Received March 16, 2021, accepted April 21, 2021, date of publication May 3, 2021, date of current version May 18, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3077070

Improving the Performance of Convolutional Neural Networks by Fusing Low-Level Features With Different Scales in the Preceding Stage

XIAOHONG YU, WEI LONG, YANYAN LI, XIAOQIU SHI, AND LIN GAO, (Member, IEEE)
School of Mechanical Engineering, Sichuan University, Chengdu 610065, China

Corresponding author: Yanyan Li (yyl_scu@163.com)

This work was supported by the Science and Technology Department of Sichuan Province under Grant 2020JDRC0026.

ABSTRACT The width of convolutional neural networks (CNNs) is crucial for improving performance. Many wide CNNs use a convolutional layer to fuse multiscale features or fuse the preceding features to subsequent features. However, these CNNs rarely use blocks, which consist of a series of successive convolutional layers, to fuse multiscale features. In this paper, we propose an approach for improving performance by fusing the low-level features extracted from different blocks. We utilize five different convolutions, including 3×3, 5×5, 7×7, $5\times3 \cup 3\times5$ and $7\times3 \cup 3\times7$, to generate five low-level features, and we design two fusion strategies: low-level feature fusion (L-Fusion) and high-level feature fusion (H-Fusion). Experimental results show that the L-Fusion is more helpful for improving the performance of CNNs, and the 5×5 convolution is more suitable for multiscale feature fusion. We summarize the conclusion as a strategy that fuses multiscale features in the preceding stage of CNNs. Furthermore, we propose a new architecture to perceive the input of CNNs by using two self-governed blocks based on the strategy. Finally, we modify five off-the-shelf networks, DenseNet-BC (depth = 40), ALL-CNN-C (depth = 9), Darknet 19 (depth = 19), Resnet 18 (depth = 18) and Resnet 50 (depth = 50), by utilizing the proposed architecture to verify the conclusion, and these updated networks provide more competitive results.

INDEX TERMS Convolutional neural networks, performance, multi-scale features fusion.

I. INTRODUCTION

CNNs (convolutional neural networks) [1] were first presented in 1989, and they have demonstrated excellent performance in many visual tasks such as semantic segmentation [2], [3], image classification [4], and object detection [5], [6]. In particular, as hardware has developed, the performance of CNNs has increased dramatically due to the higher computational capacity of the hardware. However, most CNNs are still not as accurate as a human visual system. In recent years, many efforts have been made to improve the performance of CNNs by regularizing parameters [7], exploiting superior loss functions [8], modifying pooling operations [9], and designing more meaningful network architectures [10].

Some classic models have validated that the depth of a CNN is pivotal for its performance [11], [4]. Furthermore, many visual recognition tasks have benefitted from very deep networks [12], [13]. In many cases, a considerably deeper network indeed achieves better results than a shallower network, and we can easily obtain a higher-quality model by increasing the depth. However, a very deep CNN has many disadvantages. First, a deeper network will result in vanishing or exploding gradients [14], which can lead to a disconvergence of the results. In addition, a deeper network is associated with more parameters. In particular, in the late stages of networks, the number of parameters massively increases as the convolution channels increase, which increases the additional computational costs for these massive numbers of parameters. Moreover, more stacked layers hamper the performance of networks due to overfitting. Specifically, the model has a lower training error, but it has a higher testing error. Additionally, as the network depth increases, the accuracy becomes saturated and degrades rapidly [12].

To solve the problems caused by depth, [11] introduced an ''inception module'' to increase the width of a CNN and considered that visual information should be abstracted at

The associate editor coordinating the review of this manuscript and approving it for publication was Jan Chorowski.

various scales to obtain multiscale features. The ''inception module'' is computationally less expensive because it can utilize a set of small-scale convolutions to replace a larger-scale convolution in a layer. In a wide dense block [13], the features of all preceding layers are used as inputs into all subsequent layers. In some sense, increasing the width can be regarded as feature fusion since it fuses both the previous and posterior features of some blocks in stages. In a wide residual block [12], the inputs of a subsequent layer are extracted from not only its preceding layer but also from other layers preceding that layer by using shortcut connections. The two types of blocks increase their width and fuse the preceding features to their subsequent features, which dramatically enhances the performance of their respective networks. In object detection, some high-performance object detection architectures [40] harness high-resolution features with small scales, and some larger-scale features are extracted from a shallow layer. Due to leveraging the better localization properties of low-level features, these architectures markedly improve the location accuracy and recall rate.

Based on the merits of increasing the width of CNNs, we propose a novel method that adds an auxiliary block to the different blocks of a CNN to increase the width. The auxiliary block can extract different scale low-level features, which are concatenated with the different level features extracted from different blocks of the CNN. Our method can be summarized as feature fusion between different scale low-level features and different level features. Based on the feature fusion form, we design two fusion strategies—low-level feature fusion (L-Fusion) and high-level feature fusion (H-Fusion)—and five scale low-level features, which are implemented by two CNNs (Net 1 and VGG-16-V). We assess the performance of these networks on CIFAR10 and CIFAR100. Finally, we validate our conclusions on DenseNet-BC [13] (depth = 40), ALL-CNN-C [15] (depth = 9), Darknet 19 [42] (depth = 19), Resnet 18 [12] (depth = 18) and Resnet 50 [12] (depth = 50).

The contributions of this paper can be summarized as follows.

1) L-Fusion, which concatenates the low-level features extracted from Auxi-Block with the low-level features extracted from BaseNet, can efficiently enhance the performance of networks, regardless of whether the low-level features extracted from Auxi-Block are the same scale as the low-level features extracted from BaseNet.
2) In terms of feature fusion, the nonsquare convolutions $5 \times 3 \cup 3 \times 5$ and $7 \times 3 \cup 3 \times 7$ are not remarkably better than the square convolutions 3×3, 5×5 and 7×7.
3) Owing to the fewer parameters of the 5×5 convolution comparing the 7×7 convolution or a larger size convolution, we select the 5×5 convolution to extract different scale features for L-Fusion. Using L-Fusion, we modify the architectures of DenseNet-BC (40), ALL-CNN-C (9), Darknet (19), Resnet (18) and Resnet (50) to verify our conclusions. The experimental results show that the performances of the five modified networks are more competitive.
4) Only for our method does channelwise concatenation perform better than elementwise summation. Furthermore, the Concatenation + CR operations are more applicable.

The remainder of this paper is organized as follows: In section 2, we provide an overview of the related works, focusing on different modalities for improving the performance of CNNs. In section 3, we present the details of our proposed methods. In section 4, we provide the details of our experimental procedures, results and analysis. Section 5 concludes this paper.

II. RELATED WORK

Currently, CNNs have a typical structure: stacked convolutional layers are followed by one or more fully connected layers. More precisely, before going across a convolutional layer, the input parameters will implement two consecutive operations: batch normalization [7] followed by linear activation (ReLU) [16]. To reduce the dimensionality and decrease the overfitting of a large network, downsampling will be necessarily employed after a linear activation layer. Because of the typical structure in conjunction with the principle of backward propagation, there are three principal research directions to enhance the performance of deep CNNs:

1) optimizing the convolution operation and pooling operation, 2) modifying the activation function and loss function, and 3) increasing the depth and width.

A. OPTIMIZING THE POOLING OPERATION AND CONVOLUTION OPERATION

In CNNs, convolution is the most basic operation, and a pooling layer always follows a certain convolutional layer. These layers are very important to extract semantic features and reduce the parameters of the model. Based on the sparsity of the activations in a pooling region, [9] proposed a sparsity-based stochastic pooling mechanism that integrated the advantages of max pooling, average pooling, and stochastic pooling for CNNs to improve the recognition accuracy. According to the observation that a ranking list was invariant when the activation operation changed the values of a pooling region, [17] compared three new pooling methods, including rank-based average pooling, rank-based weighted pooling and rank-based stochastic pooling, with the conventional pooling operation on four benchmark image datasets, and the results showed that rank-based pooling outperformed the existing pooling methods in classification performance. Due to the existence of redundancy in a standard convolution, which can be divided into spatial and channel domains, [18] achieved superior performance by relaxing the sparsity of the convolution in the spatial domains and reducing the redundancy in the channel domains. Replacing the conventional convolutional layer with a multilayer perceptron, [19] designed a novel deep network by stacking multiple perceptrons and a global average pooling layer. Reference [20]

proposed replacing the dense shallow multilayer perceptron with a sparse shallow multilayer perceptron to decrease the number of parameters and extract local features in channel domains and in spatial domains. To make full use of the spatial features and temporal features for video action recognition, [44] designed trajectory pooling and line pooling to fuse spatial and temporal information.

B. MODIFYING THE ACTIVATION FUNCTION AND LOSS FUNCTION

The activation function is crucial for a neural network since it determines whether a neuron works, and the loss between the output value of the network and the label value is very important for backpropagation. Many research efforts have been made in these two respects to improve network performance. Reference [21] improved the fine-grained image classification accuracy by applying a generalized large-margin loss to AlexNet, GoogLeNet and VGG. Reference [22] proposed a deep CNN with an associated objective function that consisted of a max-margin objective, a max-correlation objective and a correntropy loss for a minimum score of positive labels, a latent semantic space and a minimal training loss. To eliminate the requirement of carefully tuning the learning rate to prevent exploding gradients, [23] designed a multitask loss function to conduct joint training for classification and a bounding box regression. Reference [24] designed a fast exponentially linear unit with a rectified linear unit (ReLU) and an exponential linear unit (ELU). It used the ELU on the negative part and the ReLU on the positive part to accelerate the calculation speed and improve the robustness of the network. Reference [25] employed different activation functions, including the rectified linear unit (ReLU), leaky ReLU (LReLU), parametric ReLU (PReLU) and exponential linear unit (ELU), in different convolutional layers to extract better information.

C. INCREASING THE DEPTH AND WIDTH

In 2012, researchers began to study the importance of the depth (the number of layers) of CNNs. By using an ablation study and occlusion experiment, [26] demonstrated that the depth of networks, rather than any individual section, was vital to their performance. Reference [27] increased the depth by using a set of subnetworks to extract complementary features and the same number of classifiers based on random projections.

The width (the number of units of each layer) is as important as the depth for the performance of CNNs. Reference [10] introduced a novel network form by stacking a series of ''inception modules'', which consisted of a group of different scale convolutions in one layer. This approach increased both the depth and the width of the network, which improved the classification performance. Subsequently, an increasing number of researchers have studied how the width of a network influences network performance, and the ''inception module'' has been continuously upgraded. By factorizing convolutions of the original ''inception module'' [10] and adding batch normalization to diminish the computational cost, [52] proposed two modified architectures: Inception v2 and Inception v3. Reference [53] constructed the Inception v4 architecture by simplifying the architecture of Inception v3 and adding more ''inception modules''.

Reference [12] added a skip connection that bypassed the nonlinear transformation with an identity shortcut: $X_l = H(X_{l-1}) + X_{l-1}$. Reference [13] introduced a dense connection that connected any layer to all subsequent layers: $X_l = H([X_0, X_1, \ldots, X_{l-1}])$, where X_l denotes the features extracted from the ℓ^{th} layer (or the input feature if the layer is input layer); $H(\cdot)$ represents the output of a composite function of some consecutive operations, such as batch normalization, ReLU, and convolution; and $[X_0, X_1, \ldots, X_{l-1}]$ represents the concatenation of the features abstracted by layers $0, 1, \ldots, l-1$. To improve the adaptation of the inception architectures to new tasks, [48] exploited a split–transform–merge strategy to design a new module. The module divided the channels of a residual unit [12] into a number of channel sets with the same topology and then summed the outputs of these sets. Reference [49] designed a ''res2net module'' that replaced the 3×3 filters of a residual unit with smaller filter groups. The receptive fields of these filter groups were different; hence, the ''res2net module'' could obtain multiscale representations. Reference [50] proposed a squeeze-and-excitation (SE) block that selectively emphasized informative features and suppressed fewer useful features by using a feature recalibration mechanism. The mechanism obtained the channelwise weights of the input of an SE block by using global average pooling and a gating mechanism and then output features by using channelwise multiplication between the input and the weights. To dispose of the difficulties caused by increasing the depth and width of CNNs, [51] designed a family of ''polyinceptions'' to improve the performance from the perspective of exploring structural diversity. These ''polyinceptions'' replaced a residual unit [12] with a polynomial inception unit. Furthermore, [51] introduced two ''inception-resnet units'' by combining the inception architecture with residual connections.

Inspired by the human retina mechanism, [28] suggested a coupled convolutional layer that consisted of a set of mutually constrained convolutions. Reference [29] used a fully convolutional two-stream fusion network to extract deep features from input images and user interactions individually, which achieved better image segmentation performance. Instead of using large-scale convolutions to construct a deep CNN directly, [30] proposed a series of cascaded subpatch convolutions that included a small-scale convolution and a 1×1 convolution, and the new architecture was more robust for classification due to the merits of the cascaded subpatch convolutions. Reference [31] proposed a wide multiscale contrast network that was composed of three networks with identical structures and three subnetworks to extract multiscale and multilevel features for salient object detection. Reference [32] used two 1×1 convolutions with different channels to generate two different channel-domain features,

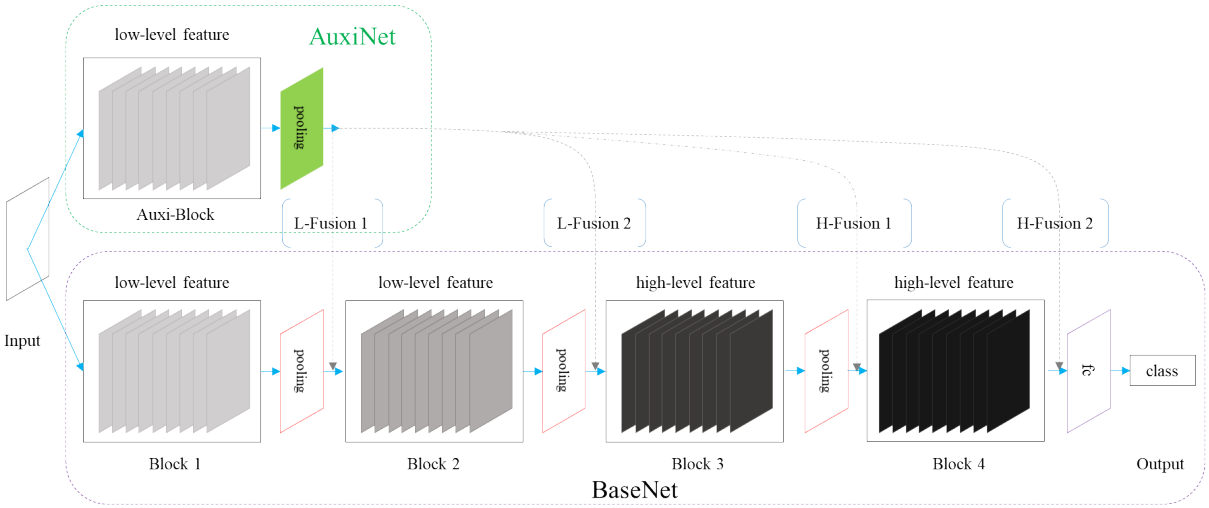


FIGURE 1. We build BaseNet and AuxiNet for multiscale feature fusion. BaseNet consists of 4 blocks, and each block has many stacked convolution layers without downsampling. Namely, the features generated by every convolution layer in a block have the same size. AuxiNet consists of only one block, which is exploited to extract the low-level features at different scales. We define two feature fusion strategies, L-Fusion and H-Fusion, which fuse the low-level features extracted from AuxiNet to the low-level features and the high-level features extracted from BaseNet, respectively.

and then the two features were propagated to multiply spatial selective modules to obtain different spatial-domain information. Based on the relevance of multiple modalities, [41] extracted multiple features at several different convolutional layers from different modality-specific CNNs by exploiting modality-dedicated embedding layers, and then weighted feature fusion was employed for biometric identification. Reference [43] constructed a deep multitask learning framework by fusing certain intermediate layers of an off-the-shelf CNN to optimize a multitask learning algorithm, and the fused network performed better than an individual learning network without feature fusion. To maintain a spatially more accurate heatmap, [54] proposed a new architecture called a high-resolution network (HRNet) by combining a set of high-to-low subnetworks with different resolutions in parallel. Between these subnetworks, different resolution information could be fused repeatedly; inside each subnetwork, every convolutional layer had the same resolution.

III. PROPOSED METHOD

To decrease the number of network parameters, the current prevailing design of CNNs implements a downsampling operation (such as max pooling and average pooling) after some convolutional layers to obtain downsampled features. We divide a CNN into different blocks on the basis of downsampling. If we use downsampling at an early or a late stage of the network, we will obtain a shallow feature (low-level) or deep feature (high-level), respectively. As shown in Fig. 1, BaseNet consists of 4 blocks. We treat the features in a block, which is a series of stacked consecutive convolutional layers, without downsampling as the same size; we regard the features as the same scale if they are extracted from the same scale convolutions. The features extracted by Block 1 or Block 2 are the shallow features (low-level), and the features extracted by Block 3 or Block 4 are the deep features (high-level).

A. DIFFERENT FEATURE FUSION STRATEGIES

To some extent, increasing the width of the network can be regarded as feature fusion because the features extracted by many different convolutions in one layer or by multiple parallel subnetworks will finally be fused. The level features are important for classification tasks. In every block of a network, the features produced by preceding convolutional layers can be fused to the features extracted from the subsequent layers to improve the robustness of the network, which has been verified by [12] and [13]. To study the influence of fusing low-level features with high-level features, we design four different feature fusion strategies, which are shown in Fig. 1. We add another block that can generate different scale low-level features using different scale convolutions. For the sake of clarity, we call the adding block Auxi-Block, which is shown in Fig. 1. Auxi-Block has the same number of convolution layers as Block 1, which is employed to extract low-level features, but the scale of the convolution in each layer of Auxi-Block is different from Block 1. Hence, as shown in Fig. 1, we can fuse the features extracted from Auxi-Block to the low-level features and high-level features extracted from Block 1, Block 2, Block 3 and Block 4. We call the four aforementioned fusions L-Fusion 1, L-Fusion 2, H-Fusion 1, and H-Fusion 2. These four fusion strategies not only increase the width of the network but also use multiscale features, which intuitively enhance the performance of CNNs.

We define an input feature $\boldsymbol{X} \in R^{W\times H\times D}$, where W represents the width, H represents the height and D represents

the channels of the feature. We define the filter of the first convolution in a block as $K_1 \in R^{(K\times K\times D)\times C}$, and we define the remaining filters of the block as $K_l \in R^{(K\times K\times C)\times C}$, where $l \in 2, 3, \ldots, n$ represent the layers, C represent the channels and K represents the kernel size. When $\boldsymbol{X}$ passes through a block (a sequence of convolution layers), the output of every layer can be computed by

$$\left|\begin{array}{l} X_1 = F[Conv2d\ (\boldsymbol{X} * K_1) + b_1] \ \in R^{W\times H\times C} \\ X_2 = F[Conv2d\ (X_1 * K_2) + b_2] \ \in R^{W\times H\times C} \\ \qquad\qquad \ldots\ldots \\ X_l = F[Conv2d\ (X_{l-1} * K_l) + b_l] \in R^{W\times H\times C} \end{array}\right| \tag{1}$$

When X_l passes through a downsampling layer with $stride = S$, the output is $X'_l \in R^{C\times(W\times H)/S}$. In Fig. 1, the output of Block i is $O_i \in R_i^{W_i\times H} \times C_i$, where $i \in 1, 2, 3, 4$. In addition, the output of Auxi-Block is $O_a \in R^{W_a\times H_a\times C_a}$. L-Fusion and H-Fusion refer to the concatenation of the features produced by different blocks. We can implement L-Fusion 1 directly as the same size features, but we must construct a translation layer including a 1×1 convolution and $S \times S$ pooling to translate the size and channels of the low-level features for concatenation. Therefore, the output of L-fusion and H-fusion can be expressed as Equation (2), where *cat* in the four formulas implies concatenation.

$$\left|\begin{array}{l} O_{L-Fusion1} = cat\ (O_a, O_1) \ \in R^{W_1\times H_1\times(C_a+C_1)} \\ O_{L-Fusion2} = cat\ (O_a, O_2) \ \in R^{W_2\times H_2\times(C_a+C_2)} \\ O_{H-Fusion1} = cat\ (O_a, O_3) \ \in R^{W_3\times H_3\times(C_a+C_3)} \\ O_{H-Fusion2} = cat\ (O_a, O_4) \ \in R^{W_4\times H_4\times(C_a+C_4)} \end{array}\right| \tag{2}$$

Take L-Fusion 1 as an example. We hypothesize that Auxi-Block and Block 1 have $L > 1$ layers, from the 1st layer to the Lth layer. Forward propagation can be computed in (3), as shown at the bottom of the page.

When the outputs of Auxi-Block and Block 1 are concatenated, the fusion feature can be computed by Equation (4). Concatenation means that one weight matrix is stacked on another weight matrix; therefore, the loss of backward propagation influences X_L and X'_L independently. Namely, the loss is propagated back to the input in two pipelines, and the input is perceived by two self-governed blocks rather than a single block.

$$Fusion = F\left[cat(X_L, X'_L) * w + b\right] \tag{4}$$

B. MULTISCALE FEATURES

The convolution operation is a crucial element of deep learning structures since a number of filters slide across the input image [33]; hence, the filters are pivotal for the convolution operation. The larger a scale filter is, the larger the receptive field is. In some cases, increasing the scale of the convolution filter indeed improves the classification accuracy. For instance, many CNNs use a large filter, such as 11×11 [4] and 7×7 [5], in the first convolutional layer to increase the receptive field. However, the large scale means more parameters. For example, a single 7×7 convolution requires $7^2C^2 = 49C^2$ parameters, whereas a single 3×3 convolution requires only $3^2C^2 = 9C^2$ parameters, where C is the number of channels of the convolution. Additionally, [11] considered that a stack of two 3×3 convolutional layers (without spatial pooling in between) has an effective receptive field of 5×5, and three 3×3 convolutional layers have an effective receptive field of 7×7. Hence, the 3×3 convolution is better than the 7×7 convolution or a larger scale convolution, and it is the most common convolution in CNNs.

There are few networks that consist of a set of larger scale convolutional layers entirely, such as 7×7 and 9×9. However, visual information with various scales helps improve the performance of CNNs. Therefore, we cannot discard large-scale convolutions. We select 5 convolution scales, including $3 \times 3, 5 \times 5, 7 \times 7, 5 \times 3 \cup 3 \times 5$ and $7 \times 3 \cup 3 \times 7$, to extract the different scale features. We use the nonsquare convolutions $5 \times 3 \cup 3 \times 5$ and $7 \times 3 \cup 3 \times 7$, where $\cup$ means implementing the 3×5 convolution and 5×3 convolution in parallel, based on the observation that the objects in an image can be rectangular.

Due to the small number of convolutional layers and the few channels in each convolutional layer in the preceding stages or blocks before the first downsampling, we stack a series of convolutional layers with a large-scale filter to construct a block rather than utilizing a single large-scale convolutional layer. The block not only ensures that we extract different scale features but also adds fewer parameters, which makes it the most different from other methods.

IV. EXPERIMENTAL RESULTS

In the first stage, according to the aforementioned definitions in section 3, we design two networks, Net 1 and VGG-16-V, based on the architectures of [13] and [11], respectively, to address the two important problems: 1) whether the fusion of multiscale features achieves better performance and 2) which scale feature is more effective. The configurations of the two networks and the feature fusion strategies are shown in Fig. 2 and Fig. 3, respectively. There are four

$$\left|\begin{array}{ll} X_1 = F\left[Conv2d\ (\boldsymbol{X} * K_1) + b_1\right] & X'_1 = F\left[Conv2d\ \left(\boldsymbol{X} * K'_1\right) + b'_1\right] \\ X_2 = F\left[Conv2d\ (X_1 * K_2) + b_2\right] & \\ & X'_2 = F\left[Conv2d\ \left(X'_1 *'_2\right) + b'_2\right] \\ \vdots & \vdots \\ X_L = F\left[Conv2d\ (X_{L-1} * K_L) + b_L\right] & X'_L = F\left[Conv2d\ \left(X'_{L-1} * K'_L\right) + b'_L\right] \end{array}\right| \tag{3}$$

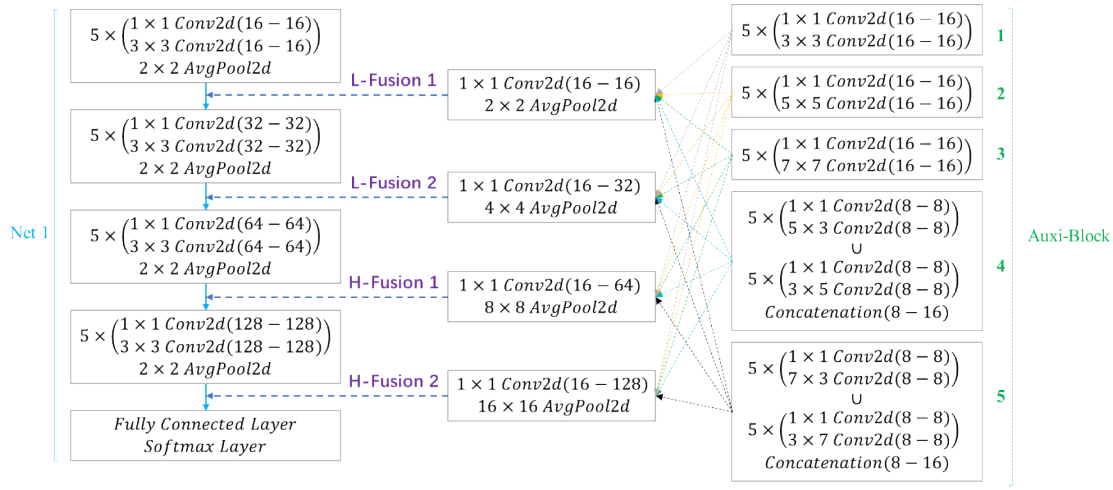


FIGURE 2. Feature fusion strategies for Net 1. We select Auxi-Blocks 1, 2, 3, 4 and 5 in turn to extract five different scale features. Then, we use every feature to implement four feature fusion strategies. We ignore the first convolution layer of Net 1 and Auxi-Block.

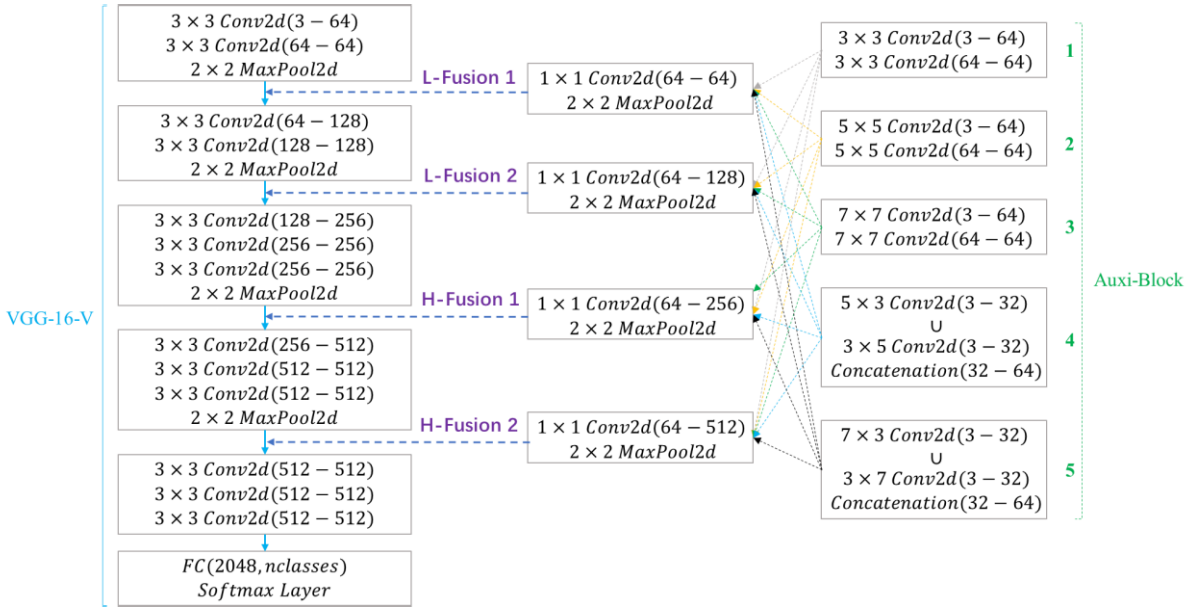


FIGURE 3. Architecture of VGG-16-V, which is a variant of VGG-16. We also design the same features with different scales and the same fusion strategies as Net 1. Note that the "Conv2d" layer shown in the figure corresponds to the sequence BN-ReLU-Conv2d.

feature fusion strategies at every scale, as shown in Fig. 1. Hence, there are 20 feature fusion experiments in total for a network. In the second stage, we select five networks, DenseNet-BC (depth = 40), ALL-CNN-C (depth = 9), Darknet 19 (depth = 19), Resnet 18 (depth = 18) and Resnet 50 (depth = 50), to verify the conclusions generated in the first stage.

A. DATASETS

We evaluate the proposed fusion strategies in the first stage on two standard benchmark datasets [36]: CIFAR10 and CIFAR100. These datasets both contain 50K training images and 10K test images, but they consist of 10 categories and 100 categories, respectively. We use all the 50K training images for training without validation during the training stage, and the 10K test images are used for testing during the testing stage. We normalize the data using the channel means and standard deviations as in [46]. During training, we adopt a data augmentation scheme with random cropping, random horizontal flips and normalization [47], which has been widely used for the two datasets [11], [12], [19] to obtain two augmented datasets that we call CIFAR10+ and CIFAR100+, respectively. During testing, we only normalize the data by using the channel means and standard deviations [47].

We adopt the ILSVRC 2012 classification dataset [45], which consists of 1.2 million images for training and 50,000 for validation from 1,000 classes, to further validate

TABLE 1. Configurations of Net 1, Net 2 and Net 3. Note that the "Conv2d" layer shown in the table corresponds to the sequence BN-ReLU-Conv2d except for the first convolutional layer. We build Net 2 and Net 3 to determine whether the large scale of the feature itself or the fusion strategy of multiscale features attains better performance. We only modify the first convolutional layer and the structure of Block 1 to construct Net 2 and Net 3. With the application of a large-scale convolution, the receptive fields of Net 2 and Net 3 are larger than that of Net 1.

Layers	Output Size	Net 1	Net 2	Net 3
First convolution	32×32	3×3 Conv2d	5×5 Conv2d	7×7 Conv2d
Block 1	32×32	$5\times\begin{pmatrix}1\times1\ \text{Conv2d}\\3\times3\ \text{Conv2d}\end{pmatrix}$	$5\times\begin{pmatrix}1\times1\ \text{Conv2d}\\5\times5\ \text{Conv2d}\end{pmatrix}$	$5\times\begin{pmatrix}1\times1\ \text{Conv2d}\\7\times7\ \text{Conv2d}\end{pmatrix}$
Transition	32×32	1×1 Conv2d		
Layer 1	16×16	AvgPool2d		
Block 2	16×16	$5\times\begin{pmatrix}1\times1\ \text{Conv2d}\\3\times3\ \text{Conv2d}\end{pmatrix}$		
Transition	16×16	1×1 Conv2d		
Layer 2	8×8	AvgPool2d		
Block 3	8×8	$5\times\begin{pmatrix}1\times1\ \text{Conv2d}\\3\times3\ \text{Conv2d}\end{pmatrix}$		
Transition	8×8	1×1 Conv2d		
Layer 3	4×4	AvgPool2d		
Block 4	4×4	$5\times\begin{pmatrix}1\times1\ \text{Conv2d}\\3\times3\ \text{Conv2d}\end{pmatrix}$		
Classification Layer	2×2	AvgPool2d		
	1×1	Fully connected Softmax		

the effectiveness of our proposed method. We use the training set during the training stage and report the classification errors on the validation set. During training, we randomly crop the size of the training images to 224 × 224 and exploit random horizontal flips and normalization (mean = [0.485,0.456,0.406], std = [0.229,0.224,0.225]). We do not leverage scale augmentation and standard color augmentation. During testing, we adopt standard single-crop testing and apply a center crop to resize the validation images to 224 × 224 only.

B. TRAINING

We implement these proposed networks on the PyTorch framework and two NVIDIA GeForce RTX 2080 GPUs. The weight initialization strategy is also introduced in [37]. For CIFAR10 and CIFAR100, all the networks are trained using stochastic gradient descent (SGD), cross-entropy loss and the ReLU. The weight decay, momentum and initial learning rate are set to 0.0001, 0.9 and 0.1, respectively. All the models are trained for 300 epochs, the learning rate is divided by 10 at the 150th and 225th epochs, and the batch size is 256. Based on Darknet 19 [42], Resnet 18 [12] and Resnet 50 [12], we implement a fusion operation on the ILSVRC 2012 classification dataset and train the three networks and their correspondingly modified networks only for 90 epochs with stochastic gradient descent (SGD), cross-entropy loss and ReLU. For Darknet 19 [42], leaky ReLU (negative slope equals 0.1) replaces ReLU. The initial learning rate is set to 0.1 and is divided by 10 at the 30th, 60th and 75th epochs. The weight decay and momentum are set to 0.0001 and 0.9, respectively. Due to the limitation of the GPU memory, the batch size is set to 64.

TABLE 2. Classification error (%) and parameters (M) for CIFAR10+ and CIFAR100+ using Net 1, Net 2 and Net 3. Because of employing a larger scale convolution, Net 2 and Net 3 have more parameters than Net 1, but Net 2 and Net 3 have lower classification accuracy. Hence, for the CIFAR dataset, a larger scale convolution is not necessary for improving the performance of CNNs.

Method	CIFAR10+		CIFAR100+	
	Error	Params	Error	Params
Net 1	9.82	1.11	37.50	1.12
Net 2	9.83	1.12	41.22	1.14
Net 3	10.49	1.16	41.17	1.17

C. CLASSIFICATION RESULTS OF NET 1 AND VGG-16-V

We construct two CNNs, Net 1 and VGG-16-V, to implement the feature fusion strategies. The input image size is 32 × 32 and is RGB, which is the same as the images of CIFAR10 and CIFAR100.

Net 1, which is inspired by the architecture of [19], consists of 45 convolutional layers, and we build its variants Net 2 and Net 3. One of our purposes of this paper is to study the advantage of multiscale feature fusion, but we also seek to answer whether large-scale feature or multiscale feature fusion increases performance more. Therefore, we construct Net 2 and Net 3 to observe the performance of the large features. Their configurations are shown in Table 1. Except for the different convolutions in the first block, the other settings are the same in Net 1, Net 2 and Net 3. The convolutions of the first block are 3 × 3, 5 × 5 and 7 × 7, respectively. The results are shown in Table 2.

We construct another network based on VGG-16 [11], which is a famous network that has been widely utilized for

TABLE 3. Parameters (M) on CIFAR10+ and CIFAR100+. Based on Net 1, we compare the four fusion strategies with the five features extracted from different Auxi-Blocks. Fusing Auxi-Block into Net 1 does not dramatically increase the number of parameters, even if Auxi-Block uses a larger scale convolution: 7 × 7 convolution.

Method	CIFAR10+					CIFAR100+				
Auxi-Block	3×3	5×5	7×7	5×3U3×5	7×3U3×7	3×3	5×5	7×7	5×3U3×5	7×3U3×7
L-Fusion 1	1.21	1.23	1.26	1.24	1.27	1.24	1.27	1.30	1.27	1.29
L-Fusion 2	1.21	1.23	1.26	1.24	1.27	1.25	1.27	1.30	1.27	1.29
H-Fusion 1	1.21	1.23	1.26	1.24	1.26	1.25	1.27	1.30	1.28	1.29
H-Fusion 2	1.21	1.23	1.26	1.24	1.26	1.25	1.28	1.30	1.28	1.29

TABLE 4. Classification error (%) on CIFAR10+ and CIFAR100+. The results that are better than those of Net 1 are in blue and bolded. All L-Fusions improve the performance of Net 1. However, most H-Fusions decrease the classification accuracy of Net 1.

Method	CIFAR10+					CIFAR100+				
Auxi-Block	3×3	5×5	7×7	5×3U3×5	7×3U3×7	3×3	5×5	7×7	5×3U3×5	7×3U3×7
L-Fusion 1	**9.23**	**8.70**	**9.51**	**8.74**	**9.10**	**35.49**	**36.08**	**36.97**	**36.40**	**35.16**
L-Fusion 2	**9.01**	**9.45**	**8.78**	**8.71**	**8.81**	**35.47**	**34.96**	**34.71**	**34.22**	**35.20**
H-Fusion 1	11.47	10.69	11.20	10.40	10.04	37.87	37.67	**36.94**	**35.59**	**36.64**
H-Fusion 2	14.54	16.04	15.26	14.04	14.84	46.36	44.44	44.68	39.94	42.29

TABLE 5. Parameters (M) on CIFAR10+ and CIFAR100+. Using VGG-16-V, we compare the four fusion strategies with five types of features. We find that H-Fusion has more parameters than L-Fusion, and the increase in parameters is more obvious as the number of channels increases.

Method	CIFAR10+					CIFAR100+				
VGG-16-V	14.74					14.93				
Auxi-Block	3×3	5×5	7×7	5×3U3×5	7×3U3×7	3×3	5×5	7×7	5×3U3×5	7×3U3×7
L-Fusion 1	14.86	14.93	15.03	14.96	15.00	15.05	15.11	15.21	15.14	15.19
L-Fusion 2	15.09	15.15	15.26	15.18	15.23	15.27	15.34	15.44	15.37	15.42
H-Fusion 1	15.98	16.05	16.15	16.09	16.13	16.16	16.23	16.34	16.27	16.32
H-Fusion 2	17.18	17.24	17.34	17.30	17.35	17.36	17.43	17.53	17.48	17.54

other CNNs, such as in [34] and [35]. Because of its good classification performance, we chose VGG-16 to implement the proposed methods. We use only one fully connected layer rather than three fully connected layers, and we call this VGG-16 variant VGG-16-V, as shown in Fig. 3.

Based on the Net 1 and VGG-16-V architectures, we design four fusion strategies, and each strategy leverages five features at five scales. The details of the two networks and configurations are shown in Fig. 2 and Fig. 3, respectively. We select the 3×3 convolution; two larger size convolutions, 5×5 and 7×7; and two unconventional convolutions, $5 \times 3 \cup 3 \times 5$ and $7 \times 3 \cup 3 \times 7$. We do not use much larger convolutions, such as 9×9 and 11×11, due to the considerable increases in the numbers of parameters.

Table 4 shows the classification errors of the four fusion strategies when the features extracted from different Auxi-Blocks are concatenated with the features generated from the different blocks of Net 1. Comparing the second column and the fourth column of Table 2, we find that all L-Fusion strategies improve the performance for both CIFRA10+ and CIFRA100+. On CIFAR10+, the best result shows that we reduce the error by 1.12% by adding the 5×5 convolution. On CIFRA100+, we improve the accuracy by 2.79% compared to the best result by using the 5×5 convolution. However, in terms of most H-Fusion strategies, multiscale feature fusion has no effect. Furthermore, the H-Fusion strategies lead to much worse results.

For VGG-16-V, the results of the experiments are shown in Table 6. Similarly, we find that all L-Fusion strategies can improve the performance for both CIFRA10+ and CIFRA100+. Furthermore, most H-Fusion strategies lead to poor performance. In terms of the best result, on CIFRA10+, we reduce the error by 1.59% by using the $7 \times 3 \cup 3 \times 7$ convolution. In addition, on CIFRA100+, we improve the accuracy by 3.3% compared to the best result by adding the $7 \times 3 \cup 3 \times 7$ convolution.

D. FUSION OPERATION

From Table 4 and Table 6, we find that all L-Fusion strategies result in better performance. Specifically, multiscale low-level feature fusion can remarkably improve the performance of CNNs. We consider these results to have very important statistical meaning. Moreover, we do not find that the non-square convolutions, $5 \times 3 \cup 3 \times 5$ and $7 \times 3 \cup 3 \times 7$, are notably better than the square convolutions, 3×3, 5×5 and 7×7. To take advantage of multiscale feature fusion and to keep as few parameters as possible, we select only the 5×5 convolution to extract different low-level features for feature fusion.

TABLE 6. Classification error (%) on CIFAR10+ and CIFAR100+. The results that are better than those of VGG-16-V are in blue and bolded. The results show the same pattern as Table 4. All L-Fusions are beneficial for performance, but most H-Fusions hinder performance.

Method	CIFAR10+					CIFAR100+				
VGG-16-V	7.83					29.76				
Auxi-Block	3×3	5×5	7×7	5×3U3×5	7×3U3×7	3×3	5×5	7×7	5×3U3×5	7×3U3×7
L-Fusion 1	**7.21**	**6.67**	**6.71**	**6.70**	**6.28**	**26.66**	**27.67**	**27.77**	**27.01**	**26.29**
L-Fusion 2	**7.70**	**6.95**	**6.86**	**7.03**	**6.24**	**27.29**	**28.84**	**28.37**	**28.22**	**26.46**
H-Fusion 1	9.55	9.18	9.63	9.91	8.58	31.53	32.06	33.52	**28.58**	29.83
H-Fusion 2	12.07	8.39	11.37	11.56	**6.39**	31.45	31.76	31.77	33.22	30.41

TABLE 7. CR denotes a nonlinear activation operation, Conv2d (1 × 1)-ReLU; Sum represents an elementwise summation and Cat means a channelwise concatenation. All the results are implemented in the VGG-16-V and L-Fusion 1 architectures discussed in subsection C.

Method	VGG-16-V: L-Fusion 1							
Dataset	CIFAR10+				CIFAR100+			
CR		√		√		√		√
Sum			√	√			√	√
Cat	√	√			√	√		
Params	14.93	14.95	14.86	14.86	15.11	15.13	15.04	15.04
Error	6.67	**6.29**	7.02	**6.32**	27.67	**27.08**	28.57	**27.73**

As we discussed above, feature fusion is very important for enhancing the performance of CNNs. In all the aforementioned experiments, we apply channelwise concatenation to perform the feature fusion operation. Nevertheless, there is another principal fusion operation: channelwise summation. In this study, we harness the two operations to research which style is more suitable for the method we proposed and implement the two operations in VGG-16-V. Simultaneously, we emulated a multilayer perceptron (MLP) introduced by [19]. Following two fusion operations, we also implement a nonlinear activation operation: a 1 × 1 convolutional layer followed by a ReLU function. The results of the experiments are shown in Table 7.

As depicted in Table 7, by integrating the CR operation, concatenation and summation were observed to improve performance. This result illustrated that the CR operation is helpful for improving performance. Moreover, the concatenation outperforms the elementwise summation, especially on CIFAR100+. We considered that the features extracted by different blocks are discriminative and independent. Elementwise summation will mutually disturb features, while concatenation can maintain the independence between features. Hence, we suggest using Concatenation + CR fusion operations and implementing these operations on the subsequent verification networks.

E. CLASSIFICATION RESULTS OF FIVE VERIFICATION NETWORKS

To further support our hypotheses, we verify the fusion strategy on DenseNet-BC (depth = 40) [13] and ALL-CNN-C (depth = 9) [15] due to their different depths and high classification accuracy. We select the CIFAR dataset to evaluate the classification performance. The configurations of the two verification networks are shown in Table 8, and the results are shown in Table 9. Because the transition layer of Block 1 of DenseNet-BC has the same function as Concatenation + CR fusion operations, we delete it.

TABLE 8. We modify the architectures of DenseNet-BC (depth = 40) and ALL-CNN-C (depth = 9) by adding the same number of convolutional layers (the bold parts) in the first stage of the two networks. We select the 5 × 5 convolution to obtain large-scale features and then implement the L-Fusion strategy. We concatenate the features generated by Block 1 and Auxi-Block and then implement Concatenation + CR fusion operations.

DenseNet-BC(κ=12)	ALL-CNN-C
Input size: 32×32	Input size: 32×32
3×3 Conv2d 6× (1×1 Conv2d, 3×3 Conv2d) **5×5 Conv2d** **6× (1×1 Conv2d, 5×5 Conv2d)**	3×3 Conv2d, **5×5 Conv2d** 3×3 Conv2d, **5×5 Conv2d** 3×3 Conv2d, **5×5 Conv2d**
Concatenation + CR	**Concatenation + CR**
6× (1×1 Conv2d, 3×3 Conv2d) 1×1 Conv2d 6× (1×1 Conv2d, 3×3 Conv2d) AvgPool Linear Softmax	3×3 Conv2d 3×3 Conv2d 3×3 Conv2d 3×3 Conv2d 1×1 Conv2d 1×1 Conv2d AvgPool Softmax

From Table 9, we find that the approach we proposed significantly improves the performance of DenseNet-BC and ALL-CNN-C. On CIFAR10+, we improve the accuracy of DenseNet-BC and ALL-CNN-C by 0.76% and 1.15%, respectively. On CIFRA100+, we reduce the error of DenseNet-BC and ALL-CNN-C by 2.25% and 4.68%, respectively. Although the number of parameters has increased greatly, the computational cost is very low.

We select the ILSVRC 2012 classification dataset to observe the applicability of the proposed approach to a larger

TABLE 9. Error rates (%) and parameters (M) on the CIFAR datasets. The results show that fusing different scale low-level features in the preceding stage of the two networks can dramatically improve the performance. The classification accuracy of the modified DenseNet-BC (40) outperforms the classification accuracy of ResNet-110 but with fewer convolutional layers and parameters.

Method	CIFAR10+		CIFAR100+	
	Error	Params	Error	Params
DenseNet-BC (40)	6.70	0.18	29.02	0.19
DenseNet-BC (40): L-Fusion	**5.94**	0.32	**26.77**	0.34
ALL-CNN-C [15]	7.25	1.4	33.71	1.4
ALL-CNN-C: L-Fusion	**6.10**	2.04	**29.03**	2.06

TABLE 10. Parameters (M) and error (%, single-crop testing) on ImageNet validation. During the training stage, the input size of these six networks is 224 × 224. During the testing stage, we only adopt single-crop testing and obtain the classification errors at a single size: 224 × 224. The bold black parts show the structures added for feature fusion. Because of the simple parallel structure added, Darknet-19-Fusion has better results than Darknet 19. Similarly, by appending a parallel block in the preceding stage, Resnet-18-Fusion and Resnet-50-Fusion have significantly better results than Resnet 18 and Resnet 50.

Method	Structure	Params	Top1	Top5
Darknet 19 [42]	3×3 Conv2d MaxPool 3×3 Conv2d MaxPool	20.85	27.1	8.8
Darknet-19-Fusion	3×3 Conv2d MaxPool 3×3 Conv2d MaxPool | **5×5 Conv2d** **MaxPool** **5×5 Conv2d** **MaxPool** **Concatenation + CR**	20.91	**26.8**	**8.6**
Resnet 18	7×7 Conv2d MaxPool 2× (3×3 Conv2d, 3×3 Conv2d)	11.69	30.43[2]	10.76[2]
Resnet-18- Fusion	7×7 Conv2d MaxPool 2× (3×3 Conv2d, 3×3 Conv2d) | **5×5 Conv2d** **MaxPool** **2×** (**5×5 Conv2d**, **5×5 Conv2d**) **Concatenation + CR**	12.20	**28.84**	**9.80**
Resnet 50	7×7 Conv2d MaxPool 3× (1×1 Conv2d, 3×3 Conv2d, 1×1 Conv2d)	25.56	24.7[1]	7.8[1]
Resnet-50-Fusion	7×7 Conv2d MaxPool 3× (1×1 Conv2d, 3×3 Conv2d, 1×1 Conv2d) | **5×5 Conv2d** **MaxPool** **3×** (**1×1 Conv2d**, **5×5 Conv2d**, **1×1 Conv2d**) **Concatenation + CR**	26.11	**24.3**	**7.2**

dataset. Darknet 19 [42], Resnet 18 [12] and Resnet 50 [12] have good performance on the ILSVRC 2012 classification dataset and have different numbers of convolutional layers. Based on the L-Fusion 1 architecture, we append an Auxi-Block to the three networks to construct Darknet-19-Fusion, Resnet-18-Fusion and Resnet-50-Fusion, and the specific structures are shown in Table 10. We only show the changes brought to these three networks by the proposed method, and the remaining architectures of these three networks remain unchanged. We append two parallel 5 × 5 convolutional layers to form Darknet-19-Fusion, and we add one parallel block that replaces the 7 × 7 convolutional layer and all 3 × 3 convolutional layers with a 5 × 5 convolutional layer to form Resnet-18-Fusion and Resnet-50-Fusion. The classification results are shown in Table 10. We find that the performance of Darknet 19 has been slightly improved by only adding two parallel convolutional layers and concatenation + CR fusion operations. Similarly, by appending a parallel block in the preceding stage, Resnet-18-Fusion and Resnet-50-Fusion have significantly better results than Resnet 18 and Resnet 50. For Resnet-18-Fusion, we improve the top-1 accuracy and top-5 accuracy by 1.59% and 0.96%, respectively. In terms of Resnet-50-Fusion, we reduce the top-1 error and top-5 error by 0.4% and 0.6%, respectively.

F. APPLICABILITY

The residual block [38], dense block [13] and inception module [11] are fundamental components used to construct high-performing architectures, especially the first design. These three layouts fuse features on a layer basis. The residual

block and dense block integrate the features extracted from a certain layer in front with the features extracted from the layers behind, and the inception module exploits multiple filters in a layer to represent the features. Nevertheless, we consider fusing the features extracted from different blocks, and the fusing operation is only used in the preceding stages of CNNs. This approach provides the most discrimination between the state-of-the-art method and our method.

In this paper, one of the intentions is to study whether the fusion of low-level features at different scales into different stages of a CNN improves its performance. The experimental results show that L-Fusion is helpful for enhancing the performance of CNNs. Moreover, by applying the L-Fusion 1 structure, which is shown in Fig. 1, to five verification networks, we further verify our conclusions. The Auxi-Block shown in Fig. 4 is easy to build and only adds a small overhead over the model parameters and computation. Hence, our method can update some off-the-shelf networks by elaborately designing the structure we proposed. We suggest building an Auxi-Block according to the structure of the first block of a CNN; however, we cannot ensure that our approach will work when the first block has too many convolutional layers.

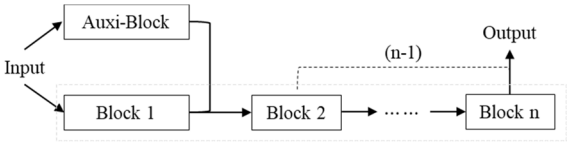


FIGURE 4. Proposed architecture. Auxi-Block and Block 1 are two parallel blocks. In terms of an off-the-shelf network, we can improve its performance simply by adding an Auxi-Block that has the same architecture as Block 1, except for a different scale convolution. The fusion operation coincides with the human retina mechanism introduced in [28], which considered that there are two types of ganglion cells with respect to the receptive field.

G. RESULTS ANALYSIS

CNNs can learn a hierarchy of features [39]. CNNs represent the low-level features that are visually recognizable in the preceding stage and represent the high-level features that are semantically recognizable in subsequent stages. L-Fusion can lead CNNs to learn low-level features with different scales. Extracting multiscale low-level features is why L-Fusion can enhance performance. The results shown in Tables 4, 6, 9 and 10 confirm this conclusion. Additionally, the effect of dense connectivity [13] is another reason that L-Fusion improves performance.

High-level features are gradually learned from low-level features [30]. According to the high-level features, the network can determine the object in an image. We consider that the semantically recognizable features will be turbulent if we fuse the low-level features with the high-level features for inference. Namely, the fused high-level features include strong and weak semantic features simultaneously. Therefore, H-Fusion will result in poor performance, and its results are shown in Table 4 and Table 6.

V. CONCLUSION

In this paper, we divide a CNN into different blocks according to the size of the features to obtain low-level and high-level features for feature fusion. We design two fusion strategies, L-Fusion and H-Fusion, to assess the influence of feature fusion at different stages. We select five low-level features with different scales to determine the advantage of multiscale feature fusion. L-Fusion, which fuses a low-level feature with different scales extracted from an auxiliary block to the low-level features extracted by a CNN, is observed to improve performance. The auxiliary block can be built according to the structure of the first block of a CNN. We validate the conclusion on five CNNs with high classification accuracy, and the experimental results show that our method achieves state-of-the-art performance. Simultaneously, the proposed architecture will not substantially increase the parameter of a CNN because the fusion operation takes place in the preceding stage.

1 https://github.com/KaimingHe/deep-residual-networks/blob/master/README.md#results

2 https://github.com/facebookarchive/fb.resnet.torch

REFERENCES

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[2] 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.

[3] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4438–4446.

[4] 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.

[5] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. Comput. Vis. Pattern Recognit.*, 2013, pp. 1–16.

[6] 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)*, Jun. 2016, pp. 779–788.

[7] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[8] 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.*, Jun. 2014, pp. 580–587.

[9] Z. Song, Y. Liu, R. Song, Z. Chen, J. Yang, C. Zhang, and Q. Jiang, "A sparsity-based stochastic pooling mechanism for deep convolutional neural networks," *Neural Netw.*, vol. 105, pp. 340–345, Sep. 2018.

[10] 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)*, Jun. 2015, pp. 1–9.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

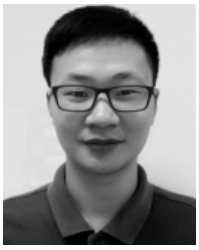
[13] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[14] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[15] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," 2014, *arXiv:1412.6806*. [Online]. Available: http://arxiv.org/abs/1412.6806
[16] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist. (AISTATS)*, vol. 15, 2011, pp. 315–323.
[17] Z. Shi, Y. Ye, and Y. Wu, "Rank-based pooling for deep convolutional neural networks," *Neural Netw.*, vol. 83, pp. 21–31, Nov. 2016.
[18] G. Xie, K. Yang, T. Zhang, J. Wang, and J. Lai, "Balanced decoupled spatial convolution for CNNs," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3419–3432, Nov. 2019.
[19] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*. [Online]. Available: https://arxiv.org/abs/arXiv:1312.4400
[20] Y. Pang, M. Sun, X. Jiang, and X. Li, "Convolution in convolution for network in network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1587–1597, May 2018.
[21] W. Shi, Y. Gong, X. Tao, D. Cheng, and N. Zheng, "Fine-grained image classification using modified DCNNs trained by cascaded softmax and generalized large-margin losses," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 3, pp. 683–694, Mar. 2019.
[22] W. Shi, Y. Gong, X. Tao, and N. Zheng, "Training DCNN by combining max-margin, max-correlation objectives, and correntropy loss for multilabel image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 2896–2908, Jul. 2018.
[23] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.
[24] Z. Qiumei, T. Dan, and W. Fenghua, "Improved convolutional neural network based on fast exponentially linear unit activation function," *IEEE Access*, vol. 7, pp. 151359–151367, 2019.
[25] X. Zou, Z. Wang, Q. Li, and W. Sheng, "Integration of residual network and convolutional neural network along with various activation functions and global pooling for time series classification," *Neurocomputing*, vol. 367, pp. 39–45, Nov. 2019.
[26] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 818–833.
[27] J. Zheng, X. Cao, B. Zhang, X. Zhen, and X. Su, "Deep ensemble machine for video classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 2, pp. 553–565, Feb. 2019.
[28] K. Uchida, M. Tanaka, and M. Okutomi, "Coupled convolution layer for convolutional neural network," *Neural Netw.*, vol. 105, pp. 197–205, Sep. 2018.
[29] Y. Hu, A. Soltoggio, R. Lock, and S. Carter, "A fully convolutional two-stream fusion network for interactive image segmentation," *Neural Netw.*, vol. 109, pp. 31–42, Jan. 2019.
[30] X. Jiang, Y. Pang, M. Sun, and X. Li, "Cascaded subpatch networks for effective CNNs," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 2684–2694, Jul. 2018.
[31] H. Wang, L. Dai, Y. Cai, X. Sun, and L. Chen, "Salient object detection based on multi-scale contrast," *Neural Netw.*, vol. 101, pp. 47–56, May 2018.
[32] C. Xu, X. Wang, and Y. Yang, "Selective multi-scale feature learning by discriminative local representation," *IEEE Access*, vol. 7, pp. 127327–127338, 2019.
[33] M. Sarıgül, B. M. Ozyildirim, and M. Avci, "Differential convolutional neural network," *Neural Netw.*, vol. 116, pp. 279–287, Aug. 2019.
[34] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
[35] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2016, pp. 21–37.
[36] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, 2012, pp. 54–57.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Dec. 2015, pp. 1026–1034.
[38] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 630–645.
[39] T. Nitta, "Resolution of singularities introduced by hierarchical structure in deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2282–2293, Oct. 2017.
[40] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2999–3007.
[41] S. Soleymani, A. Dabouei, H. Kazemi, J. Dawson, and N. M. Nasrabadi, "Multi-level feature abstraction from convolutional neural networks for multimodal biometric identification," in *Proc. 24th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2018, pp. 3469–3476.
[42] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6517–6525.
[43] R. Ranjan, V. M. Patel, and R. Chellappa, "HyperFace: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 1, pp. 121–135, Jan. 2019.
[44] S. Zhao, Y. Liu, Y. Han, R. Hong, Q. Hu, and Q. Tian, "Pooling the convolutional layers in deep ConvNets for video action recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 8, pp. 1839–1849, Aug. 2018.
[45] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," 2014, *arXiv:1409.0575*. [Online]. Available: https://arxiv.org/abs/1409.0575
[46] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," 2015, *arXiv:1507.06228*. [Online]. Available: https://arxiv.org/abs/1507.06228
[47] G. Pleiss, D. Chen, G. Huang, T. Li, L. van der Maaten, and K. Q. Weinberger, "Memory-efficient implementation of DenseNets," 2017, *arXiv:1707.06990*. [Online]. Available: http://arxiv.org/abs/1707.06990
[48] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5987–5995.
[49] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2Net: A new multi-scale backbone architecture," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 2, pp. 652–662, Feb. 2021.
[50] J. Hu, L. Shen, G. Sun, and S. Albanie, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 7132–7141.
[51] X. Zhang, Z. Li, C. C. Loy, and D. Lin, "PolyNet: A pursuit of structural diversity in very deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3900–3908.
[52] 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)*, Jun. 2016, pp. 2818–2826.
[53] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 1–7.
[54] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5686–5696.

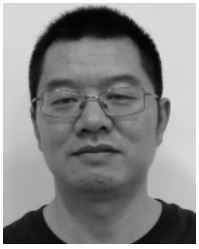
XIAOHONG YU received the B.S. degree in power machinery and engineering from the Hebei University of Technology, Tianjin, China, in 2015. He is currently pursuing the Ph.D. degree with Sichuan University, Chengdu, China. His research interests include machine vision, vehicle/traffic intelligence technology, deep learning manufacturing logistics, and planning control.

WEI LONG received the B.S. and M.S. degrees in aeroengine control engineering from Northwestern Polytechnical University, and the Ph.D. degree in mechanical manufacturing and automation from Sichuan University, Chengdu, China, in 1998. He is currently a Professor with Sichuan University. He has published about 130 articles in domestic and international journals and conferences. He has presided over or participated in more than 50 scientific research projects, published eight academic works, and obtained five invention patents. His research interests include industrial equipment automation and numerical control technology, mechanical equipment safety evaluation and reliability analysis, enterprise information control, vehicle/traffic intelligence technology, manufacturing logistics, and planning control.

YANYAN LI received the Ph.D. degree from Sichuan University. She has published about 16 articles in intelligence transportation system and obtained five invention patents. Her research interests include machine vision, vehicle/traffic intelligence technology, deep learning manufacturing logistics, and planning control.



XIAOQIU SHI received the B.S. and M.S. degrees from the School of Manufacturing Science and Engineering, Southwest University of Science and Technology, Mianyang, China, in 2010 and 2015, respectively. He is currently pursuing the Ph.D. degree with Sichuan University, Chengdu, China. His research interests include intelligent algorithm and complex networks.



LIN GAO (Member, IEEE) received the B.S. degree in material forming and control engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2002, and the M.S. degree in control engineering from the Wuhan University of Technology, Wuhan, in 2007. He is currently pursuing the Ph.D. degree with Sichuan University, Chengdu, China. He is currently an Associate Professor with Hubei Minzu University, Enshi, China. His research interests include digital image processing, deep learning, and embedded systems.