Fuzzy Sets and Systems 477 (2024) 108744

Contents lists available at ScienceDirect

Fuzzy Sets and Systems

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

# A novel deep belief network architecture with interval type-2 fuzzy set based uncertain parameters towards enhanced learning

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

[a] School of Technology and Innovations, University of Vaasa, Wolffintie 34, Vaasa FI-65200, Finland
[b] Department of Computer Science, South Asian University, Rajpur Road, Maidan Garhi, New Delhi 110068, India

ARTICLE INFO

*Keywords:*
Deep neural networks
Deep belief networks
Restricted Boltzmann machine
Interval type-2 fuzzy sets
Contrastive divergence

ABSTRACT

This paper proposes a novel Deep Belief Network (DBN) architecture, the 'Interval Type-2 Fuzzy DBN (IT2FDBN)', which models the weights and biases with IT2 FSs. Thus, it introduces a novel algorithm for augmented deep leaning, which has the capability to address all the limitations of the classical DBN (CDBN) and T1 fuzzy DBN (T1FDBN). We comparatively evaluate the performance of the IT2FDBN by conducting experiments using the popular MNIST handwritten digit recognition datasets. Additionally, to demonstrate its robustness and generalization capabilities, we also conduct experiments taking two noisy variants of MNIST dataset, viz. the MNIST with AWGN (additive white Gaussian noise) and the MNIST with motion blur. We conduct extensive simulations by considering different combinations of nodes in the hidden layers of the DBN for better model selection. We thoroughly compare the results using well-known performance measures such as root mean square error (RMSE) and Error rate. We show that, in terms of RMSE values and error rates, the proposed IT2FDBN outperforms both T1FDBN and CDBN across all the three datasets. Further, we also provide the results of convergence, runtime-based comparison, and statistical analysis in support of our proposal.

## 1. Introduction

The resurgence of deep learning (DL) models has been accredited to Hinton and his co researchers [1,2]. It includes the introduction of deep belief networks (DBNs) and unsupervised pre-training. Over the years DBN's have been successfully used in achieving remarkable results in dimensionality reduction [1], semantic hashing [3], motion capture [4], data classification [5], etc. DL as a whole can address the problems of learning complex nonlinear architectures which has set new benchmarks in terms of its huge applicability in a number of real-life problems such as sentiment classification [6], speech recognition [7], time-series forecasting [8] etc. DBNs are formed by stacking several Restricted Boltzmann Machines (RBMs), which are generative, neural networks having two layers [9], in which the first layer is formed by the visible units, which correspond to the components of an observation, whereas the hidden units, which constitute the second layer, model dependencies between the components of observations [10]. In a DBN, the states inferred by the hidden units of an RBM can be thought of as the inputs for the visible units of another RBM that can learn to model dependencies between the hidden units of the former machine. Therefore, DBNs may have one input layer, one output layer, and multiple hidden layers. Fig. 1 shows such a DBN with one visible layer, three hidden layers and one output layer. The top layer in a DBN is an undirected RBM and each pair of the layer is considered as an individual RBM. The hidden layer of the last RBM in the DBN is

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

https://doi.org/10.1016/j.fss.2023.108744
Received 6 June 2023; Received in revised form 11 September 2023; Accepted 6 October 2023
Available online 10 October 2023

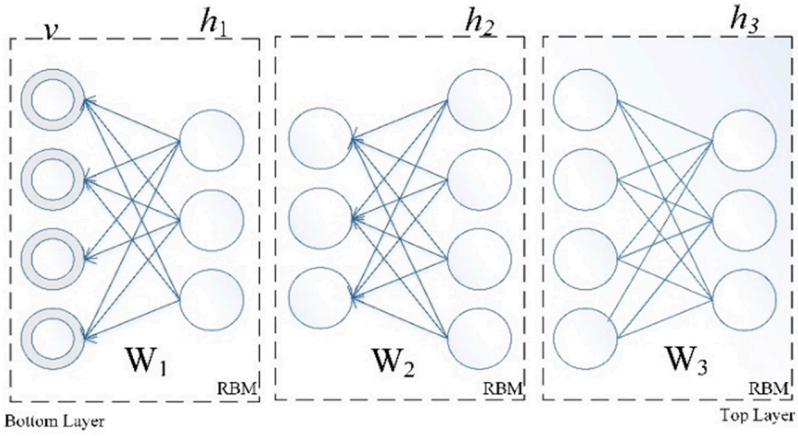


**Fig. 1.** A typical deep-belief network.

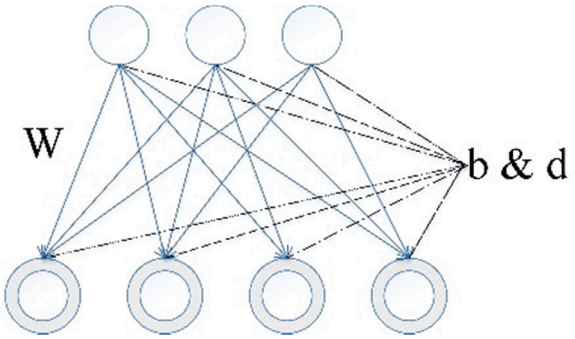


**Fig. 2.** Representation of biases in a DBN.

ultimately the output layer. Here, $v$ represents the visible layer, $\{h_1, h_2, h_3\}$ are the hidden layers and $W = \{W_1, W_2, W_3\}$ represents the weights associated with the respective layers. The corresponding biases to each node are represented by b and d as shown in the Fig. 2. The DBN hence formed has two undirected top layers, but the lower layers have top-down directed connections.

Training of DBNs refers to adjusting the weights and the biases, collectively called "parameters'', of the network in such a way that the probability distribution represented by the machine closely resembles the provided data. The training of complete DBN follows the greedy approach. At first, features are learned from the visible layer using the "contrastive divergence (CD)'' algorithm [2]. Then, the features of features are learned in the second hidden layer while the activation values of the previously trained features are treated as visible units. In the end, a complete DBN is trained once the learning of the final hidden layer is finished. This efficient greedy approach is followed by another learning algorithm called "classical backpropagation [11] to fine-tune the entire network, in which the DBN slightly adjusts the weights obtained during the pre-training phase. Here, the first trained RBM finds the local optimum and these values are taken as input to the next stacked RBM, which again tries to find the local optimum. Thus, after the final training, it is very likely that a global optimum is achieved.

In each learning step of a network, errors are calculated after random weight initialization. Weights are then updated to help the network to predict the better output with enhanced performance during learning. The weights are chosen sequentially as they learn from the previous training. Thus, at each node in a typical DBN, various computations are performed based on the selection of these weights. Since, DBNs converge to a global minimum, different initializations of the weights will cause the loss function to be minimized to a different value. The initialization of the weights and the biases during training are very crucial for the DBN to produce the output with better accuracy. In DBNs and other similar architectures, these parameter initializations are usually made randomly assuming them to be real numbers [6,7,12], and more recently as type-1 fuzzy sets (T1 FSs) [13,14]. However, consideration of real-valued and even T1 fuzzy weights and biases hinders the performance of the DBN and has the following major disadvantages:

(i) It makes the classical DBNs (CDBNs) and T1 fuzzy DBNs (T1FDBNs) limited in their representational and generalization capabilities.
(ii) It disregards the uncertainty associated in the problem variables, which usually originates from multiple sources, a common scenario in many real-life problems.
(iii) It affects the robustness of the DBN as it limits the training with noisy data.

(iv) It makes the learning spaces of the weights and the biases quite small.

All the above limitations are originated from the fact that while modelling the weights and biases with the standard or T1 FSs, it becomes difficult to come up with an exact membership function for a particular parameter. Instead, it is quite natural for the weights and biases to lie within some intervals. Also, the dimension of weights and biases in the complex deep neural networks is so huge that it is impossible for traditional FSs to model the whole uncertainty space properly. However, interval valued FSs (IVFSs), which considers membership degrees as intervals, provides better capturing of parameter variability, robustness to noise, and more flexibility to model the uncertainty [15–17]. IVFSs allow us to consider the scenarios where the exact membership function values are difficult to estimate [17,18]. Accordingly, IVFSs are more suitable to be considered to represent the weights and biases of the deep architectures as they possess the abilities to overcome all the above listed limitations of their real-valued and T1 FS-based counterparts. Moreover, as Sola et al. in [19,20] further argued that interval type-2 fuzzy sets (IT2 FSs) offer a more general presentation of the IVFSs, and since comparatively more and more works in the literature are considering this terminology (of IT2 FSs), we propose to consider IT2 FSs for modelling the weights and biases of deep architectures. In addition, IT2 FSs also handles interpretability issues better and provides enhanced information representation [21–26].

Therefore, this paper models the weights and biases of DBN with IT2 FSs to address all the above limitations and proposes a novel DBN architecture, called Interval Type-2 Fuzzy DBN (IT2FDBN) which introduces a novel algorithm for augmented deep leaning. Since we consider IT2 FSs to represent the weights and the biases, we can say that the proposed IT2FDBN is the generalized DBN, of which T1FDBN and CDBN are two special cases. Also, it has been shown in a number of recent studies ([21] -[28]) that IT2 FSs are more suitable to model the uncertainties originating from multiple sources, so the proposed IT2FDBN may be more robust than both T1FDBN and CDBN. Mendel, in one of his very recent works [27], has shown that the state space is sculpted better using IT2 FSs instead of T1 FSs, which therefore suggests improved performance by our proposed IT2DBN, because it provides wider learning spaces for the weights and the biases.

We have implemented the IT2FDBN (as well as the T1FDBN and CDBN) for experiments and utilized the popular MNIST handwriting recognition datasets along with two other modified versions that include additive noise, MNIST with AWGN (additive white Gaussian noise) and MNIST with motion blur. Results are thoroughly compared using well-known performance measures such as root mean square error (RMSE) and error rate. It has been found that, across all three datasets, the proposed IT2FDBN outperforms both T1FDBN and CDBN in terms of RMSE values and error rates. Moreover, higher improvements in the RMSE given by IT2FDBN over both T1FDBN and CDBN in the cases of MNIST with AWGN and MNIST with motion blur datasets demonstrates that it is more robust than its other two competitors in handling noisy and corrupt data. Additionally, we have investigated the model selection problem from the context of the IT2FDBN, T1FDBN and CDBN. It has been found that for all the architectures, the optimal model is when there are 600 units in the hidden layers. We have also experimented the convergence of training and test data for different epochs. Further, we have provided the running time comparison of the CDBN, T1FDBN, and IT2FDBN for different number of nodes in the hidden layer. Two well-known tests: the Kruskal-Wallis test and the Wilcoxcon Test are considered to establish the statistical significance of the proposed approach. The major contributions of this paper may be summarized as follows:

1) It proposes a novel algorithm, IT2FDBN, for augmented deep leaning, which considers uncertain weights and biases as IT2 FSs to provide novel formulations for the energy functions and the learning rules.
2) It proposes a novel algorithm to generate IT2 FSs to be used as DBN parameters.
3) IT2FDBN outperforms both CDBN and T1FDBN in terms of the performance measures and generalization capability and ensures enhanced deep learning.
4) The proposed approach demonstrates its robustness when experimented on the popular MNIST dataset and two of its noisy variants: MNIST with additive Gaussian noise, and MNIST with motion blur.
5) Rigorous experiments are performed to empirically identify the best model in terms of optimal number of nodes in the hidden layers of a DBN.

The rest of the paper is organized as follows. Section 2 provides a brief review of the related research works and a comparative overview of the state-of-the-art T2 FSs-based DL approaches. Section 3 presents the mathematical foundations required to understand the approach proposed in this paper. Section 4 demonstrates the problem formulation and proposed technique. Section 5 explains the experimental setup and the results. Finally, Section 6 discusses the outcomes and proposes some future research.

## 2. Literature survey

Due to the recent boost in their usage by the research community, authors now focus on better learning algorithms for RBMs and DBNs [2,28]. A novel learning algorithm for RBMs, with many hidden layer variables, was proposed by Salakhutdinov and Hinton [29]. Hinton also gave the most commonly used method for training RBMs, the CD algorithm in [9]. Moreover, there are a few conventional domains where RBMs have found applicability, such as combinatorial optimization [30], biomedical sciences [31], Ganglion cell population activity [32], knowledge extraction [33], and learning informative features [34]. An extended survey on the soft computing techniques and applications used with deep learning approaches is given by Zhang et al. in [35]. A problem concerning the process of fine tuning the parameters in DBNs was studied by Papa et al. in [36]. The works on active learning in the domain of multimedia annotation and retrieval with the application in content-based retrieval was explored by Wang and Hua in [37]. Han et al. [38] proposed a unique method to train the neural network (NN) in such a way that only the important connections were learned and

**Table 1**
Comparative overview of the current work with respect to other state-of-the-art T2 FSs-based DL approaches.

| Ref. | Fuzzy Methodology | DL Model | Application | Dataset | Noisy dataset considered | Whether model selection is done? | Convergence test done? | Runtime comparison provided? | Statistical analysis included? |
|---|---|---|---|---|---|---|---|---|---|
| [48] | T2 FSs | Deep CNN | Classification | Medical imagining datasets | No | Yes | Yes, 60 epochs | No | Yes, Wilcoxon signed-rank test |
| [49] | Type-2 FT and (FET) | Deep CNN | Segmentation | MED-NODE. SD-260, & SD-198 | Yes, images with nonuniform background illumination | No | Yes, 150 epochs | Yes | Yes, *t*-test to compare the mean of the metrices |
| [50] | IT2 FSs | RBMs | Classification | MNIST & FASHION-MNIST | Yes, salt and pepper noise | No | Yes, 20 epochs | No | No |
| [51] | T2 FLSs | Multi-layer perceptron NN using RBMs | hyperchaotic Chen systems | Traditional induction motor signalling data | Yes, a white noise with mean/variance of 0/0.5 is considered | No | No | Yes | No |
| [52] | SIT2-FRU | DNN | Classification | MNIST, Quickdraw, Pictionary & CIFAR-10 | No | No | Yes, for 20 epochs & decrease the learning rate by a factor of 10 | No | No |
| [53] | Chaotic T2 transient-fuzzy approach | DNN | Financial Prediction | Financial forecast time-series data | No | Yes, for each hidden layer, 400 LORS hidden nodes are used | Yes, 500 epochs, low RMSE in 70 epochs | No | No |
| [54] | Interval T2 FCM & FNN | ResU-segNet | Segmentation & Classification | INbreast and a private database by Hospital | No | No | Yes, 300 epochs | No | No |
| [55] | FT2FDNN | 3D-CNN & DNN | Classification | Real-time image/video data of facial expression | No | No | Yes, 100 epochs | No | Friedman 2-way statistical test |
| [56] | T2 FSs | LSTM | Speech enhancement | LibriSpeech database | Yes, dataset is mixed with 15 noise types from the DEMAND database | No | Yes, 10 epochs | Yes | Yes, ANOVA test |
| [57] | IT2 FSs | DBN | Classification | MNIST | No | No | No | No | No |
| **Proposed IT2FDBN** | IT2 FSs for uncertain weights and biases | DBN | Digit Recognition, classification | MNIST, MNIST with AWGN, MNIST with Motion blur | Yes, MSNIT with AWGN & MNIST with Motion blur | Yes, 200, 400, 600, 800, 1000, 1200, and 1600 no. of nodes considered in each hidden layer | Yes, convergence of training & test data for epochs: 100, 200, 400, 800, & 1000 | Yes, run time of CDBN, T1FDBN, and IT2FDBN for different nodes | Yes, Kruskal-Wallis test and Wilcoxcon Test |

stored. Gong et al. [39] proposed a multi-objective sparse feature learning based model which was entirely grounded on auto encoders. Further, they converted the constrained single-objective optimization problem to a multi-objective optimization problem (MOOP). The learning procedure in sparse response deep belief network could be used to design a MO induced learning technique as proposed in [39].

For training RBMs, Tanaka and Okutomi [12] focused on the Bernoulli distribution out of the many available probability distributions like the binary [1], Gaussian [40], etc. They proposed a novel inference for an RBM by considering the probabilities of the Bernoulli distribution as expectation of random variables. This was in contrast to the proposal made by Hinton in [9], where he used probabilities without sampling when hidden units were driven by reconstructions, and always used the stochastic binary states when hidden states are driven by data. In general formulations, the probabilities of the Bernoulli distribution were considered as expectations of the random variables, i.e. all calculations with the probabilities perform the expectation before applying the activation function. In their paper, Tanaka and Okutomi, applied the activation function and then proceeded to expectation. However, implementation of such an inference was inflexible. So, they also presented a closed form approximation of their inference method. Expectation was computed using the closed form approximation as mentioned by Wang and Manning [41]. Pre-training of the inference was performed by the CD learning algorithm.

For any DBN framework, parameter initialization is one of the crucial tasks. In the earlier models such as McCulloch and Pitts neuron [42], the weights were assigned by the human operator. Rosenblatt's perceptron learned the weights based on the input values from various categories such as 0 and 1. Bengio [31] discussed that weights need to be initialized cautiously to break the symmetry during the back propagation. Some techniques were discussed in the literature for basic weight initialization strategies. LeCun et al. [43] and Glorot and Bengio proposed the selection of weights from a uniform distribution in the range $\left[\frac{-1}{(fan-in)}, \frac{1}{(fan-in)}\right]$. Here, $(fan-in)$ is the number of inputs to a hidden unit. One of the other techniques used for weight initialization is the orthogonal random matrix initialization [44].

There has also been some research regarding the uncertainty modelling in different DL models. Chen et al. [13] proposed a Fuzzy RBM (FRBM) for the Enhancement of Deep Learning, where all the parameters of RBM were modelled as type-1 fuzzy numbers. Further, extending their own work, authors proposed learning algorithms for symmetric triangular fuzzy numbers (STFNs), asymmetric triangular fuzzy numbers (ATFNs) and Gaussian fuzzy numbers for the FRBM in [14]. Zhou et al. [6] proposed FDBN 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. In all the above-mentioned works, the authors used T1 FSs in the framework of the learning to model the parameters. There were no noticeable contributions where non-traditional FSs such as T2 FSs were used for parameter initialization in DNN's. Recently, IT2 FSs with NNs, called IT2FNNs were also successfully utilized in the time-series prediction application [45,46]. Extending the work in similar application domain, Castillo et al. [47] introduced Interval Type-3 FSs in ensemble for NNs.

In the literature, there were few hybrid models which considered T2 FSs for some other aspects of the problems. However, they did not model network parameters using T2 FSs. In a mixed framework of using T2 fuzzy uncertainty modelling and DL, Hermessi et al. [48] utilized DL with transfer learning to train a deep convolutional neural network (CNN) for soft tissue sarcomas classification. For improving classification accuracy, they used T2 fuzzy based fusion method for magnetic resonance images. In similar medical image classification task, Venugopal et al. [49] proposed deep threshold prediction network and utilized type-2 fuzzy-based thresholding (Type-2 FT) and fuzzy entropy thresholding (FET) for comparison purpose only. IT2 FSs and T2 FLSs were used with the RBM's which are the underlying model of DBN [50,51]. The T2 fuzzy variants such as Interval Type-2 Fuzzy Rule-Based Unit (SIT2-FRU) [52] and Chaotic T2 transient-fuzzy approach [53] were used with DNN in applications of financial prediction and classification, respectively. For network training, Lee [53] used 400 Lee Oscillator-based Recurrent Neural Network System (LORS) hidden nodes for each hidden layer. Shen et al. [54] utilized IT2 possibilistic fuzzy c-means (FCM) & Fuzzy NN (FNN) in CNN based on residual learning & U-Net (i.e., ResU-segNet) for breast cancer diagnosis. Ghosh et al. [55] explored image processing for studying the emotional changes of gamers using 3-dimensional convolutional neural network (3D-CNN) which is fused with Type-2 Fuzzy Deep Neural Network (FT2FDNN) for EEG signal processing. Recently, Saleem et al. [56] proposed a U-shaped low-complexity type-2 fuzzy LSTM neural network for speech enhancement.

Some early results on the use of T2 FSs in deep learning which considered only one baseline dataset with only one network configuration without any convergence test, runtime comparison and statistical test was presented by Shukla et al. at the Fuzz-IEEE [57]. However, the current paper provides detailed mathematical formulations of our proposed IT2FDBN approach. It includes exhaustive explanations of the background, motivation, and step-wise working of the proposed IT2FDBN approach. Here, the proposed approach has been presented both in flow-chart and pseudocode formats, and all its steps are elaborately discussed. It has considered three different datasets: the MNIST dataset, and two noisy datasets: the MNIST with AWGN dataset and the MNIST with motion blur dataset. The deep architecture proposed in the current work considered two hidden layers having varying number of nodes for better model selection. Specifically, 200, 400, 600, 800, 1000, 1200, and 1600 numbers of nodes in each hidden layer are tested to identify

the best network architecture. Moreover, current work has experimented the convergence of training and test data for different epochs: 100, 200, 400, 600, 800, and 1000. Importantly, it provides the running time comparison of the CDBN, T1FDBN, and IT2FDBN for different number of nodes in the hidden layer. Additionally, in this work, statistical significance of the proposed approach is established taking two popular tests: the Kruskal-Wallis test and the Wilcoxcon Test.

Table 1 lists down some most recent works which used the T2 fuzzy sets within DL frameworks, and hence presents a comparative overview of the current work with respect to other state-of-the-art approaches in terms of their various details such as used methodology, data set, application considered, etc.

## 3. Mathematical foundations of fuzzy sets

In this section, we give the basic definitions and mathematical formulations related to the T1 FS, T2 FS, IT2 FS and its FOU.

**Type-1 Fuzzy Sets:** In traditional or ordinary fuzzy sets, membership grades for each element are articulated as crisp values. Membership grade is the degree to which an element belongs to a set. A T1 FS '$T$' can be expressed mathematically as:

$$T = \{(x,\ \mu_T(x))|\ x \in X\} \tag{1}$$

or

$$T = \int_{x \in X} \mu_T(x) / x \tag{2}$$

here, $\mu_T(x)$ is the membership degree of an element $x$ for each $x \in X$. When $X$ is discrete, replace $\int$ in Eq. (2) with $\sum$.

**Type-2 Fuzzy Sets** [58–63]: Represented by $\widetilde{T}$, A T2 FS is characterized by a higher order membership function $\mu_{\widetilde{T}}(x,u)$, referred to as type-2 membership function (T2 MF), where $x \in X$ and $u \in [0,1]$, i.e.:

$$\widetilde{T} = \{((x,u), \mu_{\widetilde{T}}(x,u))|\ x \in X, u \in [0,1]\} \tag{3}$$

here, $x$ is a primary variable and $u$ is a secondary variable for each $x \in X$ and $0 \leq \mu_{\widetilde{T}}(x,u) \leq 1$. $X$ denotes the universe of discourse and primary membership degree of $x$ is represented by $J_x$, where [64]:

$$J_x = \{(x,u)|u \in [0,1],\ \mu_{\widetilde{T}}(x,u) > 0\} \tag{4}$$

Moreover, for continuous cases, $\widetilde{T}$ may be expressed as:

$$\widetilde{T} = \int_{x \in X} \int_{u \in [0,1]} \mu_{\widetilde{T}}(x,u) / (x,u) \tag{5}$$

**Interval type-2 fuzzy sets (IT2 FSs)** [62,63]: IT2 FSs are a special case of T2 FSs, for which the membership grade is 1 over the entire primary domain. They are denoted here as $\widetilde{T}_I$ i.e.:

$$\widetilde{T}_I = \{((x,u),\ 1)|\ x \in X,\ u \in\ [0,1]\} \tag{6}$$

**Footprint of Uncertainty (FOU):** An IT2 FS is characterized by its FOU, which is bounded by two T1 MFs called the lower membership function (LMF) and the upper membership function (UMF), denoted $u_1$ and $u_2$, respectively. The shaded area in Fig. 3 denotes an FOU bounded by two trapezoidal shaped LMF and UMF. Here, $\left(\underline{a}_L,\ \delta,\ \underline{a}',\ \underline{a}_R\right)$ and $(\bar{a}_L,\ \bar{a},\ \bar{a}',\bar{a}_R)$ denote the parametric representations for

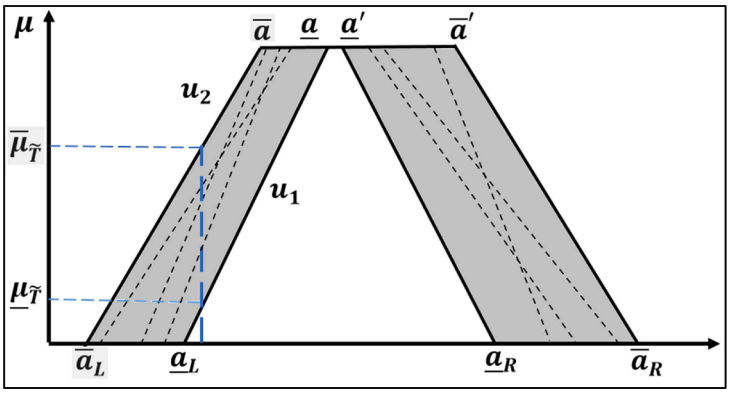


**Fig. 3.** A typical IT2 FS.

the *LMF*($x$) and the *UMF*($x$), respectively. Therefore, we will always have two corresponding membership values for each point on the $x$-axis in the range of $0 \leq \mu_{\tilde{T}}(x) \leq 1$, represented as $\mu_{\tilde{T}}(x) = \left[\underline{\mu}_{\tilde{T}}, \bar{\mu}_{\tilde{T}}\right]$, as shown in Fig. 3. The two trapezoidal MFs, LMF and UMF, are:

$$\underline{\mu}_{\tilde{T}}(x) = \begin{cases} \dfrac{x - \underline{a}_L}{\underline{a} - \underline{a}_L}, & \underline{a}_L \leq x \leq \underline{a} \\ \delta \ \underline{a} \leq x \leq \underline{a}' \\ \dfrac{\underline{a}_R - x}{\underline{a}_R - \underline{a}}, & \underline{a}' \leq x \leq \underline{a}_R \end{cases} \tag{7}$$

$$\bar{\mu}_{\tilde{T}}(x) = \begin{cases} \dfrac{x - \bar{a}_L}{\bar{a} - \bar{a}_L}, & \bar{a}_L \leq x \leq \bar{a} \\ 1 \ \bar{a} \leq x \leq \bar{a}' \\ \dfrac{\bar{a}_R - x}{\bar{a}_R - \bar{a}}, & \bar{a}' \leq x \leq \bar{a}_R \end{cases} \tag{8}$$

## 4. Proposed interval type −2 fuzzy deep belief networks

This section presents our proposed technique to enhance DL in DBNs by modelling the uncertain network parameters using IT2 FSs. As explained earlier, DBNs are formed by stacked RBMs. These networks, which are trained using backpropagation and gradient based methods, suffer from the vanishing gradient problem where the magnitude of the gradient becomes either extremely small or extremely large in the initial steps or towards the last steps in the network [65]. To avoid such problems during the training of RBMs, Hinton proposed the CD algorithm for training RBMs [9]. The key idea is to perform alternating steps of 'updates' and 'reconstructions' to train the stacked RBMs in a DBN. For any RBM, the energy of the joint configurations of the visible and the hidden units with IT2 fuzzy weights and biases is given by[1]:

$$\tilde{E}(v, h; \tilde{W}, \tilde{b}\tilde{d}) = -h^T \tilde{W} v - \tilde{b}^T h - \tilde{d}^T v \tag{9}$$

where, $\tilde{E}$ is the IT2 fuzzy energy function. Every possible pair of the visible and the hidden states are assigned probabilities. A joint probability, denoted by $p$, with any vector argument $o$, may be defined as follows:

$$p(o = 1) = \prod_i p(o_i = 1) \tag{10}$$

Thus, using the definition of the IT2 fuzzy energy function given in Eq. (9), this probability may be defined as follows:

$$p(v, h; \tilde{W}, \tilde{b}, \ \tilde{d}) = \frac{1}{\tilde{Z}} e^{-\tilde{E}(v,h;\tilde{W},\tilde{b}, \ \tilde{d})} \tag{11}$$

where, $\tilde{Z} = \sum_{v,h} e^{-\tilde{E}(v,h;\tilde{W},\tilde{b}, \ \tilde{d})}$ is the partition function, which sums over all the possible hidden and the visible vectors. Now, IT2 fuzzy sets can be used in different ways. In this paper, which is a first attempt at using them, we proceed as follows:

(i) Let $\tilde{W}$ denote a $(n_v \times n_h)$ matrix of IT2 wt, $\tilde{w}_{ij}$, with MF $\mu_{\tilde{w}_{ij}}(w_{ij})$ and LMF and UMF $\underline{\mu}_{\tilde{w}_{ij}}(w_{ij})$ and $\bar{\mu}_{\tilde{w}_{ij}}(w_{ij})$, respectively. The centroid of $\mu_{\tilde{w}_{ij}}(w_{ij})$ is $C_{\tilde{w}_{ij}}$, and can be computed by means of an algorithm such as Enhanced Karnik-Mendel (EKM) [66], and is an interval of real numbers i.e. $C_{\tilde{w}_{ij}} = [c_l(\tilde{w}_{ij}), \ c_r(\tilde{w}_{ij})]$. The COG of $C_{\tilde{w}_{ij}}$ is $c_{\tilde{w}_{ij}}$, where $c_{\tilde{w}_{ij}} = \frac{[c_l(\tilde{w}_{ij}) + \ c_r(\tilde{w}_{ij})]}{2}$. A very good approximation to $c_{\tilde{w}_{ij}}$, denoted $\hat{c}_{\tilde{w}_{ij}}$, that avoids the iterative computations of EKM, is given by the so-called improved Nie-Tan (INT) algorithm, and is given below in our Algorithm 2. The weight matrix that is used in the paper is $W$, whose elements are $\hat{c}_{\tilde{w}_{ij}}$ $(i = 1, 2, \ldots n_v; j = 1, 2, \ldots n_h)$.

(ii) Let $\tilde{b}$ and $\tilde{d}$ be $(n_h \times 1)$ and $(n_v \times 1)$ vectors of IT2 biases $\tilde{b}_k$ and $\tilde{d}_m$, with MFs $\mu_{\tilde{b}_k}(b_k)$ and $\mu_{\tilde{d}_m}(d_m)$, respectively, and respective LMFs and UMFs $\underline{\mu}_{\tilde{b}_k}(b_k), \underline{\mu}_{\tilde{d}_m}(d_m), \underline{\mu}_{\tilde{b}_k}(b_k)$, and $\bar{\mu}_{\tilde{d}_m}(d_m)$. The centroids of $\mu_{\tilde{b}_k}(b_k)$ and $\mu_{\tilde{d}_m}(d_m)$, $C_{\tilde{b}_k}$ and $C_{\tilde{d}_m}$, can also be computed by means of an algorithm such as EKM, and are intervals of real numbers, i.e. $C_{\tilde{b}_k} = [c_l(\tilde{b}_k), \ c_r(\tilde{b}_k)]$ and $C_{\tilde{d}_m} = [c_l(\tilde{d}_m), \ c_r(\tilde{d}_m)]$. The COGs of $C_{\tilde{b}_k}$ and $C_{\tilde{d}_m}$ are $c_{\tilde{b}_k} = [c_l(\tilde{b}_k) + \ c_r(\tilde{b}_k)]/2$ and $c_{\tilde{d}_m} = [c_l(\tilde{d}_m) + \ c_r(\tilde{d}_m)]/2$. Very good approximations to (estimates of) $c_{\tilde{b}_k}$ and $c_{\tilde{d}_m}$, denoted $\hat{c}_{\tilde{b}_k}$ and

[1] (9) and (10) are expressive equations, but one not meant for computations; how their right-hand sides are computed is explained below and lead to computational equations as (12) and (13).

$\widehat{c}_{\widetilde{d}_m}$, that avoid the iterative computations of the EKM, are also given by the INT algorithm (Algorithm 2). The bias vectors that are used in this paper are $b$ and $d$, whose elements are $\widehat{c}_{\widetilde{b}_k}$ and $\widehat{c}_{\widetilde{d}_m}$ $(k = 1, 2, \ldots n_v; m = 1, 2, \ldots n_h)$, respectively.

Now, using the quantities explained above, the computational versions of Eqs. (9) and (11) are:

$$\widetilde{E}(v, h; \widetilde{W}, \widetilde{b}, \widetilde{d}) \rightarrow E(v, h,\ W, b, d) = h^T W v - b^T h - d^T v \tag{12}$$

$$p(v, h; \widetilde{W}, \widetilde{b},\ \widetilde{d}) \rightarrow p(v, h,\ W, b, d) = \frac{1}{Z} e^{-E(v,h;W,b,\ d)} \tag{13}$$

The probability of the training data can be increased by fine-tuning the weights and biases to lower the energy of that data. Also, weights are adjusted to raise the energy of other data points which have low energies and make a major impact to the partition function. For that, we compute the derivative of the log probability with respect to the weights as follows:

$$\frac{\partial\ logp(v)}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \tag{14}$$

where the angle brackets represent the average (or expectation) of the training data identified by the subscripts. Thus, Eq. (14) forms the learning rule of the training data for the stochastic steepest ascent in the log probability by multiplying with the learning rate '$\in$':

$$\Delta w_{ij} = \in \left( \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \right) \tag{15}$$

Since RBM is the restricted version of the BM, there are no links between the hidden units, so the unbiased sample of $\langle v_i h_j \rangle_{data}$ can easily be computed. Given any randomly selected training case of the visible units, the binary state of each hidden unit is set to 1 with a probability given by:

$$P(h = 1|v) = \sigma\left(\widetilde{W}^T v + \widetilde{b}\right) \rightarrow \sigma\left(W^T v + b\right) \tag{16}$$

Also, given the hidden vector, the unbiased sample of the state of visible unit can easily be computed using Eq. (17):

$$P(v = 1|h) = \sigma\left(\widetilde{W}^T h + \widetilde{d}\right) \rightarrow \sigma\left(W^T h + d\right) \tag{17}$$

In Eqs. (16) and (17), $\sigma(\cdot)$ is the sigmoidal (logistic) activation function, which is given by:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{18}$$

The computation of the unbiased sample of $\langle v_i h_j \rangle_{model}$, is a difficult task [9]. Thus, Hinton [2] proposed a faster learning algorithm called contrastive divergence (CD) algorithm which starts by clamping the training vector to the states of the visible units of the RBM. The binary states of all the hidden units are then computed using Eq. (16). After this, a reconstruction is produced by setting each visible unit, $v_i$, to 1 with a probability given by Eq. (17). Eventually, the change in the weight of the interconnection between the units $i$ and $j$ may be defined as follows:

$$\Delta w_{ij} = \in \left( \left\langle v_i^0 h_j^0 \right\rangle_{data} - \left\langle v_i^1 h_j^1 \right\rangle_{reconstruction} \right) \tag{19}$$

This procedure is followed until the thermal equilibrium is attained at the $n^{th}$ state. The entire process of CD training is depicted in the Fig. 4.

The training data in our framework is represented by the state $v_0$. From this state, hidden state $h_0$ is calculated which helps in the reconstruction of $v_1$ and the associated binary state of the hidden unit $h_1$. The input is considered as a random binary variable and the associated probabilities are not provided explicitly. In such a case, the instance of the training data is selected as the probability, since the input is either 0 or 1. Hence, $h_0$ is evaluated using this updated inference where the average probability is the output. Similarly, other states including $v_1$ and $h_1$ are also evaluated. Since the hidden layer (average probability) is completely inferred, it is used as the

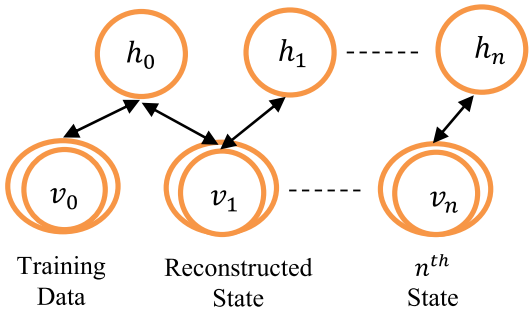


**Fig. 4.** Reconstruction process in CD learning.

probability of the random binary state. The energy Eq. (12) and conditional probability Eqs. (16) & (17) are required to compute the optimal weights. The connection weight between first layer (say, $W_1$) is frozen and sample $h_1$ is used from $p(h_1|v) = p(h_1|v,\ W_1\ )$ as the data for training the next layer. Then, the parameter vector $W_2$ (weights between second layers) that defines the second layer feature is frozen and uses the samples $h_2$ from $p(h_2|h_1) = p(h_2|h_1,\ W_2\ )$ as the data for training the third layer. Basically, these equations help in computing the optimal weights, which later helps in the inference.

In the classical inferences of RBM, the expectations of probabilities is computed before the activation function, according to which, the probability of setting the $(u+1)th$ node to 1 is given by:

$$P(\vartheta_{u+1} = 1) = \sigma\left(\widetilde{W}_u^T \widetilde{E}_{P(\vartheta_u)}[\vartheta_u] + \ \widetilde{b}_u\right) \tag{20}$$

here, $\vartheta_u$ is the associated random binary variable. However, it is preferable to apply the activation function before proceeding to the expectation [12]. Accordingly, probabilities of the $(u+1)th$ nodes can now be inferred, as follows:

$$P(\vartheta_{u+1} = 1) = \widetilde{E}_{P(\vartheta_u)}\left(\widetilde{\sigma}\left(\widetilde{W}_u^T[\vartheta_u]\right) + \ \widetilde{b}_u\right) \rightarrow E_{P(\vartheta_u)}\left(\sigma\left(W_u^T[\vartheta_u]\right) + \ b_u\right) \tag{21}$$

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. The DBN, which is formed using stacked RBMs, is pre-trained using the CD algorithm. Each individual RBM is trained, which is then used to train the subsequent RBM. This eventually pre-trains the entire DBN. Essential patterns in the data can then be detected from this pre-trained network. The training of the DBN is completed after fine-tuning the network by producing a very small set of labelled samples to associate labels with the features and the patterns.

We now present a novel learning algorithm for enhanced learning in the DBN, termed as Interval type-2 Fuzzy DBN (IT2FDBN). The complete procedure is depicted in the Fig. 5 as a flowchart. The steps of are explained as follows:

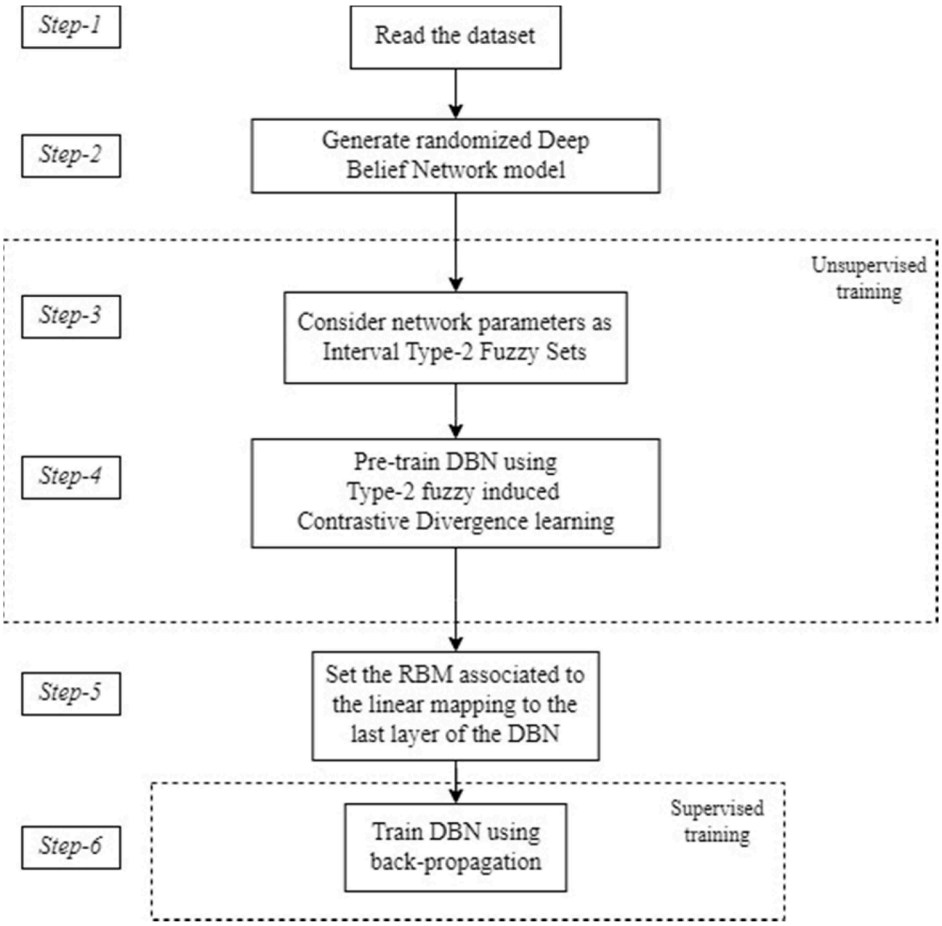


**Fig. 5.** Flowchart of the proposed IT2FDBN.

***Step-1***: Pre-processing of the data is done to make it readable.

***Step-2***: The parameters of the RBMs are initialized randomly to eventually initialize the entire DBN. We consider Bernoulli distribution for the RBMs in this paper. Moreover, the architecture of the DBN is also initialized. Our DBN model has the standard visible layer and an output layer, and in between them we have two hidden layers with the same number of nodes in each. We have considered various number of nodes in the hidden layers to select the optimal network model.

***Step-3***: The uncertain weights and biases are initialized with the IT2 fuzzy weights and biases. The approach is summarized in Algorithm 1. Thus, the network parameters $W$, $b$, and $c$ are considered as IT2 fuzzy parameters and are denoted as $\widetilde{W}$, $\widetilde{b}$ and $\widetilde{c}$.

***Step-4:*** The COG values are computed and are used for the further computations to pre-train the network using the CD learning approach. This learning is now type-2 fuzzy induced CD learning. The COG is calculated for the equations which use IT2 fuzzy weights and biases. To compute COG, we use the INT algorithm given in Algorithm 2. It is the improved version of NT algorithm [67] which is constructed from the lower and upper MFs of an IT2 FS by taking their average and then computing the centroid to get the crisp value. The computational cost for the COG output is significantly reduced due to a smaller number of iterations in the process.

***Step-5***: To complete the network, the linear mapped RBM is added to the last layer. This mapping is just an option. However, the linear mapping is the way to initialize the initial parameters.

**Algorithm 1**
Interval Type-2 Fuzzy Network Parameters.

***Input:*** *Randomly chosen input weights and biases;* $x \in (0,1)$
***Output:*** *Interval type-2 fuzzy weights and biases*
*Method Generation ()*
*{*
*Step 1: Get a single crisp input* $x \in (0,1)$*; x is a parameter used in training RBM .*
*Step 2: The uncertainty of x can be defined by a uniform distribution* $U(y-x;r) = r$ *for* $y \in [x-r, x+r]$*, where r is any randomly generated constant factor. The two extreme points of the distribution can be regarded as* $\bar{a}_L$ *and* $\bar{a}_R$
*Step 3: Parameters* $\bar{a}$ *and* $\bar{a}'$ *are computed randomly between the interval* $[\bar{a}_L, \bar{a}]$ *and* $[\bar{a}', \bar{a}_R]$*, respectively.*
*Step 4: Design a T1 FMF using Eq. (7).*
*Step 5: This T1 FMF becomes the UMF of the IT2 FMF.*
*Step 6: Choose a scaling factor '*$\gamma$*' between 0 and 1.*
*Step 7: Design the LMF by scaling the UMF (generated in Step 4) by* $\gamma$*. The points* $\underline{a}_L$ *and* $\underline{a}_R$ *are formed between the parameters* $[x, \bar{a}_L]$ *and* $[\bar{a}_R, x]$ *for every cases.*
*Step 8: Combine the LMF and UMF to obtain the IT2 MF that is used as parameters of DBN.*
*}*

**Algorithm 2**
Improved NT (INT) algorithm.

***Input:*** *IT2 fuzzy network parameters*
***Output:*** *Crisp output*

*Step 1:* The average ($a_i$) of the LMF and UMF of the IT2 fuzzy set $\widetilde{T}$ at each $x_i$ is computed using:

$$a_i = \frac{\underline{\mu}_{\widetilde{T}}(x_i) + \bar{\mu}_{\widetilde{T}}(x_i)}{2} i = 1, \ldots N$$

*Step 2:* Calculate average NT output as:

$$a_{NT} = \frac{\sum_{i=1}^{N} x_i a_i}{\sum_{i=1}^{N} a_i} = \frac{\sum_{i=1}^{N} x_i \left[\underline{\mu}_{\widetilde{T}}(x_i) + \bar{\mu}_{\widetilde{T}}(x_i)\right]}{\sum_{i=1}^{N} \left[\underline{\mu}_{\widetilde{T}}(x_i) + \bar{\mu}_{\widetilde{T}}(x_i)\right]}$$

*Step 3:* Find $k (1 \le k \le N-1)$ such that $x_k \le a_{NT} \le x_{k+1}$.

*Step 4*:

$$\text{Calculate } \delta = \frac{2\left(\sum_{i=1}^{k} (a_{NT} - x_i)\underline{\mu}_{\widetilde{T}}(x_i)\right) + 2\left(\sum_{i=k+1}^{N} (a_{NT} - x_i)\underline{\mu}_{\widetilde{T}}(x_i)\right)}{\left(\sum_{i=1}^{k} \left(\underline{\mu}_{\widetilde{T}}(x_i) + \underline{\mu}_{\widetilde{T}}(x_i)\right)\right)^2}$$

$$\text{X} \left(\sum_{i=1}^{k} (\underline{\mu}(x_i) - \underline{\mu}_{\widetilde{T}}(x_i)\right) - \sum_{i=k+1}^{N} \left(\bar{\mu}_{\widetilde{T}}(x_i) - \underline{\mu}_{\widetilde{T}}(x_i)\right)$$

*Step 5:* Calculate improved value $a_{INT} = a_{NT} + \delta$

**Algorithm 3**
Interval type-2 fuzzy deep learning (IT2FDBN).

***Input:*** *Training data samples, test data Samples, learning rate 'ε'*
***Output:*** *Enhanced trained network.*
*Method IT2FDBN ()*
*{*
*Step 1: Initialize Deep belief network model randomly* i.e. *all the internal RBM's are initialized.*
*Step 2: Perform pre-training of the initialized DBN*
*Step 2a: While learning, apply Algorithm 1 for initialising IT2 fuzzy weights and biases.*
*Step 2b: Apply type-2 fuzzy induced CD learning algorithm in an unsupervised manner, using Algorithm 2 for COG computations.*
*Step 3: Assign linear mapped RBM to the last layer of the formed DBN.*
*Step 4: Perform Training on the DBN model using back-propagation algorithm*
*Step 5: Compute RMSE and Error Rate for conventional, type-1 fuzzy inference and interval type-2 fuzzy inference.*
*}*

***Step-6***: The training process of the DBN is completed by fine-tuning the network in a supervised manner using a classical back-propagation algorithm where derivatives are propagated for each hidden unit and a local minimum is attained.

Different steps of the IT2FDBN are explicitly shown as the Algorithm 3.

## 5. Experimental results and discussion

This section demonstrates the efficacy of the proposed IT2FDBN approach in enhancing the learning capabilities of the DBN. A DBN with two-hidden-layers has been considered, where the input layer has 784 nodes, output layer has 10 nodes and for the two hidden layers we have experimented with various combinations of nodes. We have conducted our experiments using well-known benchmark datasets and compared the results considering two popular performance measures. Brief descriptions on the used datasets and the performance metrics are given below, after which we provide the experimental results. The computing server systems used for the experiments were having 64-bit windows operating systems and Intel(R) Xeon(R) CPU E5–2650 processors with 2.00 GHz frequency, 48 GB RAM (random access memory) and 2 GB NVIDIA GeForce GTX 960 graphical processing units (GPUs).

### 5.1. Data sets

We have used the handwritten data set of MNIST [30], which is the most widely used dataset for neural network related experiments. The MNIST is a handwritten digit database consisting of black and white images of size $28 \times 28$, which are linearized as vectors of size $1 \times 784$. Each image contains a digit 0 to 9 comprised of 10 classes [68]. The dataset has a total of 60,000 training images and 10,000 test images which form a 2D vector of size $60{,}000 \times 784$ and $10{,}000 \times 784$, respectively. For this dataset, we have experimented with 200, 400, 600, 800, 1000, 1200, and 1600 numbers of nodes in each hidden layer to identify the best network architecture and to verify the validity of the proposed approach. For the statistical analysis, we performed 100 Monte-Carlo simulation.

Additionally, we used two noisy variants of MNIST dataset, called n-MNIST datasets [69], which are basically modified versions of the original MNIST dataset with added noise. These datasets are MNIST with additive white Gaussian noise (AWGN) and MNIST with motion blur. Both of these datasets also have the same total of 60,000 training samples and 10,000 test samples with same data dimensions.

### 5.2. Performance measures

Two performance metrics, viz., the RMSE and the Error Rate, have been employed for comparing the performance of the algorithms. They are defined as follows:

$$RMSE = \sqrt{\frac{1}{O_c N_d} \sum_{k=1}^{N_d} \| y_k - F(x_k) \|_2^2} \tag{22}$$

$$Error\ Rate = \frac{I_N}{N_d} \tag{23}$$

where, $N_d$ is the number of data instances, $O_c$ is the number of output classes, $I_N$ is the number of incorrect inferences. $x_k$ represents $k^{th}$ input data, $y_k$ represents ground truth output and $F$ represents inference of the trained DBN. The less the RMSE and Error Rate, less will be miss classification (more correctly predicted values), thus better classification and regeneration of MNIST digits. Next, we discuss the experiments and the results obtained using the considered datasets.

### 5.3. Results from the original MNIST dataset

Our DBN model has $\{784 - H1 - H2 - 10\}$ nodes, where 784 nodes are in the first layer, equal number of nodes are considered in the hidden layers ($H1$ and $H2$), and 10 nodes are in the output layer. In the very beginning, we conducted experiments to identify the best

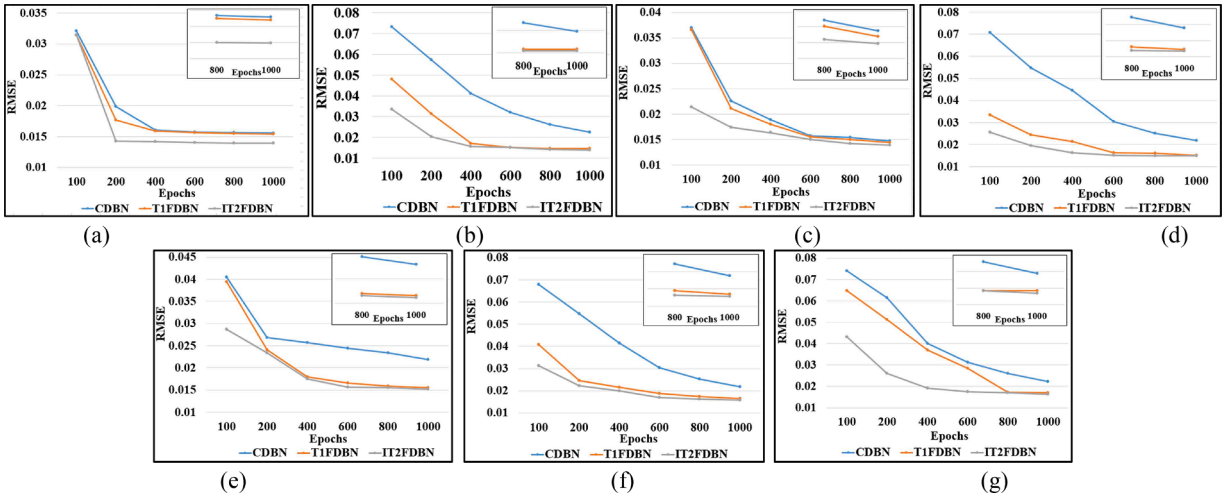


**Fig. 6.** Convergence of training data with number of units in the hidden layers as: (a) 200; (b) 400; (c) 600; (d) 800; (e) 1000; (f) 1200; (g) 1600.

model for IT2FDBN, T1FDBN and CDBN. To do this, a number of models with different number of nodes {200, 400, 600, 800, 1000, 1200, 1600} were considered for experimentations to select the best model in terms of optimal number of units in the hidden layers. All of our experiments were repeated over several epoch values {100, 200, 400, 600, 800, 1000}. Fig. 6 shows the convergence of training data for different epochs and different numbers of units in the hidden layers.

In the training data, the RMSE values from the proposed IT2FDBN is better than the T1FDBN and CDBN for all the cases of hidden layer nodes. Each figure contains a snippet (inset) for RMSE values of the 800 and 1000 epochs, which verifies the noticeable improvement in case of IT2FDBN. From all the considered cases for the hidden layer nodes, the optimal RMSE of 0.0139 is observed for IT2FDBN with 600 nodes in hidden layer, whereas T1FDBN and CDBN attains the RMSEs of 0.0144 and 0.0147, respectively. Starting from 200 nodes in the hidden layer the RMSE value gets better till 600 nodes, but after that the RMSE value increases for all the DBNs as the error increase with the increasing number of hidden layers [14]. Similar behaviour can be seen for the test data as IT2FDBN shows improved performance over the CDBN and T1FDBN with 600 number of nodes in the hidden layer. Fig. 7 shows the convergence of the test data over varying number of epochs.

The variations in the results for IT2FDBN are marginally better as compared to other two DBNs. From Fig. 7(c), we can see that IT2FDBN results in RMSE of 0.0530, which is slightly better than the 0.0541 and 0.0551 RMSEs of T1FDBN and CDBN. In Fig. 8, we have compiled the RMSE results for the training data and the test data with all number of considered nodes in the hidden layers with 1000 epochs.

Although the best performance is achieved when the nodes in the hidden layers are 600, the enhanced performance of the IT2FDBN is consistently better also for other numbers of nodes in the hidden layer. Fig. 9 shows a bar graph comparison of improvement of the IT2FDBN over T1FDBN and CDBN with training and test data (600 hidden layer nodes). For the training data, IT2FDBN has an improvement of 5.44 % and 3.30 % over T1FDBN and CDBN. Similarly, for test data, IT2FDBN shows an improvement of 3.74 % and

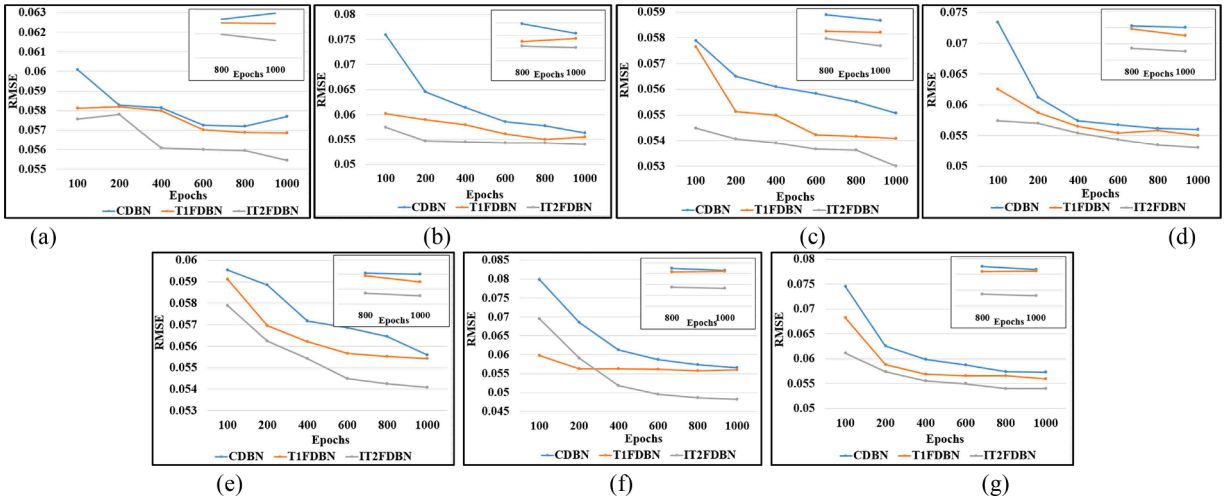


**Fig. 7.** Convergence of test data with number of units in the hidden layers as: (a) 200; (b) 400; (c) 600; (d) 800; (e) 1000; (f) 1200; (g) 1600.

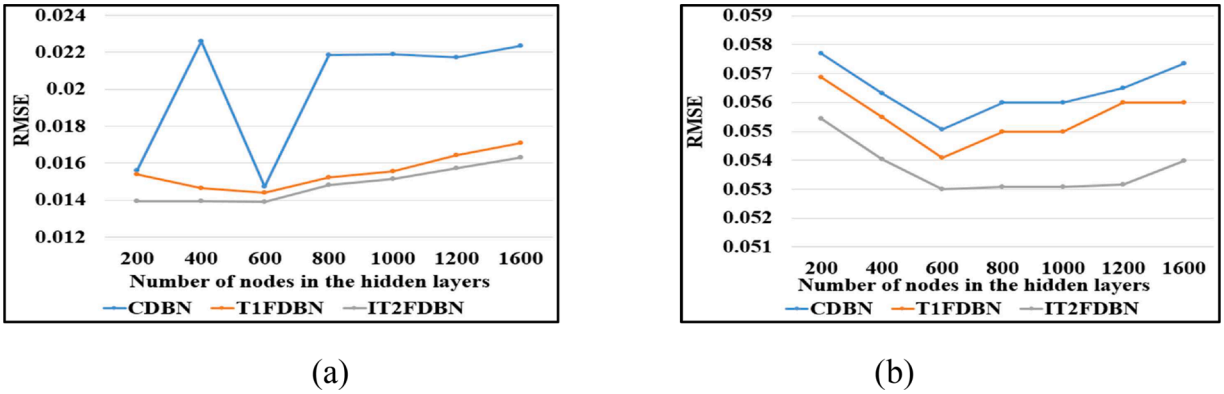


**Fig. 8.** RMSE of CDBN, T1FDBN, and IT2FDBN for (a) the training data; and (b) the test data.

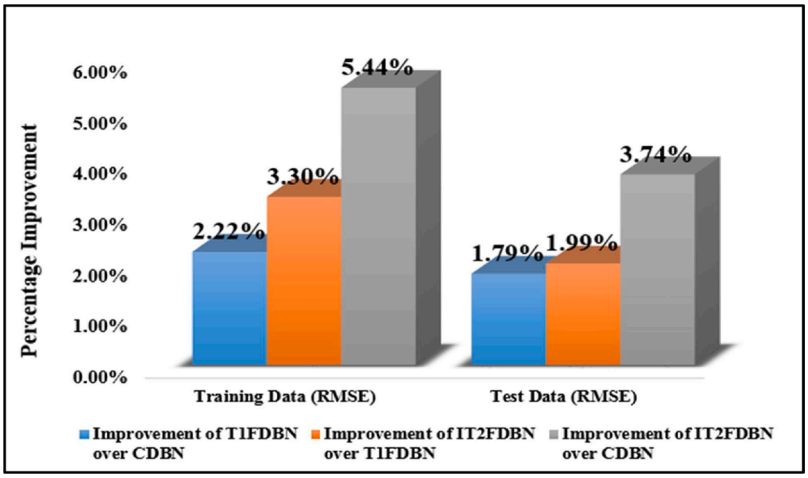


**Fig. 9.** Comparison of improvements in the RMSE of the training data and the test data for MNIST.

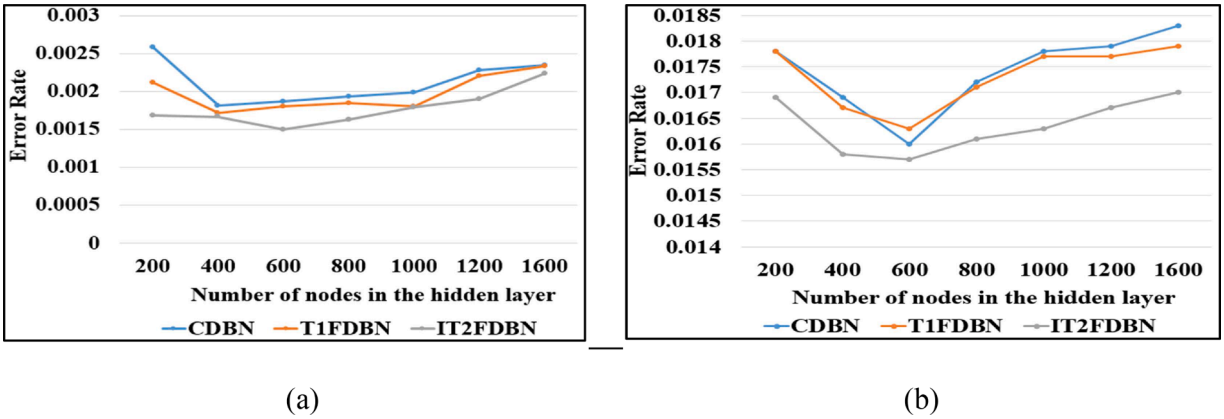


**Fig. 10.** Error rate of CDBN, T1FDBN, and IT2FDBN for (a) the training data; and (b) the test data.

1.99 % as compared to the T1FDBN and CDBN. Thus, we see that, within the context of the model selection problem, all three DBN architecture IT2FDBN, T1FDBN, and CDBN have their best results with 600 nodes in the hidden layers.

Similar behaviour of IT2FDBN can be seen for the error rates in the training data and the test data, as values in case of IT2FDBN are slightly better when compared to the T1FDBN and CDBN. The Error Rates of the training data among CDBN, T1FDBN, and IT2FDBN are shown in Fig 10(a). The smallest Error is found for IT2FDBN with a value of 0.0015 for the 600 number of nodes in the hidden layer. With the same number of nodes, 0.0018 and 0.001867 Error Rate values are obtained by T1FDBN and CDBN, respectively. Although

**Table 2**
Comparison of the RMSE and the error rate for the MNIST Dataset for 600 hidden nodes.

| | Training Data | | Test Data | |
|---|---|---|---|---|
| Average of | RMSE | Error Rate | RMSE | Error Rate |
| CDBN | 0.01472 | 0.00186 | 0.05465 | 0.0164 |
| T1FDBN | 0.01439 | 0.00180 | 0.05401 | 0.0162 |
| IT2FDBN | 0.01392 | 0.00150 | 0.05370 | 0.0157 |

the least error is observed at 600 number of hidden nodes, it can be observed that this enhanced performance of IT2FDBN is also consistent for other number of considered hidden nodes.

For the test data, Error Rate is also least for IT2FDBN with 600 number of hidden layer nodes, which is 0.0157, while the CDBN and T1FDBN attains the Error Rate of 0.0162 and 0.0164, respectively. Thus, IT2FDBN results are 3.09 % and 4.27 % better than the T1FDBN and CDBN. Fig. 10(b) shows the comparison in the Error Rates of CDBN, T1FDBN, and IT2FDBN; it is evident that the variations in the results for IT2FDBN are more enhanced as compared to other DBNs. Also, the enhanced performance of the IT2FDBN is consistent for every number of nodes in the hidden layer we have experimented. Table 2 gives the RMSE values and the Error Rates with the MNIST dataset for both training and testing for 600 hidden nodes.

### *5.4. Results from the MNIST dataset with additive white Gaussian noise (AWGN)*

This dataset was created using an additive white Gaussian noise having a signal-to-noise ratio of 9.5. A substantial amount of background noise is emulated. We have conducted experiments with CDBN, T1FDBN and the proposed IT2FDBN on this dataset maintaining 600 nodes for the hidden layers. Table 3 shows the average values and standard deviation (SD) of the RMSE and Error rate for the training and the test data. The SD is very low which shows that there is less variations in the outcome and results are consistent.

As can be seen from Table 3, T1FBDN shows better performance as compared to the CDBN in terms of RMSE. However, IT2FDBN surpasses both T1FDBN and CDBN with a RMSE value of 0.0139. Bar graph comparisons of the improvements in the RMSE of the training and the test data among the CDBN, T1FDBN and IT2FDBN are shown in Fig. 11. There is 0.90 % improvement with T1FDBN over the CDBN, while its 2.89 % improvement of IT2FDBN over T1FDBN. Overall, IT2FDBN shows 3.77 % improvement in the RMSE over CDBN. Furthermore, in the test data also, results show better performance with IT2FDBN as compared to the other two DBNs in RMSE.

**Table 3**
Comparison of the RMSE and the error rate for the MNIST Dataset with AWGN for 600 hidden nodes.

| | Training Data | | | | Test Data | | | |
|---|---|---|---|---|---|---|---|---|
| | RMSE | | Error Rate | | RMSE | | Error Rate | |
| | Average | SD | Average | SD | Average | SD | Average | SD |
| CDBN | 0.01472 | 0.00017 | 0.00186 | 0.00023 | 0.05465 | 0.0015 | 0.0164 | 0.00032 |
| T1FDBN | 0.01439 | 0.00033 | 0.0018 | 0.00028 | 0.05401 | 0.0011 | 0.0162 | 0.00072 |
| IT2FDBN | 0.01392 | 0.00018 | 0.0015 | 0.00018 | 0.0537 | 0.0014 | 0.0157 | 0.00148 |

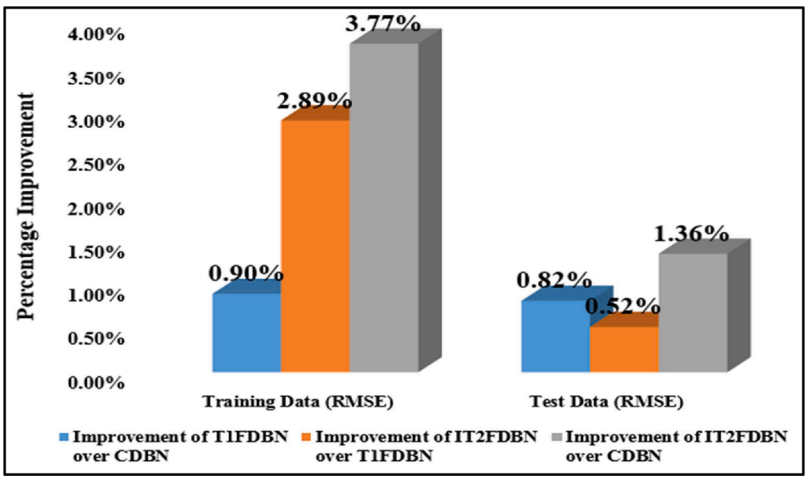


**Fig. 11.** Comparison of improvements in the RMSE of the training data and the test data for the MNIST dataset with AWGN.

**Table 4**
Comparison of the RMSE and error rate for the MNIST dataset with motion blur for 600 hidden nodes.

| | Training Data | | | | Test Data | | | |
|---|---|---|---|---|---|---|---|---|
| | RMSE | | Error Rate | | RMSE | | Error Rate | |
| | Average | SD | Average | SD | Average | SD | Average | SD |
| CDBN | 0.0292 | 0.00024 | 0.0074 | 0.00014 | 0.08898 | 0.00028 | 0.0437 | 0.00022 |
| T1FDBN | 0.0289 | 0.00025 | 0.0076 | 0.00032 | 0.08825 | 0.00037 | 0.0436 | 0.00026 |
| IT2FDBN | 0.0281 | 0.00016 | 0.0068 | 0.00019 | 0.08779 | 0.00032 | 0.0429 | 0.00031 |

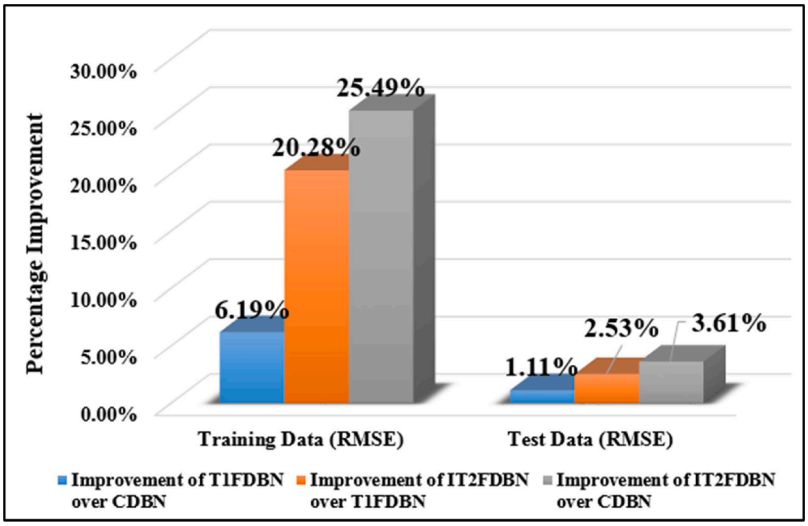


**Fig. 12.** Comparison of the RMSE of the training data and the test data for the MNIST dataset with motion blur.

### 5.5. Result from the MNIST dataset with motion blur

This dataset is generated by linearly moving a camera for $p$ pixels having some $\theta$ degree angle. Here, $p$ is chosen as 5 pixels and $\theta$ as 15° in the counter-clockwise direction. The average values and SD of RMSE and the Error Rate for the training data and the test data are shown in Table 4. In this case, IT2FDBN also outperforms the CDBN and the T1FDBN. The difference between the RMSE of the T1FDBN and IT2FDBN is quite noticeable though. For the T1FDBN, RMSE is 0.0289 whereas it is 0.0281 for the IT2FDBN. Also, for this dataset, SD is very low which signifies the less variations in the results.

Fig. 12 shows the comparative improvements of the RMSE for all the three DBNs as bar graphs. For the training data, the T1FDBN performed better than the conventional DBN as shown by improved RMSE of 6.19 %. Moreover, IT2FDBN has enhanced the performance with 20.28 % as compared to the T1FDBN; however, there is 25.49 % improvement in RMSE with IT2FDBN as compared to the CDBN. Furthermore, test data also shows improvement of the DBN framework with IT2FDBN as shown in the bar graph on the right side of the Fig. 12.

### 5.6. Discussion

From our experimental results we draw the following conclusions:

(i) IT2FDBN shows better deep learning capabilities because the RMSE values and the error rates obtained with it are better than those obtained from T1FDBN and CDBN.
(ii) For all three architectures, the best value of the RMSE is obtained when the hidden layers have 600 nodes. Thus, we can say that the model having hidden layers with 600 nodes is the best model, which we maintained for our experimentations with the remaining two datasets.
(iii) There are noticeable improvements of the RMSE obtained with the IT2FDBN over T1FDBN and CDBN for two noisy datasets: MNIST with AWGN and MNIST with motion blur; thus, the proposed IT2FDBN is more robust than both T1FDBN and CDBN.
(iv) Regardless of the number of nodes in the hidden layers, IT2FDBN has shown consistent enhanced performance, as compared to T1FDBN and CDBN.

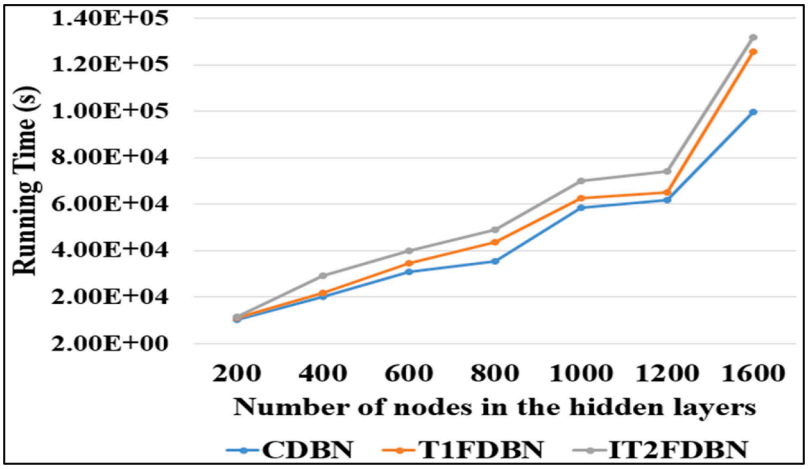


**Fig. 13.** Running time of CDBN, T1FDBN, and IT2FDBN for different number of nodes for1000 epochs.

**Table 5**
p values from the Kruskal-Wallis test.

| Dataset | Training data | Test data |
|---|---|---|
| MNIST | 1.41E-10 | 2.39E-07 |
| MNIST with AWGN | 3.40E-10 | 1.35E-05 |
| MNIST with motion blur | 4.06E-12 | 4.08E-11 |

The running times (in seconds) of the CDBN, T1FDBN, and IT2FDBN for the different numbers of nodes in the hidden layers, for 1000 epochs, are shown in Fig. 13. Due to the higher complexity and more number of parameters in IT2FSs, IT2FDBN took more time in execution as compared to its counterparts. This behaviour is consistent for all numbers of nodes. Additionally, the running time increases as the number of nodes in the hidden layers increased.

To establish the statistical significance of our findings, we performed the non-parametric Kruskal-Wallis test to analyse the results with p values as an evaluation criterion. This is a one-way hypothesis test for the equivalence of the means of all the possible iterations and gives a p value specifying if two independent variables are considerably different. The p values specify either "likely" or "unlikely" of an event with some probability. The significant level ($\alpha$) is generally kept at 0.05 to test the significance of the results. If the p value generated from the test is less than or equal to $\alpha$, then it is "likely" and the null hypothesis is rejected, i.e. results are statistically significant and medians of the three approaches are equal; whereas, if the p value is greater than $\alpha$, then it is "unlikely" and the null hypothesis is accepted, i.e. we cannot argue that results are statistically significant.

Table 5 shows the p values from the Kruskal-Wallis test for our three data sets. All p values are less than $\alpha$. The distributions of the results from CDBN, T1FDBN, and IT2FDBN with MNIST, MNIST with AWGN, and MNIST with motion blur are shown in the box plots of Fig. 14(a)-(f).

We have also performed the Wilcoxon statistical test between each of the DBNs which compares only two approaches and returns the p-value between them. It explicitly assumes two samples from the different approaches as independent. Its null hypothesis is that weather the medians are equal from the continuous distribution. The results are compiled in Table 6, from which we can make similar inference as has been made above for the Kruskal-Wallis test.

## 6. Conclusion

A novel approach for augmented deep learning is proposed in this paper which considers IT2 FSs to model the uncertain parameters viz. fuzzy weights and fuzzy biases of the DBN. This approach has led to the introduction of the novel deep architecture known as IT2FDBN. Experiments have been conducted using three popular datasets: MNIST handwriting recognition, MNIST with additive white Gaussian noise (AWGN) and MNIST with motion blur. To demonstrate the efficacy of IT2FDBN, comparative analyses have been performed with T1FDBN and CDBN. From the experimental simulations, it has been observed that the proposed IT2FDBN outperforms both T1FDBN and CDBN with improvements both in the RMSE and the error rates for training and test data. Based on the exhaustive experiments, we have established that the proposed architecture performed better when the number of nodes in the hidden layers is 600. This has empirically confirmed the best model in terms of optimal number of nodes in the hidden layers of a DBN. Moreover, the proposed IT2FDBN has also showed enhanced learning capability for any number of hidden layers as compared to the CDBN and

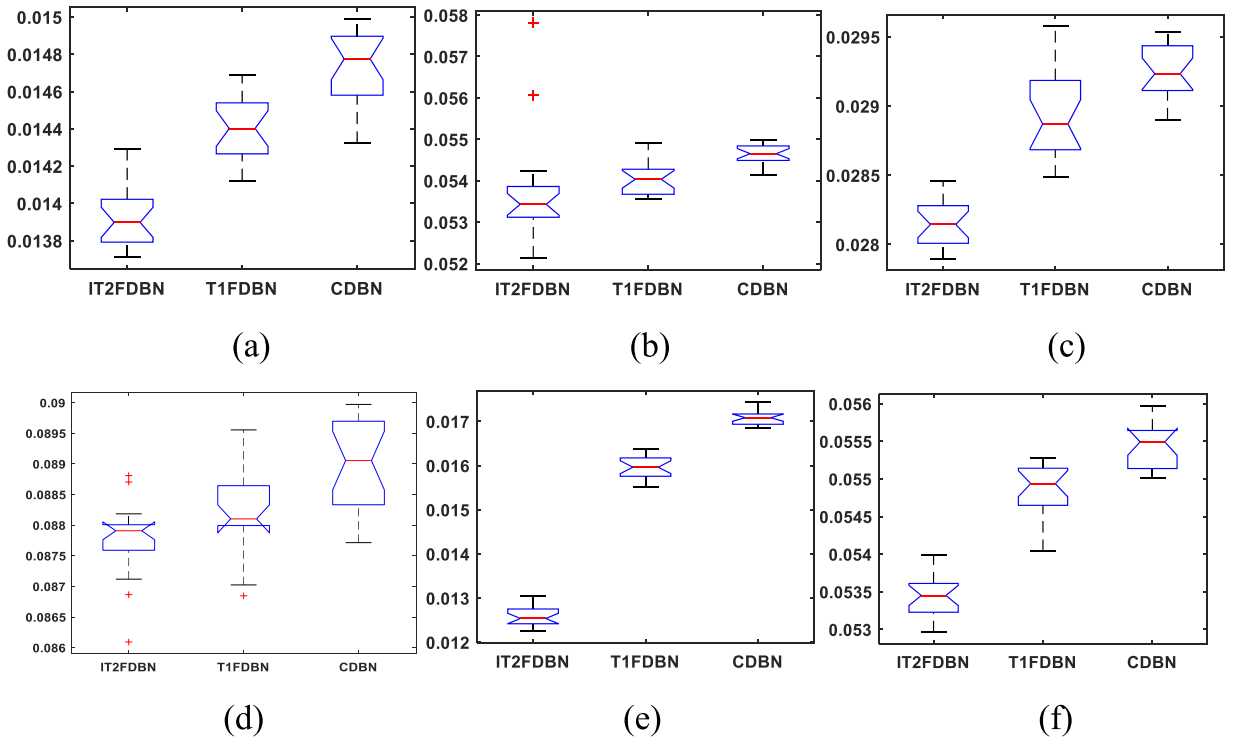


**Fig. 14.** Box plot of the RMSE's of the CDBN, T1FDBN, and IT2FDBN for (a) MNSIT training data, (b) MNIST test data, (c) MNSIT with AWGN training data, (d) MNIST with AWGN test data, (e) MNSIT with motion blur training data, (f) MNIST with motion blur test data.

**Table 6**
p-values from the Wilcoxon test.

| Datasets | | CDBN vs. T1FDBN | CFDBN vs. IT2FDBN | T1FDBN vs. IT2FDBN |
|---|---|---|---|---|
| MNIST | Training | 6.58E-05 | 6.80E-08 | 2.22E-07 |
| | Test | 5.17E-06 | 1.81E-05 | 0.0018 |
| MNIST with AWGN | Training | 0.0021 | 6.80E-08 | 6.80E-08 |
| | Test | 0.0047 | 1.8074e-05 | 0.0071 |
| MNIST with motion blur | Training | 6.80E-08 | 6.80E-08 | 6.80E-08 |
| | Test | 1.41E-05 | 6.80E-08 | 6.80E-08 |

T1FDBN. Similar performance has been observed for the two noisy variants of MNIST i.e., MNIST with AWGN and MNIST with motion blur, demonstrating the model's robustness in handling noisy and corrupt data.

Future work shall focus on IT2 fuzzy DBN when the data sets are associated with different kinds of uncertainties, e.g., the simultaneous presence of both blur and measurement noise. It shall also include a more comprehensive study that would focus on performance improvement versus the amount of motion blur. Further,research may also explore the propagation of IT2 FSs based formulations over the layers, from visible to hidden and consider other recent learning algorithms.

## 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 thankful to the editors and the reviewers for their constructive opinions and suggestions which have helped them in improving the manuscript quality immensely. Authors are very grateful to Prof. J. M. Mendel and wishes to thankfully acknowledge his several important suggestions during their numerous interactions while conducting the research related experiments and particularly during manuscript preparation.

## References


[1] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[2] G.E. Hinton, S. Osindero, Y.W. The, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[3] R. Salakhutdinov, G.E. Hinton, Semantic hashing, Int. J. Approx. Reason. 50 (7) (2009) 969–978.
[4] G.W. Taylor, G.E. Hinton, S.T. Roweis, Modeling human motion using binary latent variables, in: Advances in Neural Information Processing Systems, 2007, pp. 1345–1352.
[5] Y. Deng, Z. Ren, Y. Kong, F. Bao, Q. Dai, A hierarchical fused fuzzy deep neural network for data classification, IEEE Trans. Fuzzy Syst. 25 (4) (2017) 1006–1012.
[6] S. Zhou, Q. Chen, X. Wang, Fuzzy deep belief networks for semi-supervised sentiment classification, Neurocomputing 131 (2014) 312–322.
[7] G.E. Hinton, L. Deng, D. Yu, G.E. Dahl, A. Mohamed, N. Jaitly, A. Senior, et al., Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups, IEEE Signal Process. Mag. 29 (6) (2012) 82–97.
[8] T. Kuremoto, S. Kimura, K. Kobayashi, M. Obayash, Time series forecasting using a deep belief network with restricted Boltzmann machines, Neurocomputing 137 (2014) 47–56.
[9] G.E. Hinton, A practical guide to training restricted Boltzmann machines, in: Neural Networks: Tricks of the Trade: Second Edition, Berlin, Heidelberg, Springer Berlin Heidelberg, 2012, pp. 599–619.
[10] A. Fischer, C. Igel, An introduction to restricted Boltzmann machines, in: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, 2012, pp. 14–36.
[11] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536.
[12] M. Tanaka, O. Masatoshi, A novel inference of a restricted Boltzmann machine, in: Pattern Recognition (ICPR), 2014 22nd International Conference on, IEEE, 2014, pp. 1526–1531.
[13] C.L.P. Chen, Z. C.-Yang, C. Long, G. Min, Fuzzy restricted Boltzmann machine for the enhancement of deep learning, IEEE Trans. Fuzzy Syst. 23 (6) (2015) 2163–2173.
[14] S. Feng, C.L.P. Chen, A fuzzy restricted Boltzmann machine: novel learning algorithms based on crisp possibilistic mean value of fuzzy numbers, IEEE Trans. Fuzzy Syst. 26 (1) (2016) 117–130.
[15] R. Sambuc, Function Φ-Flous, Application a l'aide au Diagnostic en Pathologie Thyroidienne [Ph. D. thesis], 1975.
[16] K.U. Jahn, Intervall-wertige Mengen, Math. Nachr. 68 (1) (1975) 115–132.
[17] F. Herrera, On the usefulness of interval valued fuzzy sets for learning fuzzy rule based classification systems, in: Eurofuse 2011: Workshop on Fuzzy Methods for Knowledge-Based Systems, Springer Berlin Heidelberg, 2012, pp. 3–4.
[18] H. Bustince, Interval-valued fuzzy sets in soft computing, Int. J. Comput. Intell. Syst. 3 (2) (2010) 215–222.
[19] H.B. Sola, J. Fernandez, H. Hagras, F. Herrera, M. Pagola, E. Barrenechea, Interval type-2 fuzzy sets are generalization of interval-valued fuzzy sets: toward a wider view on their relationship, IEEE Trans. Fuzzy Syst. 23 (5) (2014) 1876–1882.
[20] J.M. Mendel, H. Hagras, H. Bustince, F. Herrera, Comments on "Interval type-2 fuzzy sets are generalization of interval-valued fuzzy sets: towards a wide view on their relationship, IEEE Trans. Fuzzy Syst. 24 (1) (2015) 249–250.
[21] P.K. Muhuri, Z. Ashraf, Q.M.D. Lohani, Multi-objective reliability-redundancy allocation problem with interval type-2 fuzzy uncertainty, IEEE Trans. Fuzzy Syst.. 26 (3) (2017) 1339–1355.
[22] C.I. Gonzalez, P. Melin, J.R. Castro, O. Castillo, Optimization of interval type-2 fuzzy systems for image edge detection, Appl. Soft Comput. 47 (2016) 631–643.
[23] O. Castillo, P. Melin, A review on interval type-2 fuzzy logic applications in intelligent control, Inf. Sci. 279 (2014) 615–631.
[24] 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.
[25] A.K. Shukla, P.K. Muhuri, Big-data clustering with interval type-2 fuzzy uncertainty modeling in gene expression datasets, Eng. Appl. Artif. Intell. 77 (2019) 268–282.
[26] A.K. Shukla, R. Nath, P.K. Muhuri, Q.M .D.. Lohani, Energy efficient multi-objective scheduling of tasks with interval type-2 fuzzy timing constraints in an Industry 4.0 ecosystem, Eng. Appl. Artif. Intell. 87 (2020), 103257.
[27] J.M. Mendel, Explaining the performance potential of rule-based fuzzy systems as a greater sculpting of the state space, IEEE Trans. Fuzzy Syst.. 26 (4) (2017) 2362–2373.
[28] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Advances in Neural Information Processing Systems 19, 2006.
[29] R. Salakhutdinov, G. Hinton, An efficient learning procedure for deep Boltzmann machines, Neural Comput. 24 (8) (2012) 1967–2006.
[30] M. Probst, F. Rothlauf, J. Grahl, Scalability of using Restricted Boltzmann Machines for combinatorial optimization, Eur. J. Oper. Res. 256 (2) (2017) 368–383.
[31] Y. Bengio, Practical recommendations for gradient-based training of deep architectures, in: Neural Networks: Tricks of the Trade, Berlin, Heidelberg, Springer, 2012, pp. 437–478.
[32] M. Zanotto, R. Volpi, A. Maccione, L. Berdondini, D. Sona, and V. Murino, Modeling retinal ganglion cell population activity with restricted Boltzmann machines, arXiv preprint arXiv:1701.02898 (2017).
[33] S. Tran, Representation Decomposition for Knowledge Extraction and Sharing using Restricted Boltzmann Machines, 2016.
[34] J.M. Tomczak, Learning informative features from restricted Boltzmann machines, Neural Process. Lett. 44 (3) (2016) 735–750.
[35] J. Zhang, C. Tao, P. Wang, A review of soft computing based on deep learning, in: Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII), 2016, pp. 136–144.
[36] J.P. Papa, W. Scheirer, D.D. Cox, Fine-tuning deep belief networks using harmony search, Appl. Soft Comput. 46 (2016) 875–885.
[37] M. Wang, X.S. Hua, Active learning in multimedia annotation and retrieval: a survey, ACM Trans. Intell. Syst. Technol. 2 (2) (2011) 1–21.
[38] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, in: Advances in Neural Information Processing Systems, 2015, pp. 1135–1143.
[39] 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.
[40] K. Cho, A. Ilin, T. Raiko, Improved learning of Gaussian-Bernoulli restricted Boltzmann machines, in: Artificial Neural Networks and Machine Learning (ICANN), Springer, 2011, pp. 10–17.
[41] S. Wang, C. Manning, Fast dropout training, in: International Conference on Machine Learning (ICML), 2013, pp. 118–126.
[42] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (4) (1943) 115–133.
[43] Y.A. LeCun, L. Bottou, G.B. Orr, K.R. Müller, Efficient backprop, in: Neural Networks: Tricks of the Trade, Berlin Heidelberg, Springer, 2012, pp. 9–48.
[44] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.
[45] O. Castillo, J.R. Castro, P. Melin, A. Rodriguez-Diaz, Application of interval type-2 fuzzy neural networks in non-linear identification and time series prediction, Soft comput. 18 (2014) 1213–1224.
[46] P. Melin, D. Sánchez, J.C. Monica, O. Castillo, Optimization using the firefly algorithm of ensemble neural networks with type-2 fuzzy integration for COVID-19 time series prediction, Soft Comput. 27 (6) (2023) 3245–3282.
[47] O. Castillo, J.R. Castro, M. Pulido, P. Melin, Interval type-3 fuzzy aggregators for ensembles of neural networks in COVID-19 time series prediction, Eng. Appl. Artif. Intell. 114 (2022), 105110.
[48] H. Hermessi, O. Mourali, E. Zagrouba, Deep feature learning for soft tissue sarcoma classification in MR images via transfer learning, Expert Syst. Appl. 120 (2019) 116–127.
[49] V. Venugopal, J. Joseph, M.V. Das, M.K. Nath, DTP-Net: a convolutional neural network model to predict threshold for localizing the lesions on dermatological macro-images, Comput. Biol. Med. 148 (2022), 105852.

[50] M.S. Hosseini-Pozveh, M. Safayani, A. Mirzaei, Interval type-2 fuzzy restricted Boltzmann machine, IEEE Trans. Fuzzy Syst. 29 (5) (2020) 1133–1142.
[51] A. Sedaghati, N. Pariz, M. Siahi, R. Barzamini, A new fractional-order developed type-2 fuzzy control for a class of nonlinear systems, Int. J. Syst. Sci. 52 (2021) 1–19.
[52] A. Beke, T. Kumbasar, Learning with type-2 fuzzy activation functions to improve the performance of deep neural networks, Eng. Appl. Artif. Intell. 85 (2019) 372–384.
[53] R. Lee, Chaotic type-2 transient-fuzzy deep neuro-oscillatory network (CT2TFDNN) for worldwide financial prediction, IEEE Trans. Fuzzy Syst. 28 (4) (2019) 731–745.
[54] T. Shen, et al., Hierarchical fused model with deep learning and type-2 fuzzy learning for breast cancer diagnosis, IEEE Trans. Fuzzy Syst. 28 (12) (2020) 3204–3218.
[55] L. Ghosh, S. Saha, A. Konar, Decoding emotional changes of android-gamers using a fused Type-2 fuzzy deep neural network, Comput. Hum. Behav. 116 (2021), 106640.
[56] N. Saleem, M.I. Khattak, S.A. AlQahtani, A. Jan, I. Hussain, M.N. Khan, M. Dahshan, U-shaped low-complexity type-2 fuzzy LSTM neural network for speech enhancement, IEEE Access 11 (2023) 20814–20826.
[57] A.K. Shukla, T. Seth, P.K. Muhuri, Interval type-2 fuzzy sets for enhanced learning in deep belief networks, in: Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on, IEEE, 2017, pp. 1–6.
[58] Z. Jia, J. Qiao, Extension operators for type-2 fuzzy sets derived from overlap functions, Fuzzy Sets Syst. 451 (2022) 130–156.
[59] X. Wu, G. Chen, L. Wang, On union and intersection of type-2 fuzzy sets not expressible by the sup-t-norm extension principle, Fuzzy Sets Syst. 441 (2022) 241–261.
[60] J.M. Mendel, R. John, Type-2 Fuzzy Sets made simple, IEEE Trans. Fuzzy Syst. 10 (2) (2002) 117–127.
[61] R. John, Type-2 Fuzzy Sets: an Appraisal of theory and applications, Int. J. Uncertain. Fuzziness Knowl. Based Syst. 6 (6) (1998) 563–576.
[62] J.M. Mendel, R. John, F. Liu, Interval type-2 fuzzy logic systems made simple, IEEE Trans. Fuzzy Syst.. 14 (6) (2006) 808–821.
[63] J.M. Mendel, Uncertain Rule-Based Fuzzy systems, Introduction and New Directions, 2nd ed., Springer, 2017.
[64] J.M. Mendel, M.R. Rajati, P. Sussner, On clarifying some definitions and notations used for type-2 fuzzy sets as well as some recommended notational changes, Inf. Sci. 340 (2016) 337–345.
[65] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, A Field Guide to Dynamical Recurrent Neural Networks, 2001.
[66] D. Wu, J.M. Mendel, Enhanced Karnik-Mendel algorithms, IEEE Trans. Fuzzy Syst. 17 (4) (2009) 923–934.
[67] M. Nie, W.W. Tan, Towards an efficient type-reduction method for interval type-2 fuzzy logic systems, in: Proceedings under IEEE International Conference on Fuzzy Systems, 2008, pp. 1425–1432.
[68] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[69] S. Basu, M. Karki, S. Ganguly, R. DiBiano, S. Mukhopadhyay, S. Gayaka, R. Kannan, R. Nemani, Learning sparse feature representations using probabilistic quadtrees and deep belief nets, Neural Process. Lett. 45 (3) (2017) 855–867.