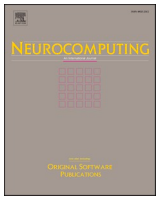
Neurocomputing 614 (2025) 128716

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Deep belief network with fuzzy parameters and its membership function sensitivity analysis

Amit K. Shukla [a,b,*], Pranab K. Muhuri [c]

[a] School of Technology and Innovations, University of Vaasa, Wolffintie 34, Vaasa FI-65200, Finland
[b] Faculty of Information Technology, University of Jyväskylä, Box 35 (Agora), Jyväskylä 40014, Finland
[c] Dept. of Computer Science, South Asian University, Rajpur Road, Maidan Garhi, New Delhi 110068, India

ARTICLE INFO

Keywords:
Deep learning
Deep belief networks
Restricted Boltzmann machine
Fuzzy sets
Type-1 fuzzy sets
Contrastive divergence

ABSTRACT

Over the last few years, deep belief networks (DBNs) have been extensively utilized for efficient and reliable performance in several complex systems. One critical factor contributing to the enhanced learning of the DBN layers is the handling of network parameters, such as weights and biases. The efficient training of these parameters significantly influences the overall enhanced performance of the DBN. However, the initialization of these parameters is often random, and the data samples are normally corrupted by unwanted noise. This causes the uncertainty to arise among weights and biases of the DBNs, which ultimately hinders the performance of the network. To address this challenge, we propose a novel DBN model with weights and biases represented using fuzzy sets. The approach systematically handles inherent uncertainties in parameters resulting in a more robust and reliable training process. We show the working of the proposed algorithm considering four widely used benchmark datasets such as: MNSIT, n-MNIST (MNIST with additive white Gaussian noise (AWGN) and MNIST with motion blur) and CIFAR-10. The experimental results show superiority of the proposed approach as compared to classical DBN in terms of robustness and enhanced performance. Moreover, it has the capability to produce equivalent results with a smaller number of nodes in the hidden layer; thus, reducing the computational complexity of the network architecture. Additionally, we also study the sensitivity analysis for stability and consistency by considering different membership functions to model the uncertain weights and biases. Further, we establish the statistical significance of the obtained results by conducting both one-way and Kruskal-Wallis analyses of variance tests.

1. Introduction

The functioning of human brain could be simulated by various machine learning (ML) algorithms. The accuracy of human like thinking from these algorithms has evolved over the years. One of the recent approaches that has contributed immensely in this area is known as Deep Learning (DL). Earlier, ML approaches had a drawback of restricted machinery, small datasets and less resources. However, in recent times, many such limitations have been addressed and this is one of the reasons of the popularity of the DL approaches. Moreover, these approaches provide enhanced results with improved learning capabilities. At the core of these DL approaches, there is a particular network architecture which plays the pivotal role. Few such popular examples are: Deep Neural Networks, Deep Belief Networks, Convolutional Neural Networks, Recurrent Neural Networks, etc.

The Deep Belief Networks (DBNs) characterize a probability distribution and are formed by stacking several trained Restricted Boltzmann Machines (RBMs) that are generative and stochastic neural networks (NNs). Various distributions of RBMs were employed in the literature such as: Gaussian distribution [23,8], Bernoulli distribution [40], binary distribution etc. The conditional probability is used to depict the inferred state of the hidden units, which is then utilized as the input to the next layer for unsupervised training. This approach contrasts with the proposal made by Hinton, where he used probabilities without sampling when hidden units were driven by reconstructions and always used stochastic binary states when hidden states are driven by data [21]. Furthermore, the expectation is taken after the activation function in calculations.

As stated earlier, the DBN is formed using RBMs which follows a two-step training process. First, the RBM is initialized with the initial

* Corresponding author.
E-mail address: amit.shukla@uwasa.fi (A.K. Shukla).

https://doi.org/10.1016/j.neucom.2024.128716
Received 1 November 2022; Received in revised form 3 October 2024; Accepted 5 October 2024
Available online 11 October 2024
0925-2312/© 2024 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

parameters and trained with the original data. The training is unsupervised using contrastive divergence (CD) algorithm [20]. The features learned from the first RBM is utilized as the input to the next RBM and so on to the last RBM. After this complete process, feature extraction process from the input data is complete [3]. This unsupervised training is regarded as the pre-training of a DBN. Second, fine-tuning is performed with the help of a supervised back-propagation (BP) algorithm [37], which is used to quickly compute the derivatives and gradient descent with respect to the weights. The weight matrix is updated using gradient-descent algorithm. Since the error is propagated in the opposite direction, there is a negligible effect on the parameters of the RBM. The DBN follows the greedy training approach where the local optima are found over all the RBMs and after the complete training the final global optima is obtained.

This greedy approach is sensitive to the diverse initialization of the network parameters such as weights and biases which are assigned randomly with real values. Better initialization is very crucial for better accuracy of the classification result. However, these parameters are highly prone to the noises in the environment of a real-world applications. Such noises and random selection of the parameters results in the inclusion of uncertainty. The random and uncertain information in any computational system may suitably be modeled using FSs, which are often represented through different membership functions (MFs). A MF is a curve that defines how each point in the input space (or sometimes referred to as the universe of discourse) is mapped to a membership value between 0 and 1. Some widely used MFs are Triangular MF (TMF), Trapezoidal MF (TrMF), Semi-Elliptic MF (SEMF) [34], and Gaussian MF (GMF).

In this paper, we have proposed a novel approach for enhanced learning in DBNs by modeling the uncertain weights and biases using fuzzy sets. For this, we have also introduced novel formulation of the energy functions and inferences in DBN with the help of fuzzy parameters. We have considered the expectation of the random variables as the probabilities of the Bernoulli distribution. Furthermore, an algorithm is proposed for generating fuzzy membership function. The experimental results on the various datasets have shown that the proposed approach overcomes the performance of the classical DBN. Various other MF shapes are also considered to model the network parameters such as: TMF, TrMF, SEMF, and GMF. These shapes are experimented to examine the sensitivity of the membership function modeled for the uncertain parameters in a DBN.

The major contribution of this work is compiled as follows:

1. A novel approach is proposed for enhancing the learning outcome in DBNs by modeling the uncertain weights and biases with fuzzy sets.
2. We conducted a thorough sensitivity analysis using different membership functions of Triangular MF (TMF), Trapezoidal MF (TrMF), Semi-Elliptic MF (SEMF), and Gaussian MF (GMF) to model the uncertain weights and biases.
3. A novel algorithm is proposed to generate different fuzzy network parameters having different MF shapes.
4. Along with MNIST and CIFAR-10 datasets, the robustness of the proposed approach is also experimented on two noisy variants of the MNIST dataset: the MNIST with AWGN (additive white Gaussian noise) and the MNIST with motion blur (a.k.a. the n-MNIST).
5. We have conducted extensive simulations by considering different combinations of nodes in the hidden layers of the DBN for better model selection. It directly contributes to reduce the computational complexity of the Fuzzy DBN architecture.
6. The model performance is evaluated with the help of RMSE, Error rate, Log loss and classification accuracy.
7. The multiple simulations are validated through the statistical significance tests of both one-way and Kruskal-Wallis analyses of variance tests.

The rest of the paper is organized as follows. Section 2 provides the preliminaries and the background work in this area. In section 3, we have presented the problem formulation and the proposed technique is in detail. Section 4 explains the experimental setup and results explanation. At last, Section 5 concludes the paper and provides the future perspective of this approach.

2. Preliminaries and background work

This section first presents the brief descriptions of the underlying architectures considered in the proposed approach i.e., RBM and DBN. Then, it concisely reviews the background research works and the related studies with respect to the scope of the paper.

2.1. Restricted Boltzmann machines (RBMs)

RBM is simply a two-layer architecture with a visible and a hidden layer. The inter layer connections are undirected and have no intra neuron connections. A typical RBM is demonstrated in Fig. 1, where v is the visible layer, h is the hidden layer, and W represent the weight matrix.

2.2. Deep belief networks (DBNs)

A DBN architecture consists typically of a visible layer and at least two hidden layers, where the last hidden layer is the output layer. The first two layers forms the first RBM, next two-layer form second RBM and so on. The DBN suffers from the ambiguity in the initial guess for the back-propagation algorithm while learning. However, this problem was significantly reduced after the introduction of pre-training of stacked RBMs by CD learning algorithm. This pre-training of RBMs is an essential process in the construction of DBN. After this training, the states of the hidden units are taken as the training data of the next layer of RBM. When the training of these RBMs is finished, they are stacked together to form DBN. Now, this complete DBN is trained using classical back propagation algorithm.

While constructing a DBN, the output from the hidden layer of one RBM is used as the input to the visible layer of the next RBM [23], and so on. Fig. 2 represents the visualization of a deep belief network, formed by stacking two RBMs, where in the first RBM, W_1 depicts its set of weights and W_2 represents the set of weights for the second RBM.

2.3. Review of the related studies

We could find a number of research work in the literature where the authors used various deep learning techniques for improved learning. Several learning algorithms were proposed for RBMs and DBNs among which Hinton contributed the most [20,22,4,42]. Salakhutdinov and Hinton proposed a novel learning algorithm for RBMs [38]. Later, Hinton also provided the better learning algorithm of the RBMs i.e., contrastive divergence (CD) learning algorithm [20]. Papa et al. in [35]

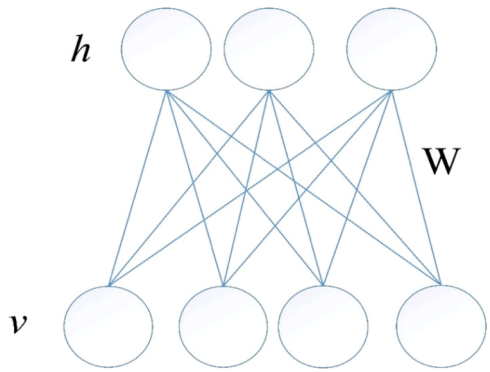


Fig. 1. Representation of a typical RBM.

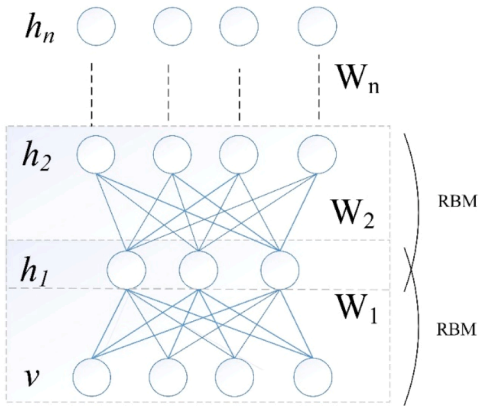


Fig. 2. Visualization of a deep belief network.

studied the problem during fine-tuning in DBNs. The machine learning problem of acoustic modeling using DBN was discussed by Mohamed et al. in [33]. A smart learning approach to learn only important connections in neural network was proposed by Han et al. in [17]. An autoencoder based multi-objective sparse feature learning model was discussed by Gong et al. in [15]. Moreover, the authors transformed the constrained single-objective optimization problem to a multi-objective optimization problem (MOOP). A temperature based RBM was proposed by Li et al. [31] in which the authors argued that the firing neurons in the hidden layers could have significant effect due to the variations in the temperature. Desjardins et al. [11] identified that altering Gibbs sampling in RBM resulted in poor classifications. Thus, they proposed a novel tempered Markov Chain Monte-Carlo approach for sampling in RBMs.

In the literature, researchers modeled weights with FSs in classical NNs. Ishibushi et al. [18] proposed the preliminary idea to model the fuzzy weights and biases as triangular fuzzy numbers. Later, Ishibushi et al. [5] extended the work to propose a novel algorithm for the FNNs. There are other noticeable works which considered FSs to model weights in NN [26,48,31]. The study of modular NNs in multimodal biometry with the comparison among the fuzzy inference systems was studied by Hidalgo et al. [19]. An adaptive neuro-complex-fuzzy-inferential system (ANCFIS) was proposed by Chen et al. [6], where authors applied the neuro-fuzzy system to the machine learning problem of time-series forecasting.

Zhou et al. [48] introduced a special case where the uncertainty in the sentiment analysis is modeled using FMFs for each class of reviews. A novel inference for RBMs with expectation of random variables considered as Bernoulli distribution was proposed by Tanaka and Okutomi in [40]. The explanation of closed form approximation to calculate the expectation was provided by Wang and Manning [44]. The uncertainty aspect was investigated by many researchers in the literature.

Recently, a Fuzzy Restricted Boltzmann Machine (FRBM) architecture was proposed by Chen et al. [7] which considered fuzzy weights and biases as triangular fuzzy numbers. Later, the same authors extended the work for symmetric triangular fuzzy numbers (STFNs), asymmetric triangular fuzzy numbers (ATFNs) and Gaussian fuzzy numbers for the FRBM in [13]. Zhou et al. [41] proposed fuzzy deep belief network for semi-supervised sentiment classification problem. However, the membership function was created for each class of reviews, not for the parameters in the learning of the DBN. BP algorithm was used by Feuring [14] to compute the lower and upper limits of weights in the network. Recently, Chu et al. [9] proposed a variation in the learning approach by using the pairwise constraints with Gaussian visible units in the RBMs. This approach resulted in the enhancement of the expression ability of classical RBMs. Zhao et al. [47] proposed a Deep Belief Network with Fuzzy Sets (DBN-FS) which is used on the real datasets for sensor bases risk perception for driver in the harsh environment of snow and ice. Liu et al. [32] and Cui et al. [10] introduced Propelled Multiple Fusion Deep Belief Network (PMF-DBN) and DBN with fuzzy ranking feature optimization for Weld defect and Seabed sediment classification, respectively. Cui et al. [10] used the real data from Multibeam Echo-Sounding System (MBES) and ground-truth sediment samples from the southern Irish Sea provided by the British Geological Survey (BGS).

Table 1 presents the comparative analysis of our proposed approach and the related FDBN approaches. Our approach has considered various MF shapes such as trapezoidal, triangular, Gaussian, and Semi-elliptic for the fuzzy weights and biases. These are the most common and relevant shapes available in the literature, which could be used in almost any real-world applicative context [34,25,1,36]. Moreover, we have considered the symmetric as well as asymmetric membership function shapes generated by our novel membership function generation algorithm, which is inclusive of all the possible uncertain scenarios. In the fine-tuning stage of the DBN, our approach has used the stochastic gradient descent (SGD) or backpropagation approach. A numerical example demonstrates the validation of the fuzzy energy function is provided in the supplementary material. In the experimentation part, the proposed approach is applied on the dataset with added noises of white Gaussian noise and motion blur, which is undoubtedly very crucial for evaluating the robustness of the fuzzy approach. Moreover, we have also considered the most widely used CIFAR-10 classification dataset for the reliability of the proposed approach. For an optimal model selection, all possible combinations of a number of hidden layers are considered to establish the network parameters empirically. All these experiments are evaluated for several epochs to get the best classification results.

In comparison, other approaches in Table 1 are particular to a specific set of problems with only a specific set of network parameters. They are not extensive in choosing the optimal number of network parameters. Apart from our proposed approach, fuzzy weights and biases are modeled with triangular MF by Feng et al., Samui et al., and Jiang et al. Among them only Samui et al. considered asymmetric shapes, but with only triangular MF. In conclusion, our approach provides the more generic fuzzy sets-based modeling of network parameters in DBN, which considers several symmetric and asymmetric membership functions for the network parameters and establish the best possible among them. Moreover, extensive experimentations provide a perspective for readers to choose an optimal network architecture.

3. Proposed fuzzy deep belief network

This section develops the basic understanding of the procedures involved in the learning of DBN. Then, we propose the formulations for the fuzzy sets for enhanced deep learning algorithm. Each RBMs of a DBN are associated with some weights and biases which determines the energy of a joint configuration of the visible and the hidden units.

We first present the flowchart of the novel algorithm for enhanced learning in the DBN. The flowchart is demonstrated in Fig. 3, followed by the step-by-step description. The step-wise explanation of the approach is explained below as:

Table 1
Comparative analysis of related Fuzzy DBN approaches[a].

Paper	Methodology proposed	Fuzzy parameters/ Fuzzy approach	Network Arcitecture	Application	Datasets
Zhou et al. [48]	FDBN and Active fuzzy DBN	Reviews and theirmembership values	Specific: 100–100–200–2 and50–50–200–2; First 3 are hidden layers and last is output layer	Sentiment classification	Movie review dataset, books, DVDs, electronics, and kitchen appliances
Jiang et al. [27]	FDNN & fuzzy Gaussian-Bernoulli RBM	Network parameters modeled using TFN	Three hidden layers as: 170–1700–17 and 150–1500–15	Click-through rate (CTR) prediction	Criteo Display Advertising Challenge dataset
Samui et al. [39]	FRBM with Symmetric & Asymmetric TFN	Network weights and biases	Number of hidden units: 64, 128, 256, 512	Speech enhancement framework	speech & noise specific datasets: TIMIT, Voice Box
Wang et al. [45]	Information Geometry Enhanced Fuzzy Deep Belief Networks (IGEF–DBN)	Fuzzy rules learned by FCM	50–50–200–6(6)−6–2, 50–50–200–8(8)−8–2, 100–100–200–5(5)−5–2, 50–50–200–12(12)−12–2, and 50–50–200–8(8)−8–2	Sentiment classification	Sentiment classification data sets: BOO, DVD, ELE, and KIT
Zhang et al. [46]	DBN-based TSK fuzzy classifier	Fuzzy rules, Gaussian MF	Specifically selected hidden nodes for different methods	Indoor user movement prediction	MovementAAL_RSS
Feng et al. [12]	FDBN with symmetric triangular MF	Only triangular fuzzy weights and biases	Consideration of specific number of hidden layers for respective models.	High-dimensional classification problem	MNIST and added noise such as: Salt & pepper and gaussian, NORB dataset, & 15 Scene dataset
Jisha and Vimal [29]	Fuzzy DBN	Intuitionistic FuzzyMutual Information	Not mentioned	Credit card fraudulent detection	Credit card datasets
Zhao et al. [47]	DBN-FS	Traditional FSs	4 hidden layers each with 20 nodes	Sensor-based risk perception	Real-world driving experiments
Liu et al. [32]	PMF-DBN	FCM and fuzzy classifiers applied at each network layer output	Three-layer DBN with 128, 200, 200 neurons in the hidden layers	Weld defect detection and classification	X-ray images of petroleum steel pipe welds
Cui et al. [10]	DBN with fuzzy ranking feature optimization	Fuzzy Ranking (FR) for feature selection and optimization	Two hidden layers with 80 neurons each during pretraining, one hidden layer during fine-tuning	Seabed sediment classification	MBES data and ground-truth sediment samples from the BGS
Proposed Approach	DBN with fuzzy parameters, novel generation of fuzzy MFs, MF sensitivity analysis	Trapazoidal, triangular, Gaussian and Semi-elliptic fuzzy weights and biases	Several hidden layers and several nodes are considered for better model selection.	Classification	1. MNIST2. MNIST with additive white gaussian noise3. MNIST with motion blur4. CiFAR−10 dataset

[a] The research focus in this paper is limited to modelling network parameters of DBN as T1 FSs and membership function sensitivity analysis. Accordingly. consideration of higher FS variants are beyond the scope of the current study.

Step 1: The inner RBMs in a DBN are initialized randomly. The Bernoulli distribution is considered for a RBM.

Step *2:* To pre-train DBN, RBM is pre-trained using contrastive divergence learning algorithm. During pre-training, the weights and biases of the RBM are fuzzified.

Step 3: The network is pre-trained using the CD algorithm in unsupervised manner.

Step 4: The RBM associated to the linear mapping is attached to the last layer in DBN.

Step 5: Finally, the supervised learning of back propagation algorithm is applied to fine-tune the network.

The energy function for an RBM is given by the following equation:

$$E(v,h;W,b,c) = -h^T W v - c^T v - b^T h \tag{1}$$

$$= -\sum_j \sum_k W_{jk} h_j v_k - \sum_k c_k v_k - \sum_j c_j v_j \tag{2}$$

where, W is the weight between that layers, h and v are the vector representing the states of the hidden and visible layers, where h_j and v_k denotes the states at jth and kth units. The biases associated with the hidden and visible layer are denoted by b and c, respectively.

The network probabilities for every pair of visible and hidden states are given by:

$$p(v,h;W,\quad b,\quad c) = \frac{1}{\tilde{\tilde{Z}}} e^{-E(v,h;W,\quad b,\quad c)} \tag{3}$$

where $Z = \sum_{v,h} e^{-E(v,h;W,\quad b,\quad c)}$ is the crisp partition function, which sums over all possible hidden and visible vectors. The inferences used in the RBM is the conditional inference of either v given h or h given v. The full conditional distribution of h given the visible layer v can be written as:

$$p(h|v) = \prod_j p(h_j|v) \tag{4}$$

For the individual distributions, $p(h_j|v)$, h_j is a random variable with binary values so it's a Bernoulli such that the probability of h_j being equal to 1 has a form:

$$p(h_j = 1|v) = \sigma(b_j + W_j v) \tag{5}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ represents the sigmoidal activation function.

Similarly, the conditional distribution for v given h is as follows:

$$p(v|h) = \prod_k p(v_k|h) \tag{6}$$

$$p(v_k = 1|h) = \sigma(c_k + h^T W_k) \tag{7}$$

The inference proposed in [4] is given by the equation:

$$P(\vartheta_{u+1} = 1) = E_{P(\vartheta_u)}\left(\sigma\left(W_u^T[\vartheta_u]\right) + \quad b_u\right) \tag{8}$$

While training the classical DBN (CDBN) architecture, the weights and biases are initialized randomly with real numbers which are generally associated with uncertainty due to the various possible values.

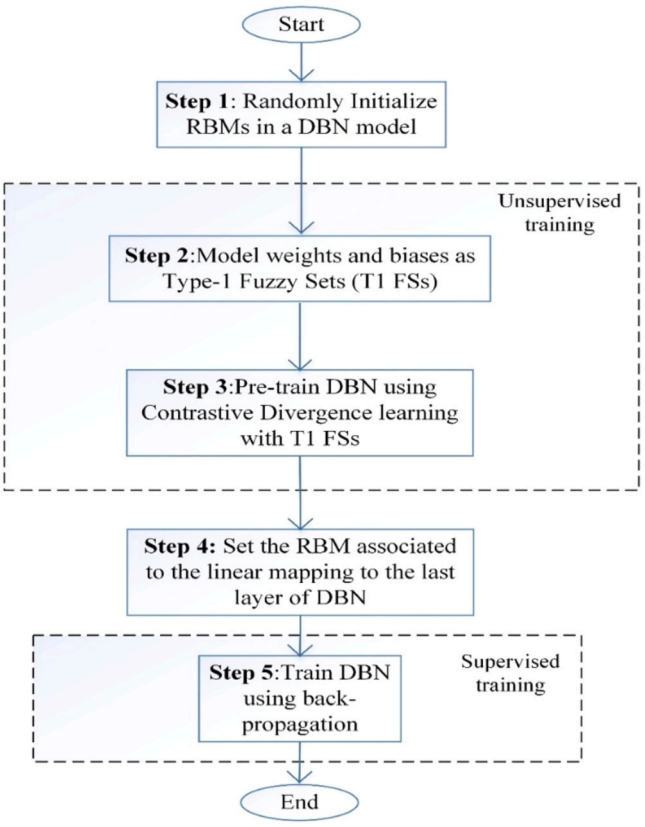


Fig. 3. Flowchart of the proposed approach.

This may hinder the overall performance of a DBN. Thus, to model these uncertain parameters, we have used FSs to model the uncertain weights and biases of the DBN. FSs are characterized by the membership function which is modeled over the membership grade which is defined as the degree to which an element belongs to a set. If the membership grades for the elements in a fuzzy set are expressed in terms of crisp values, then the fuzzy set is called an ordinary or type-1 (T1) fuzzy set. A FS 'F' can be expressed mathematically as follows:

$$F = \{(x, \quad \mu_F(x))|\forall x \in X\} \tag{9}$$

or

$$F = \int_{x \in X} \mu_F(x)/x \tag{10}$$

Here, $\mu_F(x)$ is the membership degree of an element x for each $x \in X$. Eq. (9) is applicable when x is discrete, whereas for the continuous cases Eq. (10) is valid. In practice, triangular and trapezoidal MFs are used in literature to model the fuzzy sets. Since there is randomness involved while initializing the weights and biases, we have considered four different shapes of the MFs in this work to analyze the sensitivity of these shaped over the outcome.

Let's suppose 'w' denotes the randomly assigned crisp weight, m_L denotes left spread of the MF, and m_R denotes right spread of the MF. The membership value is represented by μ, where $\mu \in [0, 1]$. Fig. 4 shows the typical representation of all these MFs. Thus, the variables associated with them, i.e. W, b, and c, are re-formulated as $\widetilde{W}, \quad \widetilde{b}$ and $\widetilde{c}$. Now, these connection weights between the visible and hidden layers form a matrix. Later, each value of this matrix is defuzified to a single crisp value which is then further used in the training of a DBN. For defuzzification, we have used the widely used centroid defuzzification method given as Algorithm 2. The general procedure to generate these MFs is compiled as Algorithm 1:

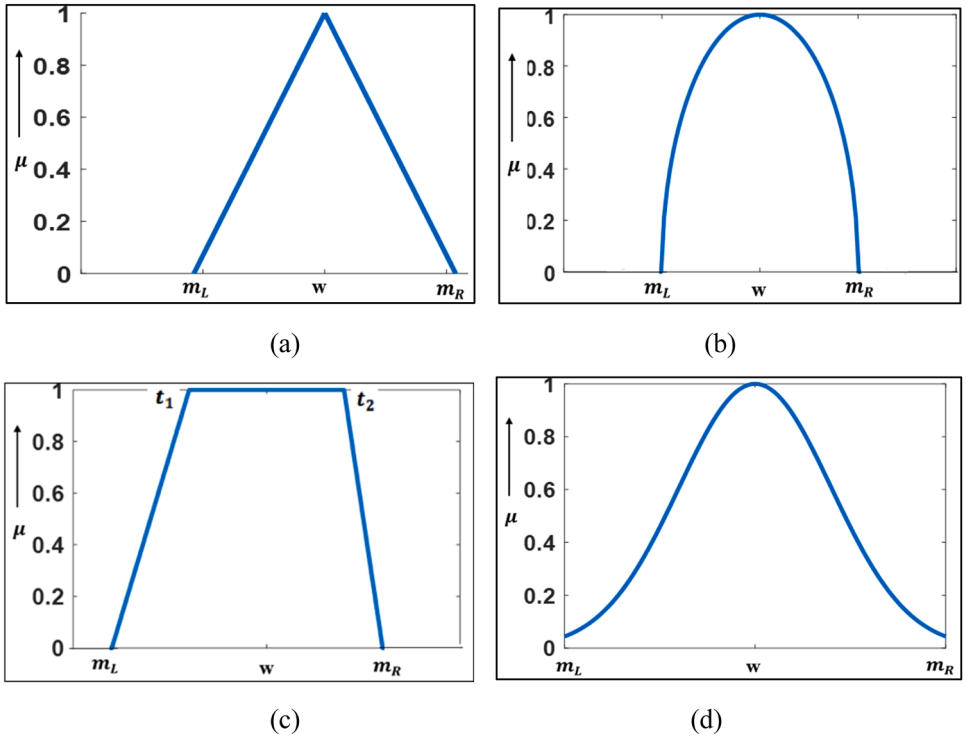


Fig. 4. MF shapes: (a) triangular, (b) semi-elliptic, (c) trapezoidal, and (d) Gaussian.

Algorithm 1. Algorithm for fuzzy membership function (FMF) generation

Input: *Crisp weights and biases*

Output: *Fuzzy membership functions*

Method FMF Generation ( ) {

Step 1: *For the input parameter, w, Steps 2-6 are repeatedly performed.*

Step 2: *A random value r is selected between [0, 1].*

Step 3: *Compute* m_L *= w-(w * r). //where* m_L *is the left spread point of the MF.*

Step 4: *Again, choose a random value between [0, 1].*

Step 5: *Compute* m_R *= w + (w * r). // where* m_R *is the right spread point of the MF*

Step 6: *Calculate triangular and trapezoidal MF using trimf (*m_L*, w,* m_R*) and trapmf (*m_L*,* t_1*,* t_2*,* m_R*), respectively //where the function 'trimf' is used to compute triangular MF from the parameters calculated in Steps 3 and 5, 'trapmf' is used to compute trapezoidal MF with* $m_L \le t_1 \le w$ *and* $w \le t_2 \le m_R$*.*

Step 7: *As left and right spread are distributed from x with some distance, we can get* r_d *as the difference between w and* m_L *(*$r_d = w - m_L$*)*

Step 8: *Now that we have the parameters, we can design the SEMF as follows:*

$$SEMF(x;C_d,r_d) \quad \mu_E(x)=\begin{cases}\sqrt{1-\frac{(C_d-x)^2}{r_d^2}} & if\ C_d-r_d \le x \le C_d+r_d \\ 0 & otherwise\end{cases}$$

where $\mu_E(x)$ *is the membership grade of x on X. Here* $[C_d - r_d, C_d + r_d]$ *is the 0-cut of SEMF and* C_d *is the center of the ellipse.*

Step 9: *Plot GMF using:* $\mu_E(x) = exp\left(-\frac{(k_i-x)^2}{2\sigma_i^2}\right)$*,* k_i *and* σ_i *are the center and width of the MF,*

}

Algorithm 2. Centroid defuzzification algorithm

Input*: T1 fuzzified parameters;* $x \in [m_L, m_R]$

Output*: Defuzzified crisp output*

Method T1 MF Defuzzification ( )

{

For i=1 to N // N is the number of input points in x

{

1. Compute upper= $\sum_{i=1}^{N} x_i\mu_i$ *// μ_i is the membership value of the ith index.*

2. Then calculate lower= $\sum_{i=1}^{N} \mu_i$

}

Compute defuzz = upper/lower;

}

Now, the output of first RBM is used as an input to the next RBM and the new weights are taken as the transpose of the previous layer weights. The whole training proceeds and the weights are tuned until the optimal weights are computed. For the stacked RBM in the DBN, the fuzzy energy function '$\widetilde{E}$' of the joint configurations of the 'v' visible and the 'h' hidden units with fuzzy weights and biases is defined as:

$$\widetilde{E}(v,h;\widetilde{W},\widetilde{b},\widetilde{c}) = -h^T\widetilde{W}v - \widetilde{b}^T h - \widetilde{c}^T v \tag{11}$$

where $\widetilde{W}$ is the fuzzy weights and $\widetilde{b}$ and $\widetilde{c}$ are the fuzzy biases.

All the pairs of the visible and the hidden units are associated with some probabilities and utilizing the above defined energy function, the probability is defined as:

$$p(v,h;\widetilde{W},\widetilde{b},\widetilde{c}) = \frac{1}{\widetilde{Z}} e^{-\widetilde{E}(v,h;\widetilde{W},\widetilde{b},\widetilde{c})} \tag{12}$$

Here, $\widetilde{Z} = \sum_{v,h} e^{-\widetilde{E}(v,h;\widetilde{W},\widetilde{b},\widetilde{c})}$ is defined as the fuzzy partition function, which is the summation of every possible hidden and the visible layer vectors.

The conditional probability of the hidden units (given visible units) and visible units (given hidden units) is expressed as:

$$p(h_j = 1|v) = \sigma(\widetilde{b}_j + \widetilde{W}_j v) \tag{13}$$

$$p(v_k = 1|h) = \sigma(\widetilde{c}_k + h^T\widetilde{W}_k) \tag{14}$$

Here, $\widetilde{\sigma}(\bullet)$ represents the fuzzy logistic function with fuzzy arguments.

The activation function is applied before proceeding to the expectation [40]. Accordingly, probabilities of the $(u+1)th$ nodes could now be inferred as follows:

$$\widetilde{P}(\vartheta_{u+1} = 1) = \widetilde{E}_{\widetilde{P}(\vartheta_u)}\left(\widetilde{\sigma}\left(\widetilde{W}_u^T[\vartheta_u]\right) + \quad \widetilde{b}_u\right) \tag{15}$$

where ϑ_u is the associated random binary variable. Moreover, the conditional probabilities of the $(u+1)th$ node are evaluated considering all the probable combinations of the binary states. Further, the expectation of finally evaluated conditional probabilities is executed to infer the probabilities of the $(u+1)th$ node.

Now, we define the basic understanding of contrastive divergence (CD) algorithm used in the training of RBM as proposed by Hinton. First, the training vector is fixed with the states of the visible units of the RBM. Then, we calculate the binary states of the hidden units. The probability in Eq. (14) is used for the reconstruction process. Thus, the change in fuzzy weights between the two units, lets say i and j, is defined as follows:

$$\Delta\widetilde{w_{ij}} = \in \left(< v_i^n h_j^n > - < v_i^{n+1} h_j^{n+1} > \right) \tag{16}$$

where $\in$ is the learning rate and the superscripts n and $n+1$ represents the 'reconstruction' steps. This is a repetitive procedure till the equilibrium is attained at the n^{th} state. CD training is shown in Fig. 5.

In CD training, we first calculate the hidden state h_0 from the training data v_0, followed by reconstructed v_1. For this calculation, training data is considered as an instance to be evaluated which is considered as the probability of that training data. Thus, h_0 is evaluated using the inference. Now, as the average probability is the output of the inference, v_1 and h_1 are calculated accordingly.

After the CD training of the RBMs, standard back-propagation learning is applied to the complete DBN [44]. Now, in the learning stage, parameters such as weights and biases play a very important role. Since, the

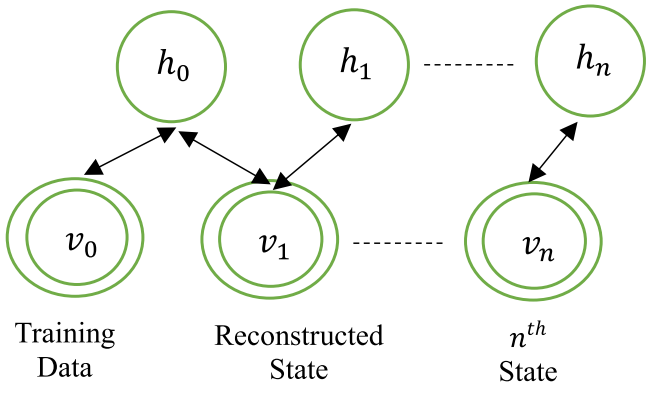


Fig. 5. Reconstruction process of contrastive divergence algorithm.

network converge to a local minimum different parameters initialization will result in the minimization of the loss function to different values. These parameters are considered as the fuzzy parameters and the underlying steps of the proposed approach is compiled as Algorithm 3**:**

Algorithm 3. Proposed approach

Input: *Training data samples, Test data Samples, Learning rate '$\in$', Crisp parameters: W, b, and c*

Output: *RMSE and Error rate*

Method ()

{

Step 1: Initialize all the inner RBMs in the DBN model randomly.

Step 2: Start pre-training DBN after the initialization

{

Pre-train RBM

{

Use CD algorithm to train the RBMs using unsupervised learning.

During learning, consider weights and biases as fuzzy parameters. (Algorithm 1)

Perform defuzzification to get the crisp output. (Algorithm 2)

}

}

Step 3: Assign the RBM associated to the linear mapping of the last layer in the DBN

Step 4: Train the formed DBN using back-propagation algorithm

Step 5: Calculate the RMSE and the Error Rate as follows:

$$RMSE = \sqrt{\frac{1}{O_c N_d}\sum_{k=1}^{N_d}\|y_k - F(x_k)\|_2^2}$$

$$Error\ Rate = \frac{I_N}{N_d}$$

$$Log\ Loss = -\frac{1}{N_d}\sum_{k=1}^{N_d}\sum_{c=1}^{O_c} y_{kc}\log(p_{kc})$$

$$Accuracy = {Correct\ predictions}/{N_d}$$

//where N_d, represents the data instances, O_c indicates number of output classes, number of incorrect inferences are represented as I_N, y_{kc} is 1 if the true class if kth instance is c, and p_{kc} is predicted probability. Further, x_k represents kth input data and y_k depicts ground truth output and F represents inference of the trained DBN

}

4. Experimentation and results

In this section, we have performed the experimentation of our proposed approach on the widely used datasets. First the considered network architecture and parameters are described, followed by the details of all the datasets. Next few sections discuss and analyze the results obtained on all those datasets. The results of the proposed approach (considering weights and biases as TrMF, TMF, SEMF, and GMF) are compared with CDBN approach. The CDBN performed the training and testing with the crisp real values. The last sub-section provides the results of the statistical significance analyses. The experiments are conducted on a workstation with Intel(R) Xeon(R) CPU E5–2650 processors with 2.00 GHz frequency, 48 GB random access memory, NVIDIA 2 GB GeForce GTX 960 graphical processing unit, and a 64-bit windows operating system.

4.1. Network architecture

The proposed approach is a four-layer model with one visible layer, two hidden layers (with same number of nodes), and the last layer act as the output layer. For instance, {V-H_1-H_2-C} represents this model where, V is visible layer, H_1 and H_2 are hidden layers, and C is last classification layer. A rigorous analysis is executed by performing a model-selection approach which considers 200, 400, 600, 800, and 1000 no. of nodes in the hidden layer. In these hidden layers, where the abstraction occurs, we have considered same number of nodes in each layer. First, we considered 200–200 (H_1-H_2) number of nodes in the hidden layer, then various other number of nodes are considered such as: 400–400, 600–600, 800–800, and 1000–1000. Each experiment with respective number of hidden nodes is performed till 1000 epochs.

4.2. Datasets

First, we have considered the widely used digit recognition MNIST classification dataset. We have performed the rigorous experimentations on this dataset for epochs from 100 till 1000. The empirical results on the MNIST dataset with our proposed model concluded that 600 nodes resulted in the best results. Therefore, for the other datasets we have fixed the number of nodes in the hidden layers as 600 and experimented till 1000 epochs. The other datasets are: two n-MNIST (MNIST with additive white Gaussian noise (AWGN) and MNIST with motion blur) and a CIFAR-10 dataset.

4.3. Evaluation metric

In this paper, we have used a probabilistic deep belief network for the problem of classification. First, we computed the root mean square error (RMSE) and Error Rate to evaluate the results from the proposed model. The outcome from the RMSE are typically probabilities and it is used to measure the error between the predicted probabilities and the actual labels. We chose RMSE since it provides a direct measure of the difference between the predicted and actual values. This is particularly useful for evaluating the fuzzy DBN (with noisy data) like probabilistic model performance in terms of regression-like tasks within the classification framework. RMSE provides a measure of error across all classes by penalizing larger errors more heavily because of the squaring of differences. In addition, error rate is also computed for a straightforward indication of the quantity of incorrect predictions. Though, RMSE is well suited to provide insights on how well the predicted probabilities match the actual distribution, however, it doesn't directly replicate the quality of the classifier in making correct predictions.

Therefore, considering the limitation of the RMSE and with the aim to assess model performance under noisy data conditions, we also considered metrics like log loss (Cross-entropy loss) [16] and accuracy to complement the overall evaluation of the classification results. The log loss quantifies the difference between the predicted probability distribution and the actual distribution, which signifies a better prediction of the model has achieved a lower log loss value. Consequently, it indicates that a higher probability is assigned to the correct classes. All the formulas for these metrices is presented in Algorithm 3.

4.4. Experimental analysis: MNIST dataset

The first dataset is the MNIST [30] dataset which is most widely used in the experimentation of any machine learning approach. It's a handwritten digit database consisting of black and white images of size 28 × 28. These images are linearized as vectors of size 1 × 784. Each image holds a digit 0–9 containing 10 classes. The dataset has a total of 60,000 training images and 10,000 test images which form a 2D vector of size 60,000 × 784 and 10,000 × 784, respectively. Table 2 and Table 3 presents the RMSE values of the training data and the test data for the considered number of hidden layer nodes over 100, 200, 400, 600, 800 and 1000 epochs. From the analysis, it is evident that the RMSE

Table 2
RMSE of the training data.

Hidden layer nodes	Epochs	Training data RMSE				
		CDBN	GMF	SEMF	TMF	TrMF
200	**100**	0.0320816	0.0320000	0.0319110	0.0314819	0.0315136
	200	0.0198498	0.0224670	0.0216100	0.0177093	0.0202171
	400	0.0160780	0.0170561	0.0169703	0.0159193	0.0161853
	600	0.0157501	0.0152490	0.0162300	0.0156293	0.0155687
	800	0.0156920	0.0154179	0.0157560	0.0155000	0.0154557
	1000	0.0155903	0.0156102	0.0155920	0.0154090	**0.0149141**
400	**100**	0.0733036	0.0360847	0.0347149	0.0478910	0.0337825
	200	0.0574025	0.0208486	0.0201659	0.0312900	0.0210433
	400	0.0411634	0.0155259	0.0158703	0.0172040	0.0161870
	600	0.0320507	0.0152490	0.0147159	0.0151018	0.0152439
	800	0.0262516	0.0154179	0.0136560	0.0147062	0.0144196
	1000	0.0226005	0.0149353	0.0146011	0.0146383	**0.0145785**
600	**100**	0.0369591	0.0360619	0.0361569	0.0365927	0.0368138
	200	0.0225985	0.0217387	0.0215482	0.0211350	0.0220687
	400	0.0189540	0.0173546	0.0164183	0.0180270	0.0164916
	600	0.0157529	0.0154909	0.0155016	0.0155707	0.0159021
	800	0.0154210	0.0149469	0.0148636	0.0150102	0.0147488
	1000	0.0147229	0.0148276	0.0143000	0.0143951	**0.0142300**
800	**100**	0.0708755	0.0398030	0.0378807	0.0335480	0.0382285
	200	0.0547427	0.0267800	0.0236483	0.0245410	0.0231188
	400	0.0445640	0.0215400	0.0165271	0.0214500	0.0174574
	600	0.0305241	0.0171111	0.0161211	0.0162516	0.0157056
	800	0.0252186	0.0163437	0.0159437	0.0160643	0.0154255
	1000	0.0218444	0.0157794	0.0152000	0.0152441	**0.0151230**
1000	**100**	0.0404848	0.0402130	0.0394556	0.0394885	0.0420688
	200	0.0268023	0.0240000	0.0239409	0.0241346	0.0241545
	400	0.0256810	0.0174398	0.0174866	0.0179117	0.0173796
	600	0.0244320	0.0169011	0.0161333	0.0165569	0.016755
	800	0.0234210	0.0160529	0.0156678	0.0159249	0.0164113
	1000	0.0218750	0.0158883	0.0155015	0.0155611	**0.0153260**

Table 3
RMSE of the test data.

Hidden layer nodes	Epochs	Test data RMSE				
		CDBN	GMF	SEMF	TMF	TrMF
200	**100**	0.0600902	0.0593692	0.0587429	0.0581166	0.0574903
	200	0.0582902	0.0605033	0.0593522	0.0582011	0.0570500
	400	0.0581572	0.0596846	0.0588423	0.0580000	0.0571577
	600	0.0572554	0.0579285	0.0574743	0.0570201	0.0565659
	800	0.0572132	0.0578519	0.0573765	0.0569011	0.0564257
	1000	0.0577020	0.0590428	0.0579523	**0.0568618**	0.0557713
400	**100**	0.0758892	0.0589504	0.0578576	0.0601780	0.0569614
	200	0.0645848	0.0554651	0.0544203	0.0589410	0.0563450
	400	0.0614035	0.0555263	0.0559448	0.0579780	0.0549595
	600	0.0585828	0.0559036	0.0553937	0.0561520	0.0560338
	800	0.0578000	0.0560022	0.0552326	0.0550427	0.0543561
	1000	0.0563065	0.0559823	0.0551860	0.0555033	**0.055175**
600	**100**	0.0578826	0.058446	0.0580339	0.0576421	0.056672
	200	0.0565044	0.0559557	0.0553728	0.0551400	0.0550002
	400	0.0561000	0.0551842	0.0551688	0.0550000	0.0549265
	600	0.0558281	0.0549969	0.0547427	0.0542224	0.0543404
	800	0.0555100	0.0545056	0.0544487	0.0541716	0.0540750
	1000	0.0550690	0.0544582	0.0541210	0.0540841	**0.0537270**
800	**100**	0.0734230	0.0596170	0.0587158	0.0625140	0.0585483
	200	0.0612433	0.0577100	0.0553642	0.0587400	0.0556251
	400	0.0574100	0.0569110	0.0545919	0.0564700	0.0546981
	600	0.0567889	0.0552890	0.0548340	0.0554615	0.0554604
	800	0.0561535	0.0551000	0.0547115	0.0558096	0.0546547
	1000	0.0559937	0.0549858	0.0547001	0.0550000	**0.0546510**
1000	**100**	0.0595349	0.0599870	0.0595480	0.0591115	0.0610137
	200	0.0588592	0.0560000	0.0548423	0.0569485	0.0558234
	400	0.0571789	0.0553432	0.0558310	0.0562092	0.0549400
	600	0.0568547	0.0554817	0.0546763	0.0556550	0.0546464
	800	0.0564507	0.0553155	0.0550090	0.0555148	0.0548339
	1000	0.0555937	0.0549678	**0.0547840**	0.0554200	0.0548341

is much better for our approach as compared to the classical DBN in the training datasets. By default, it uses TrMF to model the fuzzy network parameters.

However, even with other MFs (TMF, SEMF and GMF), the network still performed better than the CDBN. The proposed approach modeled using TrMF shows marginally improved performance over the TMF, SEMF, and GMF, as even a minor improvement in the RMSE is of great importance. Interestingly, the RMSE with TrMF and TMF have negligible difference in their values. Also, for all these considered models, the RMSE values converged at 1000 epochs. Furthermore, it is observed that the RMSE values improved for the training data when the number of nodes in hidden layers are increased from 200. The RMSE of 0.01423 is obtained at 600 no. of nodes in the hidden layer with proposed approach, which is 1.16 % better than the TMF (0.01439), 0.49 % better than SEMF (0.0143), and 6.30 % better than GMF (0.01513).

However, after this, the overall RMSE values in all the MF models increased slightly. Still, TrMF has shown much better RMSE values than other MFs. Fig. 6 shows the comparison between the proposed approach (with TrMF) and CDBN with the convergence of RMSE values in the training data. This comparison is depicted only for 600 units in the hidden layers. We have not shown the RMSE values for the 100 epochs. Similar behavior can be seen for the RMSE values of the test data, which is extremely important in classification. The least RMSE value of 0.0537 is obtained for the proposed approach at the 600 no. of nodes in the hidden layers. The overall RMSE improved till 600 nodes in the hidden layer but increased after that till 1000 nodes. Although we get the best result at 600 nodes in the hidden layer with the TrMF, SEMF showed better results than TMF and GMF. The convergence of test data with 600 units in the hidden layers is shown in Fig. 7.

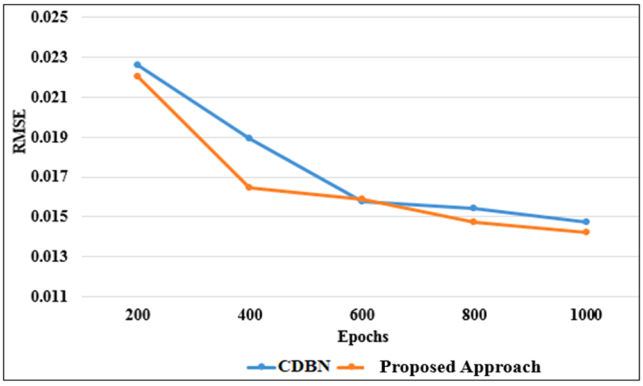


Fig. 6. RMSE vs. epochs plot for the training data.

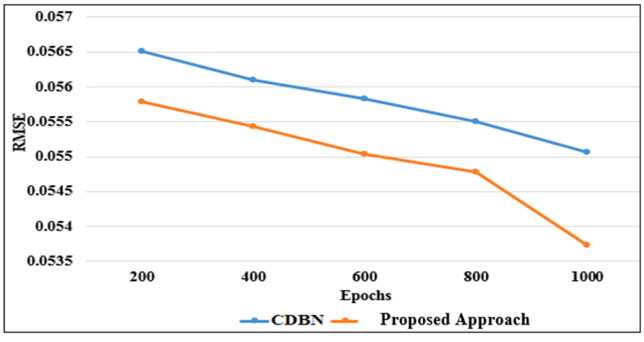


Fig. 7. RMSE vs. epochs plot for the test data.

Fig. 8 & 9 shows the comparison between our proposed approach and other MFs such as: TMF, SEMF, and GMF for the MF sensitivity analysis. A snippet is provided in the figures for the 800 and 1000 epochs to distinguish the difference. In both cases, the weights and biases modeled with TrMF performed better than when they are modeled with other considered MFs.

We have only shown the result for the Error Rate with 600 units in the hidden layers at 1000 epochs for the training data and test data in Fig. 10. We have observed that data modeled with TrMF (training data error rate = 0.0175, testing data error rate = 0.0161) has fewer incorrect inferences than the TMF, SEMF, and GMF. Additionally, we also computed two other metrics of log loss and accuracy for training and test data, which is shown in Table 4. There is again improvement with the proposed approach in terms of lower log loss and better accuracy.

4.5. *Experimental analysis: n-MNIST datasets*

To establish the validity and robustness of our proposed approach, we have also considered two additional datasets from the repository of the n-MNIST datasets [2]. The two datasets are MNIST with additive white Gaussian noise (AWGN) and MNIST with motion blur. They are formed by adding additional noise in the original MNIST dataset. The dimension and training and test data sizes are the same for both datasets. The sensitivity analysis with different membership functions for the MNIST concluded that TrDBN and TDBN gave the best results. Thus, we have used only TrMF for these two datasets, which was also earlier used as default MF for the proposed approach. It is compared with the CDBN for the 600 nodes in the hidden layer to evaluate the efficiency.

Tables 5 and 6 show the RMSE and Error Rate for the MNIST with AWGN and motion blur datasets, respectively. In the case of training data, our proposed approach, with RMSE of 0.01439 shows a 1.02 % improvement for MNIST with AWGN, while the test data has 0.8 % improvement over CDBN. While these improvements may represent small percentages, they carry significant weight and hold substantial

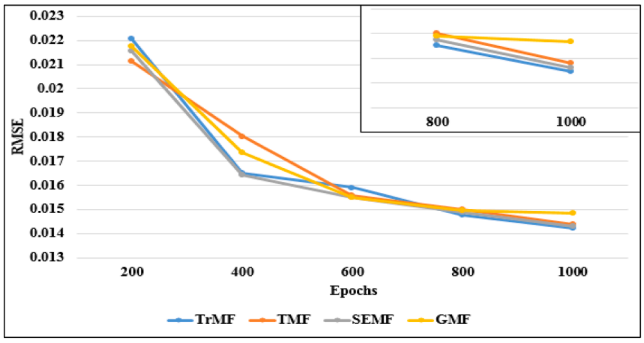


Fig. 8. RMSE vs. epochs plot for the training data with all MFs.

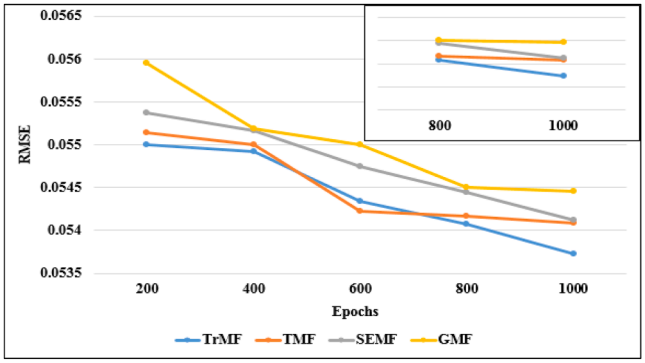


Fig. 9. RMSE vs. epochs plot for the training data with all MFs.

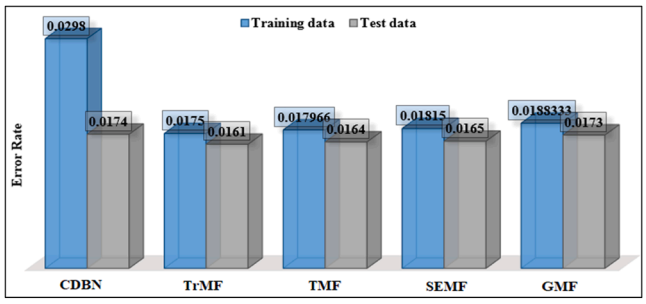


Fig. 10. Error Rate of the training data and test data with 600 no. of units in the hidden layers at 1000 epochs.

Table 4
Log loss and accuracy for MNIST dataset.

	Training		Testing	
	Log Loss	Accuracy (%)	Log Loss	Accuracy (%)
CDBN	0.0071003	99.72	0.034492	98.31
Proposed Approach	0.006887	99.87	0.034669	98.35

significance in deep learning related domains [28,24]. For the MNIST with motion blur dataset, we observe a 2.24 % and 1.17 % improvement of proposed approach over CDBN for training and test data, respectively. The improvement percentage for these two datasets is shown in Fig. 11 (a) & (b). Table 7 shows lower log loss and better accuracy values with the proposed approach for both the datasets. This reflects the similar behavior as of RMSE and Error rate.

Table 5
RMSE and error rate for the MNIST dataset with AWGN.

	Training Data		Test Data	
	RMSE	Error Rate	RMSE	Error Rate
CDBN	0.0292	0.0074	0.08898	0.0437
Proposed approach	0.0289	0.0076	0.08825	0.0436

Table 6
RMSE and error rate for the MNIST dataset with motion blur.

	Training Data		Test Data	
	RMSE	Error Rate	RMSE	Error Rate
CDBN	0.01472	0.00186	0.05465	0.0164
Proposed approach	0.01439	0.0018	0.05401	0.0162

4.6. Experimental analysis: CIFAR-10 dataset

The last dataset we experimented with our proposed approach is the CIFAR-10 dataset [43]. It consists of a total of 60,000 color images of 10 classes. The training data has 50,000 images, while the test data has 10,000 images. The dimensional data for image is $32 \times 32 \times 3$ (3072), which are the three intensity channels at each pixel. We build similar two-hidden-layer neural network, as for the MNIST datasets, with 3072–1000-1000–10 network architecture. Since this dataset is different from the MNIST, we have experimented for various number of hidden nodes till 1000 epochs. The RMSEs and the Error Rate for the training and test data are summarized in Table 8.

Figs. 12 and 13 pictorially depicts the RMSE values for the training and test data over several epochs. The major concern here is the DBN performance of the proposed approach over CDBN, which is evaluated here by better RMSE values. The improvement for the training data is higher as compared to the improvement for the test data; however, we are majorly concerned with the test data. The use of proposed approach results in better performance which implies that a similar performance can be achieved with lesser number of nodes in the hidden layers. This ultimately signifies lesser complexity of the deep learning model.

4.7. Statistical analysis

This section provides the results of the statistical tests conducted to evaluate the randomness associated with the proposed algorithm at several levels. The experiments are executed 10 times to evaluate the statistical significance of the MNIST and n-MNIST results. The testing results of RMSE and Error Rate for these ten iterations are shown in Tables 9 and 10, respectively.

We have performed two different statistical significance tests viz., the One-way ANOVA and the Kruskal-Wallis test. The principle of both these tests is same and they are used to establish the statistical significance of the results. They both return a p-value indicting the probability of the considered distributions of data under the assumption that null hypothesis is true. In the one-way ANOVA case, null hypothesis states that the means of all distributions are same while Kruskal-Wallis considers median to be equal. There is a significant level (α) which is generally assigned a value of 0.05 to test the significance of the outcome. A smaller p-value than α suggests that the observed differences between distributions are statistically significant and thus, the null hypothesis is rejected. On the contrary, a large p-value suggests insufficient evidence to reject the null hypothesis. Along with p-value, one-way ANOVA also

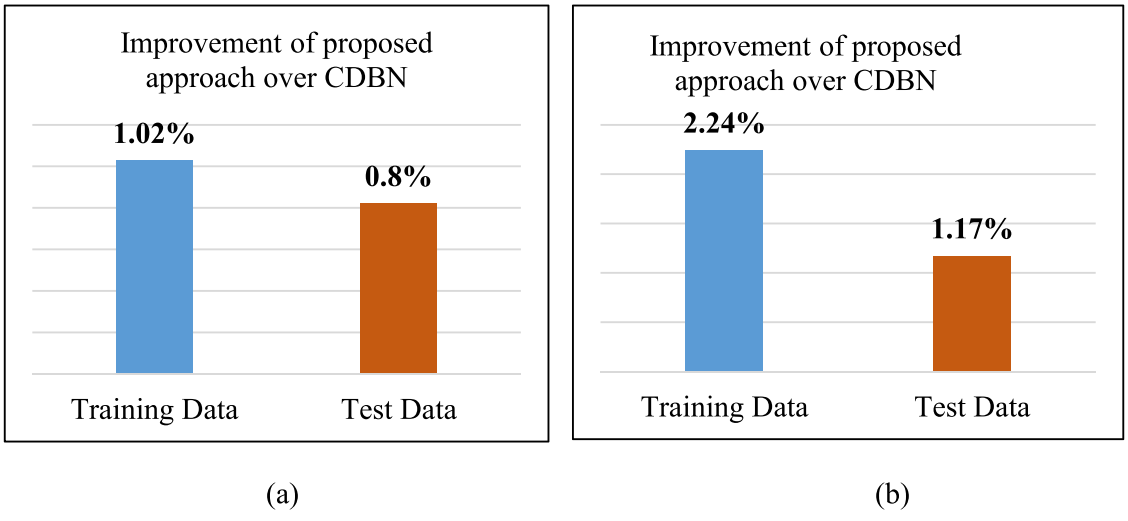


Fig. 11. Improvements percentage in the RMSE of the training data and the test data for (a) MNIST with AWGN and (b) MNIST with motion blur.

Table 7
Log loss and accuracy for MNIST with AWGN and MNIST with motion blur.

		Training		Testing	
		Log Loss	Accuracy (%)	Log Loss	Accuracy (%)
MNIST with AWGN	CDBN	0.028182	99.13	0.070866	95.75
	Proposed Approach	0.027664	99.29	0.068169	95.86
MNIST with Motion Blur	CDBN	0.0079621	99.65	0.032471	98.39
	Proposed Approach	0.0078516	99.78	0.029623	98.44

Table 8
RMSE and error rate for the CIFAR-10 dataset.

Hidden layer nodes	RMSE				Error Rate			
	Training Data		Test Data		Training Data		Test Data	
	CDBN	Proposed Approach	CDBN	Proposed Approach	CDBN	Proposed Approach	CDBN	Proposed Approach
200	0.1056	0.0945	0.2904	0.2825	0.1399	0.1359	0.5064	0.5135
400	0.0945	0.0833	0.2821	0.2719	0.1358	0.1176	0.5163	0.4873
600	0.0768	0.0759	0.2656	0.2626	0.1076	0.1062	0.4696	0.4689
800	0.0726	0.0701	0.2622	0.2601	0.1016	0.0979	0.4584	0.4493
1000	0.071	**0.0681**	0.261	**0.2591**	0.0987	**0.0942**	0.4572	**0.4479**

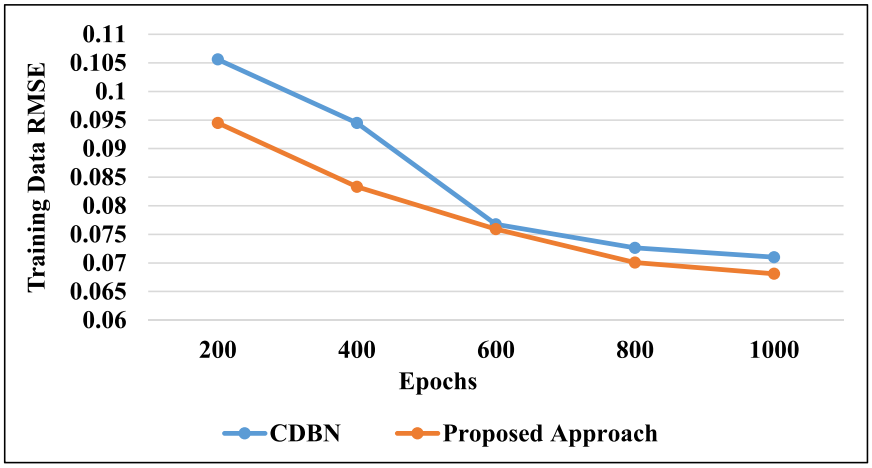


Fig. 12. RMSE vs. epochs plot for the training data.

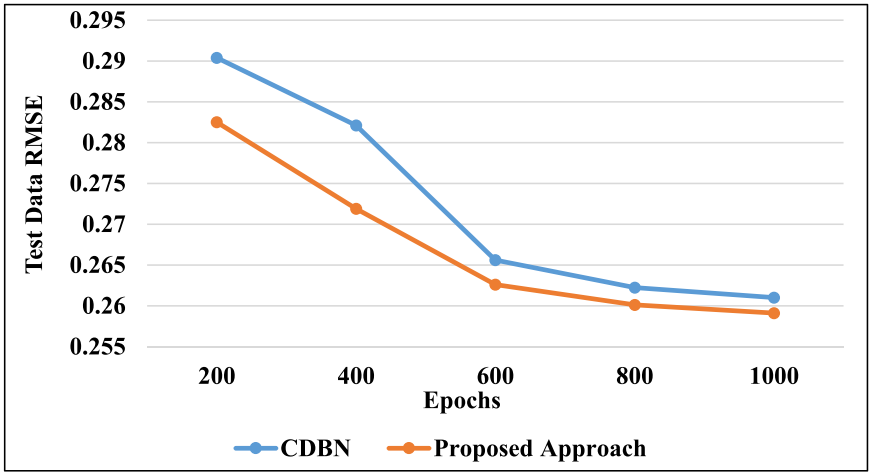


Fig. 13. RMSE vs. epochs plot for the test data.

Table 9
Testing RMSE of the MNIST and n-MNIST datasets for ten iterations.

Iteration	MNIST	MNIST with AWGN	MNIST with Motion Blur
1	0.0503276	0.0888654	0.054838900
2	0.0494991	0.0877254	0.054385100
3	0.054927	0.0883546	0.054260000
4	0.0493901	0.0877960	0.055388300
5	0.0505344	0.0882154	0.053746300
6	0.0501516	0.0888646	0.053828984
7	0.0539636	0.0883141	0.055111282
8	0.0499034	0.0886371	0.055532999
9	0.0501963	0.0889345	0.054505132
10	0.0494943	0.0881542	0.054416433

Table 10
Testing error rate of the MNIST and n-MNIST datasets for ten iterations.

Iteration	MNIST	MNIST with AWGN	MNIST with Motion Blur
1	0.0144	0.0433	0.0151
2	0.0140	0.0414	0.0147
3	0.0157	0.0420	0.0154
4	0.0137	0.0419	0.0158
5	0.0144	0.0429	0.0151
6	0.0145	0.0424	0.0151
7	0.0149	0.0429	0.0152
8	0.0147	0.0422	0.0161
9	0.0145	0.0429	0.0152
10	0.0140	0.0420	0.0156

returns a F value which is computed by comparing the variance between means of the distributions to the variance within distributions.

Table 11 compiles the results of the statistical significance tests on the testing outcomes of the MNIST and n-MINST datasets. All the p-values from both the tests are found as less than α, which establishes that the results are statistically significant. The higher values of F-Statistic, as may be seen in the Table 11, also convey the same meaning. Fig. 14 (a) comparatively shows the box plots of the RMSE values, whereas Fig. 14 (b) shows the same of the Error Rates for all experimented datasets.

Table 11
Statistical test results on testing outcome of MNIST and n-MNIST datasets.

Statistical Tests	One-way ANOVA		Kruskal-Wallis ANOVA
Testing	p-value	F-statistic	p-value
RMSE	2.62E−32	2935.41	7.16E−06
Erorr Rate	6.02E−39	9.14E+03	7.28E−06

5. Discussion, conclusion and future work

This paper introduces a novel deep belief networks model aimed at enhancing performance in deep learning through the utilization of Fuzzy Sets (FSs). In our proposed approach, these FSs are employed to model the randomly assigned and uncertain weights and biases. Various MF shapes such as TrMF, TMF, GMF, and SEMF are considered to analyze the sensitivity of the weights and biases. Extensive experiments and statistical analysis are conducted to test and analyze the effectiveness of the proposed approach over the CDBN.

Four different types of datasets are considered for the experimentations. First, the classical MNIST dataset is experimented with various MF shapes for sensitivity analysis. Considering the RMSE on test data, which is crucial for assessing performance, we observed that we attained the best RMSE with only 600 nodes in the hidden layers using the proposed approach. It signifies the lesser complex DBN architecture could be utilized when weights are modeled with FSs. This extensive experimentation also concludes that we can use TrMF or TMF to consider for the proposed approach since both resulted in approximately same outcome. Therefore, we used TrMF in the proposed approach while experimenting for other datasets. The other two noisy datasets, belonging to the class of n-MNIST, are of the same structure as MNIST but with added noise. Since the structure of the dataset is same, we considered the same 600 nodes for the hidden layers and we concluded significant improvement in the test data RMSE. Notably, a lesser number of nodes produced enhanced performance rather than the whole complex structure. For these datasets, we also computed metric of log loss (also called Cross-entropy loss), which is suitable for evaluation of probabilistic models like DBN. It quantifies the difference between the predicted probability distribution and the actual distribution. In addition, we provided the accuracy of the proposed approach as compared to CDBN. The log loss is lower with the proposed approach signifying the higher probability allocation to correct classes, which also implies better accuracy values. The last dataset is the CIFAR-10 dataset with colored

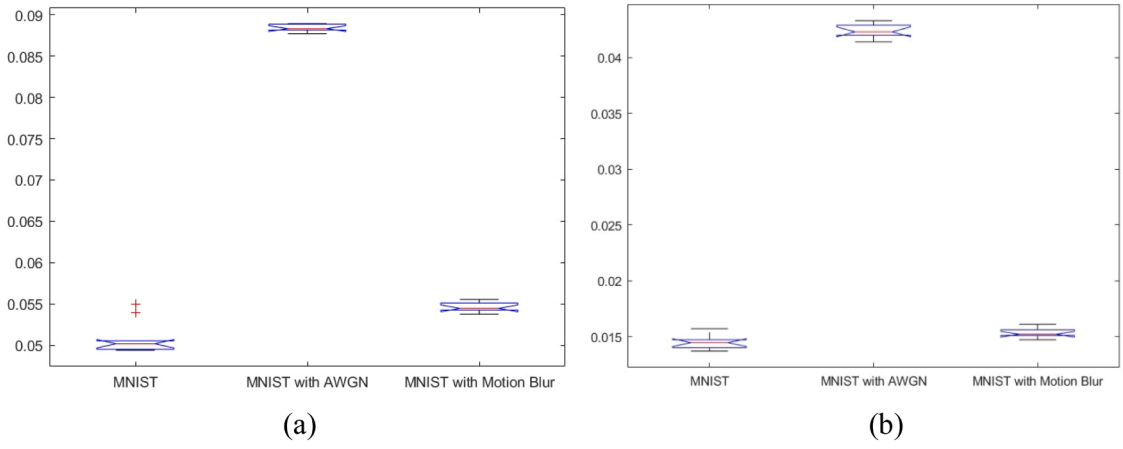


Fig. 14. Box plot of testing results for different datasets: (a) RMSE, and (b) Error Rate.

images. The proposed approach returned better performance across all experimented hidden layer nodes, as compared to the CDBN. Additionally, the one-way ANOVA and Kruskal-Wallis tests are conducted to establish the statistical significance of the results obtained from multiple iterations. Hence, we can conclude that, the proposed approach[1] outperforms the classical DBN with improved RMSE and error rates. One limitation of the proposed approach is that the MF shapes can further be tuned according to the application and dataset under study, which may provide better learning outcome.

Future research works may target extending the work to understand the behavior of FSs over the hidden layers of DBN. The work may also be extended to utilize the interpretability aspects of FSs in the deep learning architectures. Further, in terms of image re-generation task, future research may aim on regenerating images under uncertain environments. Moreover, we are of the view that more suitable uncertainty modelling techniques may yield noteworthy improvements in the learning accuracy of the DBN or other DL models, which surely offers a fertile ground for future research.

CRediT authorship contribution statement

Amit K. Shukla: Writing – review & editing, Writing – original draft, Visualization, Methodology, Investigation, Data curation, Conceptualization. **Pranab K. Muhuri:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Conceptualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Authors are sincerely thankful to the learned editors and the reviewers for all of their valuable comments, which were quite helpful in revising the manuscript, and in improving its overall technical quality.

Data availability

Data information is already mentioned in the manuscript

References

[1] M. Ashrafi, D.K. Prasad, C. Quek, IT2-GSETSK: an evolving interval Type-II TSK fuzzy neural system for online modeling of noisy data, Neurocomputing 407 (2020) 1–11.
[2] S. Basu, M. Karki, S. Ganguly, R. DiBiano, S. Mukhopadhyay, S. Gayaka, R. Nemani, Learning sparse feature representations using probabilistic quadtrees and deep belief nets, Neural Process. Lett. *45* (3) (2017) 855–867.
[3] Y. Bengio, P. Lamblin, D. Popovici, et al., Greedy layer-wise training of deep networks, Adv. Neural Inf. Process. Syst. 19 (2007) 153.
[4] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, Adv. Neural Inf. Process. Syst. 19 (2007) 153.
[5] J.J. Buckley, Y. Hayashi, Fuzzy neural networks: a survey, Fuzzy Sets Syst. 66 (1) (1994) 1–13.
[6] Zhifei Chen, Sara Aghakhani, James Man, Scott Dick, ANCFIS: a neurofuzzy architecture employing complex fuzzy sets, IEEE Trans. Fuzzy Syst. 19 (2) (2011) 305–322.
[7] C.P. Chen, C.Y. Zhang, L. Chen, M. Gan, Fuzzy restricted Boltzmann machine for the enhancement of deep learning, IEEE Trans. Fuzzy Syst. *23* (6) (2015) 2163–2173.
[8] K. Cho, A. Ilin, T. Raiko, Improved learning of Gaussian-Bernoulli restricted Boltzmann machines, Artif. Neural Netw. Mach. Learn. –ICANN 2011 (2011) 10–17.
[9] J. Chu, H. Wang, H. Meng, P. Jin, T. Li, Restricted boltzmann machines with gaussian visible units guided by pairwise constraints, IEEE Trans. Cybern. (99) (2018) 1–14.
[10] X. Cui, F. Yang, X. Wang, B. Ai, Y. Luo, D. Ma, Deep learning model for seabed sediment classification based on fuzzy ranking feature optimization, Mar. Geol. *432* (2021) 106390.
[11] G. Desjardins, A. Courville, Y. Bengio, P. Vincent, O. DelalleauTempered Markov chain Monte Carlo for training of restricted Boltzmann machines. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. pp. 145-152. .
[12] S. Feng, C.P. Chen, C.Y. Zhang, A fuzzy deep model based on fuzzy restricted boltzmann machines for high-dimensional data classification, IEEE Trans. Fuzzy Syst. *28* (7) (2019) 1344–1355.
[13] S. Feng, C.P. Chen, A fuzzy restricted boltzmann machine: novel learning algorithms based on crisp possibilistic mean value of fuzzy numbers, IEEE Trans. Fuzzy Syst. (2016).
[14] Feuring, T. (1996, June). Learning in fuzzy neural networks. In Neural Networks, 1996., IEEE International Conference on(Vol. 2, pp. 1061-1066). IEEE.
[15] M. Gong, J. Liu, H. Li, Q. Cai, L. Su, A multiobjective sparse feature learning model for deep neural networks, IEEE Trans. Neural Netw. Learn. Syst. *26* (12) (2015) 3263–3277.
[16] M. Gösgens, A. Zhiyanov, A. Tikhonov, L. Prokhorenkova, Good classification measures and how to find them, Adv. Neural Inf. Process. Syst. *34* (2021) 17136–17147.
[17] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, Adv. Neural Inf. Process. Syst. (2015) 1135–1143.
[18] Y. Hayashi, J.J. Buckley, E. Czogala, Fuzzy neural network with fuzzy signals and weights, Int. J. Intell. Syst. 8 (4) (1993) 527–537.
[19] D. Hidalgo, O. Castillo, P. Melin, Type-1 and type-2 fuzzy inference systems as integration methods in modular neural networks for multimodal biometry and its optimization with genetic algorithms, Inf. Sci. *179* (13) (2009) 2123–2145.
[20] G.E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Comput. *14* (8) (2002) 1771–1800.
[21] G. Hinton,"A practical guide to training restricted Boltzmann Machines," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep. UTML TR 2010-003, 2010.
[22] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[23] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science *313* (5786) (2006) 504–507.
[24] Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., … & Adam, H. (2019). Searching for mobilenetv3. In: Proceedings of the IEEE/CVF international conference on computer vision (pp. 1314-1324).
[25] S. Huang, G. Zhao, Z. Weng, S. Ma, Trapezoidal type-2 fuzzy inference system with tensor unfolding structure learning method, Neurocomputing 473 (2022) 54–67.
[26] H. Ishibuchi, H. Tanaka, H. Okada, An architecture of neural networks with interval weights and its application to fuzzy regression analysis, Fuzzy Sets Syst. 57 (1) (1993) 27–39.
[27] Z. Jiang, S. Gao, M. Li, An improved advertising CTR prediction approach based on the fuzzy deep neural network, PLOS One *13* (5) (2018) e0190831.
[28] M. Jiang, J. Liu, L. Zhang, C. Liu, An improved Stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms, Phys. A: Stat. Mech. Appl. 541 (2020) 122272.
[29] M.V. Jisha, D. Vimal, Population based optimized and condensed fuzzy deep belief network for credit card fraudulent detection, Int. J. Adv. Comput. Sci. Appl. *11* (10.) (2020) 14569.
[30] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[31] G. Li, L. Deng, Y. Xu, C. Wen, W. Wang, J. Pei, L. Shi, Temperature based restricted Boltzmann machines, Sci. Rep. *6* (2016) 19133.
[32] Liu, M., Li, Y., & Wang, Z. (2021). A propelled multiple fusion deep belief network for weld defects detection. In: Proceedings of the 2021 2nd International Conference on Control, Robotics and Intelligent System (pp. 141-146).
[33] A.R. Mohamed, G. Dahl, G. Hinton, Deep belief networks for phone recognition, Nips Workshop Deep Learn. Speech Recognit. Relat. Appl. 1 (9) (2009) 39.
[34] P.K. Muhuri, A.K. Shukla, Semi-elliptic membership function: Representation, generation, operations, defuzzification, ranking and its application to the real-time task scheduling problem, Eng. Appl. Artif. Intell. *60* (2017) 71–82.
[35] J.P. Papa, W. Scheirer, D.D. Cox, Fine-tuning deep belief networks using harmony search, Appl. Soft Comput. *46* (2016) 875–885.
[36] W. Pedrycz, Why triangular membership functions? Fuzzy Sets Syst. *64* (1) (1994) 21–30.
[37] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Cogn. Model. *5* (3) (1988) 1.
[38] R. Salakhutdinov, G. Hinton, Semantic hashing, RBM 500 (3) (2007) 500.
[39] S. Samui, I. Chakrabarti, S.K. Ghosh, Time–frequency masking based supervised speech enhancement framework using fuzzy deep belief network. Appl. Soft Comput. *74* (2019) 583–602.
[40] Tanaka, M., & Okutomi, M. (2014, August). A novel inference of a restricted boltzmann machine. In: Proceedings of the Pattern Recognition (ICPR), 2014 22nd International Conference on (pp. 1526-1531). IEEE.
[41] G.W. Taylor, G.E. Hinton, S.T. Roweis, Modeling human motion using binary latent variables, Adv. Neural Inf. Process. Syst. *19* (2007) 1345.
[42] G.W. Taylor, G.E. Hinton, S.T. Roweis, Modeling human motion using binary latent variables, Adv. Neural Inf. Process. Syst. *19* (2007) 1345.

[1] The MATLAB code of the proposed approach are made freely available on GitHub: https://t.ly/GGk22 for its further use, extension, and improvement by the researchers and practitioners.

[43] A. Torralba, R. Fergus, W.T. Freeman, 80 million tiny images: a large data set for nonparametric object and scene recognition, IEEE Trans. Pattern Anal. Mach. Intell. *30* (11) (2008) 1958–1970.
[44] S. Wang, C. Manning, Fast dropout training, Int. Conf. Mach. Learn. (ICML) (2013) 118–126.
[45] M. Wang, Z.H. Ning, T. Li, C.B. Xiao, Information geometry enhanced fuzzy deep belief networks for sentiment classification, Int. J. Mach. Learn. Cybern. *10* (11) (2019) 3031–3042.
[46] X. Zhang, F.L. Chung, S. Wang, An interpretable fuzzy DBN-based classifier for indoor user movement prediction in ambient assisted living applications, IEEE Trans. Ind. Inform. *16* (1) (2019) 42–53.
[47] W. Zhao, L. Xu, J. Bai, M. Ji, T. Runge, Sensor-based risk perception ability network design for drivers in snow and ice environmental freeway: a deep learning and rough sets approach, Soft Comput. *22* (2018) 1457–1466.
[48] S. Zhou, Q. Chen, X. Wang, Fuzzy deep belief networks for semi-supervised sentiment classification, Neurocomputing 131 (2014) 312–322.

Amit K. Shukla (Senior Member, IEEE) received his master's degree (gold medal holder) and Ph.D. degree in computer science from South Asian University, New Delhi, India. He is currently an Associate Professor at the School of Technology and Innovations, University of Vaasa, Finland. His research areas include: fuzzy sets and systems, anomaly detection, industry 4.0 systems, transfer learning, evolutionary optimization. Dr. Shukla was the recipient of the prestigious INSPIRE fellowship from the Department of Science and Technology, Government of India during his Ph.D. tenure. Presently, he is serving as an Associate Editor of journal of Heliyon, Elsevier, along with actively reviewing publications from various journals such as: IEEE Transactions on Fuzzy Systems, IEEE Transactions on Industrial Informatics, Information Sciences, Applied Soft Computing, Engineering Applications of Artificial Intelligence, Neural Computing and Applications, etc.

Pranab K. Muhuri (Senior Member,IEEE) was born in Chittagong, Bangladesh.He received the Ph.D. degree in ComputerEngineering in 2005 from IT-BHU [nowIndian Institute of Technology (BHU)],Varanasi, India. He is currently a Professorwith the Department of Computer Science,South Asian University, New Delhi, India,where he is leading the computational intelligence and real-timesystems research groups. Pranab's current research interests aremainly in real-time systems, fuzzy systems, evolutionaryalgorithms, perceptual computing, and machine learning.Pranab has published extensively with more than 150 papers inreputed journals and conferences including IEEE Transactionson Fuzzy Systems, IEEE Transactions on Cybernetics, IEEETransactions on Sustainable Computing, ReliabilityEngineering and Systems Safety, Fuzzy Sets and Systems, Applied Soft Computing, Computers and IndustrialEngineering, and Future Generation Computer Systems. Pranabhas guided more than 10 PhD theses and about 50 Master'-sdissertations. Currently, he is serving as editorial board memberin several journals including Applied Soft Computing andEngineering Applications of Artificial Intelligence.