International Journal of Intelligent Computing and Information Science

REDUCING ERROR RATE OF DEEP LEARNING USING AUTO ENCODER AND GENETIC ALGORITHMS

F. A. Habeeb Sherihan Abuelenin Samir Elmougy

Faculty of Computers and Information, Mansoura University, Egypt
fafh30@yahoo.com sherihan@mans.edu.eg mougy@mans.edu.eg

Abstract: *Deep Learning (DL) techniques are considered as one of machine learning classes that model hierarchical abstractions in data input with the assistance of multiple layers. DL techniques have accomplished high performance in computer vision, natural language processing and automatic speech recognition. DL combines lower modules for classifier output and raw features input to produce new features at hierarchy higher layer. Deep Auto Encoder (DAE) is a DL aims to represent data to be utilized for rebuilding and classification. It is considered as one of the powerful algorithms in DL that gives higher accuracy and best performance. The proposed method in this work is based on using DAE and Genetic Algorithm (GA) through applying split-training and merging algorithms for DL. First, the network is divided into two initialized networks using DAE. Second, both of these networks were merged using GA. This proposed approach was evaluated based on the Mixed National Institute of Standards and Technology (MNIST) dataset and the obtained results showed that it achieve a higher performance and lower error rate in the classification.*

Keyword: Deep Auto Encoder, Genetic Algorithm, Machine learning, Deep Learning.

1. Introduction

Deep Learning (DL) includes algorithms for modeling high-level data abstractions using multi-processing layers. Neural Network (NN) usually has one to two hidden layers, which are used for supervised prediction and/or classification. Support Vector Machine (SVM) is typically applied in binary classification. It is occasionally used for other supervised learning tasks. DL neural network architectures vary from the NNs because they have more hidden layers. Also, DL networks differ from both NNs and SVMs because it works for supervised or unsupervised learning tasks. The main goal of DL is to model complex-hierarchical features in data. DL is not a specific type of algorithms, such as feed forward NNs or SVMs, but a set of machine-learning algorithms. Supervised DL carries out stacking with the help of labels in the learning of each layer, which is typically a simple classifier. Combinational of the lower modules for classifier output and raw features input produced the new features for the stack classifier at hierarchy higher layer. Unsupervised DL performs stacking in a layer by layer manner that generally includes no label information. This results in rise to multiple-layers in unsupervised feature learning, as described in DBN, DAE [25]. Hybrid DL depends on the discrimination which is significant assisted with the outcomes of unsupervised and/or supervised or deep networks.

There are many DL techniques such as Deep Belief Network (DBN), Boltzmann Machine (BM), Restricted Boltzmann Machine (RBM), Deep Neural Network (DNN), and Deep Auto Encoder (DAE).

RBM is a Markov Random Field (MRF) used to designate the dependencies between a set of random variables using two architecture layers [10]. It is undirected bipartite graph model and composed of two layers: visible (v) and hidden (h) as illustrated in Figure (1). It can learn a probability distribution for a collection of inputs. It is used in the classification and feature learning applications because it is faster than the BM. The factorial nature of these conditional distributions enables efficient Gibbs sampling which forms the basis of the most widespread RBM learning algorithms such as Contrastive Divergence (CD) [24].

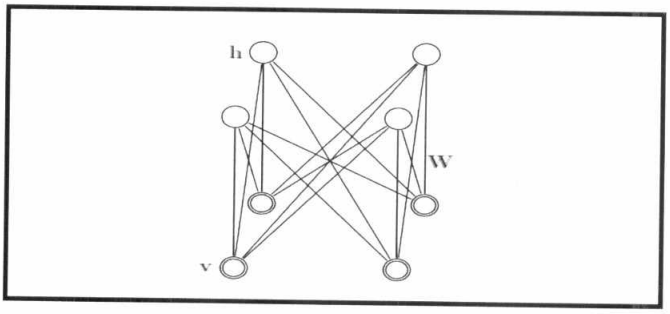

Figure (1): Restricted Boltzmann Machine (RBM) architecture **[13]**

Figure (2) illustrates RBM algorithm where each neuron has a bias to hidden binary stochastic units. The energy is given by Equation (1) [28].

$$E(v, h) = -\sum_{i \in visibal} b_i x_i - \sum_{j \in hidden} c_j h_j - \sum_{ij} w x_i h_j \quad (1)$$

where x_i and h_j donate the states of units; b_i and c_j are units biases while w_{ij} is the connection weight between them. The network gives a probability to any possible correspondence of visible and hidden vectors using Equation (2) [28].

$$p(x, h) = \frac{1}{z} e^{-E(x,h)} \quad (2)$$

where the partition function Z is given by Equations (3) and (4).

$$p(x) = \frac{1}{z} \sum_h e^{-E(x,h)} \quad (3)$$

$$p(x|h) = \prod_{i=1}^{m} p(x|h) \quad (4)$$

Stochastic steepest ascent is given in Equation [28].

$$\Delta W_{ij} = \epsilon((x_i h_j)_{data} - (x_i h_j)_{recon}) \quad (5)$$

where ϵ is the learning rate. The approximation model {W, b, c} named as θ is formed by a taking small number of Gibbs sampling.

DBN can be produced using RBMs and fine-tuning the resulting deep network with gradient descent and Back Propagation (BP). It is a generative-graphical model, which consists of hidden units

layers with linking among them but not among the units at the same layer [2].Figure (3) represents three hidden layers where every layer acquires the connections between hidden features activities in the lower layers. DBN two top layers make an undirected bipartite graph while the lowers make a directed graph [5].The coditional probability for the visibale layer is give in equation (7) and for the hidden layer is calculated from Equation (6) [20].

RBMupdate($\mathbf{x}_1, \epsilon, W, \mathbf{b}, \mathbf{c}$)
This is the RBM update procedure for binomial units. It can easily adapted to other types of units.
$\mathbf{x}_1$ is a sample from the training distribution for the RBM
ϵ is a learning rate for the stochastic gradient descent in Contrastive Divergence
W is the RBM weight matrix, of dimension (number of hidden units, number of inputs)
$\mathbf{b}$ is the RBM offset vector for input units
$\mathbf{c}$ is the RBM offset vector for hidden units
Notation: $Q(\mathbf{h}_2 = 1|\mathbf{x}_2)$ is the vector with elements $Q(\mathbf{h}_{2i} = 1|\mathbf{x}_2)$

for all hidden units i **do**
- compute $Q(\mathbf{h}_{1i} = 1|\mathbf{x}_1)$ (for binomial units, $\text{sigm}(\mathbf{c}_i + \sum_j W_{ij}\mathbf{x}_{1j})$)
- sample $\mathbf{h}_{1i} \in \{0, 1\}$ from $Q(\mathbf{h}_{1i}|\mathbf{x}_1)$

end for
for all visible units j **do**
- compute $P(\mathbf{x}_{2j} = 1|\mathbf{h}_1)$ (for binomial units, $\text{sigm}(\mathbf{b}_j + \sum_i W_{ij}\mathbf{h}_{1i})$)
- sample $\mathbf{x}_{2j} \in \{0, 1\}$ from $P(\mathbf{x}_{2j} = 1|\mathbf{h}_1)$

end for
for all hidden units i **do**
- compute $Q(\mathbf{h}_{2i} = 1|\mathbf{x}_2)$ (for binomial units, $\text{sigm}(\mathbf{c}_i + \sum_j W_{ij}\mathbf{x}_{2j})$)

end for
- $W \leftarrow W + \epsilon(\mathbf{h}_1\mathbf{x}_1' - Q(\mathbf{h}_2 = 1|\mathbf{x}_2)\mathbf{x}_2')$
- $\mathbf{b} \leftarrow \mathbf{b} + \epsilon(\mathbf{x}_1 - \mathbf{x}_2)$
- $\mathbf{c} \leftarrow \mathbf{c} + \epsilon(\mathbf{h}_1 - Q(\mathbf{h}_2 = 1|\mathbf{x}_2))$

Figure (2) Restricted Boltzmann Machine (RBM) algorithm [17]

$$p(h_j = 1|\, V) = \sigma(b_j + \sum_i v_i\, w_{ij}) \quad (6)$$

$$p(v_i = 1\,|H) = \sigma(a_i + \sum_j h_j\, w_{ij}) \quad (7)$$

The weight is updated by Equation (8) [20].

$$\Delta w_{ij} \propto < v_i h_j >_{data} - < v_i h_j >_{reconstruction} \quad (8)$$

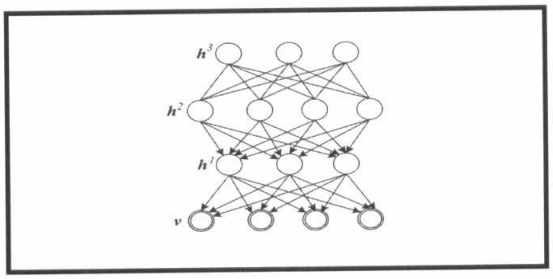

Figure (3): Deep Belief Network (DBN) [5]

The BM is a type of probabilistic NN used in density modeling [19] as represented in Figure (4). It can be represented as a MRF with only pairwise connection between units, and the units are typically binary valued. The energy is given by Equation (9) [17].

$$\text{Energy}(x, h) = -b'x - c'h - hWx - x'Ux - h'V\,h \qquad (9)$$

In general, learning is fast in "Restricted Boltzmann Machines" which has a single layer of feature detectors. RBM is a widely used special case in which P(h|x) and P(x|h) are both tractable because they are factorized in BM [8].

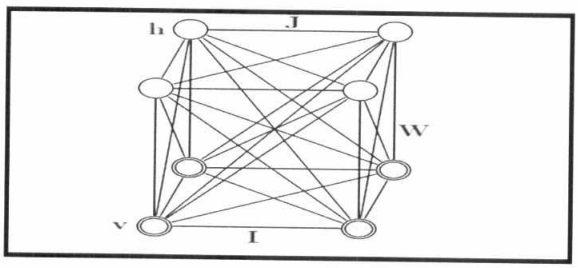

Figure (4): Boltzmann machine (BM) [13].

DNNs can model complex non-linear relationships and are normally designed as feed forward networks as shown in Figure (5). A DNN can be trained discriminatively with the standard back propagation algorithm (BP). BP is a DNN learning applied in combination with an optimization method such as gradient descent.

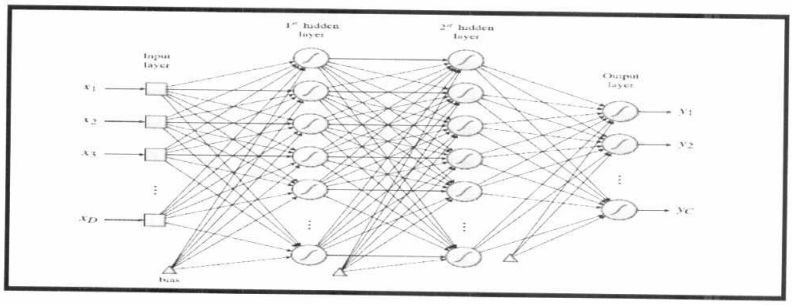

Figure (5): Architecture of Deep Neural Network (DNN) [4]

The traditional Auto Encoder (AE) is an ANN that tries to reproduce its input and the target [7]. Figure (6) illustrates the architecture of the AE.

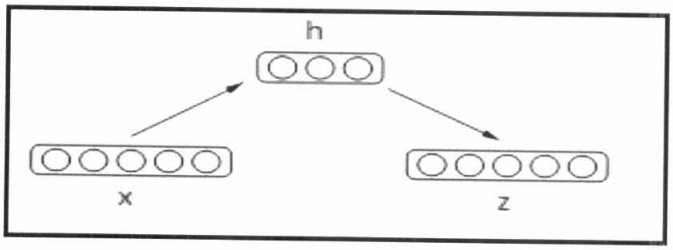

Figure (6): Auto Encoder (AE) architecture [17]

Based on original AE shown in Figure (6), its main function is to have a minimum reconstruction error. It maps input x to a hidden representation y (h(x)) then back to z (reconstruction) that minimizes this function. Updating the parameters is efficiently achieved with the Conjugate Gradient (CG); which can be efficiently implementing using the BP algorithm. DAE is a special type of DNN that is applied for learning representation of data. In this DAE, the lower layers apply the matrices for encoding input data but the upper layers decode input data through using the matrices in a reverse order. Then the DAE is fine tune using error BP to minimize the encoding errors. The probabilities of hidden units are handled as a source for training another Bernoulli-Bernoulli RBM. Detailed of AE is shown in Figure (7).

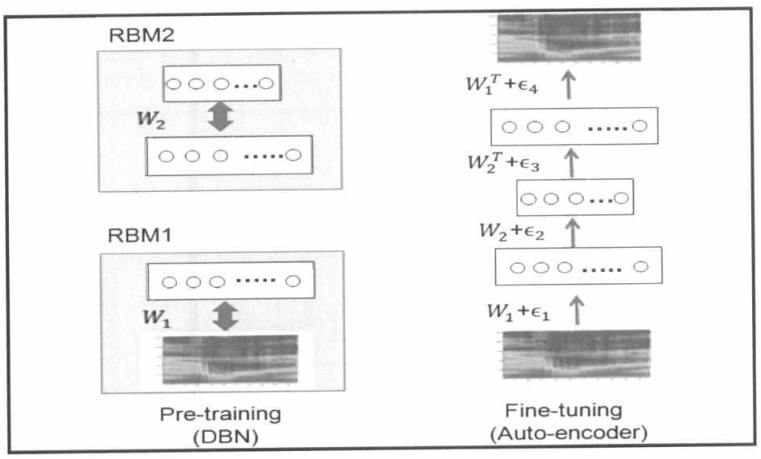

Figure (7) Detailed architecture of original Auto Encoder (AE)

A Deep Belief Net can be formed through combining two RBM's to easily infer the states of the second layer of binary hidden units from the input in a single forward pass. Stack Auto Encoder (SAE) implies a stack of signal level AE hence the SAE uses the AE described above as building blocks to create a deep network [7]. The Denoising Autoencoder (DA) is a version of AE [29] in which some parts of the input data are selected in and assigned to be zero to remove any effect of stochastically degraded input of Auto-encoder. It is stacked to build a Stacked Denoising Auto Encoder (SDA) [9]. DA output in the below layers is used to be input of higher layers. Unsupervised pre-training and supervised fine tuning [1] are SDA training processes.

Genetic Algorithm (GA) is an adaptive-heuristic search approach depends on the theory of natural selection and evolution for solving optimizing problems [27]. It was officially initiated in the USA in the 1970s by John Holland at the University of Michigan [3]. Mutation is a common operator used to help in preserving diversity in the population by finding new points in the search pace to evaluate. Crossover creates new individuals by integrating parts from two individuals. It uses a global parameter to indicate the likelihood where every variable is exchanged between two parents [26].Mutation is a common operator used to help in preserving diversity in the population by finding new points in the search pace to evaluate. It creates new individuals by making changes in a single individual. Crossover creates new individuals by integrating parts from two individuals. It does not use cut-points, but it simply uses a global parameter to indicate the likelihood, where each variable should be exchanged between two parents [3].

In Figure (8) [21], each 6×6 is composed of weighs and biases between each layer of the NN. The first column represents the bias vector and the rest of columns are the matrix weights. The crossover refers to the process of copying selected rows of parent1=*a* and parent2=*b* to the base matrix that has a value depending on the rate of crossover. The copying ration between parent *a* and b is identified as fraction ratio. Mutation occurs after the crossover with the probability of p(m). After that, validation group are prepared as a training portion and get the error rate in each spring collection. Next, two offsprings show the lowest of the error are chosen as parents for the next generations. The main objective of this process is to find out the offspring in order to reduce the function of zero and one loss [12].

In this work, GA is used with DAE to decrease the rate of error in the dataset. The second section represents related researches, the third section presents the proposed approach, the fourth shows the obtained results and their discussions and the final section represents conclusions.

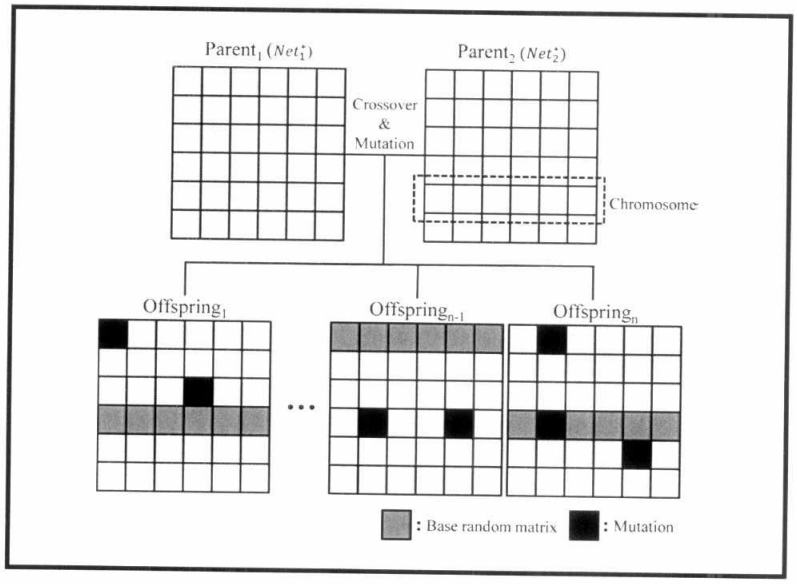

Figure (8): A simple representation of the Genetic Algorithm (GA) model [21]

2. Related Works

Iqbal [16] developed a system that provided an unsupervised deep belief network. This network can acquire information from multiple-channels and relate accurately to partial information. A correspondence layer was used to combine multiple-channels using back-fitting which helps in adding non-exist channels. Tissera and Mcdonnel [15] proposed an approach for synthesizing deep NN using Extreme Learning Machines as a stack of supervised auto encoders. The results concluded that error rate of classification was improved using integration of additional ELM train without using any label.

Qingyang and Zhang [13] proposed a method through which the weight sparse auto encoder learn the digital number outline of hand-writing instead of pen-strokes when the hidden number is smaller. The hidden layer output is the compress of the input data which gives well representation. Any increase in

the number of hidden unit results in increasing accuracy. The weight can learn the pen-strokes of hidden writing when the hidden unit number is large.

Zhao, et al. [14] proposed a method that can learn feature from data that contains non-Gaussian noise and outliers. This method combined correntropy and contractive auto-encoder into Correntropy based Contractive Auto Encoder (C-CAE). C-CAE is capable of dealing with correntropy and contractive auto-encoder. Maximum Correntropy Criterion (MCC) is adopted as a reconstruction cost function and a well-chosen penalty term is added to reconstruct cost function by replacing cross entropy with MCC.

Huang, et al. [18] presented a method depends on Chaos theory and BP ANN molding for casting wind power. Their prediction model depends on both of the best embedding dimension and the best delay time. Their method had more accuracy than using only BP ANN.Yu et al. [11] presented a stacked auto encoder model with MCC. That model showed better performance when using large amounts of outliers. It also showed that correntropy is robust to outliers so that it is promising for robust algorithm design.

3. The Proposed Methodology

In this work, we have adopted the split training and merge methodology described in [21] through applying DAE as DL and GA to reduce the errors in classification in hand-written based. Figure (9) illustrates the flowchart of this approach. The main components of this methodology are comprised of two stages. In the first stage, two networks were introduced with their related DAE. In the second stage, the set of two networks were merged using GA. The main steps can be summarized as follow:

1) The input of DAE is divided into: a) (*TR*) set to be used for training the networks, b) (*TE*) set to be used for network performance. Also, *TR* was divided into two sub-sets (TR_1 and *TR2*). The network (matrix) is divided into some batches with a fixed batch size. For example, when taken a batch of 600 is used this batch is then divided into two equal size deep networks.
2) Training data sets in DAE: both TR_1 and *TR2* are trained using DAE1and DAE2, respectively. Both networks (DAE1 and DAE2) took different biases and weights. The training outputs of these networks were referred as Net1 and Net2, respectively. This process is continual using *m*-iterations number after learning *N* layers using DAE. We have constructed N+1 ANN with these initial weights and then fine-tuned them.
3) In the following step the DAE is used with GA. First, *DAE1* and *DAE2* are used in the GA parents, then repeat the GA and get one ANN at the end and fine-tune it. Also, we use GA for crossover heuristic as a feature extractor for NN initial weights optimization. However, the presented method uses GA as processor for image segmentation after image processing using NN. It inputs x to a hidden representation y(h(x)) then back to z (reconstruction) that minimizes this function as described in Equations (10) and (11) [7].

$$h(x) = sigm(W1x + b1) \quad (10)$$
$$z(x) = sigm(W2h(x) + b2) \quad (11)$$

To train the model, the average between x and the reconstructed z with respect to the parameter should be minimized. The θ value is calculated using Equation (12) [7].

$$\theta = argmin \frac{1}{N} \sum_{i=1}^{N} L(x^{(i)}, z(x^{(i)})) \quad (12)$$

where N is the number of training samples and L is the function that measures the difference between x and z as given in Equation (13) [7].

$$L(x,z)=||x-z||^2 \quad (13)$$

Each sample contains a target classification label to be applied for the subsequent supervised classification phase.

```
Input: TR: Training Dataset,
       TE: Test Dataset,
       m:  number of generation,
       n:  number of layers
Begin

 //Split Phase
       Split TR into two training Datasets: TR1 and TR2
       Net1^0= training TR1in DAE1
       Net2^0= training TR2 in DAE2

//Merge phase
      Merge Net1^0 and Net2^0 with genetic algorithm
       m:= 0

// Iterative m-Generation
       For i:=1 to m
        Begin
              Select two fitness net
              Net^m=(Net1^m , Net2^m)
        End For

//Output Phase
Fine tune using BP and error rate of Test Set (TE)

End
```

Figure (9) Split and merge algorithm

4. Experimentation and Results

The proposed methodology was implemented in the Matlab [22] software package running on Intel core i5, 4 GB Ram, 400 GB HD, and Windows 7(64 bit). For our experiments, MNIST hand-written digit recognition database were used [23]. MINIST dataset is recognized as a network of images and each grayscale image has a digital number between 0 and 9 with a dimension of 28*28 pixels.

For training, 60,000 images were used and as a result a matrix of 60000 rows was constructed. In that matrix each image is a sequence of its constructed pixels. Accordingly, the number of rows, say x, is 60000 (number of images) and number of columns, say h, is 28*28 = 784. Another 10,000 images of the dataset were used for testing. The matrix was divided into two matrices. The first matrix, TR_1, consists of the first 30,000 images of the original matrix. The other matrix, TR_2, consists of 30,000 images of the original matrix that numbered from 30,001 to 60,000. Both TR_1 and TR_2 were trained using DAE1 and DAE2, respectively.

In the simulation, six different NNs were computed for evaluation the proposed methodology as represented in Table (1). In each of these networks, each RBM was trained at a fixed learning rate of 0.1 for 50 epochs. After the RBMs being pre-trained, BP was performed using batch size of 600 for 50

Table (1): Error rate of proposed and DAE in terms of training and testing from stage epoch (50).

Case	Layer	Proposed Training Error	DAE Training Error	Proposed Testing Error	DAE Testing Error
1	[784-100-100-500]	0.00	0.32	2.16	2.19
2	[784-200-200-1000]	0.00	0.042	1.36	1.58
3	[784-300-300-2000]	0.00	0.022	1.28	1.34
4	[784-500-500-2000]	0.00	0.00	1.18	1.29
5	[784-200-200]	0.00	0.017	1.82	2.24
6	[784-100-100]	0.123	0.007	2.79	2.54

Table (1) shows the training errors for the proposed methodology. It shows that the training errors for the networks from 1 to 5 were equal to zero. Also, the test errors for the same networks were less than their correspondence when the DAE was used. For network number 6, the DAE gave better errors for both training and testing when compared with the proposed methodology except for the network with small number of hidden layers. Figures (10 to 15) illustrate the obtained results in details.

From Figure (11), the error rate is zero in training when epoch values are ranged from 31 to 50 but when applying AE without GA, the lowest error rate is 0.0416 at epoch 50. For the testing, the lowest error rate is 1.36 is achieved at epoch 36 but when applying AE without GA, the lowest error rate is 1.48 at epoch 47.

From Figure (13), the lowest error rate is zero in training at epoch 26 and when epoch is ranged from 28 to epoch 50 but when applying AE without GA, the lowest error rate is equal to zero in training at epochs 44, 45, 46, 49, and 50. For the testing, the lowest error rate is 1.18 is achieved at epoch 50 but when applying AE without GA, the lowest error rate is 1.27 at epoch 47.

From Figure (14), the lowest error rate is zero in training when epoch is ranged from 32 to 50, but when applying AE without GA, the error rate is not equal to zero. The error rate is equal to 1.18 for the testing in epochs 39, 46, 49 while using AE without GA, the lowest error rate is equal to 1.27 at epoch 50.

Table (2) shows the error rate for the DAE and the proposed methodology with different generations. The proposed methodology includes the DAE when using 82, 500 and 10 generations applied on the network layers [784-300-700] using fixed learning rate 0.1 and batch size of 600 for 50 epochs. Figure (16) reveals that the proposed methodology with 10 generations has reduced the error more than the DAE alone.

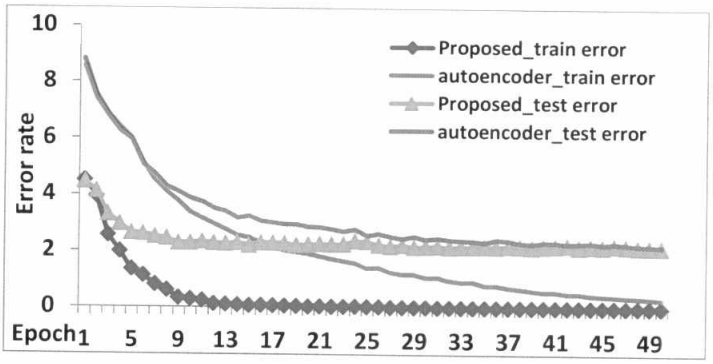

Figure (10): Error rate of proposed and DAE in terms of training and testing from layers [100-100-500]

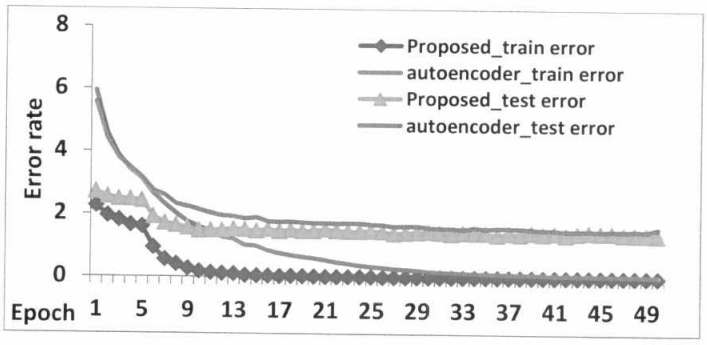

Figure (11): Error rate of proposed and DAE in terms of training and testing from layers [200-200-1000]

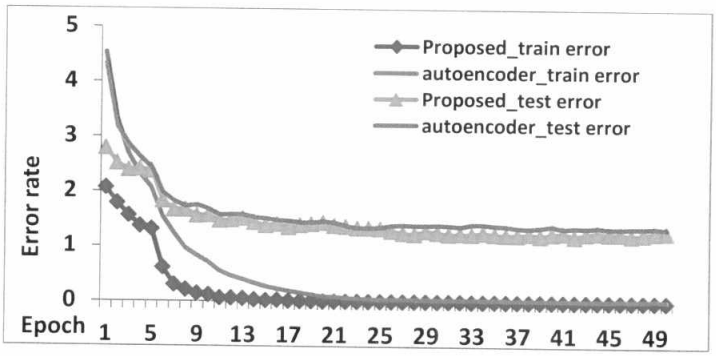

Figure (12): Error rate of proposed and DAE in terms of training and testing from layers [300-300-2000]

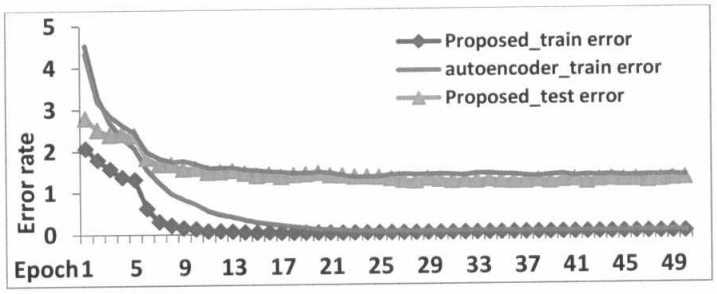

Figure (13): Error rate of proposed and DAE in terms of training and testing from layers [500-500-2000]

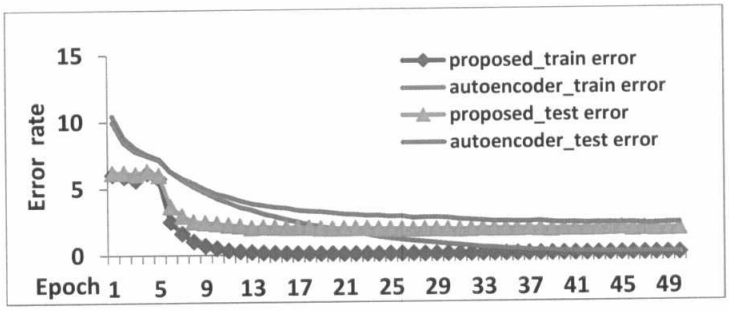

Figure (14): error rate of proposed and DAE in terms of training and testing from layers [200-200]

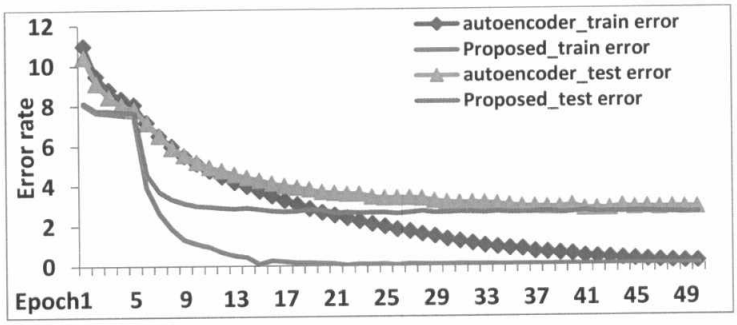

Figure (15): Error rate of proposed and DAE in terms of training and testing from layers [100-100]

Table (2) Error rate of DAE proposed with different generation

Case	Layer	Test error	train error
1	DAE	1.22	0
2	Proposed with 82 generation	1.17	0
3	Proposed with 500 generation	1.25	0
4	Proposed with 10 generation	1.15	0

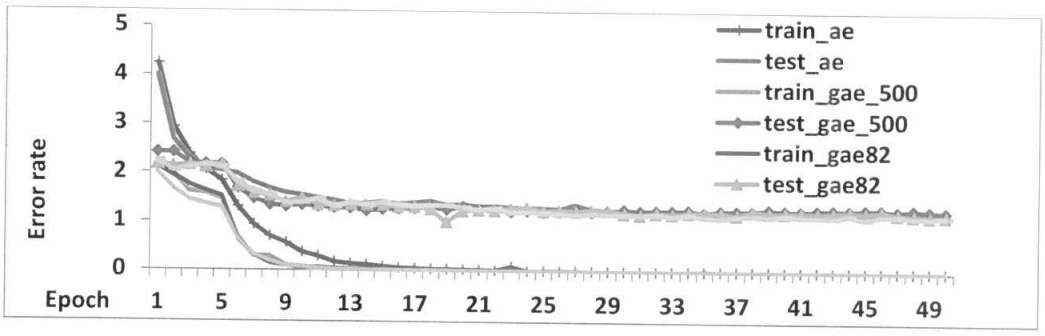

Figure (16): Error rate of DAE proposed with different generation

Conclusion

This paper proposed a methodology to reduce the error rate through using by integrating Deep Auto-encoder and Genetic algorithms. It splits the network into two networks by DAE and combined by GA. The proposed methodology successfully improved the error rate when compared with the DAE alone. This is because the use of deep learning separates the dataset into two groups and trains them separately, which results in two networks in two different features each of which has different weights and biases. The next step comes after the intermarriages between the two networks, where they are integrated with a good gene and this result in reducing the error rate.

References

1. Hinton, G. E. (2007). Learning multiple layers of representation. Trends in Cognitive Sciences, 11(10), 428-434.
2. Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. No. CU-CS-321-86, Department of Computer Science, Colorado University, USA.
3. El-sayed, N.,(2011),Probablistic machine learning with expert system to disease diagnosis, MSc thesis, Department of Computer Science, University of Mansoura, Egypt.
4. Lopes, N., and Ribeiro, B. (2015). "Machine Learning for Adaptive Many-core Machines: A Practical Approach". Springer.
5. Ling, Z. H., Deng, L., and Yu, D. (2013). Modeling spectral envelopes using restricted Boltzmann machines and deep belief networks for statistical parametric speech synthesis. IEEE Transcation on Audio Speech and Language Processing, 21(10), 2129-2139.
6. Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P. A. (2008). Extracting and composing robust features with denoising autoencoders. Proceedings of the 25th international conference on Machine learning, 1096-1103.
7. Palm, R. B. (2012). Prediction as a candidate for learning deep hierarchical models of data. MSc Thesis, Technical University of Denmark, Informatics and Mathematical Modelling, Kongens Lyngby, Denmark.
8. Hinton, G. E., Sejnowski, T. J., and Ackley, D. H. (1984). Boltzmann machines: Constraint satisfaction networks that learn. Pittsburgh, PA: Carnegie-Mellon University, Department of Computer Science.
9. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P. A. (2010). Stacked denoising autoencoders, Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research, 11: 3371-3408.

10. Hinton, G. E. (2009). Deep belief networks. Scholarpedia, 4 (5): 5947.
11. Qi, Y., Wang, Y., Zheng, X., and Wu, Z. (2014, May). Robust feature learning by stacked autoencoder with maximum correntropy criterion. In Acoustics, Speech and ICASSP (2014). IEEE International Conference .Signal Processing, 6716-6720.
12. Vapnik, V. N. (1999). An overview of statistical learning theory. IEEE Transactions, Neural Networks, 10(5), 988-999.
13. Xu, Q. and Zhang, L. (2015). The effect of different hidden unit number of sparse autoencoder. Control and Decision Conference (CCDC), May 27, China, 2464-2467.
14. Zhao, D., Guo, B., Wu, J., Ning, W., and Yan, Y. (2015). Robust feature learning by improved auto-encoder from non-Gaussian noised images. Imaging Systems and Techniques (IST), IEEE International Conference on, 1-5.
15. Tissera, M. D. and McDonnell, M. D. (2015). Deep Extreme Learning Machines for Classification. Proceedings of ELM-2014, Springer International Publishing1, 345-354.
16. Iqbal, M. S. (2015). Unsupervised Multi-modal Learning. Advances in Artificial Intelligence, Springer International Publishing, 343-346.
17. Bengio, Y. (2009). Learning deep architectures for AI. Foundations and trends® in Machine Learning, 2(1), 1-127.
18. Huang, D. Z., Gong, R. X., and Gong, S. (2014). Prediction of wind power on chaos theory BP Artificial Neural Networks approach based on Genetic Algorithm. Journal of Electrical Engineering and Technology,. 742-747.
19. Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. Cognitive science, 9(1), 147-169.
20. Raina, R., Madhavan, A., and Ng, A. Y. (2009, June). Large-scale deep unsupervised learning using graphics processors. Proceedings of the 26th annual international conference on machine learning, 873-880.
21. An, H., Shim, H. M., Na, S. I., and Lee, S. (2015). Split and merge algorithm for deep learning and its application for additional classes. Pattern Recognition Letters, 65, 137-144.
22. Salakhutdinov, R. and Hinton, G. (2010). Training a deep autoencoder or a classifier on MNIST digits. http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html Last accessed :31/3/2015.
23. LeCun, Y., Cortes, C, and Burges, C. J. C. (1998). The MINIST database of handwritten digits. Downloaded http://yann.lecun.com/exdb/mnist, Last accessed : 31/3/2015.
24. Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. Neural computation, 14(8), 1771-1800.
25. Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. Signal Processing Magazine, 29(6), 82-97.
26. Moreno-orres, J. G., Llorà, X., Goldberg, D. E., and Bhargava, R. (2013). Repairing fractures between data using genetic programming-based feature extraction: A case study in cancer diagnosis. Information Sciences, 222, 805-823.
27. Whitley, D. (1994). A genetic algorithm tutorial. Statistics and Computing, 4(2), 65-85.
28. Salakhutdinov, R. (2009). Learning deep generative models. PhD Thesis, Department of Computer Science, University of Toronto, Canda.
29. Palangi, H., Deng, L., and Ward, R. K. (2013). Learning input and recurrent weight matrices in echo state networks. arXiv preprint arXiv:1311.2987: http://arxiv.org/pdf/1311.2987.pdf.Last accessed in 28/4/2016.