A Survey on Fuzzy Deep Neural Networks

RANGAN DAS, SAGNIK SEN, and UJJWAL MAULIK, Jadavpur University

Deep neural networks are a class of powerful machine learning model that uses successive layers of non-linear processing units to extract features from data. However, the training process of such networks is quite computationally intensive and uses commonly used optimization methods that do not guarantee optimum performance. Furthermore, deep learning methods are often sensitive to noise in data and do not operate well in areas where data are incomplete. An alternative, yet little explored, method in enhancing deep learning performance is the use of fuzzy systems. Fuzzy systems have been previously used in conjunction with neural networks. This survey explores the different ways in which deep learning is improved with fuzzy logic systems. The techniques are classified based on how the two paradigms are combined. Finally, the real-life applications of the models are also explored.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**;

Additional Key Words and Phrases: Deep architecture, fuzzy systems, integrated models, ensemble models, parallel models, sequential models

ACM Reference format:
Rangan Das, Sagnik Sen, and Ujjwal Maulik. 2020. A Survey on Fuzzy Deep Neural Networks. *ACM Comput. Surv.* 53, 3, Article 54 (May 2020), 25 pages.
https://doi.org/10.1145/3369798

1 INTRODUCTION

Deep neural networks (DNNs) learn representations of the data in a hierarchical manner, building complex features from simple features. DNNs are an extension of multi-layer perceptrons (MLPs). Such networks typically organize artificial neurons in a fixed topology connected by predefined links. Neurons are non-linear processing elements are organized into columns, called layers. The output of one layer is the input of the succeeding layer. The first layer that takes the input data is called the input layer. The intermediate layers are called hidden layers, since they are hidden from the output. The last layer is called the output layer. The learning algorithm propagates the input forward through the layers of the processing elements and finds the error or the loss in the output layer [Mitchell 1997]. An increased number of layers allow the network to learn complex representations of the data, but new challenges arise when it comes to the training of such networks.

The target of training any deep neural network is to minimize the loss or error given an input vector. For every neural network, a loss function is defined that is used to calculate the loss of error for the input. The loss or the error is minimized by changing the weights of the network.

Authors' addresses: R. Das, S. Sen, and U. Maulik, Department of Computer Science and Engineering, Jadavpur University, Kolkata - 700032; emails: ragan@live.in, sagnik.sen2008@gmail.com, umaulik@cse.jdvu.ac.in.

0360-0300/2020/05-ART54 $15.00
https://doi.org/10.1145/3369798

The most common method for weight updation is by using gradient descent, which is a calculus-based method that iteratively computes the local minima of the error surface. The objective is to get to this minima by moving opposite to the direction of the slope [Karpathy 2017]. The obvious drawback of the calculus-based method is that it may not find the global minima of the error surface [Ruder 2016]. Furthermore, this process can be computationally intensive as well.

Gradient descent has three primary variations that is determined by what fraction of the input data we use to calculate the loss function. The parameters of the network are updated based on the time needed for the update as well as the volume of data used for the training process. The standard gradient descent computes the gradient of the loss function with respect to the parameters θ for the entire dataset, stochastic gradient descent (SGD) performs the update for each training example. Between these two is mini-batch gradient descent that makes the update for every batch of n tuples. However, there are some common difficulties faced by all the variants of gradient descent. These primarily include the following:

- Choosing a proper learning rate: Learning rate is a hyperparameter that the user has to supply. Finding the optimum value of the learning rate is done by trial and error. If it is too small, then the network takes a lot of time to converge, whereas a value that is too big may even prevent the network from reaching the minima. Learning rate schedules [Robbins and Monro 1985] that change the learning rate during the training process alleviate the problems a little, but these have to be defined in advance. Therefore, it becomes difficult to determine the threshold and schedules in advance without looking at the dataset. Depending on the architecture, they may also be other hyperparameters that are to be tuned manually.
- Minimizing highly non-convex error functions: Another key challenge is to avoid getting trapped in multiple local optima. The problem arises in the saddle points, where the value of the gradient is extremely small across all dimensions [Dauphin et al. 2014].

These challenges are somewhat tackled using gradient descent optimization methods such as Momentum [Qian 1999], Nesterov accelerated gradient (NAG) [Nesterov 1983], Adagrad [Duchi et al. 2011], Adadelta [Zeiler 2012], RMSprop [Tieleman and Hinton 2012], Adam [Kingma and Ba 2014], AdaMax, and Nesterov-accelerated Adaptive Momentum Estimation (Nadam). Most of these strategies aim to provide improved convergence for MLPs with a larger number of layers. Some other techniques, such as early stopping [Prechelt 1998], gradient noise [Neelakantan et al. 2015], batch normalization [Ioffe and Szegedy 2015], shuffling, and curriculum learning [Hacohen and Weinshall 2019], are used in conjunction. Furthermore, a lot of problems arise from low sample size, result from noisy samples or heterogeneous samples, or results from severe class imbalance issues [Buda et al. 2018]. These are some of the major drawbacks of deep learning techniques at present.

Besides, deep learning models needs a lot of data for training. Since it requires such copious amounts of data, the training time is also significant. Once trained, the model can be used for performing a very specific task. To deal with a slightly different problem, the entire system requires retraining. Another problem of deep learning is the computation power that is required for the training procedure, even though it is being addressed now through GPU-based parallel computing frameworks. Deep learning models work well with homogeneous features. If the features are of the same type (pixels, words counts, etc.), then the deep learning model can be easily developed. But when the features are heterogeneous, the weight updates in the network happens on different scales. Hence, the input data should be normalized using some method.

An alternative method that is seldom explored in optimizing the performance of deep neural networks is by forming a hybrid model with fuzzy systems. Fuzzy systems allow us to deal with the uncertainties and the ambiguities of real-world data. Fuzzy learning has been previously used in solving multiple real-world problems including image processing [Kwan and Cai 1994],

computer vision [Keller 1996], computational biology and bioinformatics [Jin and Wang 2008], brain and cognition modelling [Ivancevic and Ivancevic 2007], portfolio management [Wong et al. 1992; Mehlawat and Gupta 2014], and motor control [Lin et al. 2001]. Fuzzy systems have also been extensively used in combination with neural networks to give rise to neuro-fuzzy systems [Walia et al. 2015; Mitra and Hayashi 2000]. However, it has not been extensively used in conjunction with modern-day deep learning architectures.

To specifically address the drawbacks of deep learning, multiple approaches have been proposed. Some models, such as Fuzzy Restricted Boltzmann Machines [Chen et al. 2015], use the concept of fuzzy numbers to denote the network weights. An implementation of this was proposed for airline passenger profiling [Zheng et al. 2017b] and for an early warning system for industrial accidents [Zheng et al. 2017a]. Another way in which fuzzy systems are integrated into deep learning is by using fuzzy logic units instead of perceptrons in the network [Park et al. 2016], which is similar to the neuro-fuzzy approach. Fuzzy systems have also been implemented to train some of the network parameters of a deep neural network [El Hatri and Boumhidi 2018].

The article is mainly divided into two parts: The first section explores how fuzzy logic is integrated as a part of deep learning models that is presented in Section 2, whereas the second part presents models that are ensembles of deep learning and fuzzy logic methods, presented in Section 3. This sectioning is illustrated in Figure 1. Table 1 provides a chronological overview of the models discussed in the following sections.

2 INTEGRATED MODELS

This section describes in detail the models that make use of fuzzy systems as a part of the learning mechanism.

2.1 Pythagorean Fuzzy Deep Boltzmann Machines

The Pythagorean Fuzzy Deep Boltzmann Machine (PFDBM) propsed by Zheng et al. [2017b] is an extension of the Fuzzy Restricted Boltzmann Machine (FRBM) [Chen et al. 2015]. The FRBM is based on the Restricted Boltzmann Machine (RBM) [Rumelhart et al. 1986]. RBMs have been used as building blocks for deeper models [Le Roux and Bengio 2008; Srivastava and Salakhutdinov 2012; Kuremoto et al. 2014; Salakhutdinov and Hinton 2009; Mohamed et al. 2009]. Similarly, FRBM has been used as a building block for PFDBM. This model is used for airline passenger profiling where data are often incomplete and vague. Fuzzified neural networks are known to handle both labelled and unlabeled data as well as incomplete features [Yan et al. 2014]. Moreover, fuzzy deep Boltzmann machine performs parameter learning over a larger area [Chen et al. 2015]. The fuzzy DBM proposed by Zheung et al. incorporates the concepts of Deep Boltzmann Machines [Salakhutdinov and Hinton 2009], Pythagorean Fuzzy Sets [Yager 2013], and Pythagorean Fuzzy Numbers [Zhang and Xu 2014].

A standard Deep Boltzmann machine [Hinton et al. 2006] is an extension of the restricted Boltzmann machine where the number of hidden layers is more than one. The increased number of hidden layers make it a deep model and allow us to capture more complex correlations of the activities of the preceeding layers [Salakhutdinov and Larochelle 2010]. For a DBM with two hidden layers, the set of layers is $\{x, h_1, h_2\}$. This makes the free energy function as follows:

$$E(x, h_1, h_2, \theta) = -x^T W_1 h_1 - h_1^T W_2 h_2. \tag{1}$$

$\theta = [W_1, W_2]$ is the vector containing the set of parameters. Now, the probability of the model assigning a certain parameter to the input vector x is simply

$$P(v, \theta) = \frac{1}{Z(\theta)} \sum_{h1} \sum_{h2} e^{-E(x, h_1, h_2, \theta)}. \tag{2}$$

Table 1. A Chronological Overview of the Fuzzy Deep Learning Models

Year	Model	Key Feature
2014	Deep Belief Networks with Fuzzy Granulated Inputs [Zhang et al. 2014]	- First deep neural neural network architecture to use fuzzified time-series input data - Effectively reduces noise and redundant information
2014	Deep Belief Network with fuzzified outputs [Zhou et al. 2014]	- Extends a deep binary classification model by implementing fuzzy memberships.
2015	Fuzzy Restricted Boltzmann Machine [Chen et al. 2015]	- Extends the concept of RBM by implementing fuzzy weights - Building block for deeper architectures
2016	Fuzzy Deep Learning for tumor variation prediction [Park et al. 2016]	- Uses fuzzy logic operator instead of ANNs. - Reduced number of parameters drastically improves training times.
2016	Deep Learning with Fuzzy Feature Points [Wang et al. 2016]	- Uses radial basis function to fuzzify input data - Makes use of traditional deep CNN
2017	Restricted Boltzmann Machine with Fuzzy Inputs [Chopade and Narvekar 2017]	- Uses fuzzy logic to understand semantic similarity between input vectors and assign importance value to input vectors - RBM is used for feature extraction
2017	Pythagorean Fuzzy Deep Boltzmann machine [Zheng et al. 2017b]	- Extention of Fuzzy Restricted Boltzmann Machine - Uses Pythagorean fuzzy numbers as weights - Can effectively handle noisy or incomplete data - Implements a Biogeography-based learning
2017	Hierarchical Fused Fuzzy Deep Neural Network [Deng et al. 2017]	- Extracts deep representation and fuzzy membership values of data in parallel - Implements multi-modal learning
2017	Takagi Sugeno Fuzzy Deep Network [Rajurkar and Verma 2017]	- Uses Takagi Sugeno Fuzzy Inference system to map into to output in terms of "IF-THEN" rules
2018	Stacked Auto Encoder trained using Fuzzy Logic [El Hatri and Boumhidi 2018]	- Fuzzy logic system is used to adjust multiple neural network parameters - Fuzzy logic is also used to adjust learning rate
2018	Deep Convolutional Neural Networks using Weighted Fuzzy Active Shape Model [Tabrizi et al. 2018]	- Uses Gabor filter to fuzzify input data and reduce noise. - Makes use of traditional deep CNN

The basic RBM works with binary data. To work with real-valued data, the real value is transformed into a binary vector using Gaussian-Bernoulli RBM (GRBM) [Hinton and Salakhutdinov 2006]. GRBM, as the name implies, makes use of Gaussian units. The energy function is defined as [Cho et al. 2011]

$$E(x, h, \theta) = \sum_{i=1}^{D} \frac{(x_i - b_i)^2}{2\sigma_i^2} - \sum_{i=1}^{D}\sum_{j=1}^{P} W_{ij} h_j \frac{x_i}{\sigma_i^2} - \sum_{j=1}^{P} c_j h_j, \tag{3}$$

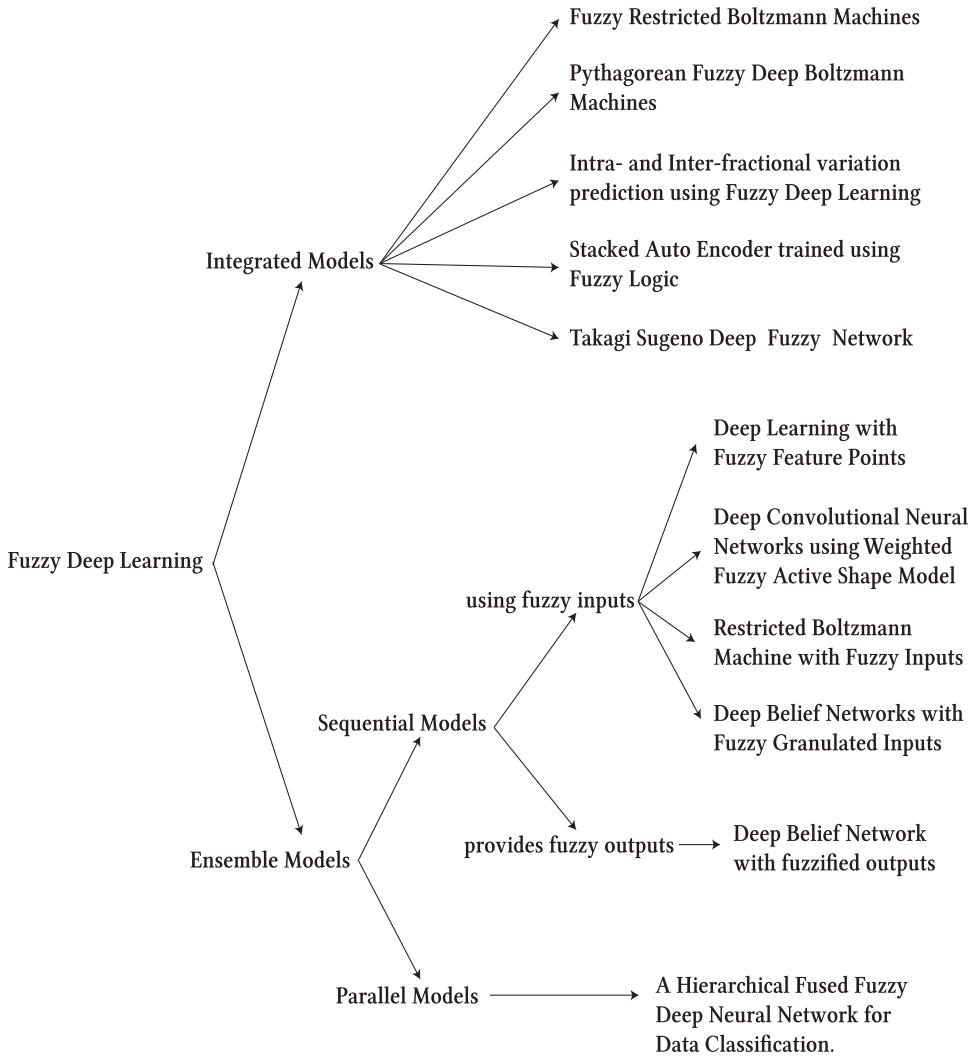

Fig. 1. Overview of the different deep learning models implementing fuzzy logic.

where the Gaussian visible neuron $x_i (1 < i < D)$ has standard deviation σ_i. b_i is the bias of each Gaussian neuron and c_i is the bias associated with each hidden neuron.

Now, the concept of the basic fuzzy sets [Zadeh 1996] is extended by Intuitionistic fuzzy set (IFS) [Atanassov 1986], which introduces a non-membership degree besides the standard membership degree. Pythagorean fuzzy sets [Yager 2014, 2013] is a further extension of IFS where the sum of the squares of the membership degree are from 0 to 1.

A Pythagorean Fuzzy Set (PFS), P, is defined as the following mathematical object:

$$P = \{< x, P(\mu_p(x), \upsilon_p(x)) > | x \in S\}. \tag{4}$$

Here, $\mu_p(x) : S \rightarrow [0, 1]$ and $\upsilon_p(x) : S \rightarrow [0, 1]$ are respectively the membership and non-membership degree of the element x to S in P, satisfying the expression $\mu_p(x)^2 + \upsilon_p(x)^2 \leq 1$. The

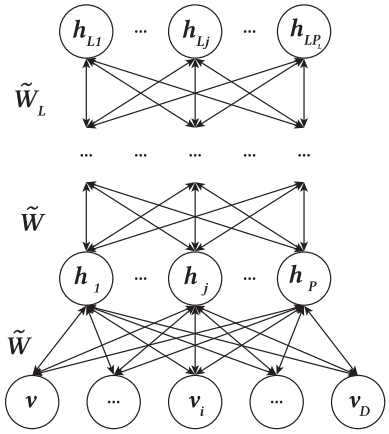

Fig. 2. Pythagorean Fuzzy Deep Belief Network (PFDBM) [Zheng et al. 2017b].

hesitant degree of $x \in X$ is written as

$$\pi_p(x) = \sqrt{1 - \mu_p^2(x) - \upsilon_p^2(x)}. \tag{5}$$

If $\beta = P\left(\mu_\beta, \upsilon_\beta\right)$ is a Pythagorean Fuzzy Number (PFN), then it satisfies $\mu_\beta, \upsilon_\beta \in [0,1]$ and $\mu_\beta^2 + \upsilon_\beta^2 \leq 1$. Using the operations defined on PFNs, two metrics [Zhang and Xu 2014] are defined that are used to rank a PFN. These are given as:

$$\begin{aligned} s(\beta) &= \mu_\beta^2 - \upsilon_\beta^2 \\ h(\beta) &= \mu_\beta^2 + \upsilon_\beta^2, \end{aligned} \tag{6}$$

where $s(\beta)$ is known as the score function and $h(\beta)$ is the accuracy function. Based on these the ranking of two PFN $\beta = P(\mu_{\beta_1}, \upsilon_{\beta_1})$ and $\beta = P(\mu_{\beta_2}, \upsilon_{\beta_2})$ is performed as follows [Ren et al. 2016]:

(1) If $s(\beta_1) < s(\beta_2)$, then $\beta_1 < \beta_2$.
(2) If $s(\beta_1) = s(\beta_2)$, then
    (a) If $h(\beta_1) < h(\beta_2)$, then $\beta_1 < \beta_2$
    (b) If $h(\beta_1) = h(\beta_2)$, then $\beta_1 = \beta_2$.

The Pythagorean Fuzzy Deep Belief Network (PFDBN) is built upon the DBM by using PFNs in place of the standard real-valued paramters. This fuzzy neural network can work with fuzzy and/or incomplete data [Ishibuchi et al. 1995, 1993]. Furthermore, (Figure 2) fuzzy parameters provide a better representation of the data using fuzzy probability [Klement et al. 1981; Yager 1999]. Moreover, the parameter learning space of the DBM increases with the introduction of the PFS [Chen et al. 2015].

The associated PFN parameters can learn new features and also sees how much a certain feature influences the output.

Given that the fuzzy parameters are $\widetilde{\theta} = [\widetilde{W_1}, \ldots, \widetilde{W_L}]$ of a PFDBM with layers, $h_1, \ldots, h_L$, then the energy function is given as [Zheng et al. 2017b]:

$$\widetilde{E}\left(v, h_1, \ldots, h_L, \widetilde{\theta}\right) = -v^T \widetilde{W_1} h_1 - \sum_{l=2}^{L} h_{l-1}^T \widetilde{W_l} h_l, \tag{7}$$

and the corresponding probability is

$$\widetilde{P}(v,\widetilde{\theta}) = \frac{1}{\widetilde{Z}(\widetilde{\theta})} \sum_{h_1} \cdots \sum_{h_L} e^{-\widetilde{E}\left(v,h_1,\ldots,h_L,\widetilde{\theta}\right)}. \tag{8}$$

Now, the objective of the learning algorithms is to learn a set of optimal parameters such that the log likelihood is maximized (or the negative log likelihood is minimized),

$$\max_{\widetilde{\theta}} \widetilde{\mathcal{L}}\left(\widetilde{\theta}, D\right) = \sum_{v \in D} \log\left(\widetilde{P}\left(v,\widetilde{\theta}\right)\right). \tag{9}$$

However, in general, fuzzy optimization problems are quite intractable. So, using the scoring functions, it is converted to a crisp optimization problem. Now, the PFDBM is trained using a combination of gradient descent and metaheuristic techniques.

The PFDBM is trained by iteratively using greedy layerwise training [Bengio et al. 2007] and a biogeography-based learning [Simon 2008] algorithm, since traditional methods are suboptimal [Chen and Wang 2012; Hu et al. 2015; Lander and Shang 2015]. Evolutionary algorithms have been used previously for ANNs, too, but not widely for DNNs [Yao 1993; Tettamanzi and Tomassini 2013; Tian et al. 2016; David and Greental 2014; Shinozaki and Watanabe 2015; Papa et al. 2016]. The biogeography-based learning algorithm is a metaheuristic optimization method that is inspired by biogeography.

The training algorithm randomly initiates a population of n solutions and a local topology. Next, it uses gradient-descent to train the network weights. Then and immigration and emigration operation, along with mutation operation, is performed. Then again, gradient descent is used to optimize the network. The old solution is replaced with a new one if the new one provides better values. Until the termination condition is satisfied, this process is continued.

2.2 Fuzzy Deep Learning for Lung Tumor Tracking

The synergy of deep learning and fuzzy logic has also been demonstrated by Park et al. [2016] in a machine that is used to predict the intra- and inter-fractional variation of lung tumours. In image-guided radiation therapy, it is important to monitor the target of the radiation very precisely to prevent damage to the surrounding healthy tissues [Yom et al. 2007; Kipritidis et al. 2015; Jan et al. 2015]. Features are extracted from respiratory signals, from which multiple metrics are first computed as done in previous studies, too [Lee et al. 2012b; Vedam et al. 2004; Lee et al. 2013; Murphy and Pokhrel 2009; Lu et al. 2006], such as "Standard deviation of time-series data," "maximum likelihood estimates," "breathing frequencies," and more. Based on these values, the patients are clustered [Lee et al. 2013] according to the similarity of the breathing features. For each patient group, the FDNN is trained to predict the two different types of variation.

The fuzzy deep learning (FDL) model is a fuzzy logic system that is organized in form of a neural network. Since the neural network is more than two layers deep, it is called a deep network. In the network [Park et al. 2016], the functional units are fuzzy rules instead of traditional artificial neurons with activation functions.

The neural network architecture gives the fuzzy logic system a self-learning feature that allows it to set its parameters automatically by using a variation of the gradient descent algorithm. In FDL, a small number of fuzzy parameters determine the weight values between the nodes. These are the prediction parameters. Furthermore, since there are fewer parameters, the performance is optimal for real-time prediction.

The network has four layers. They are summarized as follows:

Layer 1 assigns membership (as illustrated in Figure 3) values based on certain membership functions to the input. The functions take in a set of parameters $M = \{m_{i,1}, m_{i,2}, m_{i,3}\}$. Then a T-

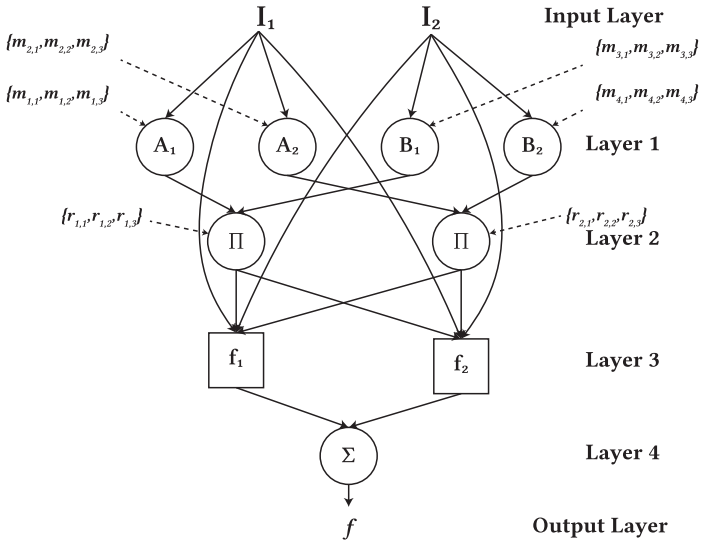

Fig. 3. Fuzzy Deep Learning architecture for tracking lung tumors [Park et al. 2016].

norm operation is performed in the next layer, that is, layer 2. In layer 3, based in the set of input parameters $R = \{r_{i,1}, r_{i,2}, r_{i,3}\}$, linear regression is done and, finally, layer 4 provides the output of the entire network by providing an additive outcome depending on the fuzzy if-then rules. According to Figure 5, the fuzzy if-then rules, which is the same as the number of nodes in layers 2 and 3, are given as follows:

1: If I_1 is A_1 and I_2 is B_1, then $f_1 = r_{1,1}I_1 + r_{1,2}I_2 + r_{1,3}$
2: If I_1 is A_2 and I_2 is B_2, then $f_2 = r_{2,1}I_1 + r_{2,2}I_2 + r_{2,3}$

Here, I_1 and I_2 are the inputs of the network, and A_i and B_i are the fuzzy sets.

Layer 1: The output of the the first layer, $O_{1,i}$, is given as:

$$O_{1,i} = \begin{cases} \mu_{A_i}(I_1) = \frac{1}{[1+|(1-m_{i,3})/m_{i,1}|]^{2m_{i,2}}} & 1 \le i \le 2 \\ \mu_{B_{i-2}}(I_2) = \frac{1}{[1+|1-m_{i,3}/m_{i,1}|]^{2m_{i,2}}} & 3 \le i \le 4 \end{cases}, \tag{10}$$

where for the fuzzy sets A_i and B_i, $\mu_{A_i}(I_1)$ and $\mu_{B_{i-2}}(I_2)$ are the membership functions. The training procedure chooses the membership parameters $m_{i,1}, m_{i,2}, m_{i,3}$.

Layer 2: Layer 2 performs a product of all the inputs coming from the preceding layer:

$$O_{2,i} = w_i = \mu_{A_i}(I_1) . \mu_{B_i}(I_2) \quad 1 \le i \le 2. \tag{11}$$

Multiplication is analogous to the T-norm operator and determines the output determines the degree of influence of the rule.

Layer 3: In the third layer, the ratio of the ith rule's degree of influence and the total degree of influence of all the rules is used to perform linear regression:

$$O_{3,i} = \frac{w_i}{\sum_j w_j}(r_{i,1}I_1 + r_{i,2}I_2 + r_{i,3}) \;\; 1 \le i \le 2, \tag{12}$$

where $\{r_{i,1}, r_{i,2}, r_{i,3}\}$ is the set of learning rate parameters determined from the training procedure.

Layer 4: This layer performs the mean as follows:

$$O_{4,1} = f = \sum \frac{w_i}{\sum_j w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}, \tag{13}$$

where f_i is the consequent of rule i.

The FDL is trained using a Hybrid Training Algorithm for Feed-forward Neural Networks as in Nasr and Chtourou [2006]. The training procedure finds the parameters set M and R, which provides the membership values and the regression parameters. These parameters are then used to train the network. Intra- and inter-fractional variation data are acquired from the CyberKnife dataset. For these data, a variable number of nodes is used. The dataset has three-dimenisonal (3D) coordinates, and for each channel a seperate network is employed. This also changes the number of nodes in each layer. For instance, both layer 2 and layer 3, in this case, will have 27 nodes. The rules are modified as follows:

Rule 1: If I_1 is A_1 and I_2 is B_1 and I_3 is C_1, then $f_1 = r_{1,1}I_1 + r_{1,2}I_2 + r_{1,3}I_3 + r_{1,4}$

Rule 27: If I_1 is A_3 and I_2 is B_3 and I_3 is C_3, then $f_27 = r_{27,1}I_1 + r_{27,2}I_2 + r_{27,3}I_3 + r_{27,4}$ Here, the input vector $\{I_1, I_2, I_3\}$ corresponds to the three channels of the CyberKnife machine. A_i, B_i, and C_i are the fuzzy sets. The computation of the output of the initial layer is now given as:

$$O_{1,i} = \begin{cases} \mu_{A_i}(I_1) = \frac{1}{[1+|(1-m_{i,3})/m_{i,1}|]^{2m_{i,2}}} & 1 \le i \le 3 \\ \mu_{B_{i-3}}(I_2) = \frac{1}{[1+|1-m_{i,3}/m_{i,1}|]^{2mi,2}} & 4 \le i \le 6 \\ \mu_{C_{i-6}}(I_3) = \frac{1}{[1+|1-m_{i,3}/m_{i,1}|]^{2mi,2}} & 7 \le i \le 9 \end{cases}, \tag{14}$$

where the different membership functions, μ_{A_i}, $\mu_{B_{i-3}}$, and $\mu_{C_{i-6}}$, are determined by the $M = \{m_{i,1}, m_{i,2}, m_{i,3}\}$. Similarly, the output of the succeeding layer is given as

$$O_{2,i} = w_i = \mu_{A_k}(I_1) . \mu_{B_l}(I_2) . \mu_{C_m}(I_3) \quad 1 \le i \le 27. \tag{15}$$

For the third layer, it is

$$O_{3,i} = \frac{w_i}{\sum_j w_j}(r_{i,1}I_1 + r_{i,2}I_2 + r_{i,3}I_3 + r_{i,4}) \ \ 1 \le i \le 2, \tag{16}$$

where $\{r_{i,1}, r_{i,2}, r_{i,3}, r_{i,4}\}$ is the set of LR parameters. Finally, the output is calculated as

$$O_{4,1} = f = \sum \frac{w_i}{\sum_j w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}, \tag{17}$$

where f_i is the consequent of rule i.

The final layer produces a single coordinate, that is, x, y, or z estimate. The IIFDL uses three FDL for each of the coordinates, allowing the estimation of 3D coordinates.

The model was tested using CyberKnife patient databases and was compared with two ANN-based models: a CNN [Lee et al. 2013] and a hybrid implementation of the extended Kalman filter (HEKF) [Lee et al. 2012a].

The IIFDL model outperformed the CNN both in terms of prediction accuracy and training speed. Since the IIFDL has fewer parameters, the training process is much faster. Prediction accuracy was measured using root mean square error for different time intervals.

2.3 Stacked Auto Encoder Trained Using Fuzzy Logic

Hatri et al. proposed a deep learning model that uses fuzzy logic to train some of the network parameters that improve the learning speed [El Hatri and Boumhidi 2018] and has a higher probability to reach a better global optimum. The deep learning network is based on stacked-auto encoders (SAE) that are trained using backpropagation but the learning rate and the momentum are determined using fuzzy logic systems. Here, three auto-encoders are used, making the network deep.

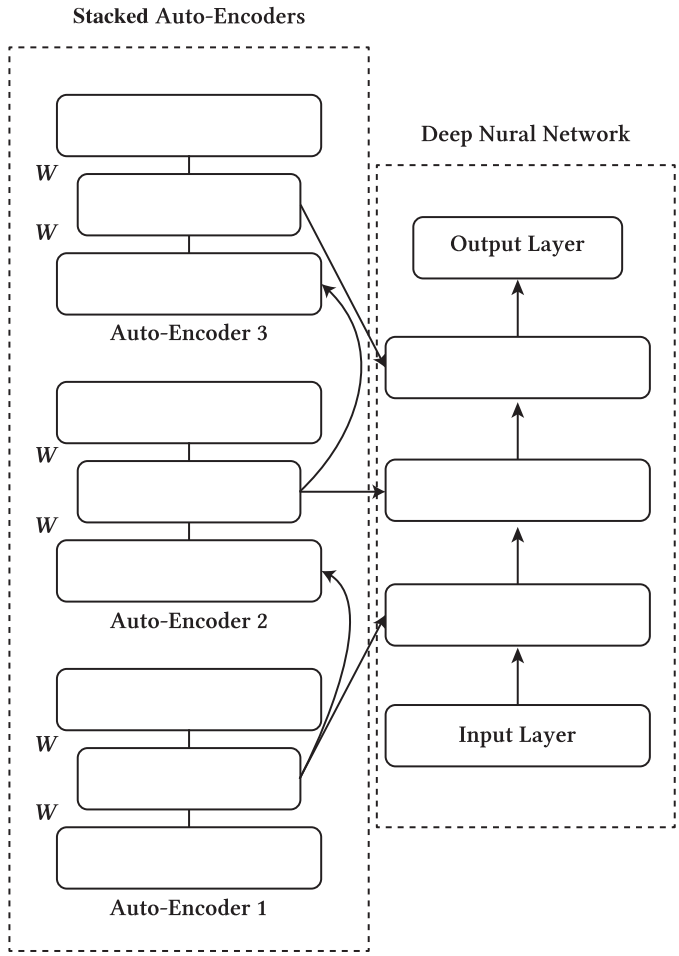

Fig. 4. Stacked auto-encoders [El Hatri and Boumhidi 2018].

A stacked auto-encoder is constructed using a series of auto-encoders (AE), one on top of the other (as illustrated in Figure 4) where the output of one auto-encoder is the input of the auto-encoder that is above it. In the simplest form, an auto-encoder is a neural network with a single hidden layer. The input layer nodes and output layer nodes are same in number. The number of nodes in the hidden layer is fewer.

The auto-encoder tries to replicate the input at the output by passing it through a fewer number of nodes. As the data are passed through the hidden layers, which are fewer in number, the network compresses the data and learns only the important features. For a set of input vectors $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots\}$ where each input $x^{(i)} \in \mathbb{R}^d$, the autoencoder first encodes every input $x^{(i)}$ to a compressed form $y(x^{(i)})$, which extracts the dominant features and then reconstructs or decodes that compressed representation into $z(x^{(i)})$. This is given as:

$$y(x) = f(W_1.x + b_1), \tag{18}$$

$$z(x) = g(W_2.y(x) + b_2), \tag{19}$$

where W_1 and b_1 are the encoding weights and biases, and W_2 and b_2 are the decoding weights and bias. The training process can be done using traditional backpropagation. The error or the loss is

calculated as

$$E\left(X,\ Z\right) = \frac{1}{2}\sum_{i-1}^{N}\left|x^i - z^i\right|^2, \tag{20}$$

where the number is samples is N, X is the sample set, and Z is the output that is constructed by the AE. x_i is an element of X, and z_i is an element of Z. When training an SAE, each autoencoder is trained individually, layer by layer. This means for a three-layer SAE, the first auto-encoder is trained first, and we obtain the weights W_1^1 and W_1^2. Then, for each AE i ($i = 1, 2, 3$), the current autoencoder is trained using the output of the preceding one, that is, y^{i-1}, in such a way that W_1^i and W_2^i are obtained. In this method, the SAE is trained.

The fine-tuning strategy that is used in this case is the simple back-propagation algorithm. During fine-tuning, from a high level, all the SAEs are treated as a single model. Therefore, the tuning process improves all the weights of the auto-encoders in a single iteration. Hence, the error of the entire network is defined as

$$E\left(t\right) = \frac{1}{2}\left|e\left(t\right)\right|^2, \tag{21}$$

where $e(t)$ is the error value. This difference between the input and the output determines the changes in the network weights, which is given as

$$\triangle w_j\left(t+1\right) = \eta\left(\frac{\delta E\left(t\right)}{\delta w_j}\right) + \alpha\triangle w_j(t), \tag{22}$$

where $\triangle w_j\left(k\right)$ is the weight in the kth iteration and $\frac{\delta E(t)}{\delta w_j}$ is the rate of change of error with respect to the weights.

Here, η is the learning rate parameter and α is a positive value called the momentum. The learning rate and the momentum is dynamically adjusted using a fuzzy logic control system. A fuzzy logic control is used to adaptively adjust the neural network parameters such that the mean squared error (MSE) is reduced [Eze et al. 2014]. To create the fuzzy logic system, four parameters are used. These include the following:

$$\begin{aligned}
\text{Relative Error} &: RE\left(t\right) = E\left(t\right) - E\left(t-1\right)\\
\text{Change in Relative error} &: CRE = RE\left(t\right) - RE\left(t-1\right)\\
\text{Sign change of error} &: SC\left(t\right) = 1 - ||\frac{1}{2}(sign(RE(t-1)) + sign(RE(t))||\\
\text{Cumulative sum of sign change} &: CSC = \sum_{m=t-4}^{t} SC\left(m\right).
\end{aligned} \tag{23}$$

Using the fuzzy logic system, the change of learning rate and the change of momentum are done.

The system was used to predict traffic incidents while using an urban traffic simulator to synthesize training and testing data. When compared to deep neural network, the FDNN had much better training times and marginally better accuracy when detecting traffic incidents.

2.4 Takagi Sugeno Deep Fuzzy Network

The Takagi Sugeno Deep Fuzzy Network (TSDFN) [Rajurkar and Verma 2017] is an attempt to make a simple and generic architecture that can be used to develop deeper networks. The proposed architecture is a three layers network built using Takagi Sugeno Fuzzy Inference System (TSFIS) [Takagi and Sugeno 1985]. The network has three layers: input, hidden, and output. In most fuzzy neural networks [Buckley and Hayashi 1994; Lee and Lee 1975; Gupta and Rao 1994; Du and Wolfe

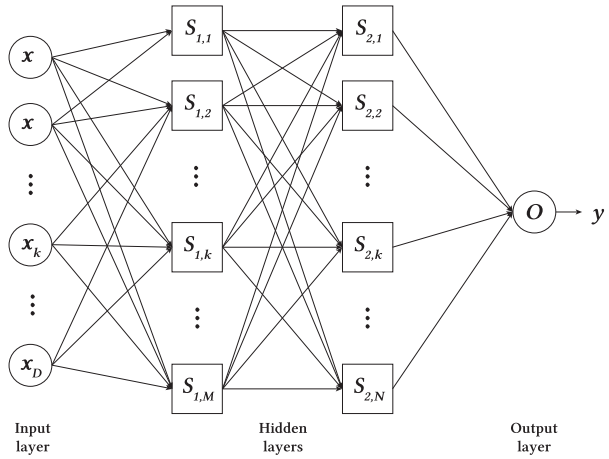

Fig. 5. A generic Takagi Sugeno Deep Fuzzy Network with two hidden layers and one output node. The number of nodes in the hidden layers and the output layer can be varied according to the application. The model in Rajurkar and Verma [2017] has only one hidden layer.

1995; Vieira et al. 2004; Chen and Teng 1995], only one fuzzy rule base is identified. These methods cannot simultaneously model (as shown in Figure 5) distinct fuzzy structures as a single fuzzy inference system is used. Therefore, for the entire system, the corresponding fuzzy membership functions are the same. When nodes of Fuzzy Inference Systems (FIS) are connected in form of layers, the deeper layers can model a more abstract fuzzy rule base from the fuzzy rule bases of the shallower layers. Hence, stacking more fuzzy inference systems can give us higher level fuzzy rule bases.

In the proposed architecture, the TSDFN is made from multiple layers of TSFIS. Every node is a completely distinct TSFIS with their own distinct parameters. Therefore, every since TSFIS also comes with its own FRB. This makes the proposed TSDFN a model with very high abstraction qualities. A fuzzy rule base of a TS FIS comes with multiple rules. Each rule is composed of two parts: a premise and a consequent. The fuzzy rule is given as

R_i^h : IF x_1 is $G_{1,i}$ AND x_2 is $G_{2,i}$ AND ... AND x_D is $G_{D,i}$ THEN y is $y_i = p_{i,0} + p_{i,1}x_1 + \cdots + p_{i,D}x_D$,

where R_i^h is the rule i of the hidden unit h, IF ... is the premise, and THEN ... is the consequent. The input k is given as x_k, where $k = (1, 2, \ldots, D)$, $G_{k,i}$ is the corresponding fuzzy membership function and $p_{i,k}$ are the coefficients of the consequent. In this case, the consequent part is a linear function. For the premise part of the FRB, "AND" is used as the fuzzy combinaion operator. The FRB maps the input and output in terms of "IF-THEN" relations that can be understood by humans.

The trainable parameters in this model include the parameters that are used to define the input membership function, the membership values of the input, and the coefficients of the consequent part. The membership function used in this case is the Gaussian membership function, since it is continuous and differentiable. The differentiability is important, since the network is trained using gradient descent and back-propagation. Other kinds of fuzzy rules and membership functions can also be used, depending on the application, as long as back-propagation is possible.

The TSDFN seems to outperform artificial neural networks having similar architectures on the Truck Backer Upper problem dataset [Wang and Mendel 1991]. TSFIS networks such as

the proposed TSDFN has much higher robustness when compared to ANNs, specially when vagueness and imprecision is introduced.

3 ENSEMBLE MODELS

The following models use fuzzy logic and deep learning in a sequential or parallel fashion. The input vector to the deep neural network is either pre-processed using a fuzzy logic system, or the input is transformed by the deep neural network and fuzzy systems in parallel.

3.1 Sequential Models

This section lists all the sequential models. Sequential models either make use of fuzzified inputs to existing deep learning architectures, or provides a fuzzy output. Based on the nature of the input and output, this section is further divided into two parts: Models with fuzzy inputs and models with fuzzy outputs.

3.1.1 Models with Fuzzy Inputs.

Deep Learning Using Fuzzy Feature Points: Wang et al. proposed a model that uses Deep Learning with Fuzzy Feature Points for damaged fingerprint classification [Wang et al. 2016]. The model essentially involves fuzzyfying the input data and then providing the input to a simple convolutional neural network. Since Wang et al. [2016] provides a very generic model that uses image data, the model can be leveraged for other visual recognition tasks, too.

Before the data is fed to a CNN, it undergoes preprocessing, feature extraction, and matching. The fuzzy features that are fed to the model is created from the preprocessed fingerprint image data. This involves the core and delta extraction, as well as endpoint and branch point extraction. To extract the core and delta points, the Poincare formula [Iwasokun and Akinyokun 2014] is used.

Once the feature points are extracted, they are fuzzified using the radial basis function. Here, the radial basis function is used for interpolation of the data points.

A radial basis function is given as $\phi(x) = \phi(||x||)$. There are various types of radial basis functions. The Gaussian type is given as $\phi(r) = e^{-(\epsilon r)^2}$, where $r = ||x - x_i||$. In this case, the "Gauss distribution function of Kriging method" is used, which is given as $\phi(r) = e^{-(r/\sigma)^2}$. The goal here is to reduce the influence certain defective fingerprint lines so that the relationship between the different feature points are highlighted. Weakening the specific positions of the feature points provide a higher degree of rotational invariance. The fuzzy graph of the feature points is then provided to a CNN [Lin et al. 2014] that predicts the class of the fingerprint. A generic deep CNN using sigmoidal activation function is used in the classification of the fuzzified data.

This provided a marginally improved recognition rate over the pre-processed original fingerprint image and is massively outperforms the CNN trained with the original fingerprint image.

Deep Convolutional Neural Networks Using Weighted Fuzzy Active Shape Model: A similar model where deep learning and fuzzy logic is used for image segmentation is proposed in Tabrizi et al. [2018]. In this case, 3D ultrasound images of kidneys are performed. The idea is quite simple: fuzzy clustering is performed on the image data for better characterization of the different parts of the images. Furthermore, to reduce the noise of the image and to enhance the edges, an omni-directional Gabor-filter is used.

The model is compared with manual segmentation of 3D ultrasound images of kidneys, and the performance of the DNN is at par with the manual process. This process can be used for segmentation of other medical images and even in other areas of computer vision.

The DNN has two convolution layers. They are succeeded by two fully connected layers. Each layer uses a rectified linear unit [Nair and Hinton 2010]. The last layer uses the softmax probability function [Bishop 2012] that is used to evaluate the position, orientation and scale. To

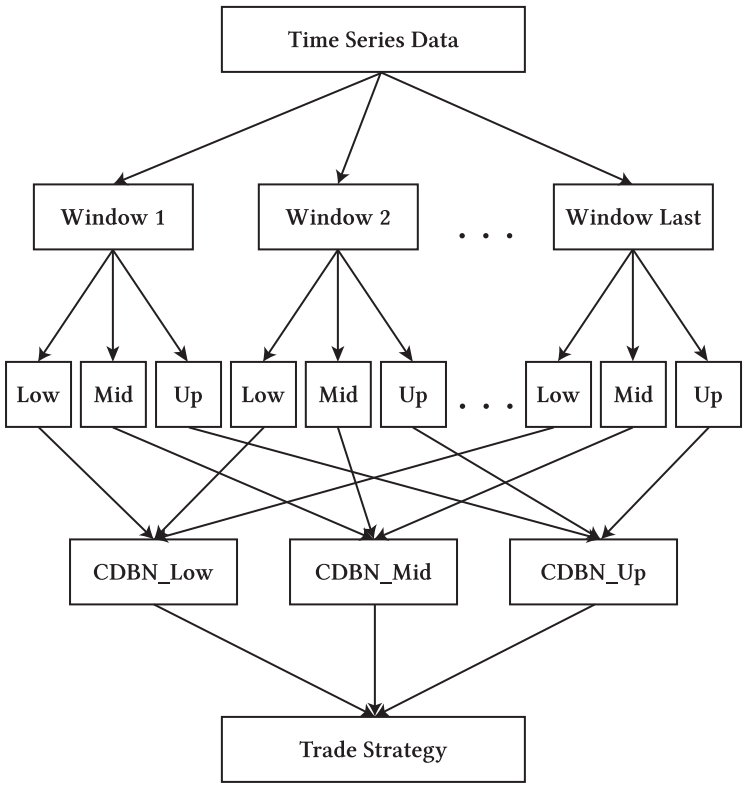

Fig. 6. Framework of the CDBN-FG that combines fuzzy granulation and deep belief networks for continuous data [Zhang et al. 2014].

reduce overfitting, a dropout [Srivastava et al. 2014] rate of 0.4 was set in the last fully connected layer.

Deep Belief Networks with Fuzzy Granulated Inputs: Zhang et al. propose a model utilizing fuzzy granulation and DBN for predicting time-series data [Zhang et al. 2014]. The concept of information granules comes from granular computing. Information granules are groups of entities that are numerically similar (Figure 6). The similarity can be because of their similar functionality, similar physical properties, or because of any other attributes that are similar numerically. The aim is to decompose a problem into simpler problems while removing some irrelevant information. In this case, fuzzy information granulation [Zadeh 1997] is done on time-series data before they are used to train and test deep belief networks for continuous data (CDBN) [Chao et al. 2011].

In fuzzy information granulation, the data are granulated into three sets, or in this context, granules, given as Low, Mid, High, using multiple different granulation methods. Ruan et al. proposed five fuzzy granulation methods in Ruan et al. [2013]: triangular membership function-based granulation, trapezoidal membership function-based granulation, Gaussian membership function-based granulation, fuzzy c-means-based granulation, and min-max-based granulation.

The time-series data are first split into multiple windows. Given that $T = t_1, t_2, \ldots, t_n$ are time-series data and w is the length of the granulation windows where $1 \leq w \leq n$, the time-series data can be split as

$$\{t_1, t_2, \ldots, t_w\}, \{t_{w+1}, t_{w+2}, \ldots, t_{w+w}\}, \ldots, \{t_{([n/w]-1)w+1}, t_{([n/w]-1)w+2}, \ldots, t_{([n/w]-1)w+w}\}. \, [n/w] \, .$$

The granulation methods are described in Zhang et al. [2014] as follows:
Triangular MF-based granulation:

$$\begin{aligned} low_i &= \frac{2\sum_{j=1}^{w/2} t^i_{j'}}{w/2} - median\left(t^i_{1'}, t^i_{2'}, \ldots, t^i_{w'}\right) \\ mid_i &= median\left(t^i_{1'}, t^i_{2'}, \ldots, t^i_{w'}\right) \\ up_i &= \frac{2\sum_{j=w/2+1}^{w} t^i_{j'}}{w/2} - median\left(t^i_{1'}, t^i_{2'}, \ldots, t^i_{w'}\right), \end{aligned} \tag{24}$$

Trapezoidal MF-based granulation:

$$\begin{aligned} low_i &= \frac{2\sum_{j=1}^{w/2} t^i_{j'}}{w/2} - t^i_{(w/2)'} \\ mid_i &= median\left(t^i_{1'}, t^i_{2'}, \ldots, t^i_{w'}\right) \\ up_i &= \frac{2\sum_{j=w/2+1}^{w} t^i_{j'}}{w/2} - t^i_{(w/2+1)'}, \end{aligned} \tag{25}$$

Minmax-based granulation function:

$$\begin{aligned} low_i &= t^i_{1'} \\ mid_i &= median\left(t^i_{1'}, t^i_{2'}, \ldots, t^i_{w'}\right) \\ \mathrm{up}_i &= t^i_{w'}. \end{aligned} \tag{26}$$

The time-series data are first split into multiple windows, and each window is granulated using fuzzy granulation techniques. There are three granules that are derived from each window. The Low, Mid, and Up granules are trained using three separate CDBNs. CDBNs employ continuous valued restricted Boltzmann machines (CRBMs) that use continuous valued stochastic units. Let s_j be the output of the neuron j, with units from neurons with state $\{s_i\}$.

$$s_j = \varphi_j\left(\sum_i w_{ij}s_i + \gamma.N_j\,(0,1)\right), \tag{27}$$

where $\varphi_j(x_j) = \theta_L + (\theta_H - \theta_L)\,.\frac{1}{1+e^{-a_j x_j}}$. The random Gaussian number $N_j\,(0,1)$ has a mean value 0 and a variance of 1. The value γ is a constant. The sigmoid function $\varphi_j(x_j)$ has θ_L and θ_H as asymptotes, and the value a_j is a parameter that is introduced to control the noise [Hu et al. 2007; Yao and Yao 2003]. The update rules for w_{ij} and a_j are given as

$$\begin{aligned} \triangle w_i j &= \eta_w\left(< s_i s_j > - < s'_i s'_j >\right) \\ \triangle a_j &= \frac{\eta_a}{a_j^2}\left(< s_j^2 > - < s_j'^2 >\right). \end{aligned} \tag{28}$$

The learning rates are η_w and η_a, s'_j denotes a single step of sampling of the state on unit j, and <.> denotes the mean. The CDBN is trained in a layer-by-layer fashion where the hidden layers or each CRBM is trained sequentially. For training a network with k input nodes, the initial training sample for $CDBN_{Low}$ is composed of $\{Low_1, Low_2, \ldots, Low_k\}$. Given this input, the CDBN is trained to predict Low_{k+1}. In the next phase, the input vector $\{Low_2, Low_3, \ldots, Low_{k+1}\}$ is used to predict the value of Low_{k+2}, and so on. The same is done with $CDBN_{Mid}$ and $CDBN_{Up}$.

When tested on stock market data, the CDBN, while using fuzzy granulation outperforms the standard CDBN. The granulation window size, as well as the membership function parameters, can be tweaked for improved performance on various datasets.

3.1.2 Models with Fuzzy Outputs.

Restricted Boltzmann Machine with Fuzzy Outputs: Chopade et al. implemented a combination of fuzzy logic and deep learning to perform document summarization [Chopade and Narvekar 2017]. Here, the summarization is done in multiple phases. First, features are extracted from the sentences and a sentence matrix is formed based on the features. Each sentence is also manually assigned an importance value. This dataset is used to train a restricted Boltzmann machine to get a more refined set of sentences. Next, the sentences are processed, and a table is created to rank the sentences using a priority value. From there, a fuzzy membership function is used to determine the sentences that will be used for summarization.

In the feature extraction phase, the features that are extracted includes title similarity, term weight, count of named entities, and the amount of numeric data. Using these features, a sentence matrix is formed.

The implementation of a restricted Boltzmann machine, in this case, is quite simple. The sentence matrix is the input of the restricted Boltzmann machine with three layers. The output of the machine is a much more refined set of sentences.

In the next stages, the sentences are processed more. Stemming, stop word removal, and part of speech tagging is performed. Other than the sentence table, two more tables are created, a seed table containing word priority values and a word table containing the word frequency values. From these three tables, the rank of each sentence is calculated. This is essentially a fuzzy membership function that determines the priority of the sentence in the document summary,

$$Rank\,(S_i) = \sum_{i=1}^{n} frequency\left(w_{ij}\right) + \sum_{i=1}^{n} membership\_with\_seed\,(w_i j), \tag{29}$$

where the total count of sentences in the sentence table is denoted by **n**, the ith sentence is S_i, and the jth word is W_{ij}. The fuzzy membership is determined by the function Membership_with_seed ().

Deep Belief Network with Fuzzified outputs: A "fuzzy deep belief network for semi-supervised sentiment classification" is a simple example of a sequential model that uses a deep belief network and fuzzy sets for sentiment classification is proposed in Zhou et al. [2014]. A Deep belief network (DBN) are trained one layer after the other, in a layerwise fashion, using a greedy layerwise training algorithm, and then are fine-tuned by standard backpropagation algorithms. In the context of sentiment classification performed in a semi-supervised fashion, fuzzy sets were used to describe the membership value with which a pattern belongs to a positive or a negative class. The model can be used for other binary class problems, too.

First, the DBN is trained with both unlabeled and labelled data. Training with unlabeled data is performed in a greedy layerwise manner by considering two adjacent layers as a restricted Boltzmann machine (RBM). Next, labelled data are used to further fine-tune the network using backpropagation and gradient descent. In this sentiment classification problem is a positive class and a negative class. Hence, a single hyperplane is needed for separating the two classes. Two membership functions are formulated that describes the membership of the input in a class and the distance of the input from the hyperplane.

The DBN [Smolensky 1986] is trained using gradient descent. Now, since there are just two classes, only one hyperplane is needed. If h^N is the last layer and z^i is the ith input vector, then

$d(z^i) = (h_1^N(z^i) - h_2^N(z^i))/\sqrt{(2)}$ is the distance between the separating line and the point $h_1^N(z^i)$. z^i is positive if $d(z^i) > 0$, and otherwise it is negative.

Now, the membership functions [Rutkowska 2012] $\mu_A(z)$ and $\mu_B(z)$ are given in Zhou et al. [2014] as

$$\mu_A(z;\beta,\gamma) = \begin{cases} S(d(z);\gamma-\beta,\gamma-\beta/2,\gamma), d(z) \le \gamma \\ 1, d(z) \ge \gamma \end{cases}, \tag{30}$$

$$\mu_B(z;\beta,-\gamma) = \begin{cases} 1, d(z) \le \gamma \\ 1 - S(d(z);\gamma,\gamma-\beta,\gamma-\beta/2), d(z) \ge \gamma \end{cases}, \tag{31}$$

where $S(d;\alpha,\beta,\gamma)$ is

$$S(d;\alpha,\beta,\gamma) = \begin{cases} 0, d \le \alpha \\ 2\left(\frac{d-\alpha}{\gamma-\alpha}\right)^2, \alpha \le d \le \beta \\ 1 - 2\left(\frac{d-\alpha}{\gamma-\alpha}\right)^2, \beta \le d \le \gamma 1, \quad d \le \gamma \end{cases}. \tag{32}$$

Now, there are two parameters to be determined, β and γ. The distance metric can be used in the following manner to get the values:

$$\begin{aligned} \gamma &= max\left|d\left(z^i\right)\right|, i = 1,\ldots,R+T \\ \beta &= \xi \times \gamma, \ \xi \ge 2. \end{aligned} \tag{33}$$

Here, ξ is a data-dependent constant used to determine the degree of separation of the classes, R and T are the number of training and test reviews, respectively.

3.2 Parallel Models

3.2.1 A Hierarchical Fused Fuzzy Deep Neural Network. Deng et al. proposed a deep learning architecture has deep learning layers and fuzzy membership functions running in parallel [Deng et al. 2017]. The deep representation and the fuzzy representation is then fused using multimodal learning techniques [Ngiam et al. 2011].

The model architecture can be divided into four sections.

Fuzzy Logic Representation: Each node of the input layer is connected to fuzzy logic nodes that computes fuzzy membership of each of the inputs, as shown in Figure 7. The input is an uni-dimensional vector. This layer determines how much a given input vector belongs to a certain set. It uses a standard Gaussian membership function [Lin et al. 2006, 2001] to do so. This layer maps the input to a real value between 0 and 1, which is basically the fuzzy membership value of the input variable. The fuzzy membership function is $u_i(.)$ and the output of the function is defined as follows:

$$o_i^{(l)} = u_i\left(a_k^{(l)}\right) = e^{-\frac{\left(a_k^{(l)}-\mu_i\right)^2}{\sigma_i^2}}, \tag{34}$$

where l denotes the position of the layer from the input layer, the input of node i is given as $a_i^{(l)}$, and the corresponding output is given as $o_i^{(l)}$. The succeeding layer performs another fuzzy operation. In this case, it is an AND operation,

$$o_i^{(l)} = \prod_j o_j^{(l-1)} \ \forall j \in \Omega_i, \tag{35}$$

where Ω_i is the set of nodes in the preceding layer that connect to the node i. The output is a real-value in the range (0,1).

Neural Representation or Deep Representation: This is a simple fully connected layer that is implemented in multi-layer perceptrons. The feed-forward layers pass the input from

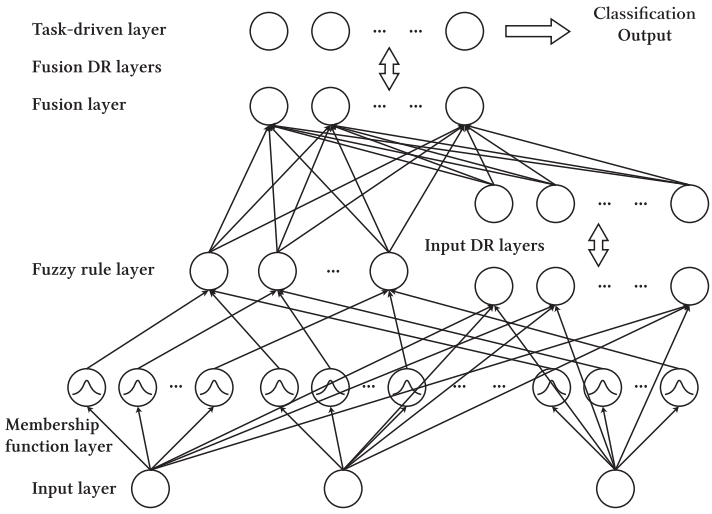

Fig. 7. Structure of the hierarchical Fused Fuzzy Deep Neural Network [Deng et al. 2017].

the preceding layer through a sigmoid function. The input of the preceding layer is the output of the layer before that times the weight and the bias. This is given as $o_i^{(l)} = \frac{1}{1+ e^{- a_i^{(l)}}}$, where $a_i^{(l)} = w_i^{(l)} o^{(l-1)} + b_i^{(l)}$. Here, $w_i^{(l)}$ is the weight of the node i on layer l, and $b_i^{(l)}$ is the corresponding bias.

Fusion layers: The fusion layers brings together the output of the fuzzy layers and the neural representation layers. The idea of fusion layers are inspired by multimodal learning [Ngiam et al. 2011]. In this set of layers, the model can learn parameters based on the classification task. Multimodal learning is a good way to represent heterogeneous data or data that have been acquired from different areas. This allows one data source to fill in the gaps that are present in the other. So, in cases where extracting feature from a single source is not possible, such learning techniques are used. In the case of this model, output of the the neural and the fuzzy layers are combined in the fusion layers. However, this is quite a trivial task. As the output of the sigmoidal nodes as well as the fuzzy nodes both lie in the range (0,1), combining the results is quite simple,

$$a_i^{(l)} = (w_d)_i^{(l)} (o_d)^{(l-1)} + \left(w_f\right)_i^{(l)} \left(o_f\right)^{(l-1)} + b_i^{(l)}. \tag{36}$$

Here, the output a_i takes in the output and connection weight of the deep representation part, given by o_d and w_d, respectively. Similarly, for the fuzzy part, the output and the weights are o_f and w_f.

Task Driven Layer: The final layer takes input from the fusion layers and provides corresponding class probabilities. Class probabilities are computed using the soft-max function. Given that f_i is the ith input, y_i is the label of the input, and $\pi_\theta(f_i)$ is the transformation of the input after passing throught the network, then the output of the task driven layer is given as:

$$\hat{y}_{ic} = p\left(y_i | f_i\right) = \frac{e^{w_c \pi_\theta(f_i) + b_c}}{\sum_c e^{w_c \pi_\theta(f_i) + b_c}}. \tag{37}$$

For a given class c, w_c is the regression coefficient and b_c is the bias. $\hat{y}_i = [\hat{y}_{i1}, \ldots, \hat{y}_{ik}]$ is the list of labels for a k-class problem.

The training is done in two distinct phases—learning and fine-tuning. Initially, all the biases are set to 0, which the weights between the nodes are randomly sampled from an uniform distribution

$U[-\frac{1}{\sqrt{n_k}}, \frac{1}{\sqrt{n_k}}]$ [Glorot and Bengio 2010], where n_k is the number of nodes in the preceding layer. All the weights between the fuzzy nodes are set as 1.

To set the parameters, that is, the mean and the standard deviation, of the fuzzy membership nodes, a fast clustering algorithm is run on the input data set. A k-means clustering is done where k is the number of classes. This clustering gives the required parameters after which the model is trained. The network is trained using back-propagation. For fine-tuning, stochastic gradient descent is used.

The model is also compared with a standard DNN as well as a sequential fuzzy DNN (SFDNN). The DNN only uses the neural representation part while the SFDNN fuzzifies data first and then uses it for prediction. Two benchmarks were used to compare the models.

The first is a natural scene image classification problem [Deng et al. 2014; Van Gemert et al. 2008]. The dataset has 4,500 images and 15 classes. Two-thirds of the data was used for training while the rest was used for testing. The FDNN had approximately 0.8% improvement over the other models.

The second benchmark is Stock Trend Prediction [Deng et al. 2015, 2016]. This was a three-class problem, where the model would predict whether the stock prices would drop, stay the same, or rise. In this case, too, the FDNN showed marginal improvement over the other competing models.

4 APPLICATIONS

Deep learning models that use fuzzy logic have been implemented in various studies and real-life applications. These are summarized below:

4.1 Traffic Control

In the case of traffic control, the objective is to predict traffic flow or detect traffic incidents. Fuzzy restricted Boltzmann machines have been used for designing models for network traffic prediction [Nie et al. 2017]. It has also been used for traffic flow prediction [Chen et al. 2018]. In Hernandez-Potiomkin et al. [2018], stacked auto-encoders used for urban traffic incident detection and traffic congestion detection. The model proposed in Deng et al. [2017] has been used for driver drowsiness detection using wearable devices [He et al. 2017]. Model with fuzzy granulated inputs have been also used in road safety analysis [Pan et al. 2017].

4.2 Surveillance and Security

Fuzzy Restricted Boltzmann Machines [Chen et al. 2015] have been the inspiration behind the Pythagorean fuzzy deep Boltzmann machine that is used for airline passenger profiling [Zheng et al. 2017b]. FRBM-based models are also used for radar target recognition [Xia et al. 2016] for classifying different types of airplanes. Since radar data tend to be noisy, the proposed FRBM model outperforms other conventional methods. Models that make use of fuzzy outputs as mentioned in Section 3.1.2 have been used for recognizing human activity in smart homes [Fang and Hu 2014].

4.3 Text Processing

In text summarization, important sentences are selected from long documents to create a small summary of the entire document. Models with fuzzified outputs as mentioned in Section 3.1.2 has been used for Extractive Text Summarization [Shirwandkar and Kulkarni 2018]. In this case, fuzzy logic is used along with a RBM-based model to enhance the summarization of textual input. Nguyen et al. [2018] proposes a fuzzy convolutional neural network for sentiment analysis. The model makes use of fuzzified features as input to the convolution layer, similar to the models with fuzzy inputs [Wang et al. 2016; Tabrizi et al. 2018; Zhang et al. 2014] as discussed in Section 3.1.1.

4.4 Healthcare

In healthcare, the data are often heterogeneous in nature. The deep learning architecture proposed in Park et al. [2016] as in Section 2.2 has been used for mortality prediction in ICUs [Davoodi and Moradi 2018]. The proposed model takes into account data from various sources such as patient's health records, the general profile of the patient, and his or her response to the treatment. The model in Section 2.2 has also been used for gestational diabetes data analytics [Moreira et al. 2018] and real-time tumor tracking [Qiao et al. 2017].

4.5 Image Processing

The model in Deng et al. [2017] has been used brain tissue image analysis [Al-Dmour and Al-Ani 2018]. It has also been used in surveillance scene representation [Ahmed et al. 2018]. Fuzzy Deep Learning proposed in Park et al. [2016] has also been used for license plate image segmentation [Rahmat et al. 2016]. Riaz et al. [2019] proposes a semi-supervised convolutional neural network that makes use of fuzzy techniques for denoising input images.

4.6 Time-series Prediction

Deep learning with fuzzy granulated inputs [Zhang et al. 2014] has been used in multiple time-series predictions. These include drought prediction [Agana and Homaifar 2017a, 2017b] using weather data. It has also been used for the prediction of algal blooms [Zhang et al. 2016]. Luo et al. [2019] proposes a six-layered model that uses interval type-2 intuitionistic fuzzy sets for regression problems and time-series prediction. This is similar to the model proposed in Section 2.4 [Rajurkar and Verma 2017].

5 DISCUSSION AND CONCLUSION

Deep learning is an area of active research, and the performance benefits that can be derived from integrating fuzzy systems can be exploited more. At present, deep learning suffers from two major drawbacks. First, training a deep neural network is quite a tedious task. The second issue is the interpretability of deep neural networks. In the case of training, even though calculus-based training methods are used at large even today, they do not guarantee optimality. Furthermore, understanding how the input is mapped to the output in a neural network is difficult. A few other drawbacks include sensitivity to noise in data, inability to work with heterogeneous, or incomplete data.

At present, the focus of most research works is on addressing these drawbacks by using fuzzy systems. It is seen that using fuzzy theory along with deep learning can improve the performance of the models where data are noisy, heterogeneous, incomplete, or vague. Fuzzy systems can be used as an integral part of deep learning models by using fuzzy parameters or by using fuzzy logic for selecting training parameters. Using fuzzy systems may increase the computational complexity, too. The availability of software platforms such as Nvidia CUDA, AMD ROCm, and Intel MKL further accelerates deep learning processes. However, the availability of such platforms for fuzzy logic systems is scarce. This increases the training times of systems that implement fuzzy logic models that make use of fuzzy weights or fuzzy parameters. Computation of the fuzzy parameters is time-consuming using the presently available architectures, even though the models provide resistance to noise and search over a wider space without getting stuck in the local optima.

Alternatively, fuzzy logic can be used alongside standard deep learning models. They can be used to process the input or output. Models can make use of fuzzified inputs coupled with standard deep learning models such as deep belief networks or convolutional neural networks. The output of the networks can also be fuzzified. This allows us to leverage the software platforms to accelerate the

training of deep neural networks using fuzzy systems. However, the fuzzification of data is often dependent on the nature of the data, and it becomes challenging to develop a generic model.

Therefore, improving the performance of fuzzy deep learning models can be further investigated in future. Since deep learning using fuzzy systems has shown resistance to noise, it can be used to defend against adversarial attacks.

REFERENCES

Norbert A. Agana and Abdollah Homaifar. 2017a. A deep learning based approach for long-term drought prediction. In *Proceedings of the IEEE Region 3 Technical, Professional, and Student Conference (SoutheastCon'17)*. IEEE, 1–8.

Norbert A. Agana and Abdollah Homaifar. 2017b. A hybrid deep belief network for long-term drought prediction. In *Proceedings of the Workshop on Mining Big Data in Climate and Environment (MBDCE'17) and the 17th SIAM International Conference on Data Mining (SDM'17)*. 27–29.

Sk Arif Ahmed, Debi Prosad Dogra, Samarjit Kar, and Partha Pratim Roy. 2018. Surveillance scene representation and trajectory abnormality detection using aggregation of multiple concepts. *Expert Syst. Appl.* 101 (2018), 43–55.

Hayat Al-Dmour and Ahmed Al-Ani. 2018. A clustering fusion technique for MR brain tissue segmentation. *Neurocomputing* 275 (2018), 546–559.

Krassimir T. Atanassov. 1986. Intuitionistic fuzzy sets.*Fuzzy Sets Syst.* 20, 1 (1986), 87–96.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2007. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*. 153–160.

Christopher M. Bishop. 2012. Pattern recognition and machine learning, 2006. *Springer* 1 (2012), 78–78.

James J. Buckley and Yoichi Hayashi. 1994. Fuzzy neural networks: A survey. *Fuzzy Sets Syst.* 66, 1 (1994), 1–13.

Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. 2018. A systematic study of the class imbalance problem in convolutional neural networks. *Neur. Netw.* 106 (2018), 249–259.

Jing Chao, Furao Shen, and Jinxi Zhao. 2011. Forecasting exchange rate with deep belief networks. In *Proceedings of the 2011 International Joint Conference on Neural Networks (IJCNN'11)*. IEEE, 1259–1266.

C. L. Philip Chen, Chun-Yang Zhang, Long Chen, and Min Gan. 2015. Fuzzy restricted Boltzmann machine for the enhancement of deep learning. *IEEE Trans. Fuzzy Syst.* 23, 6 (2015), 2163–2173.

Shengyong Chen and Zhongjie Wang. 2012. Acceleration strategies in generalized belief propagation. *IEEE Trans. Industr. Inf.* 8, 1 (2012), 41–48.

Weihong Chen, Jiyao An, Renfa Li, Li Fu, Guoqi Xie, Md Zakirul Alam Bhuiyan, and Keqin Li. 2018. A novel fuzzy deep-learning approach to traffic flow prediction with uncertain spatial–temporal data features. *Fut. Gener. Comput. Syst.* 89 (2018), 78–88.

Yie-Chien Chen and Ching-Cheng Teng. 1995. A model reference control structure using a fuzzy neural network. *Fuzzy Sets Syst.* 73, 3 (1995), 291–312.

KyungHyun Cho, Alexander Ilin, and Tapani Raiko. 2011. Improved learning of Gaussian-Bernoulli restricted Boltzmann machines. In *Proceedings of the International Conference on Artificial Neural Networks*. Springer, 10–17.

Heena A. Chopade and Meera Narvekar. 2017. Hybrid auto text summarization using deep neural network and fuzzy logic system. In *Proceedings of the International Conference on Inventive Computing and Informatics (ICICI'17)*. IEEE, 52–56.

Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. 2014. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*. 2933–2941.

Omid E. David and Iddo Greental. 2014. Genetic algorithms for evolving deep neural networks. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM, 1451–1452.

Raheleh Davoodi and Mohammad Hassan Moradi. 2018. Mortality prediction in intensive care units (ICUs) using a deep rule-based fuzzy classifier. *J. Biomed. Inf.* 79 (2018), 48–59.

Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. 2016. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Trans. Neur. Netw. Learn. Syst.* 28, 3 (2016), 653–664.

Yue Deng, Youyong Kong, Feng Bao, and Qionghai Dai. 2015. Sparse coding-inspired optimal trading system for HFT industry. *IEEE Trans. Industr. Inf.* 11, 2 (2015), 467–475.

Yue Deng, Yipeng Li, Yanjun Qian, Xiangyang Ji, and Qionghai Dai. 2014. Visual words assignment via information-theoretic manifold embedding. *IEEE Trans. Cybernet.* 44, 10 (2014), 1924–1937.

Yue Deng, Zhiquan Ren, Youyong Kong, Feng Bao, and Qionghai Dai. 2017. A hierarchical fused fuzzy deep neural network for data classification. *IEEE Trans. Fuzzy Syst.* 25, 4 (2017), 1006–1012.

Timon Chih-Ting Du and Philip M. Wolfe. 1995. The amalgamation of neural networks and fuzzy logic systemsâA survey. *Comput. Industr. Eng.* 29, 1–4 (1995), 193–197.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* 12, (Jul. 2011), 2121–2159.

Chaimae El Hatri and Jaouad Boumhidi. 2018. Fuzzy deep learning based urban traffic incident detection. *Cogn. Syst. Res.* 50 (2018), 206–213.

U. F. Eze, Igoh Emmanuel, and Etim Stephen. 2014. Fuzzy logic model for traffic congestion. *IOSR J. Mobile Comput. Appl.* 1, 1 (2014), 15–20.

Hongqing Fang and Chen Hu. 2014. Recognizing human activity in smart home using deep learning algorithm. In *Proceedings of the 33rd Chinese Control Conference*. IEEE, 4716–4720.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. 249–256.

M. M. Gupta and D. H. Rao. 1994. On the principles of fuzzy neural networks. *Fuzzy Sets Syst.* 61, 1 (1994), 1–18.

Guy Hacohen and Daphna Weinshall. 2019. On the power of curriculum learning in training deep networks. *CoRR* abs/1904.03626 (2019). arxiv:1904.03626 http://arxiv.org/abs/1904.03626.

Jibo He, William Choi, Yan Yang, Junshi Lu, Xiaohui Wu, and Kaiping Peng. 2017. Detection of driver drowsiness using wearable devices: A feasibility study of the proximity sensor. *Appl. Ergon.* 65 (2017), 473–480.

Yaroslav Hernandez-Potiomkin, Mohammad Saifuzzaman, Emmanuel Bert, Rafael Mena-Yedra, Tamara Djukic, and Jordi Casas. 2018. Unsupervised incident detection model in urban and freeway networks. In *Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC'18)*. IEEE, 1763–1769.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neur. Comput.* 18, 7 (2006), 1527–1554.

Geoffrey E. Hinton and Ruslan R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.

Qinghua Hu, Zongxia Xie, and Daren Yu. 2007. Hybrid attribute reduction based on a novel fuzzy-rough model and information granulation. *Pattern Recogn.* 40, 12 (2007), 3509–3521.

Weiming Hu, Wei Li, Xiaoqin Zhang, and Stephen Maybank. 2015. Single and multiple object tracking using a multi-feature joint sparse representation. *IEEE Trans. Pattern Anal. Mach. Intell.* 37, 4 (2015), 816–833.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

Hisao Ishibuchi, Kouichi Morioka, and I. B. Turksen. 1995. Learning by fuzzified neural networks. *Int. J. Approx. Reason.* 13, 4 (1995), 327–358.

Hisao Ishibuchi, Hideo Tanaka, and Hideiko Okada. 1993. Fuzzy neural networks with fuzzy weights and fuzzy biases. In *Proceedings of the IEEE International Conference on Neural Networks 1993*. IEEE, 1650–1655.

Vladimir G. Ivancevic and Tijana T. Ivancevic. 2007. *Neuro-fuzzy Associative Machinery for Comprehensive Brain and Cognition Modelling*. Vol. 45. Springer.

Gabriel Babatunde Iwasokun and Oluwole Charles Akinyokun. 2014. Fingerprint singular point detection based on modified poincare index method. *Int. J. Sign. Process. Image Process. Pattern Recogn.* 7, 5 (2014), 259–272.

Nuzhat Jan, Geoffrey D. Hugo, Nitai Mukhopadhyay, and Elisabeth Weiss. 2015. Respiratory motion variability of primary tumors and lymph nodes during radiotherapy of locally advanced non-small-cell lung cancers. *Radiat. Oncol.* 10, 1 (2015), 133.

Yaochu Jin and Lipo Wang. 2008. *Fuzzy Systems in Bioinformatics and Computational Biology*. Vol. 242. Springer.

Andrej Karpathy. 2017. CS231n Convolutional Neural Networks for Visual Recognition. Retrieved from http://cs231n.github.io/optimization-1/.

James M. Keller. 1996. Fuzzy logic rules in low and mid level computer vision tasks. In *Proceedings of the 1996 Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS'96)*. IEEE, 19–22.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

John Kipritidis, Geoffrey Hugo, Elisabeth Weiss, Jeffrey Williamson, and Paul J. Keall. 2015. Measuring interfraction and intrafraction lung function changes during radiation therapy using four-dimensional cone beam CT ventilation imaging. *Med. Phys.* 42, 3 (2015), 1255–1267.

Erich Peter Klement, Werner Schwyhla, and Robert Lowen. 1981. Fuzzy probability measures. *Fuzzy Sets Syst.* 5, 1 (1981), 21–30.

Takashi Kuremoto, Shinsuke Kimura, Kunikazu Kobayashi, and Masanao Obayashi. 2014. Time series forecasting using a deep belief network with restricted Boltzmann machines. *Neurocomputing* 137 (2014), 47–56.

Hon Keung Kwan and Yaling Cai. 1994. A fuzzy neural network and its application to pattern recognition. *IEEE Trans. Fuzzy Syst.* 2, 3 (1994), 185–193.

Sean Lander and Yi Shang. 2015. EvoAE–A new evolutionary method for training autoencoders for deep learning networks. In *Proceedings of the 2015 IEEE 39th Annual Computer Software and Applications Conference (COMPSAC'15)*, Vol. 2. IEEE, 790–795.

Nicolas Le Roux and Yoshua Bengio. 2008. Representational power of restricted Boltzmann machines and deep belief networks. *Neur. Comput.* 20, 6 (2008), 1631–1649.

Samuel C. Lee and Edward T. Lee. 1975. Fuzzy neural networks. *Math. Biosci.* 23, 1–2 (1975), 151–177.

Suk Jin Lee, Yuichi Motai, and Martin Murphy. 2012a. Respiratory motion estimation with hybrid implementation of extended Kalman filter. *IEEE Trans. Industr. Electron.* 59, 11 (2012), 4421–4432.

Suk Jin Lee, Yuichi Motai, Elisabeth Weiss, and Shumei S. Sun. 2012b. Irregular breathing classification from multiple patient datasets using neural networks. *IEEE Trans. Inf. Technol. Biomed.* 16, 6 (2012), 1253–1264.

Suk Jin Lee, Yuichi Motai, Elisabeth Weiss, and Shumei S. Sun. 2013. Customized prediction of respiratory motion with clustering from multiple patient interaction. *ACM Trans. Intell. Syst. Technol.* 4, 4 (2013), 69.

Chin-Teng Lin, Chang-Mao Yeh, Sheng-Fu Liang, Jen-Feng Chung, and Nimit Kumar. 2006. Support-vector-based fuzzy neural network for pattern classification. *IEEE Trans. Fuzzy Syst.* 14, 1 (2006), 31–41.

Faa-Jeng Lin, Chih-Hong Lin, and Po-Hung Shen. 2001. Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive. *IEEE Trans. Fuzzy Syst.* 9, 5 (2001), 751–759.

Hsien-I Lin, Ming-Hsiang Hsu, and Wei-Kai Chen. 2014. Human hand gesture recognition using a convolution neural network. In *Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE'14)*. IEEE, 1038–1043.

Wei Lu, Michelle M. Nystrom, Parag J. Parikh, David R. Fooshee, James P. Hubenschmidt, Jeffrey D. Bradley, and Daniel A. Low. 2006. A semi-automatic method for peak and valley detection in free-breathing respiratory waveforms. *Med. Phys.* 33, 10 (2006), 3634–3636.

Chao Luo, Chenhao Tan, Xingyuan Wang, and Yuanjie Zheng. 2019. An evolving recurrent interval type-2 intuitionistic fuzzy neural network for online learning and time series prediction. *Appl. Soft Comput.* 78 (2019), 150–163.

Mukesh Kumar Mehlawat and Pankaj Gupta. 2014. Fuzzy chance-constrained multiobjective portfolio selection model. *IEEE Trans. Fuzzy Syst.* 22, 3 (2014), 653–671.

Thomas M. Mitchell. 1997. *Machine Learning* (1 ed.). McGraw–Hill, Inc., New York, NY.

Sushmita Mitra and Yoichi Hayashi. 2000. Neuro-fuzzy rule generation: Survey in soft computing framework. *IEEE Trans. Neur. Netw.* 11, 3 (2000), 748–768.

Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton. 2009. Deep belief networks for phone recognition. In *Proceedings of the NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, Vol. 1. Vancouver, Canada, 39.

Mário W. L. Moreira, Joel J. P. C. Rodrigues, Neeraj Kumar, Jalal Al-Muhtadi, and Valeriy Korotaev. 2018. Evolutionary radial basis function network for gestational diabetes data analytics. *J. Comput. Sci.* 27 (2018), 410–417.

Martin J. Murphy and Damodar Pokhrel. 2009. Optimization of an adaptive neural network to predict breathing. *Med. Phys.* 36, 1 (2009), 40–47.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*. 807–814.

Mounir Ben Nasr and Mohamed Chtourou. 2006. A hybrid training algorithm for feedforward neural networks. *Neur. Process. Lett.* 24, 2 (2006), 107–117.

Arvind Neelakantan, Luke Vilnis, Quoc V. Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. 2015. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807* (2015).

Yurii Nesterov. 1983. A method for unconstrained convex minimization problem with the rate of convergence O (1/kˆ2). In *Doklady AN USSR*, Vol. 269. 543–547.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. 2011. Multimodal deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*. 689–696.

Tuan-Linh Nguyen, Swathi Kavuri, and Minho Lee. 2018. A fuzzy convolutional neural network for text sentiment analysis. *J. Intell. Fuzzy Syst.* 35, 6 (2018), 6025–6034.

Laisen Nie, Dingde Jiang, Shui Yu, and Houbing Song. 2017. Network traffic prediction based on deep belief network in wireless mesh backbone networks. In *Proceedings of the 2017 IEEE Wireless Communications and Networking Conference (WCNC'17)*. IEEE, 1–5.

Guangyuan Pan, Liping Fu, and Lalita Thakali. 2017. Development of a global road safety performance function using deep neural networks. *Int. J. Transport. Sci. Technol.* 6, 3 (2017), 159–173.

João Paulo Papa, Walter Scheirer, and David Daniel Cox. 2016. Fine-tuning deep belief networks using harmony search. *Appl. Soft Comput.* 46 (2016), 875–885.

Seonyeong Park, Suk Jin Lee, Elisabeth Weiss, and Yuichi Motai. 2016. Intra-and inter-fractional variation prediction of lung tumors using fuzzy deep learning. *IEEE J. Transl. Eng. Health Med.* 4 (2016), 1–12.

Lutz Prechelt. 1998. Early stopping-but when? In *Neural Networks: Tricks of the Trade*. Springer, 55–69.

Ning Qian. 1999. On the momentum term in gradient descent learning algorithms. *Neur. Netw.* 12, 1 (1999), 145–151.

Tian Qiao, Yixu Song, and Chao Ren. 2017. Real-time tumor tracking with respiratory motion based on short-term prediction. In *Proceedings of the 2017 2nd International Conference on Control, Automation and Artificial Intelligence (CAAI'17)*. Atlantis Press.

Basuki Rahmat, Endra Joelianto, I. Ketut Eddy Purnama, and Mauridhi Hery. 2016. Vehicle license plate image segmentation system using cellular neural network optimized by adaptive fuzzy and neuro-fuzzy algorithms. *Int. J. Multimedia Ubiq. Eng.* 11, 12 (2016), 383–400.

Shreedharkumar Rajurkar and Nishchal Kumar Verma. 2017. Developing deep fuzzy network with Takagi Sugeno fuzzy inference system. In *Proceedings of the 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'17)*. IEEE, 1–6.

Peijia Ren, Zeshui Xu, and Xunjie Gou. 2016. Pythagorean fuzzy TODIM approach to multi-criteria decision making. *Appl. Soft Comput.* 42 (2016), 246–259.

Saman Riaz, Ali Arshad, and Licheng Jiao. 2019. A semi-supervised CNN with fuzzy rough C-mean for image classification. *IEEE Access* 7 (2019), 49641–49652.

Herbert Robbins and Sutton Monro. 1985. A stochastic approximation method. In *Herbert Robbins Selected Papers*. Springer, 102–109.

Junhu Ruan, Xuping Wang, and Yan Shi. 2013. Developing fast predictors for large-scale time series using fuzzy granular support vector machines. *Appl. Soft Comput.* 13, 9 (2013), 3981–4000.

Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *CoRR* abs/1609.04747 (2016). arxiv:1609.04747 http://arxiv.org/abs/1609.04747.

David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group (Eds.). 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 2: Psychological and Biological Models*. MIT Press, Cambridge, MA, USA.

Danuta Rutkowska. 2012. *Neuro-fuzzy Architectures and Hybrid Learning*. Vol. 85. Physica.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Deep Boltzmann machines. In *Artificial Intelligence and Statistics*. PMLR, Florida, 448–455.

Ruslan Salakhutdinov and Hugo Larochelle. 2010. Efficient learning of deep Boltzmann machines. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. 693–700.

Takahiro Shinozaki and Shinji Watanabe. 2015. Structure discovery of deep neural network based on evolutionary algorithms. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'15)*. IEEE, 4979–4983.

Nikhil S. Shirwandkar and Samidha Kulkarni. 2018. Extractive text summarization using deep learning. In *Proceedings of the 2018 4th International Conference on Computing Communication Control and Automation (ICCUBEA'18)*. IEEE, 1–5.

Dan Simon. 2008. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* 12, 6 (2008), 702–713.

Paul Smolensky. 1986. *Information Processing in Dynamical Systems: Foundations of Harmony Theory*. Technical Report. Department of Computer Science, University of Colorado at Boulder.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 1 (2014), 1929–1958.

Nitish Srivastava and Ruslan R. Salakhutdinov. 2012. Multimodal learning with deep boltzmann machines. In *Advances in Neural Information Processing Systems*. 2222–2230.

Pooneh R. Tabrizi, Awais Mansoor, Juan J. Cerrolaza, James Jago, and Marius George Linguraru. 2018. Automatic kidney segmentation in 3D pediatric ultrasound images using deep neural networks and weighted fuzzy active shape model. In *Proceedings of the IEEE 15th International Symposium on Biomedical Imaging (ISBI'18)*. IEEE, 1170–1173.

Tomohiro Takagi and Michio Sugeno. 1985. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. Syst. Man. Cybernet.*1 (1985), 116–132.

Andrea Tettamanzi and Marco Tomassini. 2013. *Soft Computing: Integrating Evolutionary, Neural, and Fuzzy Systems*. Springer Science & Business Media.

Jin Tian, Minqiang Li, Fuzan Chen, and Nan Feng. 2016. Learning subspace-based RBFNN using coevolutionary algorithm for complex classification tasks. *IEEE Trans. Neur. Netw. Learn. Syst.* 27, 1 (2016), 47–61.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA* 4, 2 (2012), 26–31.

Jan C. Van Gemert, Jan-Mark Geusebroek, Cor J. Veenman, and Arnold W. M. Smeulders. 2008. Kernel codebooks for scene categorization. In *Proceedings of the European Conference on Computer Vision*. Springer, 696–709.

S. S. Vedam, P. J. Keall, A. Docef, D. A. Todor, V. R. Kini, and R. Mohan. 2004. Predicting respiratory motion for four-dimensional radiotherapy. *Med. Phys.* 31, 8 (2004), 2274–2283.

Jose Vieira, F. Morgado Dias, and Alexandre Mota. 2004. Neuro-fuzzy systems: A survey. In *Proceedings of the 5th WSEAS NNA International Conference on Neural Networks and Applications*. 87–92.

Navneet Walia, Harsukhpreet Singh, and Anurag Sharma. 2015. ANFIS: Adaptive neuro-fuzzy inference system-a survey. *Int. J. Comput. Appl.* 123, 13 (2015).

Li-Xin Wang and Jerry M. Mendel. 1991. *Generating Fuzzy Rules from Numerical Data, with Applications*. Signal and Image Processing Institute, University of Southern California.

Yani Wang, Zhendong Wu, and Jianwu Zhang. 2016. Damaged fingerprint classification by deep learning with fuzzy feature points. In *Proceedings of the International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI'16)*. IEEE, 280–285.

F. S. Wong, P. Z. Wang, T. H. Goh, and B. K. Quek. 1992. Fuzzy neural systems for stock selection. *Financ. Anal. J.* 48, 1 (1992), 47–52.

J. Y. Xia, X. Li, and Y. X. Liu. 2016. Application of a new restricted Boltzmann machine to radar target recognition. In *Proceedings of the 2016 Progress in Electromagnetic Research Symposium (PIERS'16)*. IEEE, 2195–2201.

Ronald R. Yager. 1999. Decision making with fuzzy probability assessments. *IEEE Trans. Fuzzy Syst.* 7, 4 (1999), 462–467.

Ronald R. Yager. 2013. Pythagorean fuzzy subsets. In *Proceedings of the IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS'13)*. IEEE, 57–61.

Ronald R. Yager. 2014. Pythagorean membership grades in multicriteria decision making. *IEEE Trans. Fuzzy Syst.* 22, 4 (2014), 958–965.

Chenggang Yan, Yongdong Zhang, Jizheng Xu, Feng Dai, Liang Li, Qionghai Dai, and Feng Wu. 2014. A highly parallel framework for HEVC coding unit partitioning tree decision on many-core processors. *IEEE Sign. Process. Lett.* 21, 5 (2014), 573–576.

JingTao Yao and Yiyu Y. Yao. 2003. Information granulation for web-based information support systems. In *Data Mining and Knowledge Discovery: Theory, Tools, and Technology V*, Vol. 5098. International Society for Optics and Photonics, 138–146.

Xin Yao. 1993. A review of evolutionary artificial neural networks. *Int. J. Intell. Syst.* 8, 4 (1993), 539–567.

Sue S. Yom, Zhongxing Liao, H. Helen Liu, Susan L. Tucker, Chao-Su Hu, Xiong Wei, Xuanming Wang, Shulian Wang, Radhe Mohan, James D. Cox, et al. 2007. Initial evaluation of treatment-related pneumonitis in advanced-stage non–small-cell lung cancer patients treated with concurrent chemotherapy and intensity-modulated radiotherapy. *Int. J. Radiat. Oncol. Biol. Phys.* 68, 1 (2007), 94–102.

Lotfi A. Zadeh. 1996. On fuzzy algorithms. In *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lotfi A. Zadeh*. World Scientific, Singapore, 127–147.

Lotfi A. Zadeh. 1997. Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy Sets Syst.* 90, 2 (1997), 111–127.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012).

Feng Zhang, Yuanyuan Wang, Minjie Cao, Xiaoxiao Sun, Zhenhong Du, Renyi Liu, and Xinyue Ye. 2016. Deep-learning-based approach for prediction of algal blooms. *Sustainability* 8, 10 (2016), 1060.

Ren Zhang, Furao Shen, and Jinxi Zhao. 2014. A model with fuzzy granulation and deep belief networks for exchange rate forecasting. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'14)*. IEEE, 366–373.

Xiaolu Zhang and Zeshui Xu. 2014. Extension of TOPSIS to multiple criteria decision making with Pythagorean fuzzy sets. *Int. J. Intell. Syst.* 29, 12 (2014), 1061–1078.

Yu-Jun Zheng, Sheng-Yong Chen, Yu Xue, and Jin-Yun Xue. 2017a. A Pythagorean-type fuzzy deep denoising autoencoder for industrial accident early warning. *IEEE Trans. Fuzzy Syst.* 25, 6 (2017), 1561–1575.

Yu-Jun Zheng, Wei-Guo Sheng, Xing-Ming Sun, and Sheng-Yong Chen. 2017b. Airline passenger profiling based on fuzzy deep machine learning. *IEEE Trans. Neur. Netw. Learn. Syst.* 28, 12 (2017), 2911–2923.

Shusen Zhou, Qingcai Chen, and Xiaolong Wang. 2014. Fuzzy deep belief networks for semi-supervised sentiment classification. *Neurocomputing* 131 (2014), 312–322.

Received May 2019; revised September 2019; accepted October 2019