T–S Fuzzy Model Based Multi-Branch Deep Network Architecture

FAGUANG WANG1,2, YUE WANG2, HONGMEI WANG1,2, AND CHAOGANG TANG3, (Member, IEEE)

1 Engineering Research Center of Intelligent Control for Underground Space, Ministry of Education, China University of Mining and Technology, Xuzhou 221116, China
2 School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China
3 School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China

Corresponding author: Hongmei Wang (whm99@cumt.edu.cn)

ABSTRACT In the traditional CNN design, the hyperparameters, such as the size of the convolutional kernel and stride, are difficult to determine. In this paper, a new convolutional network architecture, named multi-branch fuzzy architecture network (MBFAN), was proposed for this problem. In MBFAN, some branches with a certain convolutional neural network architecture are connected in parallel. In each branch, a different-sized convolutional kernel is applied. By data training and normalization, a weight is given to each branch. By these weights, the important features in the final output are strengthened. By normalization, the branches were interconnected together, making the training process more efficient. Due to overfitting, with the increase of branches, the MBFAN accuracy increases, and then decreases. The number of branches is optimized when the MBFAN accuracy is highest. On the other hand, the location of the convolutional kernel center in an image has a great influence on the convolutional results. This is also discussed in MBFAN. For the experiments, the proposed MBFAN was adopted and tested in a simple convolutional network and a VGG16 network.

INDEX TERMS Deep learning, convolutional neural network, multi-branch, fuzzy.

I. INTRODUCTION

With the development of computer technology, the amount of image data has increased rapidly. Image processing turned to be a big data problem. There are many tasks of image data processing, such as colorization, inpainting, deblurring, image classification, object detection, semantic scene labeling, and instance segmentation [1]–[3]. Compared with the histogram of oriented gradient (HOG) [4], scale-invariant feature transform (SIFT) [5], and so on, the convolutional neural network (CNN) has been broadly studied. Many different CNN architectures and methods have been proposed, for example, AlexNet, VGGNet, GAN, ResNet, and SENet [6]–[10].

CNNs have achieved great success in image classification, object detection, semantic segmentation, image compression, and other fields [11]–[16]. A CNN is generally composed of an input layer, convolutional layers, pooling layers, fully connected layers, and an output layer. The convolutional layer extracts features of an input image through a convolution operation. The pooling layer can reduce the computation of CNN training and highlight some important features in the image. Some scholars have also designed the up-sampling method based on the characteristics of the pooling layer, which was used to explain CNN [17], such as a generative adversarial network (GAN) [18], and so on [8], [19]. In general, the pooling operations are maximum pooling and average pooling [20]. There are still overlapping pooling, spatial pyramid pooling, and so on [6], [21]. A fully connected layer is a traditional neural network structure, where each neuron in the fully connected layer is connected to all neurons in the previous layer. The CNN generally ends with a classifier. The most common classification methods are binary classification, softmax, AdaBoost, k-nearest neighbors, support vector machine (SVM) and so on [22]–[24].

CNNs have many advantages in image processing. However, designing a traditional CNN for a particular task is difficult because the CNN is uninterpretable. This means that the designer cannot explain the physical significance of every layer, and computations inside a CNN are out of control. Designers cannot predict what the output will be. Therefore, the hyperparameters, layer architectures, filter size, strides, etc., are difficult to be determined before the
result is given out. Although a deconvolutional network was proposed for CNN interpretation which attempts to make the convolutional network more transparent, there is no shortcut to determining these hyperparameters directly [25], [26]. The location of the convolution center in an image has a great influence on the convolution results. By selecting different padding sizes and strides, the convolution results can also be very different. These are also hyperparameter determination problems.

For these reasons, the MBFAN was proposed for traditional convolution neural networks in this paper. MBFAN is a sub-network structure that needs to be used in combination with a mature network. There are several branches with different sizes of convolutional kernels and strides in MBFAN. If there are as many branches as possible, the best-performing convolutional kernel and stride can be contained. In the output of MBFAN, these well-performing branches take a larger proportion by the fuzzy architecture designed in our proposed model. The realization of the fuzzy part of the MBFAN was inspired by the Takagi–Sugeno (T–S) fuzzy model. This is explained in detail in Section 2.

II. RELATED WORK
A. INCEPTION NETWORKS FAMILY
Because of the great difference between training data, it is difficult to select the appropriate convolution kernel size for convolution operation. Moreover, the very deep network is easier to overfit. Inception-V1 was proposed for these problems [27]. Inception-V1 is a repetitive network. Each repetitive unit is composed of a fixed structure which has four parallel branches with $1 \times 1$ convolution, $3 \times 3$ convolution, $5 \times 5$ convolution and maximum pooling. Inception-V2 was proposed with the same architecture with Inception-V1 [28]. In Inception-V2, a factorized convolution method was applied. The larger convolution kernel was decomposed into a series of smaller ones. This method reduced the computation and improved the computing speed. In the same paper, Inception-V3 was proposed too. Based on Inception-V2, label smoothing regularization, BN-auxiliary and RMSProp were applied. To further reduce computation and improve accuracy, Inception-V4 and Inception-ResNet were proposed by conjunction with ResNet [29].

Inception Networks and Inception-ResNet are structurally distinct from MBFAN because they consist of several fixed network units in tandem repeatedly. Because there are several convolutional units in the branch of Inception-ResNet and only one convolution unit in each branch of MBFAN, the computation of MBFAN is simpler and less than Inception-ResNet.

B. SENet AND SKNet
The squeeze-and-excitation network (SENet) adaptively reallocates channel characteristic responses through the network’s global loss function by explicitly modeling interdependencies between channels [10]. The importance of each feature map is automatically acquired through learning, and then useful features are promoted according to their importance, while useless features are suppressed. SENet is composed of multiple squeeze-and-excitation (SE) blocks. The SE block can be embedded in any other network model but inevitably increases the number of network parameters and computational cost. Still, many other similar structures have been studied [30], [31].

Part of this paper is inspired by SENet, which is also a sub-network structural unit. Compared with the SENet structure, the proposed MBFAN structure emphasizes the parallel structure of multiple branches. A convolution filter is added before each branch which works with different size of convolutional kernel and strides. By this design, the feature extraction can be more efficient.

A dynamic selection mechanism, named Selective Kernel Networks (SKNets), was proposed that allows each neuron to adaptively adjust its receptive field size based on multiple scales of input information [32]. SKNet is the enhanced version of SENet. SK convolution consists of split, fuse and select. This network is repeated in structure. SKNet-50 consists of 50 SK convolutional layers, which is the typical example in reference [32]. In SKNet, the number of branches is optional, but the two-branch structure is generally used in consideration of computation. It has concluded through experiments that the larger the proportion of the recognized object in the figure, more information can be obtained from the large convolution kernel branch.

Compared with SKNet, the proposed MBFAN of this paper is not a repeated structure in the direction of network depth. Moreover, SKNet does not have an accurate standard in network depth. The proposed MBFAN can be used in combination with other mature networks, and the method of establishing multiple branches was clearly described in this article.

C. T–S FUZZY MODEL
The T–S fuzzy model was proposed in 1985 [33]. It is very useful for nonlinear system modeling and control problems [34], [35]. It has been proven that the T–S fuzzy model can approximate the most nonlinear system [36]. The T–S fuzzy model consists of fuzzy rules and their corresponding local models. The local model presents the system dynamic characteristics under the equilibrium points defined by the corresponding fuzzy rule. The global nonlinear system model is described by a combination of these local models. For a continuous-time nonlinear system, the typical T–S fuzzy rule has the following form.

The $i$-th rule: If $v_1(t)$ is $F_{i1}$ and ... and $v_k(t)$ is $F_{ik}$, then

$$\hat{x}(t) = A_ix(t) + B_iu(t)$$

where $i \in \{1, 2, \ldots, n\}$, $n$ is the number of rules, $v_1(t), v_2(t), \ldots, v_k(t)$ are premise variables which are the equilibrium points described by the system variables, $F_{ij}$ is the fuzzy set, $x(t)$ is the system variable, $u(t)$ is the system input, $A_i$ and $B_i$ are constant matrices with proper dimensions which can be calculated out in each equilibrium point.
The T–S fuzzy model for the nonlinear system can be

\[
\dot{x}(t) = \sum_{i=1}^{n} \mu_i(v(t))[A_i x(t) + B_i \mu(t)]
\]

\[
= \sum_{i=1}^{n} M_i(v(t))[A_i x(t) + B_i \mu(t)]
\]

(2)

where \( v(t) = [v_1(t), v_2(t), \ldots, v_k(t)] \), \( \mu_i(v(t)) \) is the grade of membership of \( v_j(t) \) in fuzzy set, \( \mu_i(v(t)) = \prod_{j=1}^{k} h_{ij}(v_j(t)) \), \( M_i \) is the membership function, which describes the proportion of the corresponding local model (1) in the total model, \( M_i(v(t)) \geq 0 \) and \( \sum_{i=1}^{n} M_i(v(t)) = 1 \).

In the proposed MBFAN, the training or testing image was fed to many parallel convolutional filters of different sizes. These parallel convolutional filters are called branches. They have the same function as the local model in the T–S fuzzy model. Still, there is another network in MBFAN, which gives out the proportion of each branch. This is similar to the membership function of the T–S fuzzy model. Thus, we have explained that the proposed MBFAN was inspired by the T–S fuzzy model.

III. METHODS AND MATERIALS

CNNs have been studied for years. To date, the accuracy of image processing by CNN has improved slowly because the precision is very high. Small precision improvements are also difficult. This often means more calculations. Therefore, improving accuracy and reducing computational cost are very important for image processing. For this purpose, the MBFAN method was proposed in this paper. The multi-branch architecture solved the problem of hyperparameters selection. The operating principle for these multi-branches is adopted from the fuzzy theorem. The proposed network architecture is shown in Figure 1.

A. ARCHITECTURE OF MBFAN

In Figure 1, filters 1, 2, …, n are of different convolutional kernel sizes, where n is the number of filters. The input passes through n parallel branches and ends to the normalization block. Meanwhile, \( f_1, f_2, \ldots, f_n \), are the outputs of the filters, and they are also parallel with these branches. The size of \( f_i \in [f_1, f_2, \ldots, f_n] \) depends on the size of the input size, convolutional kernel filter size, and strides. Next, \( b_i \in [b_1, b_2, \ldots, b_n] \) is a constant value and describes the weight of each branch. \( f_1, f_2, \ldots, f_n \) and the corresponding \( b_1, b_2, \ldots, b_n \) construct the fuzzy set. \( f_1, f_2, \ldots, f_n \) are called fuzzy features because the outputs of the filters are uncontrollable and uninterpretable. However, it can be believed that if there are as many different filters as possible, the best feature map can be extracted. This means if there are as many filters as possible, a \( f_j \) with in \( f_1, f_2, \ldots, f_n \) can be the best feature map, which is very helpful for image classification or other tasks in image processing. Therefore, it can be interpreted as a fuzzy set. After normalization, a constant parameter \( w_i \in [w_1, w_2, \ldots, w_n] \) is given for each branch. The normalization output \( w_i \) is calculated by

\[
w_i = \frac{b_i}{b_1 + b_2 + \cdots + b_n}.
\]

(3)

By normalization and error backpropagation training, parameters can be trained interactively between different branches. Moreover, in the fuzzy system, the membership function is obtained in the same way as in Equation (1). Therefore, the proposed method in Figure 1 is called the parallel multi-branch fuzzy architecture.

B. MBFAN FUNCTION ANALYSIS AND EXPLANATION

In the MBFAN, the problem of hyperparameters and layer architectures can be solved when there are as many branches as possible. In Figure 1, the convolutional kernel sizes of filters 1, 2, ..., filter \( \ell \), and filter \( n \) can be selected from \( 1 \times 1, 3 \times 3, \ldots, (2i-1) \times (2i-1), \ldots, (2n-1) \times (2n-1) \). When \( n \) is large enough, one or more filters that have an important effect on image processing must be included in these \( n \) filters.

As we know, there exist one or more important filters that will be helpful for our work. However, we do not know which filters belong to which branches in Figure 1. To deal with this problem, ResNet, SENet, and the attention mechanism were applied with the fuzzy theorem in this study. The branches in the red dotted rectangular box of Figure 1 are applied to
decide which filters are important for image processing. After normalization, the outputs of these branches are changed to membership, which is described by a constant from 0 to 1. This membership presents the importance of the filters. The single branch, in the red dotted rectangular box of Figure 1, is from [10], where the reasonability is analyzed. For the unit described later in our work, the outputs of the normalization part \( w_i \) in Figure 1 are called the fuzzy membership function weight.

In an image, the location of the convolution-kernel center is also important for the computation of convolution. This location is determined by the hyperparameters such as the convolutional kernel size, paddings, and stride. The optimal hyperparameters are difficult to determine because CNN is uninterpretable. However, in MBFAN, for a certain-size convolutional filter, a set of different paddings and strides can be used in different branches. The best one can still be selected by the fuzzy membership function weight.

In summary, the works of this study include:

1) As the traditional CNN is uninterpretable, the size of the convolutional kernel is difficult to determine. In MBFAN, multiple branches are applied in parallel. In each branch, a different-sized convolution kernel is applied. If sufficient branches are applied, the best one will not be missed.

2) In a training or testing image, the location of the convolution-kernel center and the size of the filter are important for the computation of convolution. In MBFAN, by setting strides, paddings, and filter size, a set of more different feature maps can be extracted from different branches.

3) Based on the optimization of weights \( w_1, w_2, \ldots, w_n \) in branches, the proportions of feature maps in each branch are adjusted in the final summary feature map. In this way, the branches with more important features can be strengthened.

4) Normalization makes the branches interconnect. By error backpropagation, the parameters of all branches are learned interactively.

C. CONSTITUTE BRANCHES WITH SAME-SIZE FILTERS

It is supposed that the proposed MBFAN of Figure 1 will show a good training result if there are as many branches as possible. However, for an image with a small size, only fewer branches can be applied. For example, for a \( 32 \times 32 \) image, the convolution filters can only be of \( \{1 \times 1, 3 \times 3, 5 \times 5, \ldots, 31 \times 31\} \), and only 16 branches can be added to the proposed MBFAN. Although a better result can be obtained by MBFAN, the effect is not significant enough. For this consideration, some complements for MBFAN are listed as follows.

1) For a certain-size filter in Figure 1, it will be applied more than once.

2) The same-padding method is applied for the convolution computation of filters in Figure 1. Following this rule, the same-size outputs of these filters are guaranteed. Note that this same-padding is different from the same-padding in TensorFlow 1.x version.

3) Different strides are applied for a certain-size convolutional kernel in different branches. For this purpose, the following equation is satisfied:

\[
\begin{align*}
S_{out} &= \text{ceil}(\frac{S_{in} - S_{filter} + 1}{S_{stride}}), \\
S_{in} &= S_{image} + \text{Padding} \\
S_{out} &= S_{image}
\end{align*}
\]

where \( S_{out} \) is the size of the filter output, \( S_{in} \) is the size of the filter input, \( S_{filter} \) is the filter size, \( S_{image} \) is the size of the image in the training or testing dataset, \( S_{stride} \) is the stride of this filter, \( \text{Padding} \) is the padding size, and \( \text{ceil} \) means obtaining the integer toward positive infinity. Note that, because same padding is applied, the size of \( S_{out} \) is same as the size of input and output of MBFAN.

Following these rules, for a certain-size convolutional kernel, a set of branches can be constituted with different strides. For example, for a \( 32 \times 32 \) CIFAR10 dataset image with \( 3 \times 3 \) size filter, the stride and padding can be \( stride = 1, \text{padding} = 2 \) or \( stride = 2, \text{padding} = 33 \). For a \( 32 \times 32 \) image, when \( stride = 3, \text{padding} = 64 \) is too large. Therefore, only \( stride = 1, \text{padding} = 2 \) are applied. With an increasing filter size, more branches can be added in. For a \( 32 \times 32 \) image, the filters can be \( \{1 \times 1, 3 \times 3, 5 \times 5, \ldots, 31 \times 31\} \), by constituting branches with same-size filters, the branch number is changed to 32. It is double that of the initial case, and the identification results can be more accurate. The illustrative experiment is shown in Section 4.

IV. EXPERIMENTS

A. TWO NETWORKS APPLIED IN THIS STUDY

In this study, two networks were applied to prove the research effect. One is a simple CNN. The other one is the typical VGG16 network [7]. These two networks follow our proposed MBFAN in Figure 1. The architectures are shown in Figure 2 and Figure 3, respectively.

In Figure 2, there are two \( 3 \times 3 \) convolution layers, which are followed by the ReLU activation function. Two max-pooling layers and one fully connected layer are applied. The output is given after the softmax layer. The number of parameters is less because of the simple architecture. In addition, it is convenient to check the padding size and easily obtain the information of the convolution center. Our design purpose is to test whether the proposed MBFAN works by comparing the case with or without MBFAN. The accuracy
of Figure 2 is not important for the experiment. Therefore, this simple architecture was adopted despite its low accuracy.

In Figure 3, the typical VGG16 network is shown. The number in brackets is the amount of the convolution kernel. Every convolution layer followed a ReLU activation function. There are 3 fully connected layers, two with 4096 connecting points, and one with 1000 connecting points. The outputs of softmax are 1000.

The simple CNN of Figure 1 is easy to program and test. The VGG16 network is a typical network that is useful for proving that the proposed MBFAN can work for other mature networks.

Notes for the experiment:
1) For comparing experiments, under the same architecture network, the same initial values were set.
2) For the proposed MBFAN, BN was applied to data processing.
3) For the experiments, the 32 × 32 CIFAR10 dataset was applied.

B. EXPERIMENTS FOR A SINGLE BRANCH AND MULTIPLE BRANCHES

To prove the efficiency of the proposed MBFAN, the comparison experiments are presented in this section. A simple CNN testing network was built by contacting the “MBFAN output” points in Figures 1 and 2. In Figure 1, there are n branches. In the experiments, n is equal to 1, 3, 16, and 32, respectively. The filter size is listed in Table 1. When n = 1, many filters can be selected. The 5 × 5 filter is selected because it has achieved a good result. When n = 32, only 16 filters are listed in Table 1 because stride = 1 and 2 are selected with one filter. This is explained in the above section. Thus, there were a total of 32 branches. The filter sizes are listed in Table 1. The experimental results are shown in Table 2 and Table 3.

In Table 2 and Table 3, two basic networks, the simple CNN of Figure 2 and the VGG16 of Figure 3, were applied in the experiment. The proposed network architecture of MBFAN, shown in Figure 1, was also used. For the VGG16 network experiments, the red dotted box part of Figure 1 was changed by 1 branch, 3 branches, 16 branches, and 32 branches of the VGG16 network. The numbers of branches applied in the experiment are listed in the second row of Table 2 and Table 3. The training and testing accuracies are listed in the third and fourth rows in Table 2 and Table 3. The “Initial Parameters” of Table 2 and Table 3 is the number of parameters for the sample CNN of Figure 2 and VGG16 of Figure 3. The “MBFAN additional parameters” of Table 2 and Table 3 is the number of additional parameters for MBFAN without the initial parameters. The total parameters of Table 2 and Table 3 are the total parameters of MBFAN. Comparing the results in Table 2 and Table 3, the following conclusions can be drawn:

1) In the simple CNN, the accuracy increases with an increase in the number of branches from branch n = 1 to n = 3.
2) In VGG16, the accuracy increases with an increase in the number of branches from branch n = 1 to n = 3.
3) The accuracies do not always increase with the number of branches. In Table 3, the accuracy of the 3 branches is better than the accuracy of the 16 branches in VGG16 network. This is caused by overfitting, which occurs in many networks and has been studied in related research.

| TABLE 1. Filter sizes and FLOPs of MBFAN. |
|---|---|---|---|
| n  | Filter size | FLOPs |
|---|---|---|
| 1  | 5x5 | 0.00046 |
| 3  | 3x3 | 0.0015 |
| 16 | 1x1 | 0.10 |
| 32 | 1x1 | 0.20 |

| TABLE 2. Experimental results for a single branch and multiple branches of simple CNN. |
|---|---|---|---|
| Number of branches | 1(5*5) | 3 | 16 | 32 |
| Training accuracy (%) | 76.30 | 76.68 | 76.11 | 75.98 |
| Testing accuracy (%) | 76.86 | 77.59 | 76.86 | 76.48 |
| Initial parameters | 25520 | 25520 | 25520 | 25520 |
| MBFAN additional parameters | 225 | 747 | 49104 | 98208 |

| TABLE 3. Experimental results for a single branch and multiple branches of VGG16. |
|---|---|---|---|
| Number of branches | 1(5*5) | 3 | 16 | 32 |
| Training accuracy (%) | 94.08 | 94.41 | 94.04 | 93.56 |
| Testing accuracy (%) | 93.56 | 93.56 | 93.56 | 93.56 |
| Initial parameters | 39917248 | 39917248 | 39917248 | 39917248 |
| MBFAN additional parameters | 225 | 747 | 49104 | 98208 |
4) Comparing with the initial parameters, with the increase in the number of branches, the number of additional parameters increases slowly.

In Table 3, top1-errors are all between SKNet-29 (top1-error 3.47) and the best result of SKNet-50 (top1-error 20.76) [32]. The reason is that MBFAN structure in this paper is grafted on VGG16, rather than a network specifically designed for a certain dataset.

Considering operational stability of the proposed MBFAN method, we have changed the learning rate from 0.01 to 0.001 in 300th epoch. The result is shown in Figure 4. It can be seen from Figure 4 that the accuracy before 300th epoch tends to change slowly, and the accuracy of the 3-branch case is better than others. After 300th epoch, the 3-branch case is still superior to others, and the accuracy will not change dramatically any more.

C. EXPERIMENTS FOR COINCIDENT CONVOLUTIONAL KERNEL CENTERS

In an image, the location of the convolution-kernel center is important for the computation of convolution. When different filters are applied to different branches, there are two cases. One is that the convolution-kernel centers are located at different points. In this case, the results are shown in Table 2 and Table 3. Another is that the convolution-kernel centers coincide. In this case, the results are shown in Table 4. In this case, the convolution-kernel centers of VGG16 can sometimes be coincident and sometimes not because the padding size cannot satisfy the convolution-kernel centers coincide and same-padding simultaneously. Therefore, only the simple CNN example was applied. For the coincidence of the convolutional kernel centers, the strides and padding size should be calculated using Equation (4). For our experiments, 32 × 32 CIFAR10 was used. The small size of the training image limits the calculation of the coincident convolutional kernel centers in 32 branches case. Thus, 32 branches case is not listed in the following table.

The results in Table 2 are the experiments with non-coincident convolutional kernel centers. Comparing Table 4 with Table 2, the following conclusions can be drawn:

1) Experiments with and without coincident convolutional kernel centers share the same number of parameters.
2) Experiments with coincident convolutional kernel centers are much less accurate than experiments with non-coincident convolutional kernel centers in all the different number of branch cases.

D. INFLUENCE WITHOUT NORMALIZATION

First, we note that this normalization does not mean batch normalization. BN has been applied in all experiments in this study. The normalization in this section refers to the normalization architecture designed in Figure 1 and Equation (3). In the proposed method, this normalization is very important because it makes the different branches interconnected. Without it, the global loss function will not make much sense. The results without this normalization are shown in Table 5 and Table 6.
In Table 2 and Table 3, normalization is applied. Comparing Table 5 and Table 6 with Table 2 and Table 3, the following conclusions can be drawn:

1) Because normalization is useless in single-branch case, the same accuracies are shown for one network with and without normalization.

2) The numbers of parameters are the same for networks with the same number of branches with and without normalization.

3) In Table 5 and Table 6, the accuracies do not always increase with the increase in the number of branches. The testing accuracies of the 3 branches are both better than the accuracies of the 16 and 32 branches in both the simple CNN and VGG16 networks.

4) Experiments without normalization in Table 5 and Table 6 are much less accurate than the experiments with normalization in Table 2 and Table 3 except for the VGG16 with 32 branches. This is caused by the parameter overfitting, which introduces the uncertain computation that we did not expect.

**E. NOTATION**

Table 2 shows that, for the CIFAR10 dataset, MBFAN with 3 branches achieved a better result on the simple CNN of Figure 2. In Table 3, MBFAN with 3 branches also achieved a better result on the VGG16 network of Figure 3. By comparing Table 2 to 6, we can conclude that the accuracy will go down a lot if there are coincident convolutional kernel centers, and the best accuracy of network without normalization is also below the corresponding best accuracy of network with normalization. Of course, the best number of branches can be obtained by more experiments where the branch number can be 5, 6, and so on. However, based on Table 2 and Table 3, we can see that the precision increases very little and the number of parameters is high. Therefore, the other experimental results are not listed in this paper. MBFAN with 3 branches can work well for the CIFAR10 dataset. Of course, for other size image dataset, the different optimal branch number will be obtained in the same way.

**V. CONCLUSION**

In this paper, a convolutional neural network called MBFAN was proposed, which solves the problem of selection of hyperparameters. The proposed MBFAN can be adopted in mature CNNs, such as VGG16. The method of determining the number of branches and the design of convolution filter were given in this paper. The proposed MBFAN was connected to a simple CNN and VGG16 network respectively for experiments. In the experiments, the influence of the number of MBFAN branches, coincidence of MBFAN filter convolution center and normalization on the accuracy was analyzed respectively. Through the experiments, the validity of the MBFAN was proven. When it is applied to a typical deep network, the increase in parameters is not large and the accuracy is much better. The results are achieved thanks to the multi-branch architecture and fuzzy weight design. Despite these advantages, the redundancy of branches and parameters is inevitable. How to further reduce this redundancy remains to be solved in our subsequent studies.

**REFERENCES**

[1] A. Y.-S. Chia, S. Zhuo, R. K. Gupta, Y.-W. Tai, S.-Y. Cho, P. Tan, and S. Lin, “Semantic colorization with Internet images,” ACM Trans. Graph., vol. 30, no. 6, pp. 1–8, Dec. 2011.

[2] A. Criminisi, P. Perez, and K. Toyama, “Object removal by exemplar-based inpainting,” in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., Madison, WI, USA, Jun. 2003, p. 2.

[3] A. Agrawal and R. Raskar, “Optimal single image capture for motion deblurring,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Miami, FL, USA, Jun. 2009, pp. 2560–2567.

[4] S. Setumin and S. A. Saundii, “Difference of Gaussian oriented gradient histogram for face sketch to photo matching,” IEEE Access, vol. 6, pp. 39344–39352, 2018.

[5] S. L. Al-khafaji, J. Zhou, A. Zia, and A.-W. C. Liew, “Spectral-spatial scale invariant feature transform for hyperspectral images,” IEEE Trans. Image Process., vol. 27, no. 2, pp. 837–850, Feb. 2018.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” Commun. ACM, vol. 60, no. 6, pp. 84–90, May 2017.

[7] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. 3rd Int. Conf. Learn. Representations, May 2015, p. 1–14.

[8] M. Mehralian and B. Karasfi, “RDCGAN: Unsupervised representation learning with regularized deep convolutional generative adversarial networks,” in Proc. 9th Conf. Artif. Intell. Robot., 2nd Asia–Pacific Int. Symp., Kish Island, Iran, Dec. 2018, pp. 31–38.

[9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[10] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Honolulu, HI, USA, Jul. 2017, pp. 2261–2269.

[11] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Columbus, OH, USA, Jun. 2014, pp. 580–587.

[13] R. Girshick, “Fast R-CNN,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Santiago, Chile, Dec. 2015, pp. 1440–1448.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 779–788.

[15] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 3431–3440.

[16] G. Toderici, S. M. O’ Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, “Variable rate image compression with recurrent neural networks,” in Proc. Int. Conf. Learn. Representations, San Juan, Puerto Rico, May 2016, p. 1–12.

[17] K. Zhang, X. Liu, J. Zhang, J. He, Y. Shi, and Z. Zhang, “CNN-based macropixel-level up-sampling for plenoptic image coding,” IEEE Access, vol. 7, pp. 80020–80026, 2019.

[18] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “Infogan: Interpretable representation learning by information maximizing generative adversarial nets,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 2180–2188.

[19] Y. Zhang, S. Liu, C. Dong, X. Zhang, and Y. Yuan, “Multiple cycle-in-cycle generative adversarial networks for unsupervised image super-resolution,” IEEE Trans. Image Process., vol. 29, pp. 1101–1112, 2020.

[20] T.-Y. Hsiao, Y.-C. Chang, H.-H. Chou, and C.-T. Chiu, “Filter-based deep compression with global average pooling for convolutional networks,” J. Syst. Archit., vol. 95, pp. 9–18, May 2019.

[21] X. Ye and Q. Zhu, “Class-incremental learning based on feature extraction of CNN with optimized softmax and one-class classifiers,” IEEE Access, vol. 7, pp. 42024–42031, 2019.
[22] W. Chen, Q. Sun, J. Wang, J.-J. Dong, and C. Xu, “A novel model based on AdaBoost and deep CNN for vehicle classification,” IEEE Access, vol. 6, pp. 60445–60455, 2018.

[23] X. Xiong, M. F. Mokbel, and W. G. Aref, “SEA-CNN: Scalable processing of continuous K-Nearest neighbor queries in spatio-temporal databases,” in Proc. 21st Int. Conf. Data Eng. (ICDE), Tokyo, Japan, Apr. 2005, pp. 643–654.

[24] J. Sun, Z. Wu, Z. Yin, and Z. Yang, “SVM-CNN-Based fusion algorithm for vehicle navigation considering atypical observations,” IEEE Signal Process. Lett., vol. 26, no. 2, pp. 212–216, Feb. 2019.

[25] M. D. Zeiler, G. W. Taylor, and R. Fergus, “Adaptive deconvolutional networks for mid and high level feature learning,” in Proc. IEEE Int. Conf. Comput. Vis., Barcelona, Spain, Nov. 2011, pp. 2018–2025.

[26] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in Proc. 13th Eur. Conf. Comput. Vis., vol. 8689, 2014, pp. 818–833.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Boston, MA, USA, Jun. 2015, pp. 1–9.

[28] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 2818–2826.

[29] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in Proc. 31st AAAI Conf. Artif. Intell., San Francisco, CA, USA, Feb. 2017, pp. 4278–4284.

[30] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Honolulu, HI, USA, Jul. 2017, pp. 5987–5995.

[31] C. Cho and S. Lee, “Effective five directional partial derivatives-based image smoothing and a parallel structure design,” IEEE Trans. Image Process., vol. 25, no. 4, pp. 1617–1625, Apr. 2016.

[32] X. Li, W. Wang, X. Hu, and J. Yang, “Selective kernel networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Long Beach, CA, USA, Jun. 2019, pp. 510–519.

[33] T. Takagi and M. Sugeno, “Fuzzy identification of systems and its applications to modeling and control,” IEEE Trans. Syst., Man, Cybern., vol. SMC-15, no. 1, pp. 116–132, Feb. 1985.

[34] K. Shi, J. Wang, Y. Tang, and S. Zhong, “Reliable asynchronous sampled-data filtering of T-S fuzzy uncertain delayed neural networks with stochastic switched topologies,” Fuzzy Sets Syst., vol. 381, pp. 1–25, Feb. 2020.

[35] X. Zhang and Z. Zhao, “Normalization and stabilization for rectangular singular fractional order T-S fuzzy systems,” Fuzzy Sets Syst., vol. 381, pp. 140–153, Feb. 2020.

[36] K. Tanaka and H. O. Wang, Fuzzy Control Systems Design and Analysis: A Linear Matrix Inequality Approach. New York, NY, USA: Wiley, 2001.

FAGUANG WANG received the Ph.D. degree from the School of Mechatronics Engineering, Changwon National University, South Korea, in 2011. He is currently a Lecturer with the School of Information and Control Engineering, China University of Mining and Technology. His research interests include the areas of deep learning and its application in target recognition, location, and grab for robot.

YUE WANG received the B.S. degree in automation from the Qingdao University of Technology, Qingdao, China. She is currently pursuing the M.S. degree in control science and engineering with the China University of Mining and Technology, Xuzhou, China. Her main area of studying includes image recognition based on convolutional neural networks.

HONGMEI WANG received the Ph.D. degree from the School of Mechatronics Engineering, Changwon National University, South Korea, in 2012. She is currently an Associate Professor with the School of Information and Control Engineering, China University of Mining and Technology. Her research interests include the areas of designing system applied for wireless communication modem and various systems required for advanced digital signal processing.

CHAOGANG TANG (Member, IEEE) received the B.S. degree from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2007, and the Ph.D. degree from the School of Information Science and Technology, University of Science and Technology of China, Hefei, China, and the Department of Computer Science, City University of Hong Kong, under a joint Ph.D. Program, in 2012. He is currently an Assistant Professor with the China University of Mining and Technology. His research interests include mobile cloud computing, fog computing, the Internet of Things, big data, and WSN.