Convolutional Fuzzy Neural Networks with Random Weights for Image Classification

Yifan Wang, Hisao Ishibuchi, *Fellow, IEEE*, Witold Pedrycz, *Life Fellow, IEEE*, Jihua Zhu, Xiangyong Cao, and Jun Wang, *Senior Member, IEEE*

***Abstract*—Deep fuzzy neural networks have established a fundamental connection between fuzzy systems and deep learning networks, serving as a crucial bridge between two research fields in computational intelligence. These hybrid networks have powerful learning capability stemming from deep neural networks while leveraging the advantages of fuzzy systems, such as robustness. Due to these benefits, deep fuzzy neural networks have recently been an emerging topic in computational intelligence. With the help of deep learning, fuzzy systems have achieved great performance on the classification task. Although fuzzy systems have been extensively investigated, they still struggle in terms of image classification. In this paper, we propose a convolutional fuzzy neural network that combines improved convolutional neural networks with a fuzzy-set-based fusion technique. Different from convolutional neural networks, filters are randomly generated in convolutional layers in our model. This operation not only leads to the fast learning of the model but also avoids some notorious problems of gradient descent procedures in conventional deep learning methods. Extensive experiments demonstrate that the proposed approach is competitive with state-of-the-art fuzzy models and deep learning models. Compared to classical deep models that require massive training data, the proposed approach works well on small datasets.**

***Index Terms*—Deep fuzzy neural networks, image classification, convolutional neural networks**

I. Introduction

DATA-DRIVEN fuzzy systems have been successfully applied to a wide range of industrial applications including automatic control, pattern recognition, and image processing [1]–[4]. With the help of machine learning, fuzzy systems have achieved outstanding performance on some benchmark learning tasks [5]. Recent developments in fuzzy models with deep learning technology have witnessed the great advantages of deep fuzzy models [6]. These deep fuzzy models revealed that the deep-stacked technology could extract features more accurately than classical fuzzy models with shallow structures, thus enhancing the classification performance on large datasets.

Although deep fuzzy models have become popular in a wide range of real-world applications, they shed no light on image classification. One major reason is that existing fuzzy models ignore the strong local correlation present in natural images, whereas pixels in the close neighborhood are highly spatially correlated [7]. On the other hand, convolutional neural networks (CNNs) are good at discovering local correlation structures in raw images and achieve promising results on large datasets [8], [9]. However, they always require massive image data for training a powerful network and they are easily fooled by adding noise to raw images. Since fuzzy systems provide a powerful fusion framework for data presentation [10], they can take advantage of different sources for final classification. This fusion strategy can improve the generalization ability of neural networks on small datasets and the robustness against noise [11]. Therefore, it is natural to ask: is there a learning model that could explore the potential benefits of combining CNNs with a fuzzy-set-based fusion strategy?

In this paper, we propose a convolutional fuzzy neural network (ConvFNN) that integrates an improved CNN framework into the fuzzy system. Different from the classical CNN training process that uses the backpropagation (BP) method to optimize the network, the proposed ConvFNN introduces randomly generated convolutional filters and uses a closed-form solution to compute the parameters of the networks. This operation leads to the fast learning of the fuzzy model and achieves excellent performance on small datasets. The training process of ConvFNN includes three steps. In the first step, the input images are transformed into several fuzzy images, each of which corresponds to one fuzzy rule. In the second step, discriminative features are extracted from each fuzzy image using cascaded convolution and pooling operations. In the third step, the output weights are learned by a novel criterion that integrates local and global learning. The entire optimization procedure does not include gradient-descent steps, thereby

This work was supported by National Natural Science Foundation of China (Grant No. 62272289, 62250710163, 62376115, 62272375), Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386). *(Corresponding Authors: Jun Wang, Hisao Ishibuchi.)*

Yifan Wang is with the School of Software Engineering, Xi'an Jiaotong University, Xi'an, 710049; Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (email: wangyifan666@stu.xjtu.edu.cn).

Hisao Ishibuchi is with Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (email: hisao@sustech.edu.cn).

Witold Pedrycz is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, T6R 2V4 AB, Canada, also with the Systems Research Institute, Polish Academy of Sciences, 21589 Warsaw, Poland, and also with the Faculty of Engineering and Natural Sciences, Department of Computer Engineering, Istinye University, 21589 Istanbul, Turkey (e-mail: wpedrycz@ualberta.ca).

Jihua Zhu is with the School of Software Engineering, Xi'an Jiaotong University, Xi'an, 710049 (email: zhujh@xjtu.edu.cn).

Xiangyong Cao is with School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China. Ministry of Education Key Lab for Intelligent Networks and Network Security, Xi'an 710049, China (email: caoxiangyong@mail.xjtu.edu.cn).

Jun Wang is with Shanghai Institute for Advanced Communication and Data Science, School of Communication and Information Engineering, Shanghai University, China (email: wangjun_shu@shu.edu.cn).

improving the training efficiency.

The main contributions and technological breakthroughs of this paper are summarized as follows:

(1) The proposed ConvFNN is a data-driven fuzzy model that automatically learns discriminant features from raw images by using convolutional and pooling operations. This is quite different from most existing fuzzy models that usually use manually extracted features. In this study, we extend fuzzy systems to raw image classification. This deep fuzzy neural network integrates fuzzy systems with deep neural networks, thereby significantly enhancing the performance of fuzzy systems in image recognition. This approach not only bridges two distinct research fields within computational intelligence but also addresses some inherent challenges.

(2) In contrast to classical deep models that always require massive data for training, the proposed ConvFNN uses random filters in the convolutional layers, which leads to the fast learning of the model. In addition, classical deep models use gradient descent to optimize the network, which suffers from some notorious problems including local minima, numerous gradient-descent searching steps, and vanishing/exploding descent with increased depth. In our model, we compute parameters through a closed-form solution thus avoiding these problems. Efficient learning with random convolutional weights allows the proposed ConvFNN to achieve good performance in scenarios where traditional deep learning networks may struggle, particularly in situations with small image datasets.

(3) Fusion is a powerful method to combine information from different sources. In this paper, the fuzzy system combines the image information from multiple neural networks and thus takes advantage of different subnetworks. The fuzzy classification is achieved by introducing a novel learning algorithm that integrates local and global learning into a unified learning criterion, which guarantees good performance on each subnetwork and the whole network. In addition to the global performance, the classification results predicted by subnetworks are also considered in the objective function. This fuzzy fusion technology provides a way for data augmentation thus enhancing the performance and making the proposed ConvFNN stable to noise.

In our experiments, the proposed approach significantly improves the testing accuracy in comparison with traditional deep learning networks (VGG16, Inception-v3, and ResNet) on JAFFE, ORL, Leaves, UMIST, and Yale datasets. The number of training image patterns is under 500 in these quite small datasets. Experimental results show the effectiveness of the proposed ConvFNN in handling small datasets, making a significant technological breakthrough. This breakthrough represents a substantial step forward in computational intelligence, opening up new possibilities for fuzzy neural networks on image recognition.

The rest of the paper is organized as follows. In Section II, we introduce the Takagi-Sugeno-Kang (TSK) fuzzy model, and the connection between the TSK fuzzy model and fuzzy neural networks (FNNs). A brief review of image classification based on deep learning is also presented. In section III, we introduce the proposed ConvFNN model in detail. Then, the classification results of the proposed ConvFNN are reported in Section IV. Finally, we conclude the paper in Section V.

II. Related Work

The proposed ConvFNN is a neural network that combines DNNs with fuzzy systems. In this section, we introduce the TSK fuzzy model and provide a brief literature review of CNNs.

A. Takagi-Sugeno-Kang Fuzzy Model

The fusion technology of the proposed ConvFNN is based on the TSK fuzzy model [12], which consists of a set of fuzzy rules. Give an input pattern $\mathbf{x}_n = (x_{n1}, x_{n2}, ..., x_{nD})$, the kth fuzzy rule of the TSK fuzzy model can be expressed as follows:

$$\begin{aligned}&\text{IF } x_1 \text{ is } A_1^k \text{ and } x_2 \text{ is } A_2^k \text{ and} \dots \text{and } x_D \text{ is } A_D^k,\\ &\text{THEN } f^k(\mathbf{x}) = p_0^k + p_1^k x_1 + \cdots + p_D^k x_D, k = 1, 2, \dots, K.\end{aligned} \tag{1}$$

where K is the number of fuzzy rules, A_d^k is a fuzzy subset for the kth rule on the dth input attribute x_d, and p_d^k is the consequent parameters for the kth rule on the dth input attribute. Here, each rule maps the input pattern $\mathbf{x}$ to a real number $f^k(\mathbf{x})$ as its output. The final output of the TSK fuzzy model is computed as

$$y = \sum_{k=1}^{K} \frac{\mu^k(\mathbf{x})}{\sum_{k'=1}^{K} \mu^{k'}(\mathbf{x})} f^k(\mathbf{x}) = \sum_{k=1}^{K} \tilde{\mu}^k(\mathbf{x}) f^k(\mathbf{x}), \tag{2}$$

where $\mu^k(\mathbf{x})$ and $\tilde{\mu}^k(\mathbf{x})$ are the firing strength and the normalized firing strength for the kth rule computed using Eqs. (3)-(4):

$$\mu^k(\mathbf{x}) = \prod_{d=1}^{D} \mu_{A_d^k}(x_d), \tag{3}$$

$$\tilde{\mu}^k(\mathbf{x}) = \mu^k(\mathbf{x}) / \sum_{k'=1}^{K} \mu^{k'}(\mathbf{x}). \tag{4}$$

In our work, the membership function is formulated as follows:

$$\mu_{A_d^k}(x_d) = \exp\left(\frac{-\left(x_d - \eta_d^k\right)^2}{2\delta_d^k}\right),\ d = 1, 2, \dots, D. \tag{5}$$

where η_d^k is the center of the kth cluster on the dth input feature and δ_d^k is the spread of the membership function. The values of η_d^k and δ_d^k can be obtained by a clustering algorithm or partition technique. In our work, we use Fuzzy C-Means (FCM) [13] to obtain η_d^k and u_n^k, which denotes the membership degree of the nth input pattern $\mathbf{x}_n = (x_{n1}, x_{n2}, ..., x_{nD})$ belonging to the kth cluster. The spread δ_d^k is computed as follows:

$$\delta_d^k = \theta \cdot \sum_{n=1}^{N} u_n^k \left(x_{nd} - \eta_d^k\right)^2 / \sum_{n=1}^{N} u_n^k, \tag{6}$$

where N is the number of input patterns, and θ is the scaling parameter that can be manually adjusted by researchers.

In this study, we only consider the case for fixed premise parameters. This implies that the centers and spreads of the

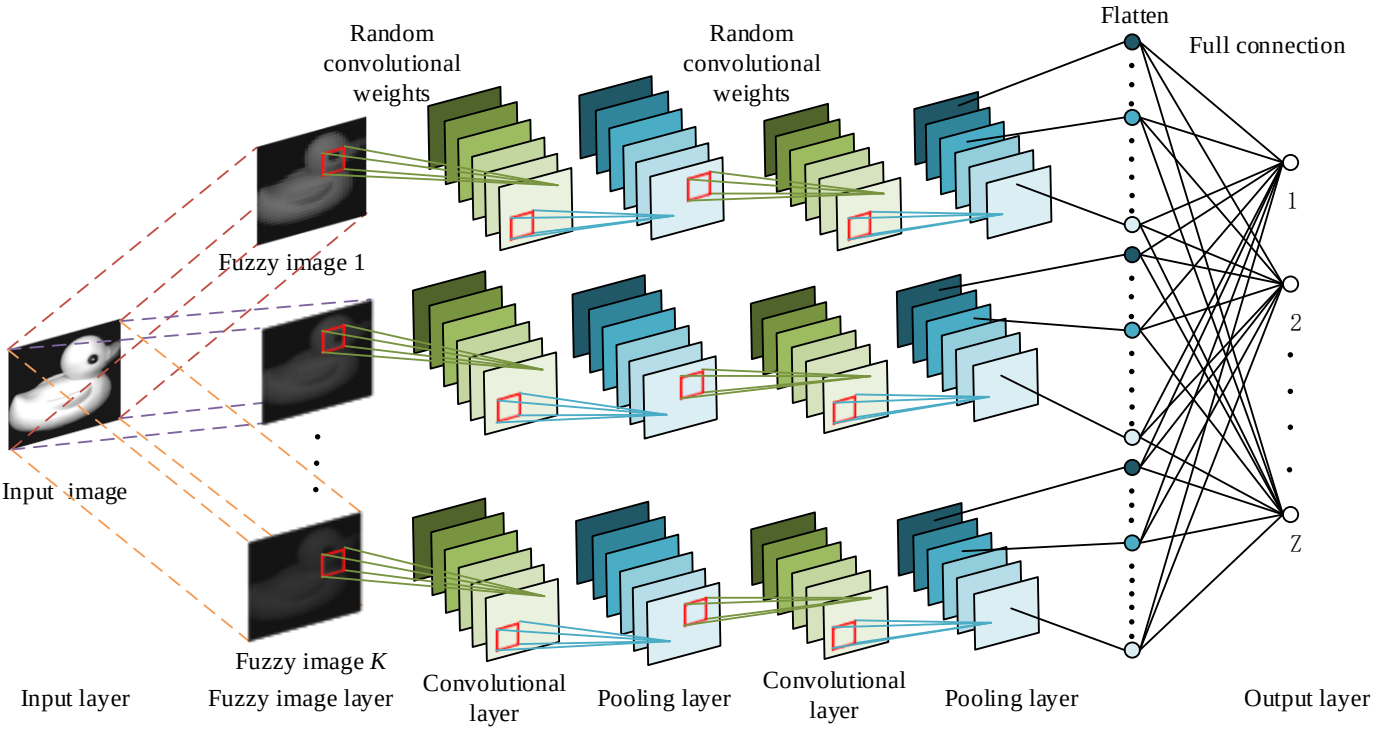


Fig. 1. The structure of ConvFNN.

Gaussian membership functions remain constant during the computation of consequent parameters. Under these conditions, the TSK model is formulated as a linear regression problem. The entire output of the TSK model is formulated as the following:

$$y = \mathbf{p}_g^T \tilde{\mathbf{x}}_g, \tag{7}$$

where $\mathbf{p}_g$ and $\tilde{\mathbf{x}}_g$ are computed as follows:

$$\mathbf{x}_e = \left(1, \mathbf{x}^T\right)^T, \tag{8}$$

$$\tilde{\mathbf{x}}^k = \tilde{\mu}^k(\mathbf{x})\mathbf{x}_e, \tag{9}$$

$$\tilde{\mathbf{x}}_g = \left(\left(\tilde{\mathbf{x}}^1\right)^T, \left(\tilde{\mathbf{x}}^2\right)^T, \ldots, \left(\tilde{\mathbf{x}}^K\right)^T\right)^T, \tag{10}$$

$$\mathbf{p}^k = \left(p_0^k, p_1^k, \ldots, p_D^k\right)^T, \tag{11}$$

$$\mathbf{p}_g = \left(\left(\mathbf{p}^1\right)^T, \left(\mathbf{p}^2\right)^T, \ldots, \left(\mathbf{p}^K\right)^T\right)^T. \tag{12}$$

To train the TSK fuzzy model, the original data are first mapped into the $K(D+1)$-dimensional feature space by using Eqs. (8)-(9). Then a linear regression problem is formulated to optimize $\mathbf{p}_g$. From the viewpoint of neural networks, the TSK fuzzy model can be considered as a three-layer fuzzy neural network with a special activation function in the hidden layer [14]. We develop a convolutional fuzzy neural network based on the fuzzy mapping in the TSK fuzzy model.

B. Convolutional Neural Networks

CNNs are characterized by three key attributes: local correlation, shared weights, and multiple layers. Different from traditional neural networks, CNNs can extract the features in images based on local correlations. CNNs can be trained using stochastic gradient descent by the backpropagation procedure method. Pioneering studies such as Neocognitron and LeNet-5 are designed for classifying handwritten digital images. With more layers, AlexNet, GoogLeNet, and VGG were proposed as winners of the ImageNet competitions. Based on a residual learning framework, ResNet has become one of the most popular deep neural networks. Recently, a deep multiview union learning network was proposed for classifying multisensor data [15]. To learn a sparse and effective feature representation, a contourlet CNN was proposed [16]. By combining multiscale geometric tools and networks, a flexible framework multiscale dynamic curvelet scattering network is proposed [17]. A deep group spatial–spectral attention fusion network is proposed for panchromatic and multispectral images, which can make up for their shortcomings in image interpretation [18].

III. The Proposed Convolutional Fuzzy Neural Networks

In this section, we explain the proposed ConvFNN in detail. The main structure of ConvFNN is given in Fig. 1 which includes an input layer, a fuzzy image layer, convolutional layers, pooling layers, and an output layer. The input layer receives images as the input data to the entire pipeline. Then, the fuzzy image layer generates several fuzzy images, each of which corresponds to one fuzzy rule. The convolutional and pooling layers subsequently perform convolution and pooling operations on the fuzzy images, respectively. Several convolutional and pooling layers can be stacked to extract effective features for classification. Finally, the output layer predicts the class label of the input pattern based on these high-level features.

A. Fuzzy Image Layer

In the antecedent part of the classical TSK fuzzy model, all attributes of the input data are used to compute the firing strength μ^k using Eq. (3). Since a large number of input attributes (i.e., the intensity of pixels) are involved for computing, Eq. (3) always includes many multiplication operations so that the firing strength tends to close to zero. To make the fuzzy model applicable to image classification, we only consider the pixels that contain useful information. Specifically, we first calculate the intensity variance $v(i,j)$ by using pixels located at (i,j) in all images. Then, we select D_{ref} reference pixels with the largest $v(i,j)$ values and further generate a subset of reference pixels for each image. Therefore, we obtain a D_{ref}-dimensional reference vector $\mathbf{r} = (r_1, r_2, ..., r_{D_{ref}})$, whose element r_d is the pixel intensity of the dth reference pixel in the subset. We use A_d^k to denote the fuzzy subset for the kth rule on r_d. Given an input image $\mathbf{X} \in \mathbb{R}^{I\times I}$, where $I \times I$ is the size of the input image, the firing strength of the kth rule is computed by using the following formulation:

$$\mu^k(\mathbf{X}) = \prod_{d=1}^{D_{\text{ref}}} \mu_{A_d^k}(r_d)\,. \tag{13}$$

We compute the membership degree $\mu_{A_d^k}(r_d)$ associated with the fuzzy set A_d^k by using the Gaussian function in Eq. (5). After that, we transform the original image $\mathbf{X}$ into K fuzzy images $\left\{\widetilde{\mathbf{X}}^k\right\}_{k=1}^{K}$ as follows:

$$\widetilde{\mathbf{X}}^k = \tilde{\mu}^k(\mathbf{X})\mathbf{X},\ k = 1, 2, \ldots, K, \tag{14}$$

where $\widetilde{\mathbf{X}}^k \in \mathbb{R}^{I\times I}$ is the kth fuzzy image, $I \times I$ is the size of the image, and $\tilde{\mu}^k(\mathbf{x})$ is the normalized firing strength of the kth rule computed using Eq. (4).

B. Stacked Convolutional and Pooling Layers

1) Convolution operation by using random weights:

Images always exhibit strong local correlations within sub-regions. That is, the pixels in a sub-region of the images are always spatially correlated. Fig. 2 shows a natural image and some associated tags. It is clear that the pixels in each rectangle are closely related and they contain high-level semantic knowledge. However, many existing fuzzy models fail to use such local correlations within images. To better utilize the local correlations of images, it is needed that the proposed fuzzy model should introduce local connections.

We compare the local and full connections in Fig. 3. Fig. 3 (a) shows a fully connected learning structure where the image pixels of the input layer are equally connected to the nodes in the next layer. However, the strong local correlations of images may make the full connections less appropriate. Different from Fig. 3 (a), Fig. 3 (b) shows the local connections between the input image pixels and the nodes in the next layer. The local connections are dense around the center of the local region and space around the boundary of the local region. This local learning structure perfectly matches the local correlation property of images. By integrating a local learning structure into our fuzzy model, we introduce the convolutional layer into the proposed fuzzy model.

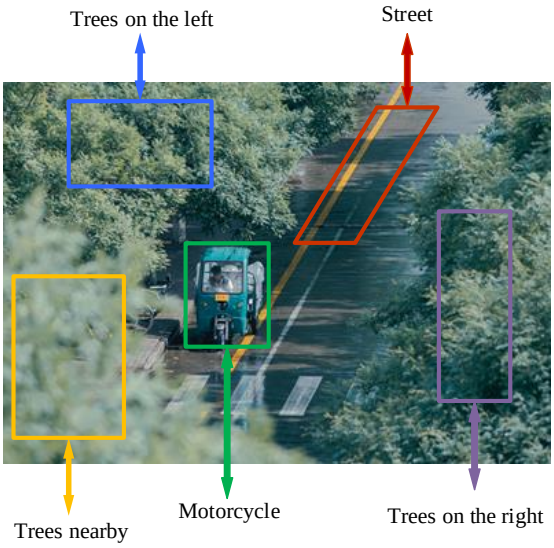


Fig. 2. A natural image and some associated tags.

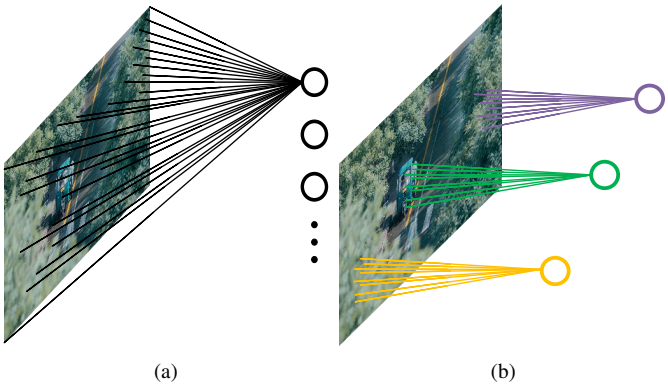


Fig. 3. Full connection and local connection.

Most CNNs use the BP algorithm for training [7], which involves numerous gradient-descent learning steps and suffers from serious problems, including slow convergence rate and local minima. In contrast to BP, some studies have shown the effectiveness of randomized learning [19]–[21]. For example, Jarrete et al. [22] found that the neural networks with random convolutional weights demonstrate comparable performance with pre-trained learning structures. Saxe et al. [23] proved that the convolutional pooling structure with random weights is translation invariant and frequency selective. Huang et al. [24] proposed the extreme learning machine for feedforward neural networks with random weights between the input and hidden layer. Motivated by these studies, we construct the proposed ConvFNN by introducing randomized learning. That is, we introduce several filters to perform the convolution operation on each fuzzy image. The weights of each filter are randomly generated. After that, we orthogonalize these weights by using singular value decomposition (SVD). The orthogonalization of the convolutional weights makes the weights spread more broadly [25], [26]. In the proposed ConvFNN, we use M different filters in each convolutional layer to ensure the diversity of extracted features.

Let the size of the fuzzy image be $I \times I$ and that of the

random filter be $F \times F$. The convolutional images in the sth convolutional layer are of the size $(I-s(F-1)) \times (I-s(F-1))$. $\mathbf{W}_m^k \in \mathbb{R}^{F \times F}$ is the weights of the filters under the kth fuzzy rule of the mth convolutional image. Let $c_{i,j,m}^k$ be the convolution node (i,j) in the mth convolutional image under the kth fuzzy rule. Then $c_{i,j,m}^k$ is calculated as follows:

$$c_{i,j,m}^k\left(\widetilde{\mathbf{X}}^k\right) = \sum_{l=1}^{F}\sum_{g=1}^{F} \widetilde{\mathbf{X}}_{i+l-1,j+g-1}^k \cdot w_{l,g,m}^k, \tag{15}$$

where $i,j = 1,2,...,I-s(F-1)$ and $\widetilde{\mathbf{X}}^k$ is the fuzzy image of the kth fuzzy rule in Eq. (14).

The convolution operation on the fuzzy image is a target-oriented self-learning procedure. This operation obtains convolutional images automatically without being elaborately designed by experienced engineers. This is different from the existing fuzzy models.

2) Square-root pooling operation:

To make the proposed ConvFNN robust to the noise and clutter [27], we use the square-root pooling operation after the convolution operation in ConvFNN. Let $h_{o,q,m}^k$ be the value of the pooling node (o,q) in the mth pooling image of the kth rule. Pooling can be performed by

$$h_{o,q,m}^k = \sqrt{\sum_{i=o-E}^{o+E}\sum_{j=q-E}^{q+E}\left(c_{i,j,m}^k\right)^2}, \tag{16}$$

where $o,q = 1,2,...,I-s(F-1)$, E is the distance between the center and the edge of the pooling area, and $c_{i,j,m}^k$ is the value of the node (i,j) in the mth convolutional image of the kth rule. If (i,j) is out of bound, $c_{i,j,m}^k = 0$.

Fig. 4 shows the details of the convolution and pooling operations on the fuzzy images. The pooling operation following the convolution operation has been proven to be frequency-selective and translational invariant, even with random weights. Furthermore, the square and summation operations in the pooling operation achieve the compactness of representation [23].

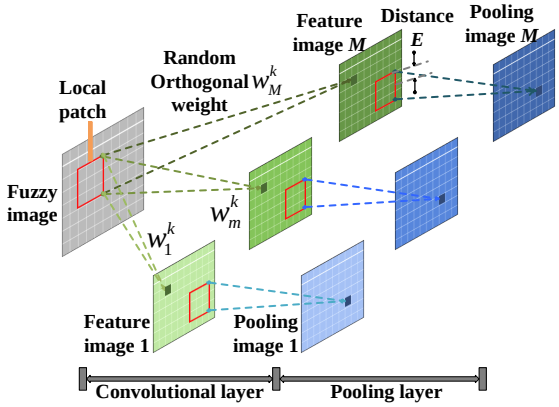


Fig. 4. Convolution and pooling operations on a fuzzy image.

C. Output Layer

The output layer is in full connection with the pooling layer. To compute the output weights, most existing fuzzy models minimize the squared errors between the outputs of the models and the true labels of the training patterns. Although these methods have satisfactory global performance on some applications, they do not guarantee good performance on local behavior [28]. In this study, we propose a novel learning algorithm that integrates local and global learning into a unified learning criterion. For each training pattern, we first compute each column of the pooling image in the last pooling layer and obtain a column vector $\mathbf{h}_n^k \in \mathbb{R}^B$, where B is the dimension of flattened pooling vector. Then we construct $\mathbf{H}^k = \left[\begin{array}{cccc}\mathbf{h}_1^k & \mathbf{h}_2^k & \dots & \mathbf{h}_N^k\end{array}\right]^T \in \mathbb{R}^{N \times B}$ for each rule and $\mathbf{H} = \left[\begin{array}{cccc}\mathbf{H}^1 & \mathbf{H}^2 & \dots & \mathbf{H}^K\end{array}\right] \in \mathbb{R}^{N \times KB}$. Here, $B = M \cdot (I-S(F-1))^2$ is the total number of nodes in all the pooling images of one fuzzy rule.

Let Z be the number of classes, y_i be the label of the ith training image and $\mathbf{y} = \left[\begin{array}{cccc} y_1 & y_2 & \dots & y_N \end{array}\right]^T \in \mathbb{R}^N$. For global learning, we define the learning criterion J_G in Eq. (17), the aim of which is to minimize the squared errors between the outputs of global learning in the proposed ConvFNN and the true labels of the training patterns:

$$J_G(\mathbf{p}) = (\mathbf{y} - \mathbf{H}\mathbf{p})^T(\mathbf{y} - \mathbf{H}\mathbf{p}), \tag{17}$$

where

$$\mathbf{p} = \left[\begin{array}{ccc}\left(\mathbf{p}^1\right)^T & \left(\mathbf{p}^2\right)^T & \dots \quad \left(\mathbf{p}^K\right)^T\end{array}\right]^T \in \mathbb{R}^{KB}. \tag{18}$$

We formulate the local learning criterion as follows:

$$J_L(\mathbf{p}) = \sum_{k=1}^{K}\left(\mathbf{y} - \mathbf{H}^k\mathbf{p}^k\right)^T \tilde{\boldsymbol{\mu}}^k \left(\mathbf{y} - \mathbf{H}^k\mathbf{p}^k\right), \tag{19}$$

where

$$\tilde{\boldsymbol{\mu}}^k = \left[\begin{array}{ccc}\tilde{\mu}_1^k & & 0 \\ & \tilde{\mu}_2^k & \\ & \ddots & \\ 0 & & \tilde{\mu}_N^k\end{array}\right] \in \mathbb{R}^{N \times N}, k = 1,2,\dots,K, \tag{20}$$

and

$$\mathbf{p}^k = \left[\begin{array}{cccc} p_1^k & p_2^k & \dots & p_B^k \end{array}\right]^T \in \mathbb{R}^B, k = 1,2,\dots,K. \tag{21}$$

The local model learns the linear approximation within each fuzzy rule. Of note, J_L in Eq. (19) can be rearranged into a complete matrix form:

$$J_L(\mathbf{p}) = (\tilde{\mathbf{y}} - \widetilde{\mathbf{H}}\mathbf{p})^T\widetilde{\mathbf{U}}(\tilde{\mathbf{y}} - \widetilde{\mathbf{H}}\mathbf{p}), \tag{22}$$

where

$$\tilde{\mathbf{y}} = \left[\begin{array}{cccc}\mathbf{y}^T & \mathbf{y}^T & \dots & \mathbf{y}^T\end{array}\right]^T \in \mathbb{R}^{NK}, \tag{23}$$

$$\widetilde{\mathbf{H}} = \left[\begin{array}{ccc}\mathbf{H}^1 & & \mathbf{0} \\ & \mathbf{H}^2 & \\ & \ddots & \\ \mathbf{0} & & \mathbf{H}^K\end{array}\right] \in \mathbb{R}^{NK \times KB}, \tag{24}$$

and

$$\widetilde{\mathbf{U}} = \left[\begin{array}{ccc}\tilde{\boldsymbol{\mu}}^1 & & \mathbf{0} \\ & \tilde{\boldsymbol{\mu}}^2 & \\ & \ddots & \\ \mathbf{0} & & \tilde{\boldsymbol{\mu}}^K\end{array}\right] \in \mathbb{R}^{NK \times NK}. \tag{25}$$

Algorithm 1: The training procedure of ConvFNN

Input: The training images $\{\mathbf{X}_n\}_{n=1}^{N}$, the corresponding class label set $\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \dots & y_N \end{bmatrix}^T$.
Output: The model of ConvFNN.

Step 1: Initialize the parameters in the structure of ConvFNN.
Step 2: Compute fuzzy images $\left\{\widetilde{\mathbf{X}}_n^k\right\}_{k=1}^{K}, n = 1, 2, \dots, N.$ in the fuzzy image layer.
Step 3: Extract features in stacked convolutional and pooling layers.
for k=*1 to* K **do**
  for m=*1 to* M **do**
    for s=*1 to* S **do**
      (3.1) Randomly generate the weights of convolutional filters and get the orthogonal matrix $\mathbf{W}_{m,n}^k \in \mathbb{R}^{F\times F}$ using SVD, where $k = 1, 2, \dots, K, m = 1, 2, \dots, M, n = 1, 2, \dots, N$.
      (3.2) Compute the convolutional node c using Eq. (15) and construct each convolutional image $\mathbf{C}_{m,n}^k \in \mathbb{R}^{(I-s(F-1))\times(I-s(F-1))}$, where $k = 1, 2, \dots, K, m = 1, 2, \dots, M, n = 1, 2, \dots, N$.
      (3.3) Compute the pooling node h using Eq. (16) and construct each pooling image $\mathbf{H}_{m,n}^k \in \mathbb{R}^{(I-s(F-1))\times(I-s(F-1))}$, where $k = 1, 2, \dots, K, m = 1, 2, \dots, M, n = 1, 2, \dots, N$.
    end
  end
end
Step 4: Compute the output weights $\mathbf{p}$ using Eq. (28).
Step 5: Return the model of ConvFNN.

We formulate the objective function for the learning of the output weights of ConvFNN as follows:

$$\begin{aligned} J(\mathbf{p}) &= \alpha J_G(\mathbf{p}) + \beta J_L(\mathbf{p}) + \mathbf{p}^T\mathbf{p} \\ &= \alpha(\mathbf{y}-\mathbf{H}\mathbf{p})^T(\mathbf{y}-\mathbf{H}\mathbf{p}) + \beta(\tilde{\mathbf{y}}-\widetilde{\mathbf{H}}\mathbf{p})^T\widetilde{\mathbf{U}}(\tilde{\mathbf{y}}-\widetilde{\mathbf{H}}\mathbf{p}) + \mathbf{p}^T\mathbf{p}. \end{aligned} \tag{26}$$

Notice that $\mathbf{p}^T\mathbf{p}$ is a regularization term that balances the variance to avoid overfitting. α and β are the parameters for global and local learning.

To minimize $J(\mathbf{p})$, we take derivatives with respect to Eq. (26) in terms of $\mathbf{p}$, set its derivatives to zero and obtain the following formulation:

$$\left(\alpha\mathbf{H}^T\mathbf{H} + \beta\widetilde{\mathbf{H}}^T\widetilde{\mathbf{U}}\widetilde{\mathbf{H}} + \mathbf{I}\right)\mathbf{p} = \alpha\mathbf{H}^T\mathbf{y} + \beta\widetilde{\mathbf{H}}^T\widetilde{\mathbf{U}}\tilde{\mathbf{y}}. \tag{27}$$

Finally, we obtain the optimal value of $\mathbf{p}$ that minimizes $J(\mathbf{p})$ by

$$\mathbf{p} = \left(\alpha\mathbf{H}^T\mathbf{H} + \beta\widetilde{\mathbf{H}}^T\widetilde{\mathbf{U}}\widetilde{\mathbf{H}} + \mathbf{I}\right)^{-1}\left(\alpha\mathbf{H}^T\mathbf{y} + \beta\widetilde{\mathbf{H}}^T\widetilde{\mathbf{U}}\tilde{\mathbf{y}}\right). \tag{28}$$

The whole training procedure of the proposed ConvFNN is summarized in Algorithm 1.

IV. Experimental Results and Analysis

We evaluate the proposed ConvFNN on both binary and multi-class classification problems. In our experiments, we downsize large images to 32×32 gray images without any other processing. The average performance of each classifier on each dataset is repeated ten times on the same training and testing sets. All algorithms are implemented with MATLAB 2021a and most of them are conducted on computers with Intel(R) Xeon(R) E5-2620v4 2.10GHz CPU and 128G memory. The experiments of ConvFNN on relatively large datasets (i.e., MNIST, SVHN, and ImageNet-based datasets) are conducted on a computer with 512G memory.

A. Binary Classification Experiments

We randomly select two classes from four datasets (i.e., MNIST, 17flowers, Caltech101, and Corel5K datasets) and construct several subsets for binary classification experiments. The MNIST is a typical handwritten digital database. All the images include the number ranging from 0 to 9. The 17flowers dataset has different kinds of common flowers in Oxford. The Caltech101 is an object image database from the real world, and each image is labeled with a single object. The Corel5K dataset is taken by Corel company and contains 5000 images that can be used for scene classification. Table I tabulates the details of the datasets in binary classification experiments.

TABLE I
Details of the Datasets for Binary Classification

Datasets	Classes	Number of training patterns	Number of testing patterns
MNISTA [29]	1/2	12700	2167
MNISTB [29]	3/8	11982	1984
17flowersA [30]	sunflowers/lily valley	118	42
17flowersB [30]	windflower/tiger lily	118	42
Caltech101A [31]	butterfly/yinyang	111	40
Caltech101B [31]	elephant/stegosaurus	90	33
Corel5KA [32]	coral/sunset	148	52
Corel5KB [32]	horse/car	148	52

For the proposed ConvFNN, all parameters are randomly initialized. Initial parameter values are shown in Table II. First, we adjust only the first parameter (i.e., the number of layers) by fixing all the other parameter values. The scope of the number of layers is $S \in \{2, 3, 4, 5, 6\}$ as shown in Table III. We can obtain five classification results from the five specifications about the number of layers. Then, we choose the value with the best performance on the testing data. In the same manner, we adjust all the other parameters one by one in the following order: Scaling parameter, size of convolutional filters, pooling

size, and the number of convolutional/pooling images. Next, we set the initial weight for global learning and the initial weight for local learning as the same value. We determine their values with the scope $\{0.005, 0.05, 0.5, 5, 50, 500, 5000\}$ by grid search and choose the value with the best testing accuracy. These parameters denote the trade-off between global/local learning and the regularization term $\mathbf{p}^T\mathbf{p}$ in Eq. (26). The coefficient of the regularization term is set to 1. As the coefficients of global and local learning vary from 0.005 to 5000, the balance between global/local learning and the regularization term changes accordingly. These changes show the effect of the regularization term with different coefficients on the performance of the classifier. After this operation, we adjust the trade-off weights between the weights for global and local learning. For example, if the value 0.5 achieves the best classification accuracy among $\{0.005, 0.05, 0.5, 5, 50, 500, 5000\}$, we will set the scope of weight for global learning as $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. Accordingly, the scope of weight for local learning is defined as $\{0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1\}$. In this case, we define the weight for global learning α and the weight for local learning β following the constraint:

$$\alpha + \beta = 1 \tag{29}$$

If the weight for global learning is 0.7, then the weight for local learning is 0.3. We choose the combination of weights for global and local learning which achieves the best accuracy. It is worth mentioning that the Eq. 29 will be changed if it is in different situations. If the value 0.05 achieves the best classification accuracy among $\{0.005, 0.05, 0.5, 5, 50, 500, 5000\}$, Eq. 29 will be changed to $\alpha + \beta = 0.1$. Thus, the scope of weight for global learning is $\{0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09\}$ and the scope of weight for local learning is $\{0.09, 0.08, 0.07, 0.06, 0.05, 0.04, 0.03, 0.02, 0.01\}$. In this study, we not only consider the balance between local learning and global learning but also consider the balance between local/global learning and the regularization term. Finally, we adjust the number of fuzzy rules.

TABLE II
INITIAL VALUES OF PARAMETERS

Order	Parameters	Values
1	Number of layers S	2
2	Scaling parameter θ	1000
3	Size of convolutional filters F	4
4	Pooling size E	3
5	Number of convolutional/pooling images M	20
6	Weight for global learning α	0.005
7	Weight for local learning β	0.005
8	Number of fuzzy rules K	3

For the other algorithms, we adjust the parameters of each model using the grid-searching strategy and choose the parameters with the best accuracy on testing sets. Table III shows the details of all the algorithms in binary classification and their parameter settings. ϵLSSLI, IQP and L2-TSK-FS are three fuzzy models based on ϵ-insensitive learning. Both ϵLSSLI and IQP use the L1-norm regularization terms, whereas L2-TSK-FS uses the L2-norm regularization terms. F-ELM is a fuzzy neural network developed based on extreme learning machine. FS-FCSVM is a SVM-based fuzzy model. LeNet-5, VGG16, Inception-v3, and ResNet18 are representative deep learning models. We use pre-trained deep models to extract features and use SVM to classify patterns. In binary classification, these nine fuzzy classification algorithms are used for comparison.

TABLE III
PARAMETER SETTINGS FOR BINARY CLASSIFICATION

Algorithms	Parameter Settings
ConvFNN (Proposed)	Number of fuzzy rules $K \in \{1, 2, 3\}$, scaling parameter $\theta \in \{10^{-1}, 10^0, 10^1, 10^2, 10^3, 10^4, 10^5\}$, number of layers $S \in \{2, 3, 4, 5, 6\}$, size of convolutional filters $F \in \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$, pooling size $E \in \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$, number of convolutional/pooling images $M \in \{10, 20, 30, 40, 50\}$, D_{ref} =5. We first set $\alpha = \beta$ and determine their values within the scope $\{0.005, 0.05, 0.5, 5, 50, 500, 5000\}$ by grid searching strategy. After that, we adjust trade-off weights between α and β .
ϵLSSLI [33]	Number of fuzzy rules $K \in \{1, 2, 3\}$,
IQP [33]	scaling parameter $\theta \in \{10^{-1}, 10^0, 10^1, 10^2, 10^3, 10^4, 10^5\}$,
L2-TSK-FS [34]	regularization parameter $\varphi \in \{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3\}$.
F-ELM [35]	Number of neural units in hidden layer $\{1000, 2000, 3000, 4000, 5000\}$, scaling parameter $\theta \in \{10^{-1}, 10^0, 10^1, 10^2, 10^3, 10^4, 10^5\}$,
FS-FCSVM [36]	Centers of Gaussian membership function $\{0, 0.25, 0.5, 0.75, 1.0\}$, proportion of unrelated features is 0.5. Number of fuzzy rules $K \in \{1, 2, 3\}$, scaling parameter $\theta \in \{10^{-1}, 10^0, 10^1, 10^2, 10^3, 10^4, 10^5\}$, regularization parameter $\varphi \in \{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3\}$.
LeNet-5 [29]	Learning rate: 1, and epoch: 5000.
VGG16+SVM [37] Inception-v3+SVM [38] ResNet18+SVM [39]	We use deep learning models to extract features, and use SVM to conduct classification.

To evaluate the performance of ConvFNN, we identify reference pixels for each image and then compute the firing strength for each rule by using the reference pixels. After that, we transform each image into several fuzzy images by using Eq. (14). For fair comparison, we perform the same operation in all the other algorithms which generate fuzzy rules.

Table IV shows the average testing accuracy of the proposed ConvFNN and nine fuzzy models for comparison. The best results on all datasets are shown in bold. One can observe that our ConvFNN is superior to the other algorithms for comparison on most datasets. The main reason is that the conventional fuzzy models are not effective on high-dimensional datasets. In contrast to those methods, our ConvFNN model learns the most significant features using stacked convolution and pooling operations based on the local correlations within images. In addition, both local and global learning are integrated into the learning criterion of ConvFNN, which also enhances the performance. Deep learning models are outstanding at extracting image features, whereas their performance tends to degrade on small datasets. This is because parameters in DNNs pre-trained on the ImageNet dataset may not be suitable for images used in this study. In addition to classification accuracy, we also present the time cost on the MNISTA and 17flowersA datasets in Table V. It is observed that traditional fuzzy algorithms execute extremely quickly on the 17flowersA dataset. ϵLSSLI is a good choice when the number of training image patterns is under 120. However, when considering both time cost and classification accuracy on the MNISTA dataset, the proposed ConvFNN proves to be an excellent option.

To further investigate how parameters affect the performance of ConvFNN, we analyze the performance sensitivity on four datasets (i.e., MNISTA, 17flowersB, Caltech101B, and Corel5KA datasets) with the six parameters (i.e., the number of fuzzy rules, the number of layers, the scaling parameter,

TABLE IV
AVERAGE TESTING ACCURACY ON BINARY CLASSIFICATION

Datasets	ConvFNN	ϵLSSLI	IQP	L2-TSK-FS	F-ELM	FS-FCSVM	LeNet-5	VGG16+SVM	Inception-v3+SVM	ResNet18+SVM
MNISTA	**99.94%**	98.94%	98.62%	91.16%	52.38%	99.49%	99.68%	99.22%	99.63%	99.49%
MNISTB	**99.95%**	96.62%	96.02%	94.06%	50.91%	97.93%	99.85%	96.57%	99.14%	97.08%
17flowersA	88.10%	**90.48%**	76.19%	88.10%	50.00%	85.71%	78.57%	71.43%	78.57%	88.10%
17flowersB	**83.33%**	71.43%	69.05%	63.57%	50.00%	73.81%	64.29%	64.29%	76.19%	57.14%
Caltech101A	**97.50%**	95.00%	95.00%	90.00%	60.00%	95.00%	95.00%	85.00%	95.00%	80.00%
Caltech101B	**87.88%**	72.73%	69.70%	72.73%	51.52%	72.73%	63.64%	57.58%	81.82%	51.52%
Corel5KA	**94.23%**	90.38%	78.85%	59.62%	50.00%	86.54%	73.08%	82.69%	86.54%	71.15%
Corel5KB	**92.31%**	75.00%	73.08%	68.65%	50.00%	76.92%	75.00%	55.77%	71.15%	53.85%

TABLE V
THE TIME COST OF ALL BINARY CLASSIFICATION ALGORITHMS ON MNISTA AND 17FLOWERSA DATASETS

Datasets	ConvFNN	ϵLSSLI	IQP	L2-TSK-FS	F-ELM	FS-FCSVM	LeNet-5	VGG16+SVM	Inception-v3+SVM	ResNet18+SVM
MNISTA	425.34s	87.19s	90.20s	2609.21s	243.29s	6.98s	72633.74s	7032.67s	6864.34s	1989.24s
17flowersA	163.17s	5.31s	7.49s	0.43s	3.59s	0.28s	6356.47s	59.84s	66.38s	20.18s

the filter size, the pooling size and the weight of trade-off). Experimental results are shown in Fig. 5. From Fig. 5 (a), one can observe that the accuracy increases with the increase in the number of fuzzy rules. When the number of fuzzy rules is set as 1, our model does not perform well on these four datasets. This observation indicates the necessity of using multiple fuzzy rules in ConvFNN. We also observe that the proposed ConvFNN can always achieve satisfactory results when the number of fuzzy rules is set as 3. Since too many clusters lead to large memory requirements in subsequent learning procedures, we cannot perform computational experiments with too many fuzzy rules for MNISTA and Caltech101B datasets. This is the reason why Fig. 5 (a) looks incomplete. Fig. 5 (b) shows that two layers are enough for all four datasets. This is because overfitting occurs when more layers are introduced. Fig. 5 (c) shows the performance of the proposed ConvFNN for various values of the scaling parameter. Good results are obtained for all datasets when the scaling parameter is in the range [1000, 100000]. Fig. 5 (d) shows how convolutional filter size affects the performance of the proposed ConvFNN. Good performance is achieved when the filter size is 8 on the Corel5KA dataset while the performance is not good when the filter size is 8 on Caltech101B and 17flowersB datasets. Fig. 5 (e) shows how pooling size affects the performance of the proposed ConvFNN. Fig. 5 (f) shows that the classification accuracy keeps the same on MNISTA, Caltech101B, and 17flowers datasets. ConvFNN achieves better classification results on the Corel5KA dataset when the value of $\alpha/(\alpha+\beta)$ is in the range [0.1, 0.5], which indicates that local learning is important in constructing fuzzy classifiers.

B. Multi-class Classification Experiments

In this subsection, we evaluate the proposed ConvFNN on other image datasets including the COIL20, COIL100, JAFFE, ORL, Leaves, MPEG7, CASIA-faceV5, UMIST, Yale, and USPS datasets for multi-class classification. The Columbia Object Image Library (COIL) dataset is used for object recognition. Every image of one object is taken at pose intervals of 5 degrees. The Japanese Female Facial Expression (JAFFE) dataset shows different facial expressions from 10 different Japanese females. The ORL dataset contains a set of face images taken by the Speech, Vision, and Robotics Group at Cambridge University. The Leaves dataset contains images of leaves against cluttered backgrounds. The MPEG7 dataset is used for evaluating shape similarity and classification. The CASIA-faceV5 Image Database Version 5.0 contains 2500 facial images which are captured using a Logitech USB camera in one session. The UMIST dataset contains face images of 20 individuals. The Yale face dataset contains grayscale images in GIF format of 15 individuals. The USPS is a handwritten recognition dataset that is automatically scanned from envelopes by the U.S. Postal Service. In addition, we use two relatively large datasets for evaluation (i.e., the number of training patterns is more than 60000). The Street View House Numbers (SVHN) dataset is a house number recognition dataset, which is collected from Google Street View. The MNIST dataset is used in binary classification experiments. The details of the above-mentioned datasets are summarized in Table VI.

TABLE VI
DETAILS OF THE DATASETS FOR MULTI-CLASS CLASSIFICATION

No.	Datasets	Number of training patterns	Number of testing patterns	Number of classes
1	COIL20 [40]	1060	380	20
2	COIL100 [40]	5354	1918	100
3	JAFFE [41]	152	61	10
4	ORL [42]	280	120	40
5	Leaves	137	49	3
6	MPEG7 [43]	980	420	70
7	CASIA-faceV5 [44]	2000	500	500
8	UMIST [45]	416	159	20
9	Yale [46]	105	60	15
10	USPS [47]	19999	1500	10
11	MNIST [29]	60000	10000	10
12	SVHN [48]	73257	26032	10

We compare the performance of ConvFNN with four classifiers in Table VII and five classifiers in Table III. ICSC is proposed by introducing incremental confidence samples into classification, which includes both adaptive adjustment and increment labeling. Including deep learning models used for comparison in binary classification, ResNet50 and ResNet101 are also used here. SVM classifier with the linear kernel is included in our experiments. In addition, we also used FS-FCSVM for comparison, which has achieved the second-best performance on most datasets for the binary classification task.

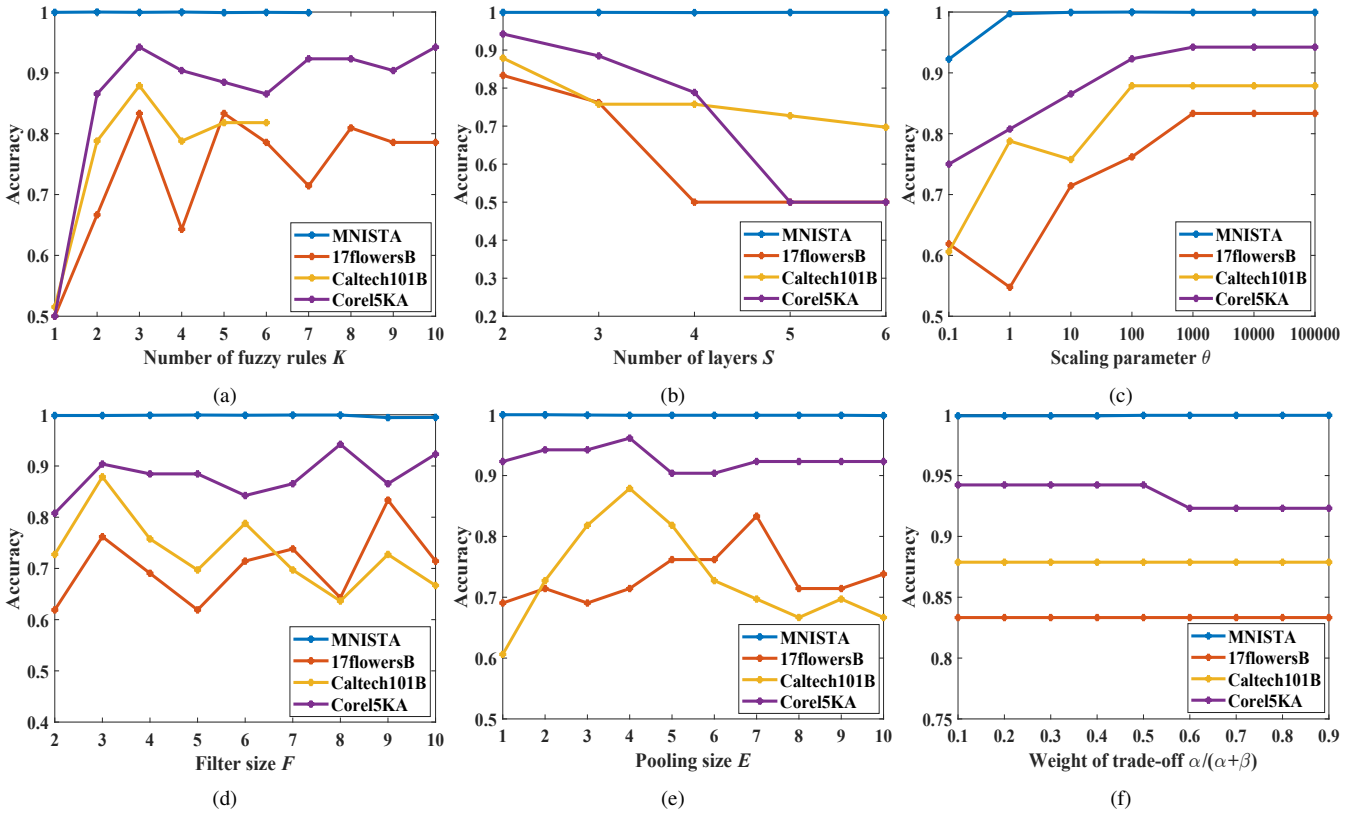


Fig. 5. Parameter sensitivity analysis of ConvFNN on binary-class datasets. (a) The performance of ConvFNN with different numbers of fuzzy rules K. (b) The performance of ConvFNN with different numbers of layers S. (c) The performance of ConvFNN with different values of scaling parameter θ. (d) The performance of ConvFNN with different filter sizes F in convolutional layers. (e) The performance of ConvFNN with different pooling sizes E in pooling layers. (f) The performance of ConvFNN with different values of $\alpha/(\alpha+\beta)$.

TABLE VII
PARAMETER SETTINGS IN MULTI-CLASS CLASSIFICATION

Algorithms	Parameter Settings
ICSC [49] [1]	Number of reduced dimensions $\{10, 30, 50, 70, 100, 200, 250\}$, Number of iterations $\{10, 20, 30, 40, 50\}$, regularization parameter $\{0.01, 0.02, 0.03, 0.04, 0.05, ..., 1\}$, Initialization of inner-class and between-class weight $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$.
ResNet50, 101+SVM [39]	We use deep learning models to extract features and use SVM to conduct classification.
SVM [50]	$C \in \{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}\}$.

[1] The time cost of ICSC on large datasets is unacceptable. Therefore, regularization parameter is set as {0.1, 0.2, 0.3,..., 1} on COIL100, CASIA-faceV5 and USPS datasets. For ICSC on USPS dataset, we just adjust one parameter and choose the parameter that achieves the best testing accuracy.

For deep learning models, we first use two methods to train the parameters in ResNet18 on seven datasets. One is using pre-trained ResNet18 to extract features and conduct classification via SVM. In this process, only the parameters in SVM are adjusted. Another method is training all the parameters in deep learning networks by Adam optimizer [51]. In the second method, available code in Pytorch is used to conduct experiments. ResNet18 is modified for dealing with 32×32 input patterns. We initially set the learning rate as 0.001 and decay the learning rate by parameter γ once the number of epochs reaches one of the milestones. In this experiment, the γ is 0.5 and the milestones are [75, 100]. That is, if the number of epochs is less than 75, the learning rate is 0.001; if the number of epochs is large than or equal to 75 and less than 100, the learning rate is 0.0005; if the number of epochs is large than or equal to 100, the learning rate is 0.00025. The maximum number of epochs is set as 200. The experimental results of ResNet18 with two different training methods are shown in Fig. 6. One can observe that the experimental results of ResNet18+SVM are clearly better than those of ResNet18 on six out of the seven datasets. This is because the training patterns of these datasets are not enough to train all the parameters in a deep-learning model. In this paper, we use pre-trained deep learning models by the ImageNet dataset to extract features. Training patterns are only used for training SVM for classification. Compared with deep learning models, LeNet-5 is quite small and the number of parameters is not high. Therefore, we use training patterns to optimize the parameters in the LeNet-5 network. Table VII shows the parameter settings in multi-class experiments.

Multi-class experimental results are shown in Table VIII. Our ConvFNN achieves the best performance on most datasets. The reason is the fact that ConvFNN utilizes the stacked convolution and pooling operation on multiple fuzzy images. It is worth mentioning that the performance of FS-FCSVM is generally better than SVM on most datasets. This observation confirms that fuzzy mapping can improve image classification accuracy. Table VIII also demonstrates that some classical deep models including VGG16+SVM, Inception-v3+SVM ResNet18+SVM, ResNet50+SVM, and ResNet101+SVM do not always achieve satisfactory results on multi-class classification problems. This is because these deep learning models

TABLE VIII
AVERAGE TESTING ACCURACY ON MULTI-CLASS CLASSIFICATION

Datasets	ConvFNN	FS-FCSVM	ICSC	SVM	LeNet-5	VGG16+SVM	Inception-v3+SVM	ResNet18+SVM	ResNet50+SVM	ResNet101+SVM
COIL20	**100.00%**	98.95%	98.68%	98.68%	94.74%	70.00%	86.84%	69.47%	87.37%	88.95%
COIL100	**91.38%**	90.67%	73.25%	90.20%	1.93%	12.20%	67.36%	5.21%	65.80%	64.08%
JAFFE	**100.00%**	**100.00%**	**100.00%**	**100.00%**	95.08%	27.87%	73.77%	32.79%	88.52%	90.16%
ORL	**100.00%**	95.00%	97.50%	94.17%	91.67%	51.67%	72.50%	61.67%	57.50%	49.17%
Leaves	**95.92%**	77.55%	85.71%	71.43%	85.71%	42.86%	79.59%	34.69%	69.39%	85.71%
MPEG7	**91.95%**	83.10%	75.00%	82.86%	79.05%	51.43%	78.33%	57.14%	77.86%	74.52%
CASIA-faceV5	**91.00%**	77.00%	82.20%	77.00%	0.20%	20.00%	41.60%	46.80%	42.00%	35.40%
UMIST	**89.94%**	77.36%	77.99%	77.36%	69.18%	14.47%	33.33%	14.47%	32.08%	31.45%
Yale	**100.00%**	96.67%	**100.00%**	95.00%	86.67%	43.33%	71.67%	76.67%	71.67%	70.00%
USPS	**98.40%**	87.33%	55.00%	86.07%	96.73%	88.20%	96.60%	92.33%	96.60%	96.33%

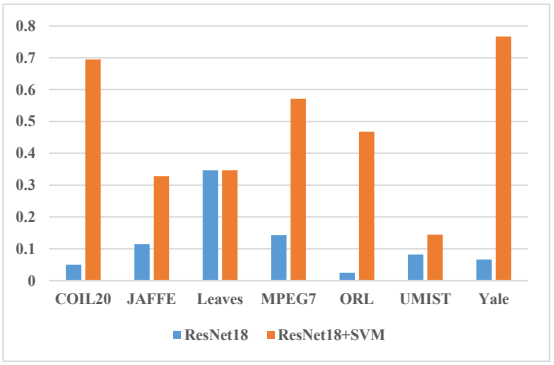


Fig. 6. Experimental results of ResNet18 and ResNet18+SVM on seven datasets.

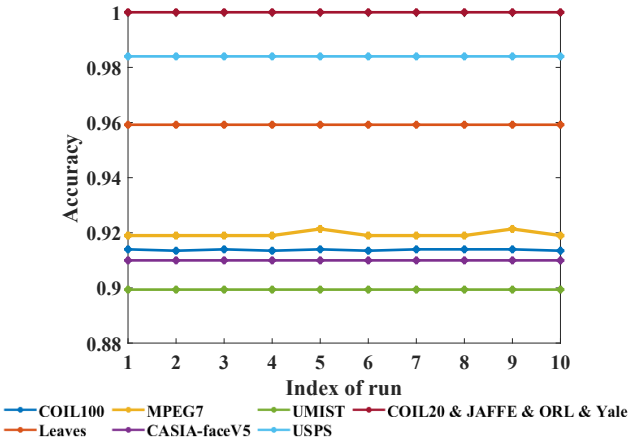


Fig. 7. The classification accuracy obtained by repeating ConvFNN ten times.

are pre-trained by the ImageNet dataset (whereas each dataset in Table VIII is not similar to the ImageNet dataset). As a result, the classification performance of deep learning models is not good. For ResNet-based models, the performance of ResNet50+SVM and ResNet101+SVM is better than that of ResNet18+SVM in most cases. More layers make higher accuracy.

To demonstrate the robustness of the proposed ConvFNN with randomly generated convolutional weights, we test the proposed approach ten times to each dataset using the same partition into the training and testing data. In each run, the convolutional weights are randomly specified. Experimental results are shown in Fig. 7. From Fig. 7, one can observe that the same or similar classification results are obtained from the ten runs for each dataset. This observation validates the robustness of our model under the randomly generated convolutional weights.

In addition to the robustness, we also evaluate the time cost for all algorithms for muti-class classification on MPEG7 and USPS datasets. Table IX shows the time cost. All algorithms are conducted on a computer with Intel(R) Xeon(R) E5-2620v4 2.10GHz CPU. FS-FCSVM and SVM finish their classification process quickly. However, the test accuracy of the proposed approach is about 10% higher than their testing accuracies. ICSC is significantly affected by the quantity of the datasets. The time cost of ICSC is unacceptable when the number of patterns is large (i.e., more than 20000). We also find that using the CPU for training LeNet-5 is time-consuming. Regarding five deep learning networks, we use pre-trained networks for extracting features and use SVM for classification. Thus, they are faster than LeNet-5. The proposed approach gets results within a quarter of the time that Inception-v3 and ResNet101 spend. For both considering the time cost and accuracy, the proposed ConvFNN is a good choice.

Next, we evaluate the significance of the regularization term. The classification results of ConvFNN without the regularization term on the JAFFE, Leaves, and USPS datasets are presented in Table X. We present both the average accuracy over ten runs and the accuracy for each individual run. The performance of ConvFNN without the regularization term is found to be inferior to that of ConvFNN with the regularization term. Large variations in accuracies are observed on the JAFFE and Leaves datasets, which brings difficulty for parameter tuning. On the USPS dataset, the accuracy is slightly lower than that of ConvFNN. This is because training data is enough for adjusting parameters in ConvFNN on the USPS dataset when there is no regularization term. Consequently, it can be inferred that the regularization term has a substantial impact on small datasets.

We further evaluate ConvFNN, LeNet-5, and five popular deep learning models (VGG16, Inception-v3, ResNet18, ResNet50, and ResNet101) on two relatively large-scale datasets (SVHN and MNIST datasets). The details of the two datasets are shown in Table VI. For deep learning models, we also use two methods to train the parameters. One is the same as the parameter settings in Table VII (i.e., only training SVM via training patterns for classification), another is training all the parameters in the deep neural networks by Adam optimizer. Experimental results are shown in Fig. 8. One can observe

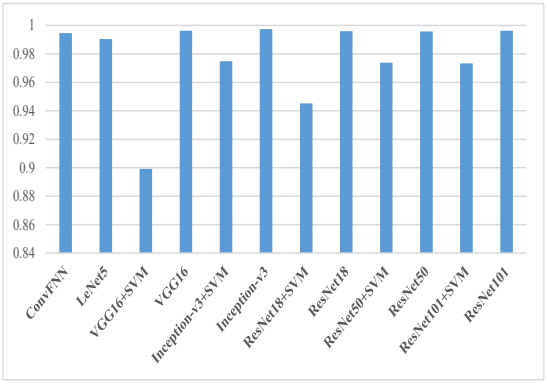
TABLE IX
THE TIME COST OF ALL MUTI-CLASS ALGORITHMS ON MPEG7 AND USPS DATASETS

Datasets	ConvFNN	FS-FCSVM	ICSC	SVM	LeNet-5	VGG16+SVM	Inception-v3+SVM	ResNet18+SVM	ResNet50+SVM	ResNet101+SVM
MPEG7	371.21s	6.59s	220.47s	2.72s	$\approx$ 1 day	498.57s	626.31s	190.71s	398.07s	640.57s
USPS	2587.54s	398.41s	21917.35s	244.96s	$>$ 1 day	11670.52s	10990.64s	3045.66s	7254.23s	12411.11s

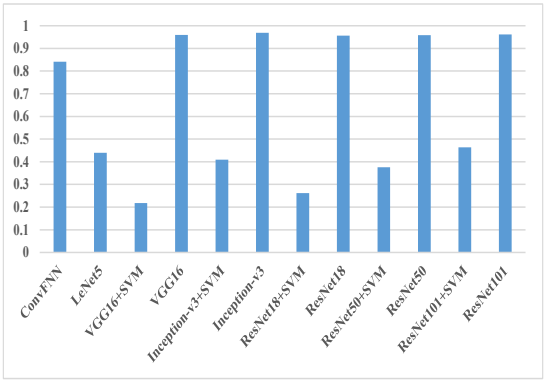
TABLE X
THE CLASSIFICATION ACCURACY OF CONVFNN WITHOUT REGULARIZATION TERM

Datasets	Average	Accuracy variations in ten runs
JAFFE	33.44%	40.98%, 32.79%, 22.95%, 19.67%, 22.95%, 60.66%, 13.11%, 37.70%, 55.74%, 27.87%
Leaves	37.55%	42.86%, 30.61%, 36.73%, 32.65%, 57.14%, 20.41%, 44.90%, 38.78%, 36.73%, 34.69%
USPS	93.57%	93.53%, 93.47%, 93.60%, 93.60%, 93.53%, 93.53%, 93.60%, 93.67%, 93.60%, 93.60%

that training all the parameters of deep learning models is a better choice for relatively large datasets. Whereas results of using pre-trained deep learning models to extract features and using SVM to classify patterns are not very good. This is because the training patterns of MNIST and SVHN are enough for adjusting the parameters in deep learning models. On the MNIST dataset, the proposed ConvFNN shows similar (but slightly inferior) performance to the fully trained deep learning models in Fig. 8 (a). On the SVHN dataset, the proposed ConvFNN is inferior to the fully trained deep learning models in Fig. 8 (b). These observations suggest that the proposed ConvFNN is not always competitive for large datasets whereas it has clear advantages in the handling of small datasets.



(a) Experimental results on MNIST dataset.



(b) Experimental results on SVHN dataset.

Fig. 8. Experimental results of ConvFNN, LeNet-5, VGG16, Inception-V3 and ResNet on MNIST and SVHN datasets.

Since ImageNet is a benchmark dataset for training deep learning models, we also test the proposed ConvFNN on the ImageNet dataset for evaluation. In the proposed ConvFNN, convolutional weights are generated randomly and the weights in the fully connected layer (i.e., weights to the output layer) are the only parameters needed to compute. This training process requires that all the training patterns should be loaded into the working memory for training in one-batch mode. We conduct these experiments on a computer with 512G memory. However, this memory size is not enough for the input patterns with the size of 64×64 since the flattened matrix in the final hidden layer is quite large. Therefore, we downsize raw images into 32×32 gray images for the proposed ConvFNN. In addition to the dimension of input patterns, the number of input patterns also significantly affects the working memory. In this paper, we test the proposed ConvFNN and four typical deep learning models (i.e., DenseNet121, ResNet101, ResNet50, and ResNet18) on nine subsets of the ImageNet dataset. First, a two-class subset is created using patterns from two classes. Then, a three-class subset is created by adding patterns from an additional class. In this manner, by adding an additional class one by one, we create subsets with 2-10 classes. The created subsets are explained as "ImageNet$Z$" which means a Z-class subset. For deep learning models, we use pre-trained neural networks with no modification.

The experimental results of the proposed ConvFNN and the four deep learning models are shown in Fig. 9. Since the labels of testing patterns on the ImageNet dataset are difficult to obtain, we train the parameters in ConvFNN by training patterns and evaluate the classification performance on validation patterns. We also evaluate the classification performance of the deep learning models on validation patterns. From Fig. 9, we can observe that the proposed ConvFNN achieves better performance than those of DenseNet121, ResNet18, and ResNet50 on the ImageNet2 dataset. With more classes, the deep learning models achieve about 80% validation accuracy; whereas the proposed ConvFNN can obtain about 60% validation accuracy. The main reason is that the classification by the deep learning models is based on 224×224 input patterns, whereas the classification by the proposed ConvFNN is based on 32×32 input patterns. This dimensionality reduction leads to severe information loss for multi-class classification. Therefore, we evaluate the classification performance of input patterns with the size of 48×48 on the ImageNet2 and ImageNet7 datasets for comparison. From Table XI, the validation accuracy is enhanced by about 4% when the size of input patterns increases from 32×32 to 48×48. With more working memory and a larger size of input patterns, the proposed ConvFNN will achieve higher accuracy.

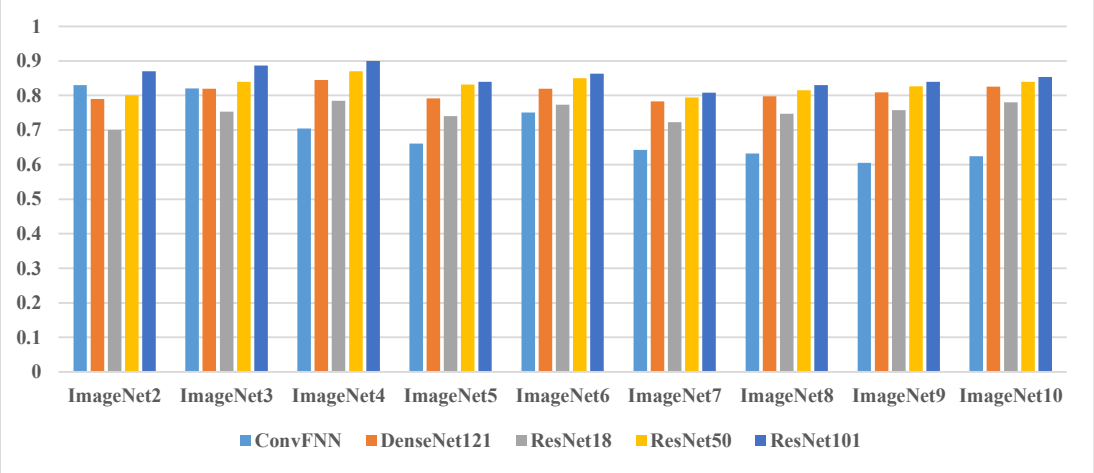


Fig. 9. Experimental results of ConvFNN, ResNet18, ResNet50, ResNet101 and DenseNet121 on the ImageNet-based datasets.

TABLE XI
VALIDATION ACCURACY WITH DIFFERENT INPUT SIZES

Datasets	Size of input patterns	Validation accuracy
ImageNet2	32×32	83.00%
	48×48	86.50%
ImageNet7	32×32	64.23%
	48×48	68.57%

C. Experiments on Image Datasets with Noise

In this experiment, we test the robustness of the proposed ConvFNN against noise. We use noisy images with different noise intensities in order to evaluate how the noise intensity affects the performance of the proposed ConvFNN. The same parameter settings as in Table III and Table VII are used in each algorithm on the noisy datasets here. Fig. 10 shows the average test accuracy repeated ten times on the Corel5KB_noise and Leaves_noise datasets. One can observe that the proposed ConvFNN has the most satisfactory results on noisy image datasets. This is because the proposed ConvFNN is developed based on fuzzy mapping before extracting features, which can be regarded as a method of data argumentation and is more robust to noise. Moreover, the proposed ConvFNN uses a novel learning criterion that integrates global and local learning for classification, which ensures the accuracy of sub-networks and the whole network simultaneously. All of these operations enhance the performance of the proposed ConvFNN on noisy images. In Fig. 10 (a), the proposed ConvFNN shows the best test accuracy. Its accuracy decreases by 15% when the noise density increases from 0% to 20%. The test accuracy of the proposed ConvFNN on noisy datasets is much better than that of some other algorithms on datasets without noise. In Fig. 10 (b), ConvFNN also shows the best results on the Leaves dataset with "salt and pepper noise". Generally, the performance of the algorithms decreases with the increase of the noise density in datasets. However, the accuracy of the proposed ConvFNN on the Leaves dataset with 15% noise density is better than that on the Leaves dataset with 5% noise density. This is in accordance with existing findings that features with noise may help improve accuracy [52].

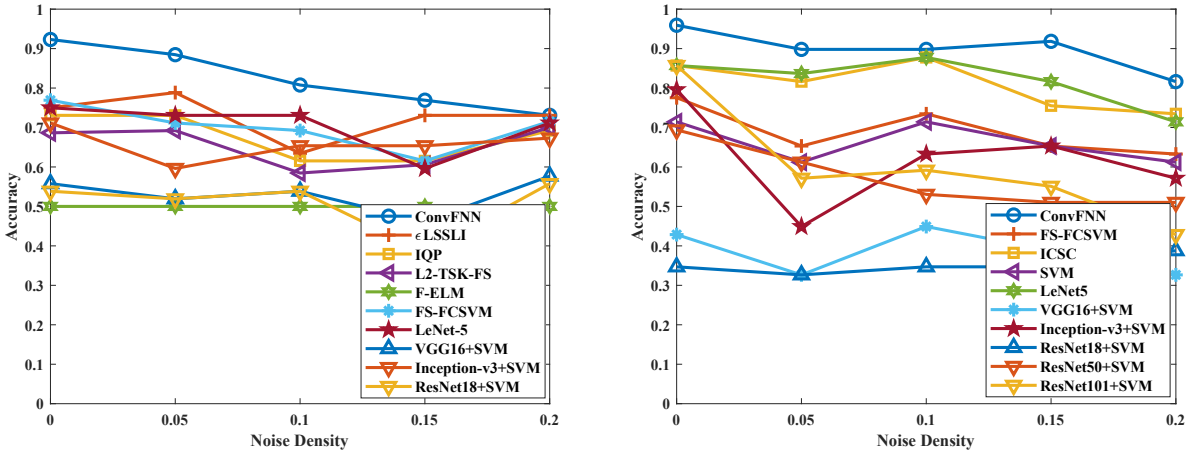
D. Discussion about Fuzzy Mapping

Firstly, we evaluate the global and local performance of the proposed ConvFNN. Here, global performance refers to the classification accuracy of the proposed ConvFNN, while local performance refers to the classification accuracy of each subnetwork for individual fuzzy rules. The classification accuracy of the proposed ConvFNN and three subnetworks are presented in Table XII. We observe that the performance of the subnetworks is slightly inferior to that of the proposed ConvFNN on most datasets. Only for the COIL100 dataset, the subnetworks show very poor classification results. This is due to the use of a relatively small number of parameters in each subnetwork for 100-class image patterns. These observations confirm that the proposed ConvFNN achieves satisfactory local performance in most cases. It is worth mentioning that the classification accuracies of ConvFNN on COIL100 and MPEG7 are slightly different between Table XII and Table VIII. This is because Table VIII shows average results over ten runs while Table XII shows single run's results.

TABLE XII
GLOBAL PERFORMANCE AND LOCAL PERFORMANCE ON MULTI-CLASS CLASSIFICATION

Datasets	ConvFNN	Subnetwork1	Subnetwork2	Subnetwork3
COIL20	100.00%	97.63%	98.68%	98.68%
COIL100	91.35%	1.20%	0.99%	1.46%
JAFFE	100.00%	100.00%	100.00%	100.00%
ORL	100.00%	98.33%	96.67%	99.17%
Leaves	95.92%	85.71%	87.76%	89.80%
MPEG7	91.90%	87.62%	89.76%	89.05%
CASIA-faceV5	91.00%	88.20%	88.40%	89.40%
UMIST	89.94%	86.16%	85.53%	89.31%
Yale	100.00%	100.00%	95.00%	96.67%
USPS	98.40%	94.73%	95.07%	95.27%

In the fuzzy mapping stage, FCM clustering is employed to compute the membership degree. Fig. 5 (a) shows the accuracy variations with the increase in the number of fuzzy rules. However, there is a special measure (i.e., the Davies-Bouldin index) to evaluate the clustering results. Table XIII presents the quality of clustering results based on Davies-Bouldin values for multi-class datasets. A smaller value indicates better



(a) Experimental results on Corel5KB dataset with "salt and pepper noise".

(b) Experimental results on the Leaves dataset with "salt and pepper noise".

Fig. 10. Experimental results on Corel5KB and Leaves datasets with "salt and pepper noise".

clustering quality. As can be observed from XIII, two fuzzy rules are often the optimal choice. To examine the impact of clustering quality on classification performance, we present the classification results for the optimal number of fuzzy rules as measured by the Davies-Bouldin index in Table XIV. The classification results are denoted as $ACC(K)$, where K represents the number of fuzzy rules. It can be observed that the performance of CovFNN with tuning parameters is generally superior to those defined by the Davies-Bouldin index. In ConvFNN, the number of fuzzy rules is equivalent to the number of subnetworks. The learning ability of three subnetworks surpasses that of two subnetworks. A larger number of fuzzy rules can enhance the performance of ConvFNN but may also lead to overfitting. Therefore, determining an appropriate number of fuzzy rules that balances classification accuracy improvement and overfitting is a challenging future research topic for the proposed ConvFNN.

TABLE XIII
DAVIES–BOULDIN VALUES OF FUZZY MAPPING ON MULTI-CLASS CLASSIFICATION

Datasets	Number of fuzzy rules								
	2	3	4	5	6	7	8	9	10
COIL20	**0.45**	0.68	1.05	1.11	0.90	0.95	1.17	1.07	1.07
COIL100	**0.46**	0.59	0.80	0.92	1.01	1.24	1.31	1.43	1.18
JAFFE	**0.56**	0.74	0.77	0.82	0.93	0.90	0.91	1.11	0.94
ORL	**0.89**	1.27	1.02	1.12	1.25	1.19	1.06	1.09	1.17
Leaves	**0.60**	1.03	1.35	1.16	1.08	1.06	1.04	1.10	1.04
MPEG7	0.70	0.62	**0.51**	0.72	0.71	0.73	0.76	0.81	0.80
CASIA-faceV5	**0.64**	0.82	0.98	1.15	1.27	1.45	1.65	1.70	1.51
UMIST	**0.46**	0.79	0.66	0.76	0.92	1.03	0.98	0.90	0.96
Yale	0.98	0.80	0.73	0.68	**0.65**	0.74	0.97	0.96	0.91
USPS	**0.81**	1.37	1.07	1.16	1.06	0.97	0.92	1.12	1.13

In addition to the number of fuzzy rules, we also examine the impact of different partitions of the input space on classification accuracy. Subtractive clustering can return computed clustering centers, which can directly define the number of clusters (i.e., fuzzy rules). The classification results of different clustering methods with the same number of fuzzy rules are presented in Table XIV. It can be observed that the choice of a partition method for the input space has only a very minor effect on the classification accuracy of ConvFNN.

TABLE XIV
CLASSIFICATION ACCURACY OF CONVFNN WITH DIFFERENT FUZZY MAPPING STRATEGIES ON MULTI-CLASS CLASSIFICATION

Datasets	Different number of fuzzy rules		Different clustering methods	
	Tuning (proposed)	Davies–Bouldin index	FCM (proposed)	Subtractive clustering
COIL20	100.00% (3)	99.21% (2)	100.00% (3)	100.00% (3)
COIL100	91.35% (3)	91.40% (2)	91.35% (3)	90.82% (3)
JAFFE	100.00% (3)	100.00% (2)	100.00% (4)	100.00% (4)
ORL	100.00% (3)	98.33% (2)	100.00% (5)	100.00% (5)
Leaves	95.92% (3)	91.84% (2)	91.84% (2)	91.84% (2)
MPEG7	91.90% (3)	91.19% (4)	91.19% (4)	91.19% (4)
CASIA-faceV5	91.00% (3)	90.60% (2)	91.00% (3)	91.00% (3)
UMIST	89.94% (3)	86.79% (2)	91.19% (4)	89.31% (4)
Yale	100.00% (3)	98.33% (6)	98.33% (7)	98.33% (7)
USPS	98.40% (3)	97.60% (2)	97.60% (2)	97.13% (2)

The above experiments focus on the impact of clustering centers, specifically the centers of membership functions. Subsequently, an evaluation of the spreads is conducted. In Eq. (6), the scaling parameter θ is used to adjust the spreads of the Gaussian membership functions. Fig. 11 shows the accuracy variations with different scaling parameter values. It can be observed that the accuracy exhibits instability when θ ranges from 0.1 to 1000. However, no change in accuracy is noted when θ is in a range of [1000, 100000]. The computation of the spreads leads to a minimal time cost of a few seconds, indicating that this tuning process does not significantly increase complexity.

In the antecedent part of the proposed ConvFNN, only several selected reference pixels are utilized to compute the membership degree. According to [20], we set the number of reference pixels to 5. Fig. 12 illustrates classification accuracy variations with different dimensions of reference pixels. The number of reference pixels is in the range of [2, 10]. It is observed that there is almost no impact on classification accuracy when the number of reference pixels is less than ten.

To this end, we conclude that fuzzy mapping is useful for improving the accuracy since the performance of the proposed ConvFNN is better than those of subnetworks. However, premise parameters have an impact on the structure of a fuzzy classifier, and consequent parameters play an important role in the classification accuracy improvement. The fuzzy

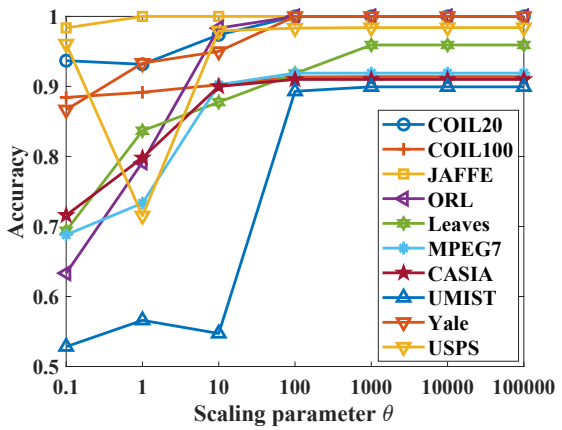


Fig. 11. The classification accuracy variations with different spreads in Gaussian membership function.

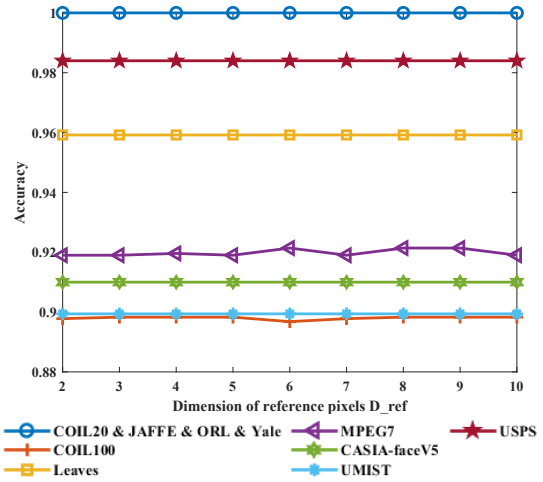


Fig. 12. The classification accuracy variations with different dimensions of reference pixels.

mapping has three contributions to the proposed ConvFNN. (1) The number of fuzzy rules controls the width of the fuzzy neural network structure. It is a fusion strategy of combining several sub-networks. (2) Fuzzy mapping is a way of data augmentation, it transforms one image into several fuzzy images with computed membership degrees. This trick is useful on quite small datasets. (3) The proposed ConvFNN can have a balance between global learning and local learning with the firing strength. This not only enhances its performance but also increases its stability in the presence of noise.

V. Conclusion

We proposed a novel ConvFNN fuzzy model for image classification, which combines an improved CNN framework with the fuzzy system. In contrast to conventional fuzzy models, the proposed ConvFNN handles the local correlation of natural images by introducing stacked convolutional and pooling operations. Moreover, the developed model used randomly generated filters in the convolution stage and a closed-form solution to compute the weights of the output layer, which leads to fast learning. Experimental results based on real-world image datasets demonstrated that the proposed ConvFNN outperforms state-of-the-art fuzzy classifiers, SVM, and classical deep learning models especially when training patterns are not enough for deep learning. The proposed ConvFNN achieves good results on small and relatively large datasets. In this paper, we explored the potential benefits of combining CNNs with fuzzy-set-based fusion techniques.

State-of-the-art deep learning models have achieved impressive results in image classification. They always require massive labeled data to train a powerful model, which is a time-consuming procedure. However, they always suffer from the over-fitting problem on small training datasets. In this paper, we have proposed ConvFNN to fill this gap. Our experimental results show that satisfactory classification performance is obtained by the proposed ConvFNN even when training patterns are not available for deep learning. Computing the weights in the fully connected layer of ConvFNN requires that all the training patterns should be loaded into the random-access-memory (RAM) for calculating the optimal weights. As a result, the working memory required by the proposed ConvFNN on a large-scale dataset can be extremely large. We will investigate this issue in the future when there will be such a large RAM machine.

References

[1] W. Pedrycz and F. Gomide, *An introduction to fuzzy sets: analysis and design*. MIT press, 1998.
[2] J. C. Bezdek, J. Keller, R. Krisnapuram, and P. Nikhil, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. New York, NY, USA: Springer Science+Business Media, 1999.
[3] H. Ishibuchi and T. Nakashima, "Effect of rule weights in fuzzy rule-based classification systems," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 4, pp. 506–515, 2001.
[4] W. Pedrycz, "Fuzzy sets in pattern recognition: Methodology and methods," *Pattern Recognition*, vol. 23, no. 1, pp. 121–146, 1990.
[5] I. Couso, C. Borgelt, E. Hüllermeier, and R. Kruse, "Fuzzy sets in data analysis: From statistical foundations to machine learning," *IEEE Computational Intelligence Magazine*, vol. 14, no. 1, pp. 31–44, 2019.
[6] B. J. Murray, M. A. Islam, A. J. Pinar, D. T. Anderson, G. J. Scott, T. C. Havens, and J. M. Keller, "Explainable AI for the choquet integral," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 4, pp. 520–529, 2021.
[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[8] Q. Lai, J. Zhou, Y. Gan, C.-M. Vong, and C. P. Chen, "Single-stage broad multi-instance multi-label learning (BMIML) with diverse inter-correlations and its application to medical image classification," *IEEE Transactions on Emerging Topics in Computational Intelligence*, Early Access.
[9] K. Yao, K. Huang, J. Sun, and A. Hussain, "Pointnu-net: Keypoint-assisted convolutional neural network for simultaneous multi-tissue histology nuclei segmentation and classification," *IEEE Transactions on Emerging Topics in Computational Intelligence*, Early Access.
[10] R. Krishnapuram and J. Lee, "Fuzzy-set-based hierarchical networks for information fusion in computer vision," *Neural Networks*, vol. 5, no. 2, pp. 335–350, 1992.
[11] S.-B. Cho and J. H. Kim, "Combining multiple neural networks by fuzzy integral for robust classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 2, pp. 380–384, 1995.
[12] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 1, pp. 116–132, 1985.
[13] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York, NY, USA: Plenum, 1981.
[14] Z. Deng, K.-S. Choi, Y. Jiang, and S. Wang, "Generalized hidden-mapping ridge regression, knowledge-leveraged inductive transfer learning for neural networks, fuzzy systems and kernel methods," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2585–2599, 2014.

[15] X. Liu, L. Jiao, L. Li, L. Cheng, F. Liu, S. Yang, and B. Hou, "Deep multiview union learning network for multisource image classification," *IEEE Transactions on Cybernetics*, vol. 52, no. 6, pp. 4534–4546, 2022.
[16] M. Liu, L. Jiao, X. Liu, L. Li, F. Liu, and S. Yang, "C-CNN: Contourlet convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 6, pp. 2636–2649, 2021.
[17] J. Gao, L. Jiao, X. Liu, L. Li, P. Chen, F. Liu, and S. Yang, "Multiscale dynamic curvelet scattering network," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2022.
[18] X. Liu, L. Li, F. Liu, B. Hou, S. Yang, and L. Jiao, "GAFnet: Group attention fusion network for pan and ms image high-resolution classification," *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 10 556–10 569, 2022.
[19] P. N. Suganthan and R. Katuwal, "On the origins of randomization-based feedforward neural networks," *Applied Soft Computing*, vol. 105, p. 107239, 2021.
[20] Y. Wang, H. Ishibuchi, M. J. Er, and J. Zhu, "Unsupervised multilayer fuzzy neural networks for image clustering," *Information Sciences*, vol. 622, pp. 682–709, 2023.
[21] Y. Wang, H. Ishibuchi, J. Zhu, Y. Wang, and T. Dai, "Unsupervised fuzzy neural network for image clustering," in *Proceedings of 2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–6.
[22] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proceedings of 2009 IEEE 12th International Conference on Computer Vision (ICCV)*, pp. 2146–2153.
[23] A. M. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng, "On random weights and unsupervised feature learning," in *Proceedings of the 28th International Conference on Machine Learning*, 2011, pp. 1089–1096.
[24] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2011.
[25] G.-B. Huang, Z. Bai, L. L. C. Kasun, and C. M. Vong, "Local receptive fields based extreme learning machine," *IEEE Computational Intelligence Magazine*, vol. 10, no. 2, pp. 18–29, 2015.
[26] J. Ngiam, Z. Chen, D. Chia, P. Koh, Q. Le, and A. Ng, "Tiled convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 23, 2010.
[27] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 111–118.
[28] J. Yen, L. Wang, and C. W. Gillespie, "Improving the interpretability of TSK fuzzy models by combining global learning and local learning," *IEEE Transactions on fuzzy Systems*, vol. 6, no. 4, pp. 530–537, 1998.
[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[30] M.-E. Nilsback and A. Zisserman, "A visual vocabulary for flower classification," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1447–1454.
[31] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.
[32] G.-H. Liu, J.-Y. Yang, and Z. Li, "Content-based image retrieval using computational visual attention model," *Pattern Recognition*, vol. 48, no. 8, pp. 2554–2566, 2015.
[33] J. Leski, "TSK-fuzzy modeling based on /spl epsiv/-insensitive learning," *IEEE Transactions on Fuzzy Systems*, vol. 13, no. 2, pp. 181–193, 2005.
[34] Z. Deng, K.-S. Choi, F.-L. Chung, and S. Wang, "Scalable TSK fuzzy modeling for very large datasets using minimal-enclosing-ball approximation," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 2, pp. 210–226, 2010.
[35] S. Y. Wong, K. S. Yap, H. J. Yap, S. C. Tan, and S. W. Chang, "On equivalence of FIS and ELM for interpretable rule-based knowledge representation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 7, pp. 1417–1430, 2014.
[36] C.-F. Juang, S.-H. Chiu, and S.-J. Shiu, "Fuzzy system learned through fuzzy clustering and support vector machine for human skin color segmentation," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 37, no. 6, pp. 1077–1087, 2007.
[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of International Conference on Learning Representations*, 2015.
[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.
[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
[40] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (COIL-100)," Department of Computer Science, Columbia University, New York, NY, USA, Technical Report CUCS-006-96, February 1996.
[41] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets," in *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 200–205.
[42] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, pp. 138–142.
[43] L. J. Latecki, R. Lakamper, and T. Eckhardt, "Shape descriptors for non-rigid shapes with a single closed contour," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2000, pp. 424–429.
[44] W. Gao, B. Cao, S. Shan, X. Chen, D. Zhou, X. Zhang, and D. Zhao, "The cas-peal large-scale chinese face database and baseline evaluations," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 38, no. 1, pp. 149–161, 2007.
[45] D. B. Graham and N. M. Allinson, *Characterising Virtual Eigensignatures for General Purpose Face Recognition*. Berlin, Heidelberg: Springer, 1998, pp. 446–456.
[46] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, 1997.
[47] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 550–554, 1994.
[48] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.
[49] S. Teng, Z. Zheng, N. Wu, L. Fei, and W. Zhang, "Domain adaptation via incremental confidence samples into classification," *International Journal of Intelligent Systems*, vol. 37, no. 1, pp. 365–385, 2022.
[50] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, 2011.
[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of International Conference on Learning Representations*, 2015.
[52] G. Huang, S. Song, J. N. D. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2405–2417, 2014.