Short Paper

A Hierarchical Fused Fuzzy Deep Neural Network for Data Classification

Yue Deng, Zhiquan Ren, Youyong Kong, Feng Bao, and Qionghai Dai

***Abstract*—Deep learning (DL) is an emerging and powerful paradigm that allows large-scale task-driven feature learning from big data. However, typical DL is a fully deterministic model that sheds no light on data uncertainty reductions. In this paper, we show how to introduce the concepts of fuzzy learning into DL to overcome the shortcomings of fixed representation. The bulk of the proposed fuzzy system is a hierarchical deep neural network that derives information from both fuzzy and neural representations. Then, the knowledge learnt from these two respective views are fused altogether forming the final data representation to be classified. The effectiveness of the model is verified on three practical tasks of image categorization, high-frequency financial data prediction and brain MRI segmentation that all contain high level of uncertainties in the raw data. The fuzzy dDL paradigm greatly outperforms other nonfuzzy and shallow learning approaches on these tasks.**

***Index Terms*—Feature learning, fuzzy neural networks, pattern classification.**

I. Introduction

Learning effective representations from real-world data is a long-standing pursuit of the machine learning community. In the big-data era, tons of data are generating in all kinds of scientific and industrial fields around the world. However, with the data quantity increasing, another critical issue of data ambiguity inevitably emerged that the big data themselves may contain high amount of noises and unpredictable uncertainties. Such feature ambiguity issues impose great challenges on the data understanding and classification tasks.

To overcome the uncertainties among the raw data, fuzzy learning (FL) [1], [2] has been established for solving many practical problems, e.g., image processing [3], portfolio management [4], [5], and motor control [6]. Fuzzy systems automatically learn the fuzzy membership functions and consequently derive fuzzy rules from a large quantity of training data [7]. The fuzzy logic values are then linearly combined in a defuzzifier forming the final decision for some specific tasks. Compared with conventional deterministic representations, the fuzzy logic representations flexibly construct fuzzy rules to reduce the uncertainties in raw data.

Manuscript received September 6, 2015; revised January 22, 2016; accepted April 20, 2016. Date of publication June 1, 2016; date of current version August 2, 2017. This work was supported by Project of the National Natural Science Foundation of China under Grant 61327902 and Grant 61120106003. The work of Y. Kong was supported under Grant BK20150650 NSF of Jiangsu Province, China. Y. Deng and Z. Ren contribute to this work equally.

Z. Ren, F. Bao, and Q. Dai are with the Automation Department, Tsinghua University, Beijing 100084, China (e-mail: renzhiquan1989@gmail.com; fbao0110@gmail.com; qhdai@tsinghua.edu.cn).

Y. Deng is with the Automation Department, Tsinghua University, Beijing 100084, China, and also with the University of California, San Francisco Medical Center, San Francisco, CA 94158 USA (e-mail: yuedeng.thu).

Y. Kong is with the School of Computer Science, Southeast University, Nanjing 210018, China (e-mail: kongyouyong@seu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2016.2574915

In the machine learning field, there are also significant progresses on the developments of neural networks [8]. Neural learning exploits the brain cognitive mechanism to hierarchically process information from layer to layer. In practice, the advantages of them are mainly summarized as two points [9]: 1) noise reduction and 2) task-driven feature learning. The noise reduction property is apparent that, with the information transformed from layer to layer, the contaminations on the raw data are sequentially decreased and removed. Meanwhile, the task-driven learning allows the knowledge to be sequentially propagated from the learning layer (e.g., the classification layer) to the data representation layers that provides a more intelligent way to automatically discover informative features from data.

In a nutshell, fuzzy and neural systems are both effective methods for data representation. There has been some early attempts in the filed that successfully combines neural learning and FL concepts. However, these approaches majorly fall into the sequential learning paradigm. For instance, the algorithm in [10] first transforms the original data into a latent space using deep learning (DL) and, then, fuzzifies the deep representation at the output layer for pattern classification. Alternatively, in the works [11], [12], fuzzy logics of the original input data are further transformed forming high-level summarization of the original fuzzy degrees. Accordingly, it is natural to ask: rather than "sequential" combinations, is there a joint learning framework that could more intelligently fuse these two respective learning views altogether?

To address this challenge, in this paper, we try to propose a fused fuzzy deep neural network (FDNN) that simultaneously extracts information from both fuzzy and neural representations (NR). The knowledge learnt from these two views are fused together in a fusion layer forming the final representation for data classification. In details, fuzzy representation reduces the uncertainties and NR removes the noises in the original data. The proposed FDNN further utilizes these two better representations forming the fused representation for final classification. Therefore, FDNN is potentially suitable to be applied to more difficult pattern classification tasks involving data ambiguity and noise. The performances of FDNN are verified on three practical but challenging data classification tasks of image categorization, brain magnetic resonance imaging (MRI) segmentation and high-frequency financial tick prediction. All these three tasks contain high level of uncertainties and noise. FDNN leads to much better performances than other state of the arts.

II. Fuzzy Deep Neural Network

A. *Model Configurations*

The FDNN is shown in Fig. 1, which is composed of four learning parts as summarized in the caption of the figure. In a nutshell, the input data (purple) follow two paths to respectively make the fuzzy logic representation (black part) and the NR (blue part). Consequently, the representations from these two views are combined together in the

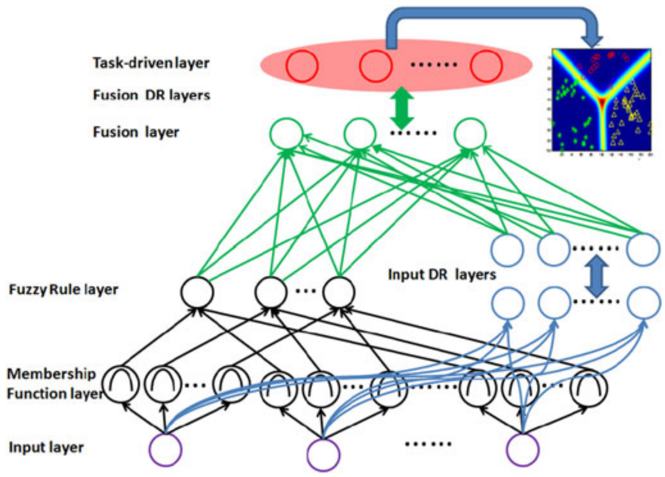

Fig. 1. Conceptual explanation of the fused FDNN. It is composed of four parts that respectively address different learning tasks, i.e., fuzzy logic representation part (black), NR part (blue), fuzzy-and-deep representation fusion part (green), and the task driven learning part (red).

fusion part (green part). In addition, the fused information of the first layer are further sequentially transformed forming the red layer at the end. The red layer is the task driven part connected with the classify to assign data points into different categories. We denote l as the layer number, $a_i^{(l)}$ is the input of the ith node and $o_i^{(l)}$ is the corresponding output. Please note, in the consequent discussions, we do not carefully distinguish different layers by using different layer symbols l . In each part, l is refereed as the currently discussed layer.

Part I—Fuzzy logic representation (black): Each node in the input layer is connected with multiple membership functions that assign linguist labels to each input variable. Here, the input variable is one dimension of the input vector. The fuzzy membership function calculates the degree that an input node belongs to a certain fuzzy set. On this layer, the ith fuzzy neuron $u_i(\cdot) : R \rightarrow [0, 1]$ maps the kth input as the fuzzy degree

$$o_i^{(l)} = u_i(a_k^{(l)}) = e^{-(a_k^{(l)} - \mu_i)^2/\sigma_i^2} \quad \forall i. \tag{1}$$

The Gaussian membership function with mean μ and variance σ^2 is utilized in our system following the suggestions of some pioneering works [6], [13]. The fuzzy rule layer performs the "AND" fuzzy logic operation, i.e., $o_i^{(l)} = \Pi_j o_j^{(l-1)} \quad \forall j \in \Omega_i$, where Ω_i defines the set of the nodes on the $(l-1)$th layer that connect to i. In this case, the $(l-1)$th layer is the input layer. The outputs of this part are the fuzzy degrees.

Part II—NR (blue): This part exploits the neural learning concept to transform the input as some high-level representations. The layers are fully connected implying each node on the (l)th layer is connected to all the nodes on the $(l-1)$th layer with parameters $\theta^{(l)} = \{\mathbf{w}^{(l)}, \mathbf{b}^{(l)}\}$, i.e.,

$$o_i^{(l)} = \frac{1}{1 + e^{-a_i^{(l)}}}, a_i^{(l)} = \mathbf{w}_i^{(l)} \mathbf{o}^{(l-1)} + b_i^{(l)} \tag{2}$$

where $\mathbf{w}_i^{(l)}$ and $b_i^{(l)}$ represent the weights and bias connecting to node i on the lth layer.

Part III—Fusion part (green): The fused concept adopted here is mainly inspired by recent successes of multimodal learning [8]. Multimodal learning believes that extracting features from a single view is not sufficient to capture the complex structure of high-content data. Accordingly, these approaches always generate multiple features from various aspects and synthesize them to a high-level representation for classification. In FDNN, we have exploited fuzzy and neural parts to respectively seek for better representations by reducing the uncertainty and noise of the input data. To better understand our model design, we suggest readers consider the output of the fuzzy part as the feature rather than its original fuzzy underlying. Moreover, the neural learning and FL parts are both configured in the neural network. Therefore, it is quite intuitive to implement the feature fusion step also in the neural network setting. In this paper, we prefer the widely used multimodel neural network structure [8] to combine neural and fuzzy representations with dense-connected fusion layers

$$\begin{aligned} o_i^{(l)} &= \frac{1}{1 + e^{-a_i^{(l)}}} \\ a_i^{(l)} &= (\mathbf{w}_d)_i^{(l)} (\mathbf{o}_d)^{(l-1)} + (\mathbf{w}_f)_i^{(l)} (\mathbf{o}_f)^{(l-1)} + b_i^{(l)}. \end{aligned} \tag{3}$$

In (3), the outputs from the deep representation part (denoted as $\mathbf{o}_d$) and fuzzy logic representation part (denoted as $\mathbf{o}_f$) are fused together with weights $\mathbf{w}_d$ and $\mathbf{w}_f$, respectively. Then, the fused information are deeply transformed by placing multiple all-connected layers after the fusion layer as in (2). The outputs combine fuzzy degrees and NRs altogether, which are no longer fuzzy degrees.

While there are alternative feature extraction approaches for deep fusion, we still prefer FL here due to the following three reasons. First, FL provides a convenient way to reduce the uncertainties among input data. Such important ambiguity reduction pursuit is the indispensable property of fuzzy system and can hardly be replaced by other learning systems. Second, the FL naturally leads to soft logistic values (fuzzy representations) in the range of $(0, 1)$. Moreover, each dimension in the neural outputs is also in the range of $(0, 1)$ [see (2)]. The quantities of fusion and neural outputs are in the similar range, making these two outputs easily fused in the fusion part. Third, the FL part allows task-driven parameter learning. Here, the exhausting hand-craft parameter tuning steps can be replaced by the intelligent data-driven learning through backpropagation.

Part IV—Task-driven part (red): The final part is the classification layer that assigns the fused representation into its corresponding category. In this paper, the soft-max function is applied to classify the data points into different classes. We denote $(\mathbf{f}_i, y_i)$ as the ith input and its corresponding label, $\pi_\Theta(\mathbf{f}_i)$ means the feed-forward transformation of the FDNN from the input layer to the last task-driven layer (red layer). Then, the following soft-max function can be used as the output layer with the cth entry calculated by

$$\hat{y}_{ic} = p(y_i|\mathbf{f}_i) = \frac{e^{\mathbf{w}_c \pi_\Theta(\mathbf{f}_i) + b_c}}{\sum_c e^{\mathbf{w}_c \pi_\Theta(\mathbf{f}_i) + b_c}} \tag{4}$$

where $\mathbf{w}_c$ and b_c represent the regression coefficient and bias of the cth class and $\hat{\mathbf{y}}_i = [\hat{y}_{i1}, \ldots \hat{y}_{ik}]$, which is the predicted labels of the neural network with k classes. Then, the mean-square-loss of the FDNN can be defined over m training samples

$$C = \frac{1}{m} \sum_i^m \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|_2^2. \tag{5}$$

B. *FDNN Training*

The training phase of FDNN contains two major steps of parameter initialization and fine tuning. The initialization steps are critical in DL because the whole learning system is not convex. Better initialization strategy may help the neural network converge to a good local minimum more efficiently [14]. In this paper, the initialization should be done for both the fuzzy and neural parts.

The weights between all the layers in the deep part, the fusion part and the classification part are initialized according to the suggestions in [15]. In details, the biases b for all the nodes are initialized as zero. Then, the weight between layers are randomly initialized according to the rule

$$w_i^{(l)} \sim U\left[-\frac{1}{\sqrt{n^{(l-1)}}}, \frac{1}{\sqrt{n^{(l-1)}}}\right] \tag{6}$$

where U is the uniform distribution and $n^{(l-1)}$ is the number of nodes on the $(l-1)$th layer. For the fusion layer, $n^{(l-1)}$ counts the total nodes number on the last layers of both the fuzzy and deep parts.

The rest of parameters requiring initialization are in the fuzzy part of FDNN. All the weights between the input "fuzzy membership layer" and the "fuzzy rule layer" are 1. Fuzzy membership nodes contain unknown fuzzy centers (μ_i) and widths (σ_i^2), where i means the ith fuzzy neuron. We follow the method in [16] to initialize the fuzzy membership nodes according to k-means results on the input data, where k is suggested the same as the class number to be classified. With the fuzzy, deep, and fusion parts well initialized, the fine tuning step is required to train the whole FDNN in a task-driven manner. The fine tuning step allows to precisely adjust the parameters in the neural network to enhance the discriminant ability of the final feature representation. To conduct this step, the famous back-propagation algorithm in (7) is used to calculate the gradient for all the parameters in the system

$$\frac{\partial C}{\partial \theta^{(l)}} = \sum_n \underbrace{\left(\frac{\partial C_n}{\partial o_i^{(l)}}\right)}_{BP} \frac{\partial o_i^{(l)}}{\partial a_i^{(l)}} \frac{\partial a_i^{(l)}}{\partial \theta^{(l)}} \tag{7}$$

where C is the loss function defined in (5). To note, θ represents the general parameters set in the FDNN. In the gradient, the first term in the bracket is the backpropagation term. The last two terms are layer-wise specified which should be derived according to the activation ($a_i^{(l)}$) and output ($o_i^{(l)}$) of the neuron. The neurons in the deep part, fusion part, and classification (task-driven) part just involve linear weights and biases, i.e., $\theta = \{b, \mathbf{w}\}$. Their gradients are not difficult to get according to (2), (3), and (4). The second type of neurons lie on the fuzzy layer which include the parameter set $\theta = (m_i, \sigma_i)$ to be adjusted. They are also easily calculated when coming to (1).

After getting the gradient for the parameter set, the stochastic gradient descenting is applied to implement the fine tuning step. The adaptation law for the parameters is given as following:

$$v(t) = \gamma v(t-1) + \alpha \frac{\partial C}{\partial \theta^{(l)}}, \theta^{(l)}(t+1) = \theta^{(l)}(t) - v(t). \tag{8}$$

In the above adaptation law, $v(t)$ is the velocity vector which is determined by the previous velocity and the current gradient. t records the iteration counts. $\gamma \in [0, 1]$ controls the information contributed by the previous gradient and $\alpha < 1$ is the learning rate. Such adaptation law is termed as momentum method for deep training [14]. In the learning phase of FDNN, we follow the suggestions in [14] to empirically set relatively small numbers for the gradient memorizing coefficient ($r = 0.1$) and learning rate ($\alpha = 0.05$). To the best of our knowledge, the theoretic proof about the convergence of deep neural network (DNN) is still an open problem in the filed. Many pioneering works could only empirically demonstrate the convergence of the neural network in experiments. In this study, we will also verify the convergence of FDNN from real data in Section III-A.

A challenge for training neural networks stems from the overfitting problem. Here, we suggest adopt the dropout strategy [17], [18] which is now becoming the benchmark method to alleviate overfitting in DNN training. In each training iteration, dropout strategy randomly selects $p\%$ of neurons and their gradients will not be updated in the current iteration. The procedures for training FDNN with dropout strategy has been summarized in Algorithm 1.

```
Algorithm 1: The training strategies for FDNN.
Input         : Training samples and their labels
                $(\mathbf{f}_i, \mathbf{y}_i), i = 1....m$, class number $k$ and
                input feature dimension $n$, training epoch
                number N.
Initialization: Initialize the parameters $\Theta_1$ in FDNN
                with two steps:
                Initialize $k \times n$ neurons on the fuzzy layer
  according to (1);
                Initialize weights of deep and fusion
  layers according to (6);
1 for $t = 1...N$ do
2     Randomly dropout $p\%$ of neurons in the FDNN and
      get FDNN$_{remain}$, the dropout neurons are labeled as
      FDNN$_{drop}$;
3     Feedforward all the training samples $\mathbf{f}_i$ through
      FDNN$_{remain}$ and get the fitting error $C$ by (5);
4     Propagate the fitting error $C$ back-through
      FDNN$_{remain}$ and apply the adaptation law in (8) to
      update the new parameter set $\hat{\theta}_{t+1}$, where $\hat{\theta}_{t+1} \in$
      FDNN$_{remain}$;
5     $\bar{\theta}_{t+1} = \bar{\theta}_t$, where $\bar{\theta}_t \in$ FDNN$_{drop}$;
6     Keep the parameters of the dropout neurons the same
      as the values in the last iteration:
      $\theta_{t+1} = \{\hat{\theta}_{t+1}, \bar{\theta}_{t+1}\}$;
7 end
Output        : The well trained FDNN with $\theta_{t+1}$;
```

TABLE I
SUMMARIZATION OF DIFFERENT FDNN CONFIGURATIONS

	Input	Fuzzy	Neural	Fusion	Output
C1	n	$k \times n$	64 (2)	64 (2)	k
C2	n	$k \times n$	128(2)	128(2)	k
C3	n	$k \times n$	256(3)	256(3)	k

III. Experimental Verifications

The experimental discussions are mainly divided into two parts. In the first part, the FDNN is tested on three benchmark datasets with the comparisons to other fuzzy and nonfuzzy classification methods. We will discuss three different configurations of FDNN and experimentally compare their performances in the consequent tests. Here, brief summarizations of different FDNN configurations have been provided in Table I. In each cell of the table, the figure shows the number of neurons on a layer and the figure in the brackets, if any, tells the number of layers of that part. In the table, n is the dimension of the input feature and k is the number of classes to be classified. The fuzzy membership function number is fixed the same as class number and it leads to a $k \times n$ neurons on the fuzzy layer. Apparently, the complexities of the FDNN increase from C1 to C3.

TABLE II
CLASSIFICATION ACCURACY OF THE DL METHODS WITH DIFFERENT CONFIGURATIONS

	C1			C2			C3		
	DNN	SFDNN	FDNN	DNN	SFDNN	FDNN	DNN	SFDNN	FDNN
Scene	70.7±0.6	70.3±0.8	71.3±0.8	71.9±1.0	71.4±0.7	72.9±0.7	72.6±1.1	72.1±0.9	73.2±0.8
HFT($\mathcal{H}$=5)	47.1±1.2	47.6±1.3	50.2±1.1	49.6±1.1	50.4±1.4	52.6±1.1	50.9±1.2	51.1±1.3	53.2±1.2
HFT($\mathcal{H}$=20)	41.3±1.5	40.8±1.6	42.4±1.4	42.8±1.7	41.9±1.2	43.7±1.5	42.5±1.7	43.4±1.5	44.2±1.6

A. *FDNN for Data Classification*

In this part, the performances of FDNN are verified on two classification tasks including natural scene image categorization [19] and stock trend prediction. The scene dataset used here contains more than 4500 natural images collected from 15 scene categories. In detail, all these 4500 images (from 15 classes) were mixed in a single dataset without overlapping. FDNN performs a 15-classes classification task on this dataset to tell the category of each scene image. We follow the kernel assignment algorithm [20] to generate a histogram for each image as feature for classification. In details, on each image, a grid-based method is used to extract the dense SIFT features on 16 $\times$ 16 pixel patches sampled every 8 pixels. Then, following the same experimental settings in [20], the local features are clustered into 200 codewords and the SIFT features are assigned to these codewords by the kernel assignment. When referring to Table I, in this task, we set $k = 15, n = 200$.

In the second application, we will apply the FDNN model to the task of financial signal prediction, which has been widely discussed in the machine learning field [21]. From the view of machine learning, this task can be well described as a three-classes classification problem. The goal is to predict, at time point t, whether the price of stock will be higher, lower or the same in the $(t + \mathcal{H})$th point where $\mathcal{H}$ is the forecasting interval. We use the high-frequency tick data of Shanghai Financial Index future (IF). The high-frequency ticks are updated twice a second accumulating more than 32 000 ticks per trading day. We follow the same implementation in [22] to extract multiple indicators from the price, volume and order book inducing to a long vector in $\mathbb{R}^{76}$. The feature extraction part is the same as in [22] and interested readers are referred there for details. Each feature dimension is normalized into the range of $[-1, 1]$. Obviously, for this task, we set the neural network $k = 3$ and $n = 76$ in Table I. Two predicting intervals will be considered, i.e., $\mathcal{H} = 5$ and $\mathcal{H} = 10$ that correspond to 5 and 10 ticks, respectively.

We will pit FDNN against other data classification and feature learning strategies. The FDNN is composed of two major parts as fuzzy part and deep part. Accordingly, we will first compare FDNN with other classic fuzzy neural networks. The support vector fuzzy neural network (SVFNN) [13] and self-constructing fuzzy neural network (SCFNN) [7] are chosen as the competitors because these two systems are the golden milestones in the field. In these two networks, the only parameter needs to be specified is the number of membership functions and it is fixed the same as our FDNN implementation. We also report the results of DNN [23] for comparisons. DNN only uses the blue part in Fig. 1 to deeply transform the input as the high-level representation, which is also implemented with three complexity levels from C1 to C3. In details, DNN is configured with the input, deep, and output parts. In addition, the results of sequential FDNN (SFDNN) [10] are also reported. Unlike the proposed FDNN that fuses fuzzy view and deep view altogether, SFDNN follows the sequential training strategy to add a fuzzy logical layer at the end of DNN. The fuzzy membership number in SFDNN is fixed as k and the deep part corresponds to the complexities from C1 to C3.

For training, in the 15 scene dataset, we follow the suggestion in [20] to randomly choose 100 images from each category as training samples, and the rest are for testing. Each random evaluation involves approximately 1500 training samples and 3000 testing samples. In the high frequency trading (HFT) problem, the model is trained and tested once a day. In details, for any trading day, a random set of 16 000 points from its previous five trading days are used for training. All the 32 000 points in the current trading day are used for testing. The model is retrained at the day level while not in the tick resolution. In practice, we retrain the prediction model after the closing time of each trading day and will apply the renewed model to all the ticks in the next trading day. All the trading days in July 2015 are involved in this experiment covering more than 60 000 ticks to be tested. For both the scene and HFT classification problems, the random training and testing processes are repeated for 20 times. The statistic results in the form of mean $\pm$ standard deviation are reported in Table II.

We also report the results produced by SVFNN, fuzzy-logic-based fuzzy neural network and support vector machine as the comparison baseline. These methods are shallow learning methods without deep configuration. They respectively get the classification accuracy of $69.4 \pm 0.9\%$, $69.1 \pm 1.1\%$, $70.4 \pm 0.8\%$ on scene dataset, $45.2 \pm 1.5\%$, $43.8 \pm 1.4\%$, $46.4 \pm 1.7\%$ on HFT ($\mathcal{H} = 5$), and $39.2 \pm 1.5\%$, $40.3 \pm 1.6\%$, $41.4 \pm 1.6\%$ on HFT($\mathcal{H} = 20$). From these tests, it is concluded that DL could improve the performances of shallow learning system. By analyzing the configurations of the neural network, we found that all the DL networks using C3 generally outperform C1 and C2. It is reasonable because C3 is the most complicated structure. We also noticed that C2 improves C1 significantly while C3 only improves C1 for a little bit. However, the training costs of a complicated neural network will be much heavier than a simple one. We adopt the early stopping strategy to train the FDNN for 40 epochs. The three configurations respectively take 7 min (C1), 19 min (C2), and 42 min (C3) to finish 40 training epochs on the stock prediction task that involve $16k$ training samples. All the algorithms are written in Python and implemented on a computer with quad-cores 2.8 GHZ CPU and 16G RAM. To note, the training process can be conducted in an offline manner. When the FDNN is well trained, it can be applied to predict the HFT ticks in real time.

To gain much deep insights on the convergence about FDNN, we have provided training details in Fig. 2, which is based on the neural network structure C2. It is drawn from training data of the first day in the HFT problem. It is interesting to note that with the increases of training epochs, the in-sample curve (training data) become flat while the out-of-sample curve (testing) suffers large oscillation at the end. This is where the overfitting phenomena occurs. Besides, the fitting errors majorly decrease in the first 40 iterations. Therefore, in this paper, only 40 training epochs are allowed. In addition, the flat trend of the in-sample curve at the tail empirically suggest the convergence of the FDNN. The whole FDNN is nonconvex and the converged point is only expected to be a local optimum.

In addition to the aforementioned general comparisons, it is also emphasized here that stock direction prediction is a widely studied topic

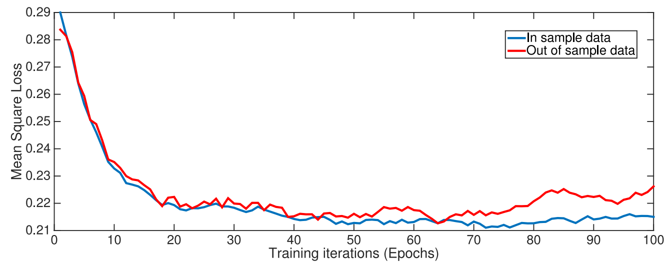

Fig. 2. Convergence analysis of the training.

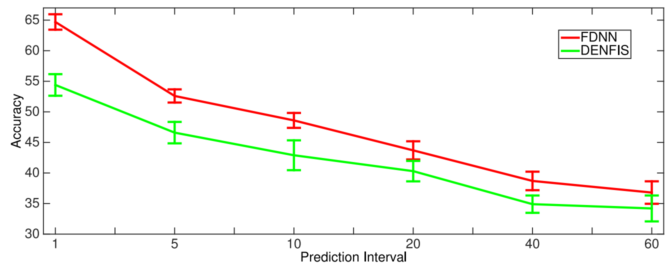

Fig. 3. Comparisons of FDNN and DENFIS.

in the FL field. Accordingly, in this study, we further compare FDNN with a famous fuzzy time series prediction system: DENFIS [16]. In the DENFIS, the fuzzy neuron number is set the same as FDNN. Then, a T–S type fuzzy inference system is established with the logistic regression as the prediction function. In the original DENFIS paper, a linear regression model is used to predict the price values in the future. However, knowing the exact price changing amount for high-frequency financial data is perhaps impractical because short-term price fluctuations are quite noisy. Accordingly, in this study, we just consider predicting the direction of the price movement instead of its exact value. Therefore, when implementing DENFIS, we replace the linear regression with a three-class logistic regression.

The dataset used here is the same as the HFT dataset in Table II. We implement FDNN and DENFIS with different predicting intervals. The experiments are repeated for 20 times with the mean and standard deviation reported in Fig. 3. By analyzing the results, we have observed that the classification accuracy gradually drops with the prediction interval increasing. However, FDNN consistently outperforms DENFIS for a large margin in all the tests. Although with a relatively large predicting interval, e.g., $\mathcal{H} >= 40$, the accuracies of FDNN are still better than random guessing (33%).

B. Brain MRI Component Segmentation

In this part, FDNN will be applied to an important clinical task [24], i.e., brain MRI segmentation, where the data contain high amount of uncertainties and noises. In general, MRI segmentation requires classifying brain tissues as white matter (WM), gray matter (GM), and cerebrospinal fluid (CSF) [24]. In this problem, the uncertainties in data mainly stem from two effects, i.e., partial volume effect and bias field effect [25]. In details, the partial volume effect is caused by the fact that different tissue types can occur in the same volume of the MRI. Due to such tissue heterogeneity, a certain region on the brain tissue can fuzzily associate with different tissue types. On the other hand, the bias effect is also a common challenge in MRI which is caused by the nonuniform magnetic field during the imaging process. The intensity value of the same tissue type can be quite different because of the bias effect. It brings a large amount of noises to the MRI image.

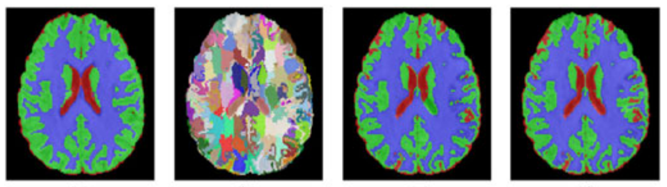

Fig. 4. Segmentation results on the brain MRI tissue of different methods. (a) Ground Truth, (b) Super-voxel, (c) DNN, (d) FDNN.

In conclusion, partial volume effect and bias filed effect, respectively, lead to the feature uncertainty and feature noises which can potentially be handled by the two views of FDNN.

The experiments are conducted on two widely used datasets from Internet Brain Segmentation Repository (IBSR) [26] and BrainWeb database [27]. These two datasets both contain 18 cases, i.e., 18 3-D images. For each 3-D image in ISBR, there are 256 slices and each slice is in the resolution of 256×128, leading to $256 \times 256 \times 128$ voxels in all. Each image in BrainWeb has 181 slices with the resolution of 217×181. In implementation, the homogenous local regions on the brain tissue are grouped together into a super-voxel according to the method in [28]. In this paper, we prefer to conduct segmentation on the super-voxel level due to the following two reasons. First, the huge number of voxel can be transferred into a small number of super-voxels, which can reduce the computational burden for classification. Second, compared with classifying voxels directly on MRI images, super-voxel aggregation can provide more wealthy information, including intensity, texture and shape features, to better discriminate each type of tissues in the classification [25]. After super-voxel aggregation, each brain MRI is composed of 3000 super-voxels and the classification is performed at the super-voxel level. An instance of super-voxels-level segmentation generated by the SLIC algorithm [28] has been provided in Fig. 4. The classification results by the proposed FDNN are also shown there. On each supervoxel, the intensity distribution histogram of 64 bins is generated to summarize the intensity information of the supervoxel. Therefore, in this task, we fill $k = 3$ and $n = 64$ in Table I.

For comparison purpose, in addition to those neural network-based approaches, the results of some other prevalent MRI segmentation methods are also reported. The comparisons include nonlocal fuzzy segmentation (NLFS) [29], Markov random fields (MRF) [30], and information theoretic segmentation (ITS) [25]. These three methods respectively exploits FL, probabilistic model, and information theory to reduce uncertainties and noises in brain tissues. In the training phase, we randomly select 200 supervoxels in each brain MRI for training and the rest 2800 supervoxels are for testing. Therefore, each dataset involves 3600 training samples and 50 400 testing samples. To evaluate the prediction accuracy, the dice similarity coefficient (DSC) [31] is exploited which can be calculated from true positive (TP), false positive (FP), and false negative (FN) rates as, DSC $= \frac{2\times\mathrm{TP}}{2\times\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$. To note, we adopt DSC rather than the classification accuracy here because DSC is the mostly used indicator to assay the performances of MRI segmentations. The random training and testing processes are repeated for 20 times and the average DSC with standard deviation are reported in Table III. In the implementations of different neural network related methods, we select the configuration C2 in Table I because it well trades off the complexity and effectiveness according to our tests in the last section.

From the results, it is apparent that FDNN produces the best results on three brain components in two datasets. It is also noted that DL approaches (DNN, SFDNN, and FDNN) generally outperform other shallow learning approaches. This finding is consistent with the previ-

TABLE III
PERFORMANCES OF DIFFERENT SEGMENTATION METHODS ON IBSR AND BRAINWEB DATASETS

	IBSR			BrainWeb		
Methods	CSF	GM	WM	CSF	GM	WM
	0.54±0.05	0.62±0.04	0.69±0.04	0.77±0.03	0.79±0.05	0.75±0.04
SCFNN	0.56±0.05	0.69±0.05	0.73±0.05	0.79±0.05	0.81±0.03	0.76±0.04
DNN	0.59±0.02	0.72±0.02	0.80±0.03	0.81±0.03	0.82±0.02	0.88±0.02
SFDNN	0.61±0.03	0.74±0.03	0.82±0.02	0.83±0.02	0.81±0.03	0.86±0.03
NLFS	0.53±0.02	0.65±0.04	0.76±0.03	0.78±0.05	0.80±0.04	0.83±0.03
MRF	0.52±0.03	0.68±0.03	0.79±0.04	0.79±0.04	0.77±0.04	0.78±0.04
ITS	0.59±0.01	0.73±0.01	0.78±0.02	0.81±0.02	0.82±0.02	0.87±0.02
FDNN	0.64±0.02	0.77± 0.02	0.84±0.01	0.86±0.03	0.87±0.02	0.91±0.01

TABLE IV
VERIFICATIONS BY ALTERNATIVE FUSION APPROACHES

	NR+IT	AE+Fuzzy	BM+Fuzzy	NR+Fuzzy
Scene	0.69	0.71	0.72	0.74
ISBR	0.71	0.75	0.75	0.77
BrainWeb	0.85	0.88	0.87	0.89

ous literatures that DL is much effective to extract informative features from raw data for pattern classifications. When performing the Student's t-test, we have got the conclusion that FDNN is significantly better than SFDNN with p value less than 0.01.

C. Alternative Fusion Approaches

The FDNN is composed of two parts: fuzzy representation and NR parts. Here, we consider other alternatives for these two parts and corresponding fused neural networks will be compared with FDNN. The fuzzy part plays the role of reducing the data ambiguity. Information theoretic (IT) learning is also a widely used approach to model the uncertainty in data. In implementation, we exploit a famous IT feature learning framework [32] to maximize the mutual information of latent representations and their labels. The IT representations will be used to replace the soft fuzzy representation in FDNN. The output dimension of the IT part is fixed the same as the fuzzy output dimensions in FDNN (see Table I). We also change the NR part and considers two other DNNs: autoencoder (AE) [33] and Boltzman machine (BM) [34]. Both AE and BM are stacked with five dense-connected hidden layers and each layer has 128 neurons. Both AE and BM are more deep than our NR part in FDNN (only has two hidden layers). The outputs of AE and BM will be fused with fuzzy outputs in FDNN. The fused neural network will be verified on scene dataset (150 training images per category) and two brain MRI datasets (1500 training supervoxels per sample). The experimental results were summarized in Table IV.

From the table, we have observed that the best performance on three datasets was obtained by FDNN, i.e., NR+Fuzzy. When using IT learning to replace fuzzy settings, the accuracies on all three datasets are decreased. The reason why FL is a better choice than other approaches has been discussed in Section II-A. On the other hand, when replacing the NR part with AE or BM, the performances are still worse than FDNN (NR+Fuzzy). This is because FDNN directly backpropagates the label information (fitting errors) back to the NR part and adjust parameters in a task-driven manner. Therefore, FDNN leads to better performances even though with a relatively shallow NR part. We have also considered the Rectified Linear Unit (ReLu) to replace the sigmoid activation in FDNN. It leads to 0.73, 0.79, and 0.89 accuracies on the Scene, ISBR, and BrainWeb datasets, respectively. Compared with the sigmoid activation, we have not observed significant improvements by using the ReLu activation.

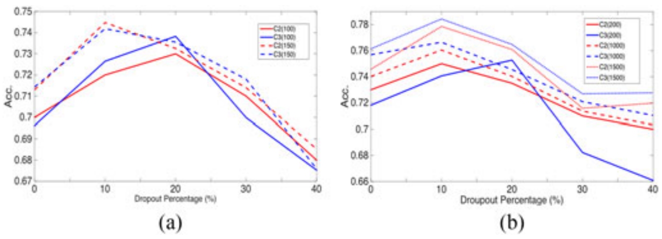

Fig. 5. Classification accuracy with different dropout ratio and training samples. In the legend, the number in the brackets represent how many training samples per image category (resp. per sample) were used to train the FDNN in the scene (resp. ISBR) datasets. (a) Scene, (b) IBSR.

D. Dropout Verifications

We demonstrate the effects of dropout in alleviating the overfitting phenomena. Here, the investigations were made on two previously discussed image datasets: 15 scene and ISBR (brain MRI). In the scene dataset, the training samples size was chosen at two levels: 100 images and 150 images per image category. In the ISBR dataset, the number of labeled supervoxels (training samples) was selected at three levels: 200, 1000, 1500 supervoxels per sample. These different training samples were used to train the FDNN at two complexity configurations: C2 and C3 (see Table I). The rest of data in these two datasets were used for testing.

We varied the dropout percentage ($p\%$ in Algorithm 1) of neurons in FDNN from 0% to 40%. The corresponding classification accuracy on two datasets were reported in Fig 5. It is observed that dropout results are much higher than training the original FDNN ($p\% = 0$). However, when too many neurons were dropped (e.g., $p\% > 20\%$), the testing accuracies also decrease. The FDNN always achieves the best accuracy when dropping out 10% neurons in each training iteration. The results also demonstrate the advantages of using more supervised samples to train FDNN. Increasing the training sample size could generally increase the classification rate on both datasets. In conclusion, either increasing the training sample size or using dropout strategy could potentially improve the FDNN training performances.

IV. CONCLUSION

This paper introduces an FDNN that hierarchically fuses the neural and fuzzy logic representations altogether for robust data classification.

In the fuzzy view, the data ambiguity is reduced by placing multiple fuzzy rules. In addition, the deep view reduces the noises in the input yielding much clean data representations to "meet" with the fuzzy logic representation. When setting the machine as a classifier, the FDNN can generate more reasonable features that achieve much better classification accuracies on the brain MRI and high-frequency financial data. The comparisons with other nonFDNNs reveal that FL is indeed a plausible way to further enhance the performances of DL. Although the focus of this paper is about data classification, the proposed FDNN is flexibly connected with other learning machines in the task-driven layer, e.g., regression, for more general feature learning applications.

REFERENCES

[1] C.-T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Trans. Comput.*, vol. 40, no. 12, pp. 1320–1336, Dec. 1991.

[2] J. J. Buckley and Y. Hayashi, "Fuzzy neural networks: A survey," *Fuzzy Sets Syst.*, vol. 66, no. 1, pp. 1–13, 1994.

[3] H. K. Kwan and Y. Cai, "A fuzzy neural network and its application to pattern recognition," *IEEE Trans. Fuzzy Syst.*, vol. 2, no. 3, pp. 185–193, Aug. 1994.

[4] F. Wong, P. Wang, T. Goh, and B. Quek, "Fuzzy neural systems for stock selection," *Financ. Anal. J.*, vol. 48, no. 1, pp. 47–52, 1992.

[5] M. Mehlawat and P. Gupta, "Fuzzy chance-constrained multiobjective portfolio selection model," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 3, pp. 653–671, Jun. 2014.

[6] F.-J. Lin, C.-H. Lin, and P.-H. Shen, "Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 5, pp. 751–759, Oct. 2001.

[7] C.-F. Juang and C.-T. Lin, "An online self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 1, pp. 12–32, Feb. 1998.

[8] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 689–696.

[9] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[10] S. Zhou, Q. Chen, and X. Wang, "Fuzzy deep belief networks for semi-supervised sentiment classification," *Neurocomputing*, vol. 131, pp. 312–322, 2014.

[11] R. Zhang, F. Shen, and J. Zhao, "A model with fuzzy granulation and deep belief networks for exchange rate forecasting," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2014, pp. 366–373.

[12] G. Padmapriya and K. Duraiswamy, "Association of deep learning algorithm with fuzzy logic for multidocument text summarization," *J. Theor. Appl. Inf. Technol.*, vol. 62, no. 1, 2014, pp. 166–173.

[13] C.-T. Lin, C.-M. Yeh, S.-F. Liang, J.-F. Chung, and N. Kumar, "Support-vector-based fuzzy neural network for pattern classification," *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 1, pp. 31–41, Feb. 2006.

[14] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 1139–1147.

[15] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. conf. Artif. Intell. Stat.*, 2010, pp. 249–256.

[16] N. Kasabov and Q. Song, "Denfis: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 144–154, Apr. 2002.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv preprint arXiv:1207.0580, 2012.

[19] Y. Deng, Y. Li, Y. Qian, X. Ji, and Q. Dai, "Visual words assignment via information-theoretic manifold embedding," *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1924–1937, Oct. 2014.

[20] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. Smeulders, "Kernel codebooks for scene categorization," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 696–709.

[21] Y. Deng, Q. Dai, R. Liu, Z. Zhang, and S. Hu, "Low-rank structure learning via nonconvex heuristic recovery," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 3, pp. 383–396, Mar. 2013.

[22] Y. Deng, Y. Kong, F. Bao, and Q. Dai, "Sparse coding-inspired optimal trading system for HFT industry," *IEEE Trans. Ind. Inf.*, vol. 11, no. 2, pp. 467–475, Apr. 2015.

[23] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010.

[24] L. He and N. A. Parikh, "Automated detection of white matter signal abnormality using t2 relaxometry: Application to brain segmentation on term MRI in very preterm infants," *Neuroimage*, vol. 64, pp. 328–340, 2013.

[25] Y. Kong, Y. Deng, and Q. Dai, "Discriminative clustering and feature selection for brain MRI segmentation," *IEEE Signal Process. Lett.*, vol. 22, no. 5, pp. 573–577, May 2015.

[26] T. Rohlfing, "Image similarity and tissue overlaps as surrogates for image registration accuracy: Widely used but unreliable," *IEEE Trans. Med. Imaging*, vol. 31, no. 2, pp. 153–163, Feb. 2012.

[27] R.-S. Kwan, A. Evans, and G. Pike, "MRI simulation-based evaluation of image-processing and classification methods," *IEEE Trans. Med. Imag.*, vol. 18, no. 11, pp. 1085–1097, Nov. 1999.

[28] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

[29] B. Caldairou, N. Passat, P. A. Habas, C. Studholme, and F. Rousseau, "A non-local fuzzy segmentation method: Application to brain MRI," *Pattern Recognit.*, vol. 44, no. 9, pp. 1916–1927, 2011.

[30] Y. Zhang, M. Brady, and S. Smith, "Segmentation of brain mr images through a hidden Markov random field model and the expectation-maximization algorithm," *IEEE Trans. Med. Imaging*, vol. 20, no. 1, pp. 45–57, Jan. 2001.

[31] B. Dogdas, D. Shattuck, and R. Leahy, "Segmentation of skull and scalp in 3-d human MRI using mathematical morphology," *Human Brain Mapping*, vol. 26, no. 4, pp. 273–285, 2005.

[32] K. Torkkola, "Feature extraction by non parametric mutual information maximization," *J. Mach. Learn. Res.*, vol. 3, pp. 1415–1438, 2003.

[33] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2011, pp. 215–223.

[34] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.