Journal of Computer Science

Original Research Paper

Cascade-Based Non-Linear Feedforward Neural Networks for Bi-Directional Memory

[1]Thipendra Pal Singh, [1]Tanupriya Choudhury and [2]Rohini A

[1]*School of Computing Science, University of Petroleum and Energy Studies, Dehradun, 248007, India*
[2]*Department of Computer Science and Engineering, Anil Neerukonda Institute of Technology and Science, Andhra Pradesh, 531162, India*

Article history
Received: 05-12-2021
Revised: 12-12-2022
Accepted: 14-12-2022

Corresponding Author:
Tanupriya Choudhury
School of Computing Science,
University of Petroleum and
Energy Studies, Dehradun,
248007, India
Email: tanupriya@ddn.upes.ac.in

Abstract: In today's world, efficient computation is the key to success in many fields. Pattern association plays an influential role in many areas of life, such as learning and memory. In complex dynamics, bidirectional associative memory has been effectively demonstrated by neural networks. However, these neural networks face challenges in terms of performance, such as computational time. A random bipolar input pattern and output pattern of a matrix with different sizes were used to analyze the background of this study. In order to address the challenges of bidirectional associative memory, nonlinear memory association is considered to be the most feasible method. In this study, we present a cascade-based non-linear feedforward neural network that performs pattern association in two passes and behaves like a Bayesian algorithm. A random pattern and English alphabets with different patterns have been used to validate the results of this approach. Using the experimental results, the study evaluated BAM's equivalent performance, pattern association, and stability.

Keywords: Cascade Feed Forward Neural Network, Bidirectional Associative Memory, Memory Association, Pattern Recognition

Introduction

For a neural network, the challenge is to design the machines as well as to apply principles that can be simplified and abstracted from neurobiological studies of the brain. These principles can be applied to similar calculations. Various types of connections must be distinguished in neural networks. The study used a feedforward graph without loops. In order to simplify this problem, the memory unit determines the output of the network based on its current input. The input and output are linked using linear and nonlinear relationships. There have been few studies focusing on feedforward and feed backward networks. Feedforward neural networks are primarily applied to pattern classification and pattern mapping. In this study, dimensionality reduction of input data was followed by pattern recognition from memory. Several properties with complex dynamics have been dealt with using nonlinear dimensionality reduction. Additionally, this neural network approach can solve high-dimensional binary association memory problems. Using delta learning rules and its variants, neural networks are trained to overcome these approaches. To distinguish between the variations of algorithms, need to assign weights and biases to the input images. ConvNets requires much less preprocessing. The convolved feature is less dimensional than the input, while the other is more dimensional. Convolved features are reduced in spatial size by the pooling layer. The maximum value from the kernel region is returned by max pooling. Nonlinear functions can be parallelized and distributed in a variety of ways. An efficient and accurate method of implementing bidirectional memory is to use Cascade-based nonlinear Feedforward Neural Networks (CFNNs). Compared to traditional computer systems, neural networks are more efficient when it comes to storing, recalling, and processing information.

A feedback neural network is a type of recurrent neural network. Through these networks, auto-associations and hetero-associations can be performed. This study examined the various input parameters involved in the cascade feedforward neural network model. For generalizing the extracted features to be used by the convolutional filters and recognizing the location in the image. Using the output of this layer, the delta rule recalculates the input weights for neurons associated with target activation. The feed forward NN accepts multiple neurons. In accordance with the model, it determines the output of the closest approximation of values.

The development of hidden layers for pattern association and bi-directional associative memory is referred to as the development of the hidden layer. It is shown in Table 1 that the number of nodes has been used in the layers of pattern using a logistic approach.

The development of a nonlinear continuous BAM in two phases has been proposed. The study characterized different phases of pattern sizes. A back propagation learning algorithm is used to analyze the pattern in the initial phase. An output layer has been associated with the forward direction after the network has been trained. Weights for forward connections are updated during training. The output pattern of the first phase is fed into the second phase. Again, with the help of the back propagation learning algorithm the network is trained, but this time in the feedback direction. As a result, the weights associated with backward connections are updated. The two phases, forward-pass, and backward-pass are implemented one after the other with asymmetric connection weights. Consequently, the weights passed in subsequent iterations of bi-directional training are recurrent. To minimize the squared error, forward and backward weights are simultaneously converged. Interconnection weights in both directions are convergent, indicating the stability of the network

Due to bidirectional associative memory, the presented pairs of the pattern will be associated both directly and indirectly. When input-output pattern pairs are associated non-linearly, storage capacity will be increase. Thus, when recalling occurs, spurious states will also be reduced. As a result of convergence, the layers can accept the nonlinear relationship of input to output without eliminating the linear relations. The simulation results show better performance and analysis of the patterns.

Review of Literature

Using recurrent neural networks with symmetric weights, bidirectional associative memories have been extensively explored in neural networks. Researchers have used several approaches to improve Kosko's original BAM (Wu and Wang, 2021). Forward and backward associative pairs of memory were discussed in the asymmetric connection of weigts (Acevedo-Mosqueda *et al*., 2013; Brachmann and Redies, 2016). Repeated the training model in the BAM multiple times with the same input-output pattern in order to optimize the correlation matrix weights (Wu *et al*., 2018; Zhang *et al*., 2020). Orthonormal vectors are obtained by adding supplementary patterns to existing patterns or by codifying them. In some studies, the learning rule of threshold has been proposed as a method to improve the recall efficiency of Kosko's BAM. The design of BAMs for the learning and recalling phases was worked on by some authors. Feedforward neural networks are applied in the pattern classification and mapping (Çakar and Cil, 2004; Rafiq *et al*., 2001). Two-layer feedforward BAMs were transformed into three-layer feedforward BAMs by delta rule learning algorithms. The recall is achieved by using only the hidden layer. A layer of input and output is used to review the development of feedforward, while a hidden layer recalls previously stored patterns. For each pair of stored in and out patterns, a fixed point of stability is shown. Additionally, two hidden layers have been considered. Input-output patterns with varying sizes are applied to train the network (Shieh and Yeh 2013; Van Nguyen *et al*., 2015). Several factors affect the performance of a feedforward network, including its architecture and algorithms. An appropriate feedforward architecture and fast learning algorithm are difficult to implement in nonlinear systems. A Cascade Feedforward Neural Network (CFNN) accommodates nonlinear relationships between inputs and outputs by removing layer relationships (Krippendorf and Syvaeri, 2020). Studies have shown that cascade feedforward neural networks are more powerful than most feedforward neural networks. As the number of layers increases, the network's power increases significantly (Kosko, 1987). With the same number of nodes and learning algorithm, cascade feedforward neural networks converge faster than feedforward neural networks (Kosko, 1988; Li *et al*., 2021; Chen *et al*., 2019; Rahman *et al*., 2021). Therefore, we can study the behavior of this newly developed BAM model with CFNN. Cohen and Grossberg (1983) theorem have been applied to analyze the study and derive the appropriate function for symmetric unidirectional associators. The neural network model of Hopfield has been enhanced to include BAMs. Bidirectional associative memory can be approached effectively using a novel method based on nonlinearity that analyzes random bipolar input and output patterns in a matrix. Using a feedforward and deep learning neural network model, (Pulickal, 2022; Jepkoech *et al*., 2022; Aburass *et al*., 2022) constructed practices in two passes and showed similarity across different applications. Based on a similarity index and the mean squared error of model behavior, (Panthong, 2022) calculated the peak signal-to-noise ratio. The results revealed existing methods for determining reliability. Li (2009a-b); Wang *et al*. (2018); Ge *et al*. (2013) has been discussed the neural networks with different application and shows the analysis in different resolutions, the drawback of those papers was not defined the correlation between the epochs. The regression algorithm has been used in the relationship between variables in the patterns to predicted values (Lai *et al*., 2005). Ozcan *et al*. (2019) Multiple delays has derived lyapunov functional using the criterion in independent delay of time series parameters. It leads the stable result in neutral systems. Shi and Zeng (2020) By Associative rule memory the weights leads the frequency of neurons. Shieh and Yeh (2013) Stability of the linear processing has achieved by the dimensionality reduction using principal component analysis. Wang and Chen (2020) By deep learning techniques parameters were accessed consistency of the results were statistically proven in the undistinguishable product.

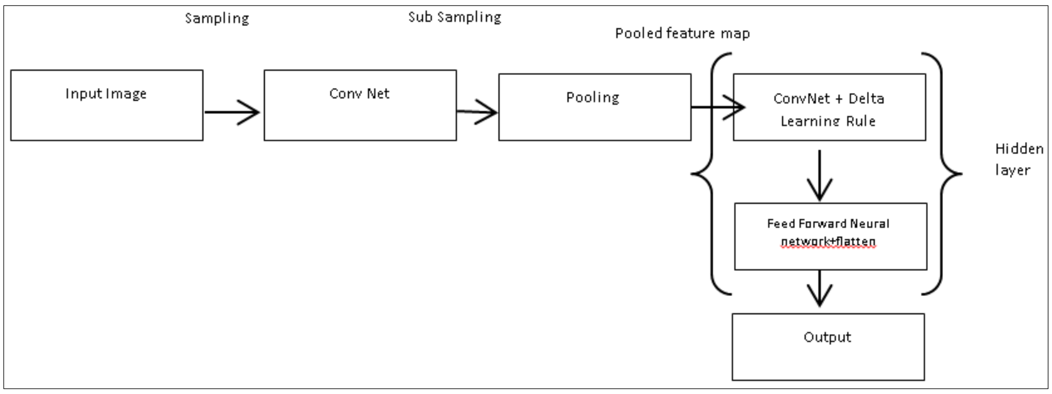


Fig. 1: Block diagram of an analysis of feature extraction through pattern recognition of CFNN-BAM

Table 1: Various parameters in the model under study

No. of hidden layers	2
No. of nodes in input layer	N
No. of nodes in output layer	M
No. of nodes in first hidden layer	13
No. of nodes in second hidden layer	7
Output function at input layer	Linear
Output function at output layer	Linear
Output functions at hidden layers	Non-linear (logistic)
Input-output patterns	Random/fixed

Design of the Proposed Network Model

Table 1, bidirectional associative memory is based on CFNN with two hidden layers. The cascade feedforward neural network is an extension of multilayer feedforward neural networks in which every layer is connected to every previous layer, including the input layer. With a three-layer CFNN, the output layer is connected to the hidden layer as well as directly to the input layer. Input-output relationships can be learned by these networks when they contain a sufficient number of hidden nodes. Figure 1 shows the CFNN's architecture.

The bipolar patterns associated with L are represented as follows:

$$\{(X_1, Y_1), (X_2, Y_2), \ldots (X_L, Y_L)\}$$

where, $X_i \in \{-1, 1\}_N$ and $Y_i \in \{-1, 1\}_M$ for $i = 1,2,3\ldots n$.

Phase-1 starts with the presentation of one input pattern from the associated patterns set, l^{th} pattern performs forward direction at any discrete point of time 't' in training, as:

$$X_i^l(t) = \{X_1^l(t), X_2^l(t), \ldots X_N^l(t)\}$$

The CFNN now obtains the desired mapping corresponding to the input pattern, as output, using the standard BP-learning rule:

$$Y_i^l(t+1) = \{Y_1^l(t+1), Y_2^l(t+1), \ldots Y_N^l(t+1)\}$$

The forward instantaneous square error is written as follows:

$$E_F^l = \frac{1}{2}\sum_{j=1}^{J}\{y_j^l(t+1) - a_j^l(t+1)\}$$

where:

$$y_j^l(t+1) = \sum f^i w_{ji} x_i + \sum f^{h1} w_{jh1} z_{h1} + f^o\left(\sum w_{jh2}\left(f^{h2}\left(\sum f^{h2} w_{h2i} x_i + \sum w_{h2h1} f^{h1}\left(\sum w_{h1i} x_i\right)\right)\right)\right)$$

During this, the update in weight vectors W^F in the forward direction takes place as follows:

a) Change in weight vector between hidden layer 2 and output is guided by:

$$\Delta w_{h2j}^F = -\eta_j \frac{\partial E_F}{\partial w_{h2j}^F}$$

Now:

$$\frac{\partial E_F}{\partial w_{h2j}^F} = \frac{\partial E_F}{\partial f^o(y_j^l)} \cdot \frac{\partial f^o(y_j^l)}{\partial y_j^l} \cdot \frac{\partial y_j^l}{\partial w_{h2j}^F}$$

$$\frac{\partial E_F}{\partial f^o(y_j^l)} = -(d_j^l - f^o(y_j^l) = -e_j^l$$

$$\frac{\partial f^o(y_j^l)}{\partial y_j^l} = f^{o\prime}(y_j^l) = f^o(y_j^l)(1 - f^o(y_j^l))$$

$$\frac{\partial y_j^l}{\partial w_{h2j}^F} = f^{h2}(Z_{h2}^l)$$

$$\frac{\partial E_F}{\partial w_{h2j}^F} = -e_j^l \cdot f^{o'}(y_j^l) \cdot f^{h2}(Z_{h2}^l)$$

$$\frac{\partial E_F}{\partial w_{h2j}^F} = -\delta_j^l f^{h2}(Z_{h2}^k)$$

Here, $\delta_j^l = e_j^l \cdot f^{o'}(y_j^l)$ represents an error and signal slope product.

Therefore:

$$\Delta w_{h2j}^F = \eta_j \delta_j^l f^{h2}(Z_{h2}^k) \tag{1}$$

b) Similarly, weight vector has changed in between the hidden layer 1 and 2 is guided by:

$$\Delta w^F_{h1h2} = -\eta_{h2}\frac{\partial E_F}{\partial w^F_{h1h2}}$$

Now:

$$\frac{\partial E_F}{\partial w^F_{h1h2}} = \frac{\partial E_F}{\partial f^{h2}\left(Z^l_{h2}\right)}\cdot\frac{\partial f^{h2}\left(Z^l_{h2}\right)}{\partial Z^l_{h2}}\cdot\frac{\partial Z^l_{h2}}{\partial w^F_{h1h2}}$$

$$\frac{\partial E_F}{\partial f^{h2}\left(Z^l_{h2}\right)} = \sum_{j=1}^{M}\left\{\frac{\partial E_F}{\partial y^l_j}\cdot\frac{\partial y^l_j}{\partial f^{h2}\left(Z^l_{h2}\right)}\right\}$$

$$\frac{\partial E_F}{\partial w^F_{h1h2}} = \sum_{j=1}^{M}\left\{\frac{\partial E_F}{\partial y^l_j}\cdot\frac{\partial y^l_j}{\partial f^{h2}\left(Z^l_{h2}\right)}\right\}\cdot f^{h2'}\left(Z^l_{h2}\right)\cdot f^{h1}\left(Z^l_{h1}\right)$$

$$\frac{\partial E_F}{\partial w^F_{h1h2}} = \sum_{j=1}^{M}\left\{\frac{\partial E_F}{\partial f^{o}\left(y^l_j\right)}\cdot\frac{\partial f^{o}\left(y^l_j\right)}{\partial y^l_j}\cdot\frac{\partial y^l_j}{f^{h2}\left(Z^l_{h2}\right)}\right\}\cdot f^{h2'}\left(Z^l_{h2}\right).f^{h1}\left(Z^l_{h1}\right)$$

$$\frac{\partial E_F}{\partial w^F_{h1h2}} = \sum_{j=1}^{M}\left\{-e^l_j.f^{o'}\left(y^l_j\right).w^F_{h1h2}\right\}.f^{h2'}\left(Z^l_{h2}\right).f^{h1}\left(Z^l_{h1}\right)$$

$$\frac{\partial E_F}{\partial w^F_{h1h2}} = -\sum_{j=1}^{M}\left\{\delta^l_j\cdot w^F_{h1h2}\right\}.f^{h2'}\left(Z^l_{h2}\right).f^{h1}\left(Z^l_{h1}\right)$$

$$\frac{\partial E_F}{\partial w^F_{h1h2}} = -\delta^k_{h2}.f^{h2'}\left(Z^l_{h2}\right).f^{h1}\left(Z^l_{h1}\right)$$

$$e^l_{h2} = \sum\nolimits_{j=1}^{M}\left\{\delta^l_j.w^F_{h1h2}\right\} \, and \, \delta^l_{h2} = e^l_{h2}.f^{h2'}\left(Z^l_{h2}\right)$$

Therefore:

$$\frac{\partial E_F}{\partial w^F_{h1h2}} = -\delta^l_{h2}.f^{h1}\left(Z^l_{h1}\right)$$

Hence:

$$\Delta w^F_{h1h2} = \eta_{h2}\delta^k_{h2}.f^{h1}\left(Z^l_{h1}\right) \tag{2}$$

c) Finally, weight changes between input and hidden layer 1 are represented as:

$$\Delta w^F_{ih1} = -\eta_{h1}\frac{\partial E_F}{\partial w^F_{ih1}}$$

Now:

$$\frac{\partial E_F}{\partial w^F_{ih1}} = \frac{\partial E_F}{\partial f^{h1}\left(Z^l_{h1}\right)}\cdot\frac{\partial f^{h1}\left(Z^l_{h1}\right)}{\partial Z^l_{h1}}\cdot\frac{\partial Z^l_{h1}}{\partial w^F_{ih1}}$$

$$\frac{\partial E_F}{\partial f^{h1}\left(Z^l_{h1}\right)} = \sum_{j=1}^{M}\left\{\frac{\partial E_F}{\partial y^l_j}.\frac{\partial y^l_j}{\partial f^{h2}\left(Z^l_{h2}\right)}.\frac{\partial f^{h2}\left(Z^l_{h2}\right)}{\partial f^{h1}\left(Z^l_{h1}\right)}\right\}$$

$$\frac{\partial E_F}{\partial w^F_{ih1}} = \sum_{j=1}^{M}\left\{\frac{\partial E_F}{\partial y^l_j}.\frac{\partial y^l_j}{\partial f^{h2}\left(Z^l_{h2}\right)}.\frac{\partial f^{h2}}{\partial f^{h1}\left(Z^l_{h1}\right)}\right\}.f^{h1}\left(Z^l_{h1}\right).f^{i}\left(x^l_i\right)$$

$$\frac{\partial E_F}{\partial w^F_{ih1}} = \sum_{j=1}^{M}\left\{-e^l_j.f^{o'}\left(y^l_j\right).\frac{\partial y^l_j}{\partial f^{h2}\left(Z^l_{h2}\right)}.\frac{\partial f^{h2}\left(Z^l_{h2}\right)}{\partial f^{h1}\left(Z^l_{h1}\right)}\right\}.f^{h1'}\left(Z^l_{h1}\right).f^{i}\left(x^l_i\right)$$

$$\frac{\partial E_F}{\partial w^F_{ih1}} = \sum_{j=1}^{M}\left\{-e^l_j.f^{o'}\left(y^l_j\right).w^F_{ih1}\right\}.f^{h1,}\left(Z^l_{h1}\right).f^{i}\left(x^l_i\right)$$

$$e^l_{h1} = \sum\nolimits_{j=1}^{M}\left\{\delta^l_j.w^F_{ih1}\right\} \, and \, \delta^l_{h1} = e^l_{h1}.f^{h1,}\left(Z^l_{h1}\right)$$

Therefore:

$$\frac{\partial E_F}{\partial w^F_{ih1}} = -\delta^l_{h1}.f^{i}\left(x^l_i\right)$$

Hence:

$$\Delta w^F_{ih1} = \eta_{h1}\delta^l_{h1}.f^{i}\left(x^l_i\right) \tag{3}$$

Therefore, phase-1 gets completed with the update in forward weight vectors. During this, the implicit relationship between the input of $[-1, 1]_N$ to the output of $[-1, 1]_M$ spaces.

The average Mean Squared Error (MSE) for all the pattern of 'L' for flattened is calculated and minimized by chain operations in the Eq. (1-3).

By the next y^l_j has represented in the pairs of associated patterns for backward pass to the time as:

$$Y^l_{i\,(t+2)} = \left\{Y^l_1\left(t+2\right), Y^l_2\left(t+2\right), ...Y^l_M\left(t+2\right)\right\}$$

Now using standard BP-learning rule, the CFNN obtains the desired mapping corresponding to the input pattern, as output:

$$X^l_{i\,(t+3)} = \left\{X^l_1\left(t+3\right), X^l_2\left(t+3\right), ...X^l_N\left(t+3\right)\right\}$$

The backward instantaneous square error can be represented as follows:

$$E^l_B = \frac{1}{2}\sum_{i=1}^{J}\left\{x^l_i\left(t+3\right) - b^l_i\left(t+3\right)\right\}$$

where:

$$x^l_i = \sum f^{oi}w_{ij}y_j + \sum f^{h2i}w_{h2i}z_{h2} + f^{i}\left(\sum w_{ih1}\left(f^{h1}\left(\sum f^{oh1}w_{h1j}y_j + \sum w_{h1h2}f^{h2}\left(\sum w_{h2j}y_j\right)\right)\right)\right)$$

During the process the backward weight vectors W^B between the different layers takes place as follows:

a) The weight vector has updated between the '*HI*' and '*I*' for backward pass is guided by:

$$\Delta w^B_{h1i} = -\eta_1\frac{\partial E_B}{\partial w^B_{h1i}}$$

Now:

$$\frac{\partial E_B}{\partial w_{h1i}^B} = \frac{\partial E_B}{\partial f^i\left(x_i^l\right)} \cdot \frac{\partial f^i\left(x_i^l\right)}{\partial x_i^l} \cdot \frac{\partial x_i^l}{\partial w_{h1i}^B}$$

$$\frac{\partial E_B}{\partial f^i\left(x_i^l\right)} = -\left(d_i^l - f^i\left(x_i^l\right)\right) = -e_i^l$$

$$\frac{\partial f^i\left(x_i^l\right)}{\partial x_i^l} = f^{i'}\left(x_i^l\right) = f^i\left(x_i^l\right)\left(1 - f^i\left(x_i^l\right)\right)$$

$$\frac{\partial x_i^l}{\partial w_{h1i}^B} = f^{h1}\left(z_{h1}^l\right)$$

$$\frac{\partial E_B}{\partial w_{h1i}^B} = -e_i^l . f^{i'}\left(x_i^l\right) . f^{h1}\left(z_{h1}^l\right)$$

$$\frac{\partial E_B}{\partial w_{h1i}^B} = -\delta_i^l f^{i'} . \left(x_i^l\right)$$

Therefore:

$$\Delta w_{h1i}^B = \eta_i \, \delta_i^l f^{i'} . \left(x_i^l\right) \tag{4}$$

b) The updating weight vector between hidden layer 2 and 1 for backward pass is guided by:

$$\Delta w_{h2h1}^B = -\eta_{h1} \frac{\partial E_B}{\partial w_{h2h1}^B}$$

Now:

$$\frac{\partial E_B}{\partial w_{h2h1}^B} = \frac{\partial E_B}{\partial f^{h1}\left(z_{h1}^l\right)} \cdot \frac{\partial f^{h1}\left(z_{h1}^l\right)}{\partial z_{h1}^l} \cdot \frac{\partial z_{h1}^l}{\partial w_{h2h1}^B}$$

$$\frac{\partial E_B}{\partial f^{h1}\left(z_{h1}^l\right)} = \sum_{i=1}^{N} \left\{ \frac{\partial E_B}{\partial x_i^l} \cdot \frac{\partial x_i^l}{\partial f^{h1}\left(z_{h1}^l\right)} \right\}$$

$$\frac{\partial E_B}{\partial w_{h2h1}^B} = \sum_{i=1}^{N} \left\{ \frac{\partial E_B}{\partial x_i^l} \cdot \frac{\partial x_i^l}{\partial f^{h1}\left(z_{h1}^l\right)} \right\} . f^{h1'}\left(z_{h1}^l\right) . f^{h2}\left(z_{h2}^l\right)$$

$$\frac{\partial E_B}{\partial w_{h2h1}^B} = \sum_{i=1}^{N} \left\{ \frac{\partial E_B}{\partial f^i\left(x_i^l\right)} \cdot \frac{\partial f^i\left(x_i^l\right)}{\partial x_i^l} \cdot \frac{\partial x_i^l}{f^{h1}\left(z_{h1}^l\right)} \right\} . f^{h1'}\left(z_{h1}^l\right) . f^{h2}\left(z_{h2}^l\right)$$

$$\frac{\partial E_B}{\partial w_{h2h1}^B} = \sum_{i=1}^{N} \left\{ -e_i^l . f^{i'}\left(x_i^l\right) . w_{h2h1}^l \right\} . f^{h1}\left(z_{h1}^l\right) . f^{h2}\left(z_{h2}^l\right)$$

$$\frac{\partial E_B}{\partial w_{h2h1}^B} = -\sum_{i=1}^{N} \left\{ \delta_i^l \cdot w_{h2h1}^B \right\} . f^{h1}\left(z_{h1}^l\right) . f^{h2}\left(z_{h2}^l\right)$$

$$\frac{\partial E_B}{\partial w_{h2h1}^B} = -\delta_{h1}^l . f^{h1'}\left(z_{h1}^l\right) . f^{h2}\left(z_{h2}^l\right)$$

$$e_{h1}^l = \sum\nolimits_{i=1}^{N} \left\{ \delta_i^l . w_{h2h1}^B \right\} and \; \delta_{h1}^l = -e_{h1}^l . f^{h1'}\left(z_{h1}^l\right)$$

Therefore:

$$\frac{\partial E_B}{\partial w_{h2h1}^B} = -\delta_{h1}^l . f^{h2}\left(z_{h2}^l\right)$$

Consequently, the weight change can be formulated as:

$$\Delta w_{h2h1}^B = \eta_{h1} \delta_{h1}^l . f^{h2}\left(z_{h2}^l\right) \tag{5}$$

A neural network's backward pass update should be guided by a weight vector, but there have been debates about how to guide such an update. The error between hidden layer 2 and the output layer is taken into account by some researchers. Others take into account the error between hidden layer 1 and the output layer. Nevertheless, we believe the weight vector should be updated based on the sum of squared errors between output and all other layers, which can be expressed as:

$$\Delta w_{jh2}^B = -\eta_{h2} \frac{\partial E_B}{\partial w_{jh2}^B}$$

Now:

$$\frac{\partial E_B}{\partial w_{jh2}^B} = \frac{\partial E_B}{\partial f^o\left(z_{h2}^l\right)} \cdot \frac{\partial f^o\left(z_{h2}^l\right)}{\partial z_{h2}^l} \cdot \frac{\partial z_{h2}^l}{\partial w_{jh2}^B}$$

$$\frac{\partial E_B}{\partial f^o\left(z_{h2}^l\right)} = \sum_{i=1}^{N} \left\{ \frac{\partial E_B}{\partial x_i^l} \cdot \frac{\partial x_i^l}{\partial f^{h1}\left(z_{h1}^l\right)} \cdot \frac{\partial f^{h1}\left(z_{h1}^l\right)}{\partial f^{h2}\left(z_{h2}^l\right)} \cdot \frac{\partial f^{h2}\left(z_{h2}^l\right)}{\partial f^o\left(z_{h2}^l\right)} \right\}$$

$$\frac{\partial E_B}{\partial w_{jh2}^B} = \sum\nolimits_{i=1}^{N} \left\{ \frac{\partial E_B}{\partial x_i^l} \cdot \frac{\partial x_i^l}{\partial f^{h1}\left(z_{h1}^l\right)} \cdot \frac{\partial f^{h1}\left(z_{h1}^l\right)}{\partial f^{h2}\left(z_{h2}^l\right)} \cdot \frac{\partial f^{h2}\left(z_{h2}^l\right)}{\partial f^o\left(z_{h2}^l\right)} \right\} . f^{h2'}\left(z_{h2}^l\right) . f^o\left(y_j^l\right)$$

$$\frac{\partial E_B}{\partial w_{jh2}^B} = \sum\nolimits_{i=1}^{N} \left\{ \frac{\partial E_B}{\partial f^i\left(z_i^l\right)} \cdot \frac{\partial f^i\left(x_i^l\right)}{\partial x_i^l} \cdot \frac{\partial x_i^l}{\partial f^{h1}\left(z_{h1}^l\right)} \cdot \frac{\partial f^{h1}\left(z_{h1}^l\right)}{\partial f^{h2}\left(z_{h2}^l\right)} \cdot \frac{\partial f^{h2}\left(z_{h2}^l\right)}{\partial f^o\left(z_{h2}^l\right)} \right\}$$
$$. f^{h2'}\left(z_{h2}^l\right) . f^o\left(y_j^l\right)$$

$$\frac{\partial E_B}{\partial w_{jh2}^B} = \sum_{i=1}^{N} \left\{ -e_i^l . f^{i'}\left(x_i^l\right) . \frac{\partial x_i^l}{\partial f^{h1}\left(z_{h1}^l\right)} \cdot \frac{\partial f^{h1}\left(z_{h1}^l\right)}{\partial f^{h2}\left(z_{h2}^l\right)} \right\} . f^{h2'}\left(z_{h2}^l\right) . f^o\left(y_j^l\right)$$

$$\frac{\partial E_B}{\partial w_{jh2}^B} = \sum_{i=1}^{N} \left\{ -e_i^l . f^{i'}\left(x_i^l\right) . w_{jh2}^B \right\} . f^{h2'}\left(z_{h2}^l\right) . f^o\left(y_j^l\right)$$

$$e_{h2}^l = \sum\nolimits_{i=1}^{N} \left\{ \delta_i^l . w_{jh2}^B \right\} and \; \delta_{h2}^l = -e_{h2}^l . f^{h2'}\left(z_{h2}^l\right)$$

Therefore:

$$\frac{\partial E_B}{\partial w_{jh2}^B} = -\delta_{h2}^l . f^o\left(z_j^l\right)$$

which leads to following weight change expression:

$$\Delta w_{jh2}^B = \eta_{h2} \delta_{h2}^l . f^o\left(z_j^l\right) \tag{6}$$

Thus, following phase 2, the backward weights are iterated and captured the function of implicit mapping to the input space $[*1, 1]_M$ to output space $[*1, 1]_N$. L patterns of average Mean Squared Error (MSE) for backward phase is calculated and minimize by chain operations of Eq. (4-6).

Finally, total (MSE) E_{total}^l for both the phases of forward and backward can be written with the expression:

$$E^l_{total} = \frac{1}{2}\left(E^l_F + E^l_B\right) \tag{7}$$

Now, in Eq. (7) the network is tuned for both the phases simultaneously, to minimize the MSE E^l_{total} by providing both patterns, *X* and *Y*, from the associated pattern pair (*X*, *Y*). Based on above equations the change in weight vectors during the tuning process can be represented as:

$$\Delta w_1 = -\frac{\partial^2 E^l_{total}}{\partial w_1^2} = -\frac{\partial E^l_{total}}{\partial w_1^F}.\frac{\partial E^l_{total}}{\partial w_1^B} = -\frac{\partial E^l_{total}}{\partial w^F_{h2j}}.\frac{\partial E^l_{total}}{\partial w^B_{h1i}} \tag{8}$$

and:

$$\Delta w_2 = -\frac{\partial^2 E^l_{total}}{\partial w_2^2} = -\frac{\partial E^l_{total}}{\partial w_2^F}.\frac{\partial E^l_{total}}{\partial w_2^B} = -\frac{\partial E^l_{total}}{\partial w^F_{h1h2}}.\frac{\partial E^l_{total}}{\partial w^B_{h2h1}} \tag{9}$$

Finally:

$$\Delta w_3 = -\frac{\partial^2 E^l_{total}}{\partial w_3^2} = -\frac{\partial E^l_{total}}{\partial w_3^F}.\frac{\partial E^l_{total}}{\partial w_3^B} = -\frac{\partial E^l_{total}}{\partial w^F_{ih1}}.\frac{\partial E^l_{total}}{\partial w^B_{jh2}} \tag{10}$$

Based on Eq. (1-10) a proposed network model is optimized in the associated pattern of memories in the training set for consideration of input and output space.

Cascade Feed Forward Neural Network-Based Bidirectional Associative Memory (CFNN-BAM) Model Implementation

The proposed Cascade Feedforward Neural Network (CFNN-BAM) model is evaluated with MATLAB 2018R on Lenovo laptops. A CFNN-BAM model is proposed with two hidden layers, H_1 and H_2, each with 13 nodes. Input output patterns with varying sizes are applied to train the network. A total of n neurons has been used for input and *m* neurons have been used for output. Symmetric saturating linear functions are employed as transfer functions. A forward pass and a backward pass are included in the simulation. As part of the forward pass, the input pattern is used for back propagation training. As a result of training, the output pattern is mapped to the corresponding output pattern. As part of the backward pass, the output pattern i.e., is presented as input and the network is trained by using a back-propagation learning algorithm to map this pattern to its corresponding associated input pattern, which is used as output in this phase. In artificial intelligence, pattern recognition plays an important role. Identifying patterns in data is necessary for a computer to be able to learn. Neural networks are one of the most common ways to accomplish this. By adjusting their own connections and weights, neural networks allow computers to learn on their own. In this way, they can identify patterns that humans would otherwise find impossible or difficult to recognize. Pattern recognition involves instructing a computer to find patterns in data sets. Based on that pattern, the computer will create a model to identify similar patterns in new data sets. This process allows computers not only to recognize patterns that have been specifically programmed into them but also allows them to learn new ones as they encounter them.

The total MSE of both passes is calculated by training the network simultaneously from both sides. This process is repeated with the stopping criteria as either the convergence of network weights takes place (from both ends) or the maximum number of iterations is achieved. In this study, the maximum number of iterations has been determined to be 100. If the regression lines are linearly associated ($R = 1$), then the input pattern is appropriately associated with the output pattern; otherwise, there is an error in association or recall.

Results and Discussion

The simulations are divided into two sets one for random patterns, the other for English alphabets. Various parameters taken in these runs are summarized in the following Tables 2-3. It is performed 10 times in each category, i.e., on random patterns and on English alphabets, to determine the average MSE and performance.

A random bipolar input pattern matrix size of 5 × 3 and a pattern of output matrix of size 4 × 3 are presented in experiment 1 for training with the proposed algorithm. Results are presented in Fig. 2(a-b). The network's worst, best, and average performance is 0.76, 0.99, and 0.92, respectively. On average, 92.0% of the patterns are associated with their matched pattern pairs. It can be used in linearly separable and takes inputs to weights along with bias. In the second experiment, the randomly generated bipolar input pattern size of 25 and output pattern size of 12 are presented to the network for training. Its best overall performance is shown in Fig. 3(a) and regression in Fig. 3(b).

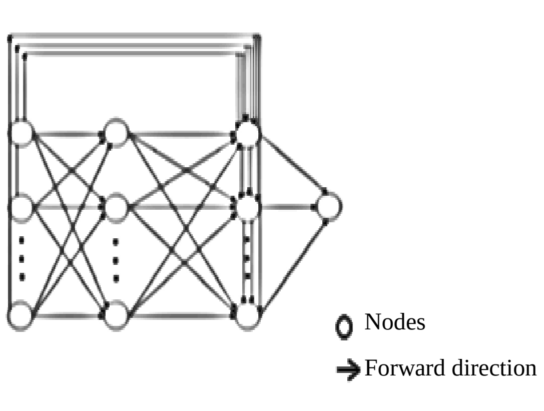


Fig. 2: Cascade feedforward neural network structure with one hidden layer

In terms of performance, the network achieved the worst performance, the best performance, and the average performance. As a result, 93% of pattern pairs are correctly associated with the network. Training the neural network in forward and backward passes together has been counted as one pass for each batch. The network does not form any loops. The hidden layers must be trained. As the validation progresses over three epochs, the generalization error is not caused by evaluation time and the model has started to increase accuracy on the performance of 9.6004e-19. An increase in epochs during the validation may result in a decrease in accuracy.

In experiment 3, the input pattern was 10 × 5 and the output pattern was 6 × 5. Figure 1 shows the regression and the best performance Fig. 4(a-b) respectively.

As associated memory, the network performs on average at 0.88, with the worst and best performances at 0.84 and 0.92, respectively. Over fitting is a critical issue on the data. In both the validation and test sets, we lose generalization capacity due to over fitting. It is necessary to keep track of epochs in order to identify any starting points that are over fitting. Due to the overfitting, the generalization performance has high on the epochs.

Simulation on English Alphabet Letters

Further, different experiments were conducted for the pattern recognition problem. For this problem, input patterns taken into consideration are capital English alphabets in the grid of 7 × 5 and their associated output patterns are small English alphabets in the grid of 5 × 5. Considering 26 English alphabet letters, the size of input pattern matrix becomes 35 × 26 and that of output pattern matrix becomes 25 × 26. The best overall performance of network as BAM is shown in Fig. 5(a). The worst, average, and best overall performance of the associative memory is 0.90, 0.95, and 0.98 respectively. The weight convergence for stability is represented in Fig. 5(b).

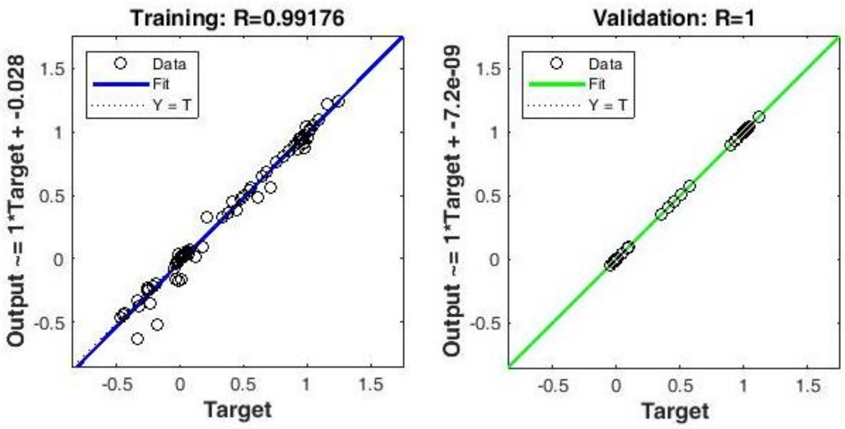
Secondly, the size of the input and output patterns of a pair needs to be considered. There are 26 alphabets, each with a pattern size of 5 × 5 for input and 7 × 5 for output. The associated pairs of performance are represented by 0.97 in Fig. 6(a). It captures patterns automatically and identifies them, even if they are partially hidden. The worst performance is 0.84 and the average performance is 0.92. Fig. 6(b) shows the convergence of weight.

Table 2: Summary of parameters of simulation on random patterns

	Input pattern size	Output pattern size	Pattern generation method
Experiment 1	5 × 3	4 × 3	Random
Experiment 2	5 × 5	4 × 3	Random
Experiment 3	10 × 5	6 × 5	Random

Table 3: Summary of parameters of simulation on English alphabet

	Input pattern size	Output pattern size	Pattern generation method
Experiment 1	7 × 5	5 × 5	Fixed
Experiment 2	5 × 5	7 × 5	Fixed
Experiment 3	9 × 5	6 × 5	Fixed



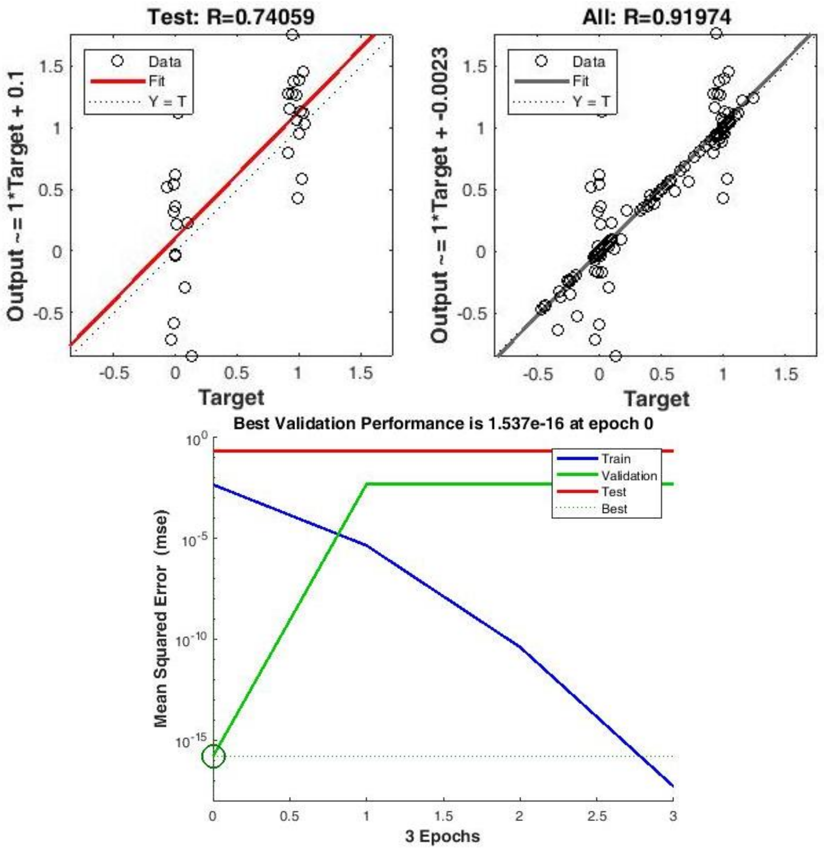
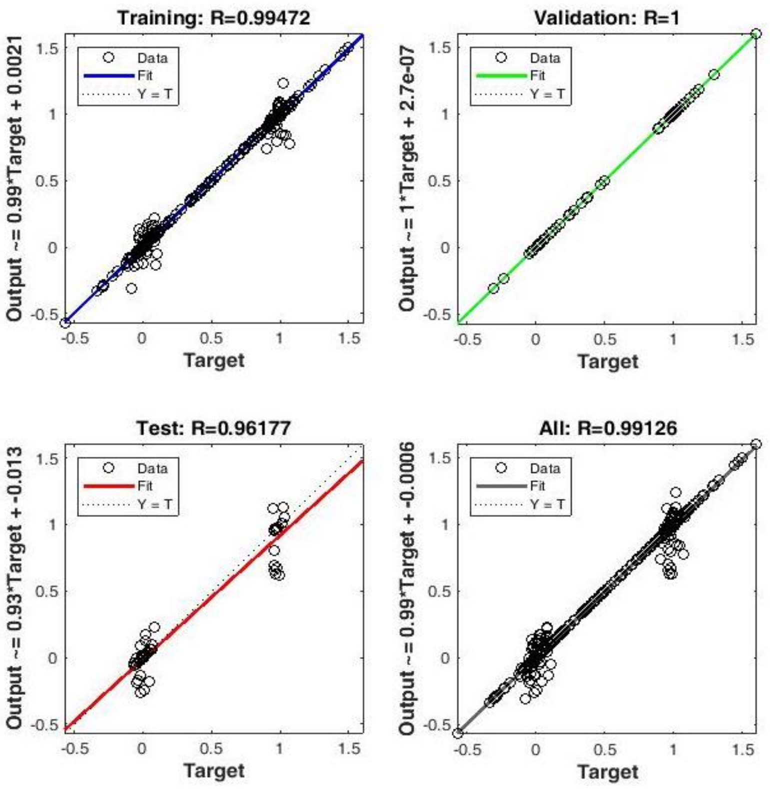


Fig. 3: (a) Regression plot and (b) performance of random pattern size of bipolar (input 5 × 3 and output 4 × 3)



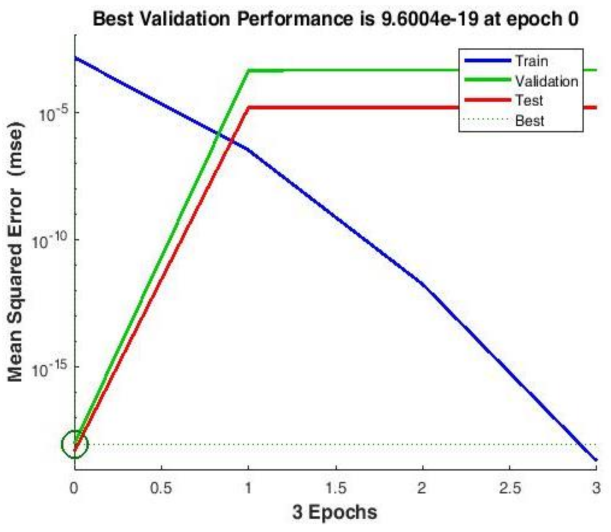
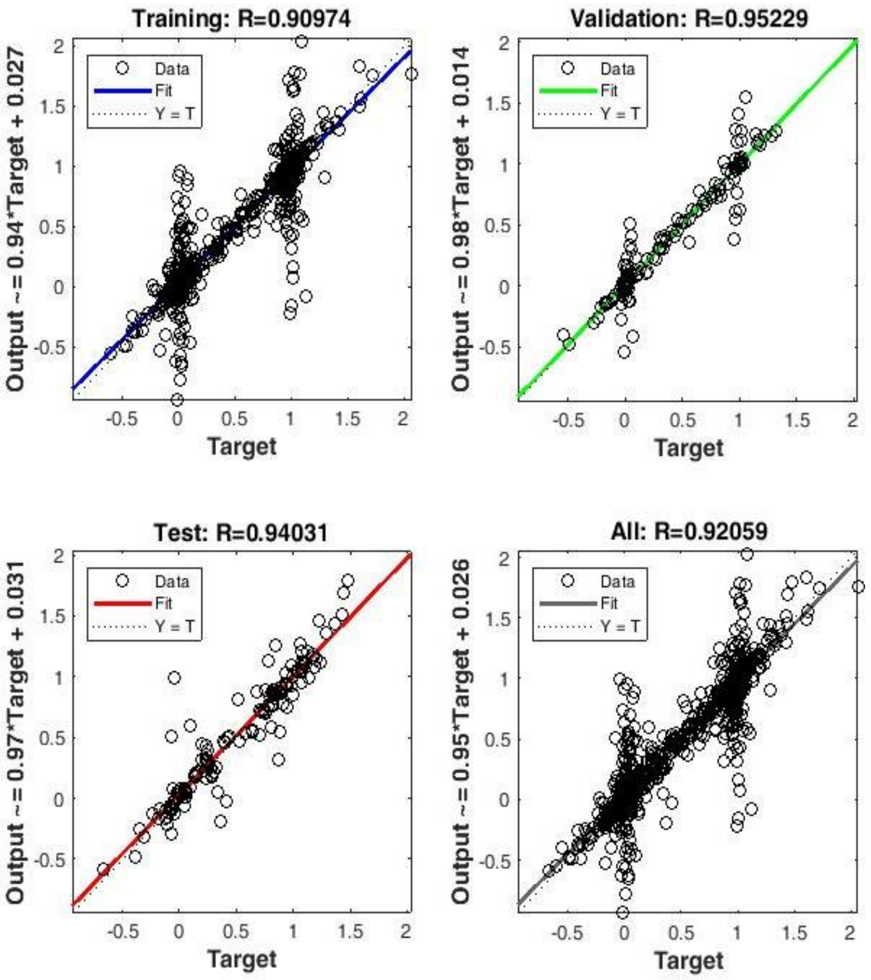


Fig. 4: (a) Regression plot and (b) performance of random pattern size of bipolar (input 5 × 5 and output 4 × 3)



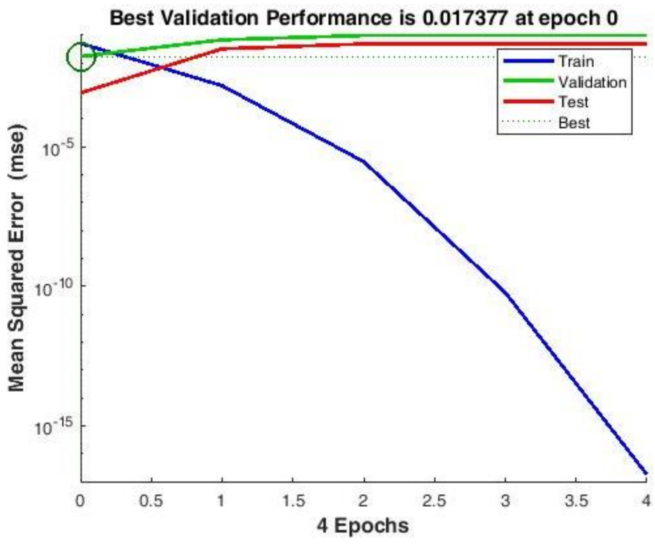
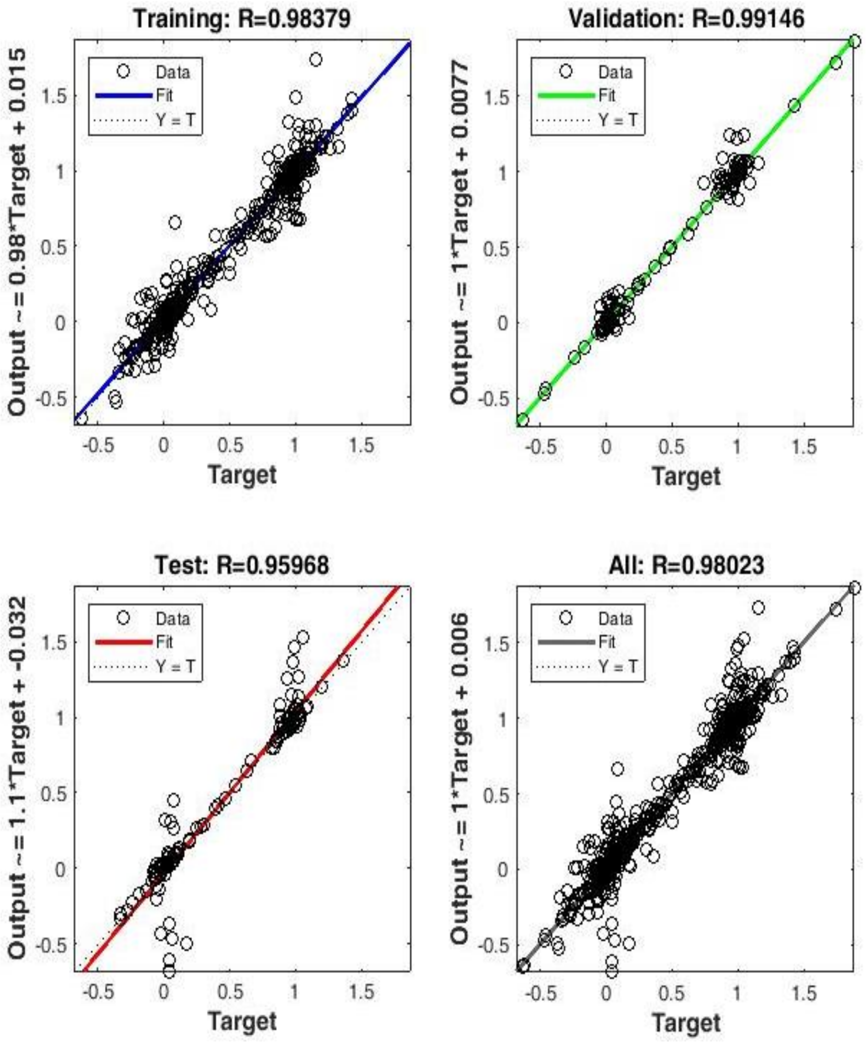


Fig. 5: (a) Regression and (b) bipolar random pattern size of performance of (input 10 × 5 and output 6 × 5)



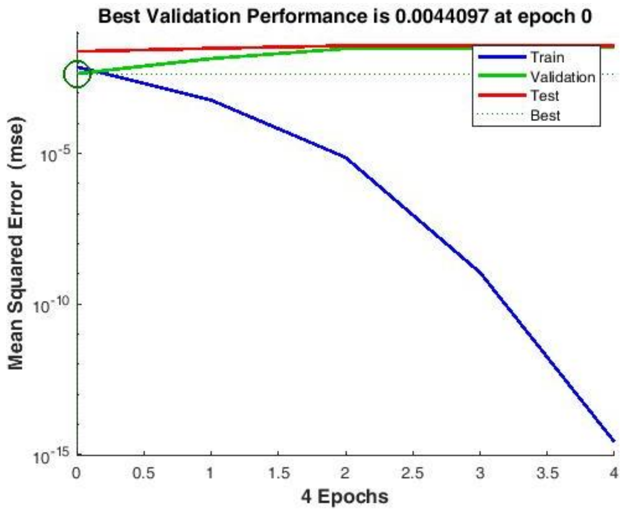
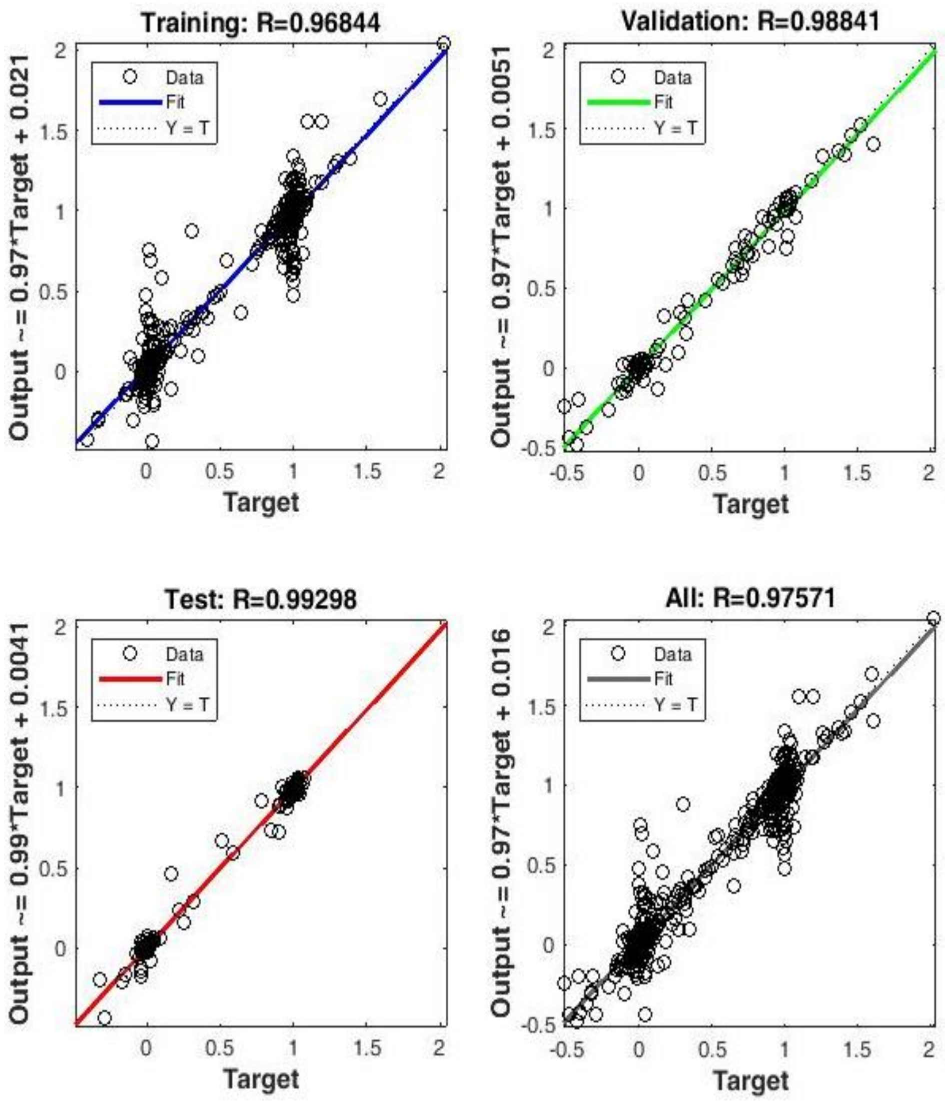


Fig. 6: (a) Regression and (b) pattern recognition of input dimension 35 × 26 and output dimension 25 × 26



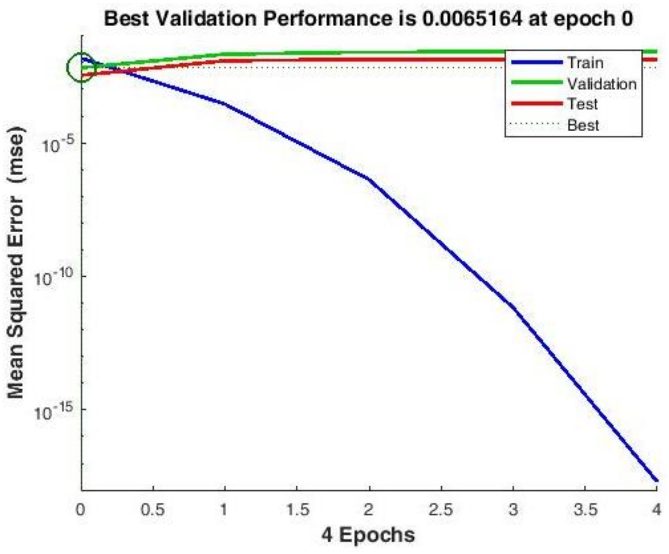
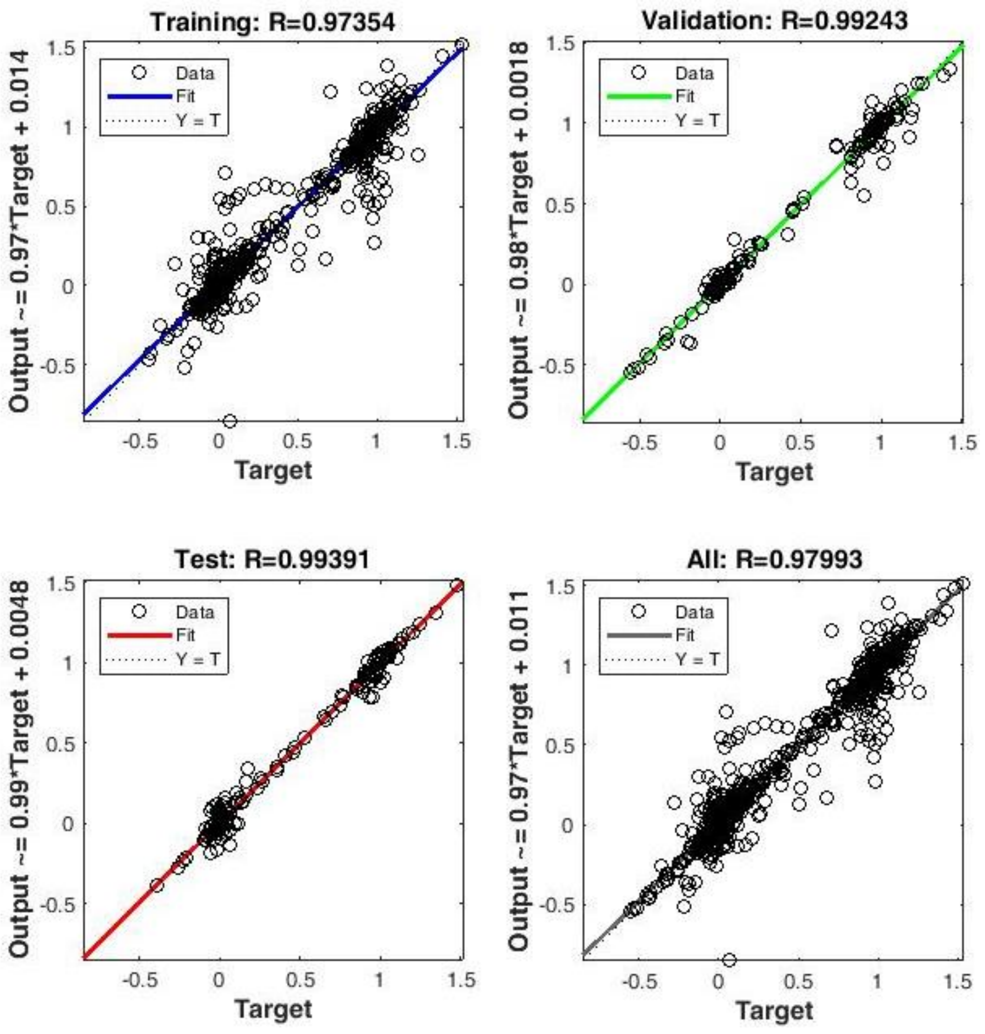


Fig. 7: (a) Regression plot and (b) input 25 × 26 and output 35 × 26 dimensions of recognition of pattern



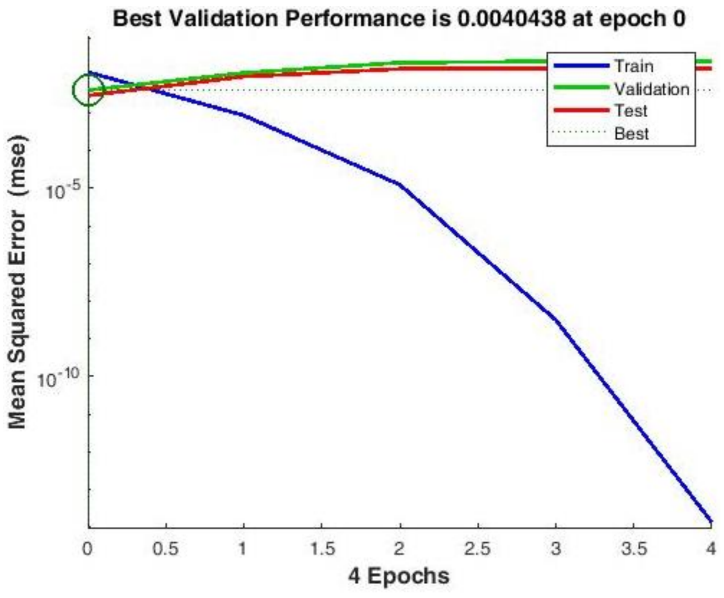


Fig. 8: (a) Regression plot and (b) input 45 × 26 and output 30 × 26 dimensions of recognition of pattern

Further, to validate the performance of associative memory for larger dimensions, input-output pattern pairs of larger size are taken into consideration. Thus, the size of the input matrix 45 × 26 and its corresponding size of the output matrix 30 × 26 are presented to the network. The association for input-output patterns is represented in Fig. 7(a) which shows the overall performance of nearly 98% is achieved with worst and average performance of 0.92 and 0.96 respectively. Figure 7(b) shows the stability of weight convergence. The error gradient is highly dependent on the learned model, though it defines less accuracy.

The regression plot shows the linear relationship between variables in the target data. Predicting patterns from data is done by fitting given values to predicted values. A regression line with values ranging from 0 to 1 indicates a strong linear relationship between the data values. All of the original points lie in a straight line. Based on values of different sizes, a model with values of different sizes is simulated and evaluated as a bidirectional associative memory. This pattern recognition system achieves good accuracy by using input dimensions of 45 × 26 and output dimensions of 30 × 26. Figures 2-7, the output value patterns show good accuracy in bi-directional input output. Figure 8(a) Regression plot shows the positive coefficient values are increasing of the dependent variables also tend to increasing in the test part of the analysis. The model values are positively correlated and strong coordination between the points in the graph. Figure 8(b) shows the input and output dimensions of 45 × 26 and 30 × 26 has given best validation in all epochs.

Conclusion

Artificial intelligence has made great strides in recent years, with machines now capable of learning and performing tasks once only reserved for humans. Identifying patterns in data is one such task, which can be used to predict the future. The purpose of this study is to examine the effectiveness of a particular type of artificial intelligence algorithm known as a neural network for pattern recognition. Two experiments will be presented first to demonstrate how neural networks recognize patterns, followed by a description of how they work. In the first experiment, input and output data are generated randomly, while the second experiment uses real-world climate research data. As a result, our neural network predicted patterns using real-world data with an average accuracy rate of 88%. As a result, it demonstrates the capability of recognizing patterns even in complex datasets containing a lot of noise. The proposed study has demonstrated significant efficiency and has achieved 98.01% accuracy on average. By average. Using the correlation between features, this approach identifies and accelerates learning. A 7 × 5 grid of capital English alphabets is used as the input pattern matrix and a 5 × 5 grid of small English alphabets is used as the output pattern matrix. The performance of associative memory for larger dimensions has been validated with 97.0% accuracy for these pairs of patterns. Object recognition, image restoration, and other complex pattern recognition problems can be solved with it.

Future Scope

As a result, the model is able to better handle linear and non-linear processing techniques for noisy inputs.

The future scope is more cognitively explicatory in linear and nonlinear processes. A directed graph labeled with labels can be used to represent relational knowledge.

Acknowledgment

We thank the University of Petroleum and Energy Studies (UPES), Dehradun 248007 India, and Anil Neerukonda Institute of Technology and Science (ANITS), Visakhapatnam 531162 Andhra Pradesh, India for jointly completed this research work.

Funding Information

We have not received any internal and external funding for this project.

Author's Contributions

Thipendra Pal Singh: Designed the whole manuscript and participated in all experiments, coordinated the analysis and contributed to the written of the manuscript.

Tanupriya Choudhury: Incorporated the literature review and completed the methods required for this study.

Rohini A: Formulated the review responses, designed the figures with high resolution tool.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Aburass, S., Huneiti, A., & Al-Zoubi, M. B. (2022). Classification of Transformed and Geometrically Distorted Images using Convolutional Neural Network.

Acevedo-Mosqueda, M. E., Yanez-Marquez, C., & Acevedo-Mosqueda, M. A. (2013). Bidirectional associative memories: Different approaches. *ACM Computing Surveys (CSUR)*, *45*(2), 1-30. https://doi.org/10.1145/2431211.2431217

Brachmann, A., & Redies, C. (2016). Using convolutional neural network filters to measure left-right mirror symmetry in images. *Symmetry*, *8*(12), 144. https://doi.org/10.3390/sym8120144

Çakar, T., & Cil, I. (2004). Artificial neural networks for design of manufacturing systems and selection of priority rules. *International Journal of Computer Integrated Manufacturing*, *17*(3), 195-211. https://doi.org/10.1080/09511920310001607078

Chen, Q., Xie, Q., Yuan, Q., Huang, H., & Li, Y. (2019). Research on a real-time monitoring method for the wear state of a tool based on a convolutional bidirectional LSTM model. *Symmetry*, *11*(10), 1233. https://doi.org/10.3390/sym11101233

Cohen, M. A., & Grossberg, S. (1983). Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, (5), 815-826. https://ieeexplore.ieee.org/abstract/document/6313075

Ge, S. S., Hang, C. C., Lee, T. H., & Zhang, T. (2013). *Stable Adaptive Neural Network Control* (Vol. *13*). Springer Science & Business Media.

Jepkoech, J., Kenduiywo, B. K., Mugo, D. M., & Tool, E. C. (2022). A Backward Regressed Capsule Neural Network for Plant Leaf Disease Detection. http://repository.chuka.ac.ke/handle/chuka/15413

Kosko, B. (1987). Adaptive bidirectional associative memories. *Applied Optics*, *26*(23), 4947-4960. https://doi.org/10.1364/AO.26.004947

Kosko, B. (1988). Bidirectional associative memories. *IEEE Transactions on Systems, man and Cybernetics*, *18*(1), 49-60. https://ieeexplore.ieee.org/abstract/document/87054

Krippendorf, S., & Syvaeri, M. (2020). Detecting symmetries with neural networks. *Machine Learning: Science and Technology*, *2*(1), 015010. https://iopscience.iop.org/article/10.1088/2632-2153/abbd2d/meta

Lai, H. H., Lin, Y. C., & Yeh, C. H. (2005). Form design of product image using grey relational analysis and neural network models. *Computers & Operations Research*, *32*(10), 2689-2711. https://doi.org/10.1016/j.cor.2004.03.021

Li, X. (2009a). Existence and global exponential stability of periodic solution for impulsive Cohen-Grossberg-type BAM neural networks with continuously distributed delays. *Applied Mathematics and Computation*, *215*(1), 292-307. https://doi.org/10.1016/j.amc.2009.05.005

Li, X. (2009b). Exponential stability of Cohen-Grossberg-type BAM neural networks with time-varying delays via impulsive control. *Neurocomputing*, *73*(1-3), 525-530. https://doi.org/10.1016/j.neucom.2009.04.022

Li, Y., Li, J., Li, J., Duan, S., Wang, L., & Guo, M. (2021). A reconfigurable bidirectional associative memory network with memristor bridge. *Neurocomputing*, *454*, 382-391. https://doi.org/10.1016/j.neucom.2021.04.077

Ozcan, N. (2019). Stability analysis of Cohen-Grossberg neural networks of neutral-type: Multiple delays case. *Neural Networks*, *113*, 20-27. https://doi.org/10.1016/j.neunet.2019.01.017

Panthong, R. (2022). Combining SMOTE and OVA with Deep Learning and Ensemble Classifiers for Multiclass Imbalanced, *Journal of Computer Science* 2022, 732-742. https://doi.org/10.3844/jcssp.2022.732.742

Pulickal, L. (2022). Image Super-Resolution using Auto-Encoders with Parallel Skip-Connections. *Journal of Computer Science* 2022, 1051-1061

Rafiq, M. Y., Bugmann, G., & Easterbrook, D. J. (2001). Neural network design for engineering applications. *Computers & Structures*, *79*(17), 1541-1552. https://doi.org/10.1016/S0045-7949(01)00039-6

Rahman, M. M., Watanobe, Y., & Nakamura, K. (2021). A bidirectional LSTM language model for code evaluation and repair. *Symmetry*, *13*(2), 247. https://doi.org/10.3390/sym13020247

Shi, J., & Zeng, Z. (2020). Design of in-situ learning bidirectional associative memory neural network circuit with memristor synapse. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *5*(5), 743-754. https://ieeexplore.ieee.org/abstract/document/9139434/

Shieh, M. D., & Yeh, Y. E. (2013). Developing a design support system for the exterior form of running shoes using partial least squares and neural networks. *Computers & Industrial Engineering*, 65(4), 704-718. https://doi.org/10.1016/j.cie.2013.05.008

Van Nguyen, N., Lee, J. W., Tyan, M., & Kim, S. (2015). Repetitively enhanced neural networks method for complex engineering design optimisation problems. *The Aeronautical Journal*, 119(1220), 1253-1270. https://doi.org/10.1017/S0001924000011234

Wang, H. H., & Chen, C. P. (2020). A case study on evolution of car styling and brand consistency using deep learning. *Symmetry*, *12*(12), 2074. https://doi.org/10.3390/sym12122074

Wang, J., Tian, L., & Zhen, Z. (2018). Global Lagrange stability for Takagi-Sugeno fuzzy Cohen-Grossberg BAM neural networks with time-varying delays. *International Journal of Control, Automation and Systems*, *16*(4), 1603-1614. https://link.springer.com/article/10.1007/s12555-017-0618-9

Wu, B., Han, S., Shin, K. G., & Lu, W. (2018). Application of artificial neural networks in design of lithium-ion batteries. *Journal of Power Sources*, *395*, 128-136. https://doi.org/10.1016/j.jpowsour.2018.05.040

Wu, D., & Wang, G. G. (2021). Causal artificial neural network and its applications in engineering design. *Engineering Applications of Artificial Intelligence*, *97*, 104089. https://doi.org/10.1016/j.engappai.2020.104089

Zhang, Y., Jin, Z., & Chen, Y. (2020). Hybrid teaching-learning-based optimization and neural network algorithm for engineering design optimization problems. *Knowledge-Based Systems*, *187*, 104836. https://doi.org/10.1016/j.knosys.2019.07