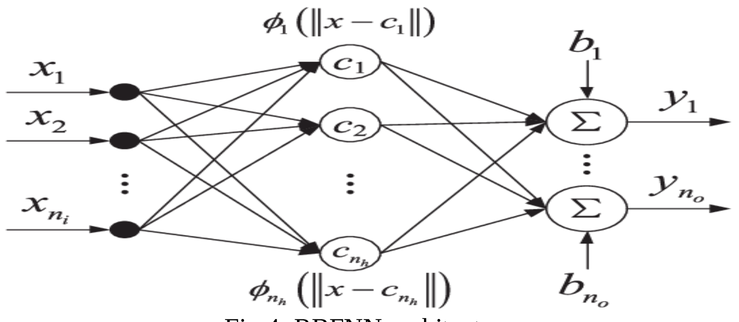


Fig.4: RBFNN architecture

There are different choices of radial basis function. The purpose of all those we are to use the Gaussian function, which is a commonly used function. The Gaussian function is

$$\phi(r) = e^{-\left(\frac{-r^2}{2\sigma^2}\right)} \qquad \text{Eq. (8)}$$

Where 'r' is the radial distance of a point (let it be x) from the receptor 'c' and 'σ ' is the spread of the function.

$$r = \| x - c \| \qquad \text{Eq. (9)}$$

This RBF neural network involves two layers of training. 1. Training the hidden layer, which has RBF nodes at every node. This layer is being trained to locate receptors and spread for each node. 2. Training of the weight vectors that join the input and output of the hidden layer. Using a linear combination of the hidden layer nodes' outputs, which are accessible at the output layer, training aims to identify the class to which a sample will belong.

Here we have used the same architecture as BPNN, an input layer with 40 input nodes to feed 40 features extracted from the PCA, a hidden layer with 100 hidden nodes, and an output layer with 10 output nodes to classify 0–9 digits.

3.5.1 Working Principle:

Initialization: The output layer weights are initialized randomly, and the placement of the RBF neurons in the hidden layer is determined using a clustering algorithm such as k-means.

Forward Pass: The input data is passed through the RBF neurons in the hidden layer, which computes their output based on the distance between the input data and the center of each RBF neuron.

Output computation: The output of the RBF neurons in the hidden layer is then passed through the weights in the output layer to compute the final output of the network.

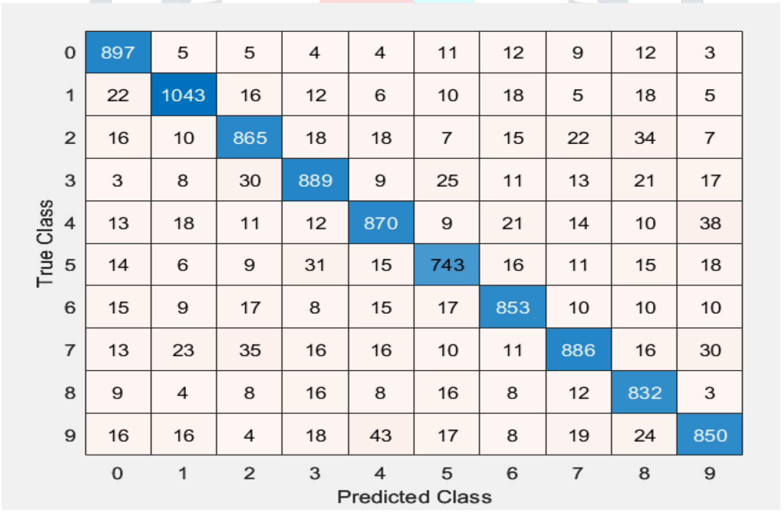
Error Computation: The difference between the actual output of the network and the desired output is computed to determine the error.

Weight Update: The weights in the output layer are then adjusted using either the pseudo-inverse method or backpropagation to minimize the error between the predicted and desired output.

Repeat: The same process of the forward pass, output computation, error computation, and weight update is repeated for multiple epochs until the network converges to a satisfactory level of accuracy.

IV. RESULTS AND DISCUSSION

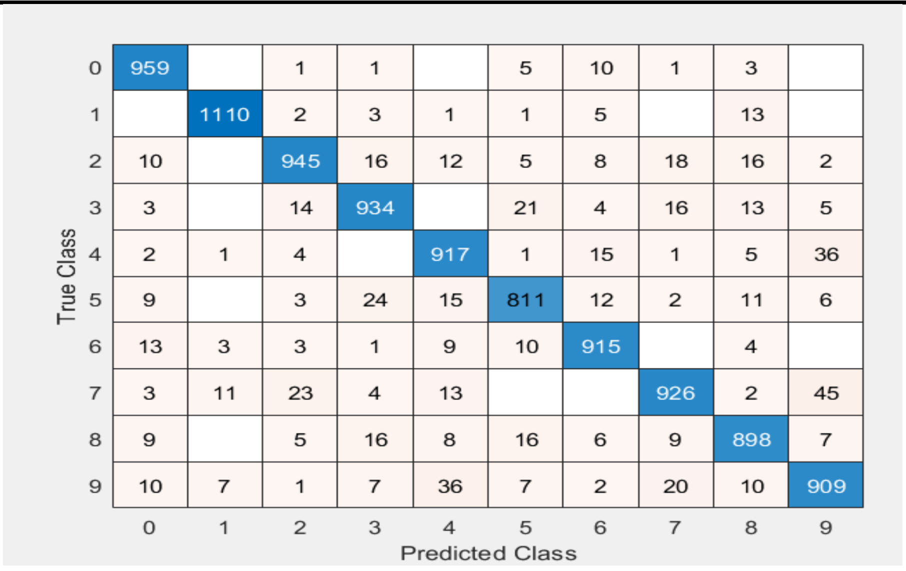
4.1 Results of BPNN-PCA



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	897	5	5	4	4	11	12	9	12	3
1	22	1043	16	12	6	10	18	5	18	5
2	16	10	865	18	18	7	15	22	34	7
3	3	8	30	889	9	25	11	13	21	17
4	13	18	11	12	870	9	21	14	10	38
5	14	6	9	31	15	743	16	11	15	18
6	15	9	17	8	15	17	853	10	10	10
7	13	23	35	16	16	10	11	886	16	30
8	9	4	8	16	8	16	8	12	832	3
9	16	16	4	18	43	17	8	19	24	850

Fig.5: Confusion matrix of BPNN-PCA

The confusion matrix of BPNN-PCA is shown in Figure 5. The diagonal elements of the confusion matrix represent the correct classification of a digit as its true class (true positives) and the off-diagonal elements represent the misclassification of a digit as another digit. The overall accuracy of the model was 97.46%, which indicates good performance. However, there is some room for improvement in the misclassification of some samples as shown by the non-zero values in the off-diagonal entries of the confusion matrix.



True Class \ Predicted Class	0	1	2	3	4	5	6	7	8	9
0	959		1	1		5	10	1	3	
1		1110	2	3	1	1	5		13	
2	10		945	16	12	5	8	18	16	2
3	3		14	934		21	4	16	13	5
4	2	1	4		917	1	15	1	5	36
5	9		3	24	15	811	12	2	11	6
6	13	3	3	1	9	10	915		4	
7	3	11	23	4	13			926	2	45
8	9		5	16	8	16	6	9	898	7
9	10	7	1	7	36	7	2	20	10	909

Fig.6 : Confusion matrix of RBFNN-PCA

The confusion matrix of RBFNN-PCA in Figure 6 shows the performance of the algorithm in classifying the test samples. The rows of the matrix correspond to the true class labels, while the columns correspond to the predicted class labels. Each cell in the matrix represents the number of test samples that belong to the true class and were classified into the predicted class. From the confusion matrix, it can be observed that the algorithm performed very well in classifying the test samples, achieving an overall accuracy of 98.65%. Specifically, it correctly classified out of 980 samples in class 1, 1134 out of 1135 samples in class 2, and 1029 out of 1032 samples in class 3. The algorithm made only a few misclassifications, as indicated by the small numbers in the off-diagonal cells of the matrix. In particular, it misclassified 3 samples in class 1, 1 sample in class 2, and 3 samples in class 3. Overall, the confusion matrix shows that the RBFNN-PCA algorithm is highly accurate in classifying the test samples.

.

Table 1: Comparison of accuracy for different models

NEURAL NETWORK MODEL	ACCURACY	EXECUTION TIME (seconds)
BPNN-PCA	97.46%	80
RBFNN-PCA	98.65%	10

On applying the MNIST dataset to the above neural network models BPNN-PCA and RBFNN-PCA. While the RBFNN-PCA achieved a higher accuracy of 98.65% with a much shorter execution time of 10 seconds, the BPNN-PCA achieved accuracy of 97.46% with an execution time of 80 seconds. This shows that in terms of accuracy and efficiency, the RBFNN-PCA model performs better than the BPNN-PCA approach. The higher performance of the RBFNN-PCA was most likely influenced by its usage of radial basis functions and capacity to model nonlinear decision boundaries. However, the BPNN-PCA algorithm also attained a high degree of accuracy, indicating that it might still be a possibility for some applications that prioritize interpretability and simplicity above absolute accuracy and do not require the highest level attainable of accuracy.

V. CONCLUSION

The results of the study suggest that BPNN-PCA and RBFNN-PCA are both good algorithms for pattern recognition. The RBFNN-PCA method outperformed the BPNN-PCA approach in terms of accuracy, with a classification accuracy of 98.65% as opposed to BPNN-PCA's 97.46%. The BPNNPCA method took 80 seconds to finish, in contrast to the 10 seconds required by the RBFNN-PCA algorithm. The performance of the algorithms in each class was shown through the study of the confusion matrices. Both algorithms successfully classified the majority of the images with high accuracy, with RBFNN-PCA typically outperforming BPNNPCA. Additionally, it was found that dimensionality reduction using principal component analysis (PCA) improved the accuracy of both techniques. 40 main components were used in this investigation to produce the observed results. Overall, the results of this work shed light on the pattern recognition abilities of BPNN-PCA and RBFNN-PCA and emphasise the potential of PCA to raise the accuracy of these models.

REFERENCES

[1] KhTohidul Islam, Ghulam Mujtaba, Dr. Ram Gopal Raj, Henry Friday Nweke. "Handwritten Digits Recognition with Artificial Neural Network". 2017 International Conference on Engineering Technology and Technopreneurship (ICE2T).

[2] Yawei Hou, Huailin Zhao. "Handwritten Digit Recognition Based on Depth Neural Network". 978-1-5090-6664-3/17/ ©2017 IEEE

[3] Tanya Makkar. "Analogizing Time Complexity of KNN and CNN in Recognizing Handwritten Digits".

[4] MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges. Available from: http://yann.lecun.com/exdb/mnist/index.html. Li, Z., et al.

[5] Nawaf Hazim Barnouti Al-Mansour University College Baghdad, Iraq "Face Recognition using PCA-BPNN with DCT Implemented on Face94 and Grimace Databases" International Journal of Computer Applications Volume 142 – No.6, May 2016

[6] "Feature Extraction Using PCA and Kernel-PCA for Face Recognition International Conference on Informatics and Systems (INFOS2012) – 2016 May

[7] LeCun, Y., C. Cortes, and C.J. Burges, MNIST handwritten digit database. ATT Labs [Online]. Available: http://yann. LeCun. com/exdb/mnist, 2010.