RS-CapsNet: An Advanced Capsule Network

SHUAI YANG,1, FEIFEI LEE1, (Member, IEEE), RAN MIAO,1, JIAWEI CAI1, LU CHEN1, WEI YAO1, KOJI KOTANI2, (Member, IEEE), AND QIU CHEN3, (Member, IEEE)
1School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China
2Department of Intelligent Mechatronics, Akita Prefectural University, Akita 015-0055, Japan
3Major of Electrical Engineering and Electronics, Graduate School of Engineering, Kogakuin University, Tokyo 163-8677, Japan
Corresponding authors: Feifei Lee (feifeilee@ieee.org) and Qiu Chen (q.chen@ieee.org)

This work was supported in part by the Program for Professor of Special Appointment (Eastern Scholar) at the Shanghai Institutions of Higher Learning, and in part by the JSPS KAKENHI under Grant 15K00159.

ABSTRACT Capsule Network is a novel and promising neural network in the field of deep learning, which has shown good performance in image classification by encoding features into capsules and constructing the part-whole relationships. However, the original Capsule Network is not suitable for the images with complex background due to its weak ability of feature extraction, a large number of training parameters, and the characteristic of explaining everything in the image. To address the above issues, we propose an advanced Capsule Network named RS-CapsNet, which uses Res2Net block to extract multi-scale features, and Squeeze-and-Excitation (SE) block to highlight useful features and suppress useless features. Meanwhile, we adopt the method of linear combination between capsules, which enhances the representation ability of capsules for detected object and reduces the number of capsules. Moreover, we propose the method of firstly constructing intermediate capsules that can represent most of the detected object, and then using the intermediate capsules and primary capsules to construct the classification capsules together. The experimental results show that proposed RS-CapsNet has better performance on CIFAR10, CIFAR100, SVHN, FashionMNIST, and AffNIST datasets, it can also provide better translation equivariance, and the number of training parameters is reduced by 65.11%.

INDEX TERMS Capsule network, multi-scale, CNNs, deep learning.

I. INTRODUCTION

In recent several years, convolutional neural networks (CNNs) [1]–[4] have been developing rapidly due to their excellent learning ability, which have been widely applied to many image tasks, including image classification [1]–[4], image retrieval [5], object detection [6], image segmentation [7], scene recognition [8], and so on. Although CNNs perform well in many fields, they also have many limitations. Pooling operation can provide a little translation invariance, but it will lose the precise location information of features and what we really need is equivariance. In addition, CNNs cannot learn the relationships between features, and they tend to memorize data rather than understanding it [9]. Therefore, CNNs need a large amount of training data.

In order to address the drawbacks of traditional CNNs, Sabour et al. [10] propose CapsNet, which abandons the pooling layer and retains the location information of features. It also replaces scalar output with vector output and increases the levels of the structure. Different from the translation invariance provided by the pooling operation, CapsNet can provide the translation equivariance. Furthermore, unlike the CNNs, CapsNet proposes the idea of using parts to construct the whole, which is realized by using capsules and dynamic routing algorithm.

However, the original CapsNet [10] has its own drawbacks. Firstly, it uses only two convolutional layers to extract image features, which is not suitable for complex images. Secondly, the size of the convolutional kernels in the original CapsNet is $9 \times 9$, which involves a large number of training parameters. Moreover, the original CapsNet tends to explain everything in the image, which is not suitable for classification tasks for images with complex backgrounds.

So as to solve the limitations of the original CapsNet, we propose the following methods. To enhance the ability of convolutional layers to extract image features, we propose to use Res2Net block [11]. Different from the traditional residual block in ResNet [4], Res2Net block uses small convolutional kernels and different groups of convolution operation. The different groups of convolutional kernels are...
tuted object and reduce the number of capsules at the same time. Proposed to enhance the ability for capsules to represent the properties that represent the detected object in the classification capsules. We employ a method of linear combination between capsules to construct the whole together. In addition, due to the use of small convolutional kernels and Res2Net block, RS-CapsNet only involves fewer parameters.

The contributions of this paper can be summarized as follows:

(I) We propose to use Res2Net block to extract multi-scale image features and increase the receptive field of each convolutional layer without involving many parameters. Because of the drawback of too many parameters in the original CapsNet, besides Res2Net block, we also use several extra convolutional layers with small convolutional kernels to extract the features and change the shape of feature maps. In order to eliminate the influence of the characteristic that original CapsNet tends to explain everything in the image on the classification task of images, we propose to use the Squeeze-and-Excitation (SE) block [12], which can highlight useful features and suppress useless features by assigning different weights to different channels of feature maps. We also employ a method of linear combination between capsules to enhance the ability for capsules to represent detected objects and reduce the number of capsules at the same pixel location. In addition, based on the idea that CapsNet uses parts to construct the whole, we think that large parts of the detected object can construct the whole better than small parts of the detected object. Thus, we propose to firstly construct intermediate capsules representing the large parts of the detected object, and then use the intermediate capsules and primary capsules representing the small parts of the detected object to construct the classification capsules together, in which the feature information of different types of capsules can complement each other.

Based on the above methods, we propose a novel Capsule Network that can extract multi-scale features of the input images, highlight useful features, and suppress useless features. By using the method of linear combination between capsules, the ability for capsules to represent the detected object can be enhanced. RS-CapsNet can also construct the classification capsules better by the idea that use large parts of the detected object and the small parts of the detected object to construct the whole together. In addition, due to the use of small convolutional kernels and Res2Net block, RS-CapsNet only involves fewer parameters.

The rest of the paper is organized as follows. In Section II, some related works are introduced. Our proposed advanced Capsule Network is described in detail in Section III. In Section IV, we demonstrate our experimental results and related analysis. Finally, the conclusion will be given in Section V.

II. RELATED WORKS

Based on the concept of “capsule” that is introduced by Hinton et al. [13] in 2010, Sabour et al. [10] propose the dynamic routing algorithm between capsules and present a novel neural network named CapsNet. Since CapsNet can overcome some drawbacks of traditional CNNs, including the loss of location information in the pooling operation, and it performs well in many classic datasets, many researchers have paid attention to this [14], [15] and are devoted to improving its algorithm [16]–[20] or architecture [21]–[23].

Based on the Sabour’s work, Hinton et al. [9] propose matrix Capsule Network, which replaces the vector capsule with matrix capsule and uses EM algorithm to update the coupling coefficients such that lower-layer capsules can be routed to the higher-layer to construct the higher-layer capsules. Xiang et al. [24] introduce Multi-Scale CapsNet (MS-CapsNet), in which the multi-scale features are extracted by multi-scale convolutional kernels and then used to construct the multi-dimension primary capsules. They also propose an improved dropout for the capsules. Deliege et al. [25] propose the HitNet, an improved CapsNet by constructing a new layer named Hit-or-Miss layer and introducing a centripetal loss function. HitNet also develops a method of hybrid data augmentation by combining the information between the data space and feature space. Rosario et al. [26] introduce the Multi-Lane CapsNet (MLCN), a novel CapsNet which consists of multiple parallel lanes. MLCN allows parallel processing, faster training, and fewer parameters. They also propose an improved dropout for the multi-lane. Cheng et al. [27] propose Complex-valued CapsNet (Cv-CapsNet). They use multi-scale complex-valued convolutional layers to extract the multi-scale complex-valued features and construct complex-valued capsules. They also generalize the dynamic routing algorithm to the complex-valued domain. The experiments show that the Cv-CapsNet needs fewer training parameters, fewer training epochs, and it performs better than the
Real-valued CapsNet. Gagana et al. [28] test the effects of different activation functions on the performance of the CapsNet, including ReLU, LReLU, and e-Swish. The experiments show that LReLU and e-Swish activation functions can better optimize the CapsNet than the ReLU activation function in aspect of better accuracy and faster convergence.

Furthermore, CapsNet has been applied in many different fields [29]–[32]. Saqur and Vivona [33] introduce the CapsGAN, a novel Generative adversarial Capsule Network that replaces the traditional convolutional neural network with the CapsNet as the discriminators. Yin et al. [34] apply the CapsNet to the hyperspectral image classification. They transfer the well-initialized parameters of the convolutional layers to the CapsNet and show that the performance on the CapsNet is better than that on the traditional CNNs.

III. PROPOSED ADVANCED CAPSULE NETWORK

In this paper, a novel Capsule Network named RS-CapsNet is proposed, in which convolutional layers are augmented by Res2Net blocks and SE blocks, and the capsule layer is based on the method of linear combination between capsules and the proposed routing process. The overview of proposed RS-CapsNet is shown in Fig. 1.

A. CONVOLUTIONAL LAYERS

Compared with the original CapsNet [10] that uses only two convolutional layers to extract the features of the input images, we use Res2Net block [11] to enhance the ability that the convolutional layers extract the features and increase the receptive fields of each convolutional layer. We also use the SE block [12] to highlight the useful features and suppress useless features. In addition, we use extra four convolutional layers to change the shape of the convolutional feature maps.

1) RES2NET BLOCK

Res2Net block is a variant of the ResNet, which uses several groups of convolutional operation and constructs hierarchical connections within one single residual block. Different from the multi-scale feature extraction methods which are in a layer-wise manner, Res2Net block extracts multi-scale features at a granular level and can increase the range of the receptive fields of each convolutional layer. The architecture of the Res2Net block is shown in Fig. 2.

In order to represent the multi-scale features at a granular level and increase the range of the receptive fields, the Res2Net block replaces the $3 \times 3$ convolutional kernels of $c$ channels with $n$ smaller groups of convolutional kernels of $m$ channels, where $c = n \times m$. These smaller groups of convolutional kernels are connected in a hierarchical manner to increase the number of scales of the features that the network outputs [11].

As shown in Fig. 2, firstly, the input is sent to a set of $1 \times 1$ convolutional kernels, and the output feature maps are divided into four groups after the $1 \times 1$ convolution. For the first group of feature maps $x_1$, there is no convolution operation to it. For the second group of feature maps $x_2$, a set of $3 \times 3$ convolutional kernels are used to extract the features from it, the output is $y_2$. Secondly, $y_2$ and the third group of feature maps $x_3$ are sent to the second set of $3 \times 3$ convolutional kernels together and the output is $y_3$. Then, $y_3$ and the fourth group of the feature maps $x_4$ are sent to the third set of the $3 \times 3$ convolutional kernels together and the output is $y_4$. Finally, output feature maps from all groups are concatenated and sent to another group of the $1 \times 1$ convolutional kernels to fuse the features. Similar to the residual block in the ResNet, Res2Net block uses the residual connection to connect the input with the output of the last group of convolution operations. Because input features can be transformed into output
features through multiple paths, so the receptive fields will increase when they pass a set of convolutional kernels. Due to the final combination effects, the output feature maps of the Res2Net block achieve multi-scale feature representation, and they also achieve the different numbers and different combinations of the receptive field sizes [11].

2) SE BLOCK

“Squeeze-and-Excitation” (SE) block is a kind of attention block based on the channels of feature maps. It can adaptively adjust the channel-wise feature responses by explicitly modelling the interdependencies between channels [12]. SE block can learn to use global information of feature maps to highlight useful features and suppress useless features. It has been demonstrated that it can significantly improve the performance of existing CNNs with a minimal computational cost. The architecture of the SE block is shown in Fig. 3.

For any feature map $G \in \mathbb{R}^{H \times W \times C}$, it can be transformed into $M \in \mathbb{R}^{H \times W \times C}$ by a convolution operation. For the generalized feature maps $M \in \mathbb{R}^{H \times W \times C}$, SE block is used to perform the feature recalibration. In order to model the interdependencies between the channels, the feature maps $M$ is firstly passed through a “squeeze” operation to squeeze the global information of each channel. SE block uses global average pooling to be the “squeeze” operation, where global average pooling operation is represented by $F_{sq}(\bullet)$. When the feature maps $M$ passes through the global average pooling, a data description $s \in \mathbb{R}^{C}$ will be produced, in which each dimension can describe the spatial information of corresponding channel.

In order to use the information that is produced by the “squeeze” operation, SE block uses “Excitation” operation which is represented by $F_{ex}(\bullet, w)$ to model interdependencies between the channels. The “Excitation” operation is to use two fully-connected layers, in which the first fully-connected layer is followed by the activation function of ReLU and the second fully-connected layer is followed by the sigmoid function. The reasons for using two fully-connected layers are that it can learn the nonlinearity between the channels and it is flexible. The reason for using sigmoid function as the activation function of the second fully-connected layer is that the SE block should ensure that multiple channels can be highlighted instead of just one channel [12].

After the “Excitation” operation, it can produce a set of channel weights $s' \in \mathbb{R}^{C}$, in which each dimension represents the feature importance of the corresponding channel. Then, multiplying the channel weights $s'$ with the input feature maps $M$ can result in the recalibrated feature maps $N$.

3) THE ARCHITECTURE OF CONVOLUTIONAL LAYERS

Due to the excellent characteristics of the Res2Net block and the SE block, we apply them to RS-CapsNet. Different from architecture of the convolutional layers in Res2Net, in RS-CapsNet, we only simply use the architecture of the Res2Net block to be the part of the convolutional layers, there is no skip connection between the two adjacent Res2Net blocks. In order to change the shape of the feature maps, we use extra four convolutional layers. After each convolutional layer, we use a Res2Net block to enhance the ability that convolutional layers extract the feature and increase the receptive fields. In addition, after the Res2Net block, we use SE block to highlight useful features and suppress useless features.

B. CAPSULE LAYERS

The idea of the original CapsNet [10] is to use the parts to construct the whole, and the primary capsules are directly used to construct the classification capsules. However, these primary capsules are reshaped from the convolutional feature maps, so they only represent the small parts of the detected object and have much redundant information. We propose to firstly construct the intermediate capsules that can represent the large parts of the detected object and then use all the intermediate capsules and primary capsules to construct the final classification capsules together. In addition, we introduce a method of linear combination between capsules to enhance the ability for capsules to represent the detected object. This method also can reduce the number of the capsules at the same pixel location.

1) THE LINEAR COMBINATION BETWEEN CAPSULES

In the traditional CNNs, fully-connected layers are used to combine the neurons in one layer with different weights into the neurons in another layer, and it can fuse the different features in one layer. In principle, it is same as the multi-layer perceptron neural network.

In RS-CapsNet, we introduce the method of the linear combination between capsules. The neurons at the same pixel
location in the feature maps represent different features of the detected object in the same receptive field, it means that the detected object in the receptive field can be represented by these features. When we group these neurons into primary capsules, the detected object in the receptive filed can be represented by these capsules. However, since the primary capsules are directly reshaped from the convolutional feature maps with a large amount of redundant information, so there may be a lot of redundant information in the primary capsules, for example, there may be some features about background information.

Fig. 4 shows the method of linear combination between capsules that are at the same pixel location. For any feature maps, we firstly reshape them into capsules, then we take each capsule as a whole and construct the fully-connected layer between the capsules which are at the same pixel location, where $D_1$ represents the dimension of the capsules, $N_1$ the number of capsules at each pixel location, and $N_2$ the number of capsules at each pixel location after the linear combination. Finally, we perform a squashing function on the capsules that are achieved by using the linear combination method so that it can shrink the length of capsules to $[0, 1)$, keep the direction of capsules constant, and provide more nonlinearity for the entire network. There are three reasons that we use the linear combination between capsules. On the one hand, it will not involve many parameters, because the linear combination between capsules only exists between all capsules which are at the same pixel location (e.g. the number of channels of the feature maps is 512 and the dimension of each capsule is 8, so there are only 64 capsules at the same pixel location.). On the other hand, it can reduce the number of the capsules which are at the same pixel location after the linear combination. What’s more, it can fuse the features of the different capsules to reduce the redundant information and enhance the fused capsules’ ability of representing the detected object.

2) PROPOSED ROUTING PROCESS

In the original CapsNet [10], the primary capsules are reshaped from the feature maps generated by the last convolutional operation, and then the primary capsules are directly used to construct the classification capsules. The routing principle between the primary capsules and the classification capsules is to use the parts to construct the whole. Because each primary capsule with a fixed dimension is reshaped from the channel-wise features that are at a pixel location, so each primary capsule only represents a part of the detected object. Compared with the effect of predicting the whole with the small parts of detected object, the effect of predicting the whole with the large parts of the detected object is better. The capsules that represent the large parts of the detected object can be constructed by applying the routing algorithm to primary capsules, it means that we can construct the intermediate capsules that represent the large parts of the detected object by increasing the capsule layers. However, it has recently been shown that directly stacking more fully-connected capsule layers will result in poor performance in the middle layers [37], so we cannot directly use intermediate capsules to construct the final classification capsules. In order to use intermediate capsules, we propose to use intermediate capsules that represent large parts of the detected object and primary capsules for small parts to construct the final classification capsules together. Although the effect of using the large parts of the detected object is better, the small parts of the detected object can represent the whole in more details, so using capsules that represent the small parts and the large parts of the detected object can complement the information each other.

In order to construct the intermediate capsules that can represent the large parts of the detected object, we firstly slice the feature maps which are produced from the last convolutional operation into many small partial feature maps, then use these small partial feature maps to construct the capsules that can represent the large parts of the detected object. For the method of slicing the feature maps, we propose to use vertical and horizontal sliding windows, which is shown in Fig. 5(b).

There are two reasons that we choose the vertical and horizontal sliding windows method. Firstly, for the object with horizontal structure, vertical structure, or symmetrical structure, the vertical and horizontal sliding windows method is more conducive to preserving the integrity of the object. Secondly, we hope to use more small partial feature maps. Compared with traditional sliding windows method as shown in Fig. 5(a), vertical and horizontal sliding windows method can provide more partial feature maps.
For partial feature maps, if we directly use them to construct the capsules representing large parts of the detected object, more capsules and more computation will be involved. Thus, we firstly use the $1 \times 1$ convolution to reduce the number of channels of original feature maps and partial feature maps. Then, the method of linear combination between capsules is used to fuse the features between different capsules at the same pixel location, so as to reduce the number of capsules. Note that, for the capsules reshaped from the original feature maps, we perform the linear combination operation to them, but we do not reduce the number of these capsules.

As shown in Fig. 6, for the feature maps $FM$, we firstly use the method of vertical and horizontal sliding windows to construct the partial feature maps $fm_i(1 \leq i \leq 6)$, then we use $1 \times 1$ convolution operation to reduce the number of channel of all partial feature maps $fm_i(1 \leq i \leq 6)$ and the original feature maps $FM$ to the half of the original number of the channels, the output of feature maps $fm_i(1 \leq i \leq 6)$ and $FM$ are $fm'_i(1 \leq i \leq 6)$ and $FM'$ respectively. For all achieved feature maps $fm'_i(1 \leq i \leq 6)$, we firstly reshape them into capsules, then use the method of linear combination between capsules to fuse features to enhance the ability for capsules to represent the detected object and reduce the number of capsules by half. For the feature maps $FM'$, we reshape them into primary capsules, then apply the method of linear combination between capsules to them, but we do not reduce the number of primary capsules, because we think the primary capsules can more represent the details of the detected object.

For the different groups of capsules which are obtained from different partial feature maps $fm'_i(1 \leq i \leq 6)$, we perform the dynamic routing algorithm to them and construct the intermediate capsules that can represent the large parts of the detected object. As shown in Fig. 6, each partial feature maps $fm'_i(1 \leq i \leq 6)$ can construct $N_3$ capsules, where the dimension of each capsule is $D_2$. Finally, we use all intermediate capsules which are constructed from partial feature maps $fm'_i(1 \leq i \leq 6)$ and primary capsules which are obtained from the feature maps $FM'$ to construct the classification capsules together.

### C. LOSS FUNCTION

Because the length of the capsule represents the probability that the detected object exists, so we use the margin loss [10] as the loss function of RS-CapsNet. The margin loss can ensure the true class capsule has a long length of the capsule and the false class capsules have short length of the capsules. It also allows multiple classes to exist.

In RS-CapsNet, the loss function is defined as:

$$L_k = T_k \max(0, m^+ - \|v_k\|^2) + \alpha(1 - T_k) \max(0, \|v_k\| - m^-)^2$$

Here $T_k = 1$ if the true class is $k$ and zero otherwise. $m^+$ and $m^-$ are the lower boundary of the true class capsule and the higher boundary of the false class capsules. The $\alpha$ factor of the loss for the false class capsules stops the initial learning from shrinking the length of the capsules [10]. We use $m^+ = 0.9$, $m^- = 0.1$, and $\alpha = 0.5$. 
IV. EXPERIMENTS

A. DATASETS

In order to test the performance of RS-CapsNet, we use CIFAR10, CIFAR100, SVHN, FashionMNIST, and AffNIST datasets. CIFAR10 dataset consists of 60000 images with the size of 32 × 32 × 3, in which 50000 images are the training images, 10000 images are the test images, and all images are divided into one of the 10 classes. CIFAR100 is similar to CIFAR10, except that it has 100 classes and each class has 600 images. There are 500 training images and 100 test images for each class. SVHN dataset includes 73257 training images and 26032 test images, all images are in the size of 32 × 32 × 3 and are divided into one of the 10 classes. FashionMNIST consists of 70000 images with the size of 28 × 28 × 1, 60000 images are training images and 10000 images are test images, they are divided into 10 classes. AffNIST is made by applying various affine transformations to the MNIST dataset. In AffNIST, each original MNIST image has been transformed into 32 different version with the size of 40 × 40 × 1 pixel large. The training data in each transformed version has 50000 images and test data has 10000 images.

B. SYSTEM SETUP

We use Tensorflow and Python to code RS-CapsNet. All experiments are implemented by using Tesla K40c with 12GB RAM. In our experiments, the mini-batch is 100 and the optimizer is Adam. The initial learning rate is 0.001, divided by 10 when the loss of each epoch does not decrease. The maximum number of training epochs on CIFAR10, CIFAR100, SVHN, and FashionMNIST are 180, 220, 120, and 140 respectively. We adopt batch normalization (BN) after each convolution and before the activation function. For the input of the network, we perform preprocessing to the input images. For the CIFAR10, CIFAR100 and FashionMNIST dataset, we randomly crop their images into 24 × 24 × 3, 24 × 24 × 3, and 24 × 24 × 1 respectively. For the SVHN dataset, we firstly resize the 32 × 32 × 3 images into 28 × 28 × 3, then randomly crop the 28 × 28 × 3 images into 24 × 24 × 3. Different from the original CapsNet [10], we do not use the reconstruction sub-network. We train the network on each dataset for 7 times. Among the obtained 7 test accuracies, we remove the maximum and the minimum test accuracy, and average the remaining 5 test accuracies to get the final result.

The Baseline-CapsNet consists of two convolutional layers and a capsule layer, which is similar to the original CapsNet [10], except that it does not have reconstruction sub-network. The first convolutional layer has 256 convolutional kernels with the size of 9 × 9, stride of 1, and ReLU activation. The second convolutional layer is a convolutional capsule layer, which consists of 32 capsules with 8 dimensions at each pixel location. Each primary capsule is obtained by a convolution operation with 8, 9 × 9 convolutional kernels and stride of 2 on the output of first convolutional layer [10]. The final classification capsule layer has 10, 16-dimension capsules, each capsule represents a class.

RS-CNN stands for the proposed network without routing process, where we replace the routing process and margin loss function with global average pooling and cross-entropy loss function.

C. EXPERIMENTAL RESULTS

1) RESULTS OF DIFFERENT METHODS COMBINATION

Table 1 shows the comparison of the network performance after applying different methods to the original CapsNet [10] without reconstruction sub-network. “Convs” represents the model that replaces the two convolutional layers that use 9 × 9 convolutional kernels with four convolutional layers that use 3 × 3 convolutional kernels. “Convs-Res2Net” represents the model that adopts the Res2Net block after each convolutional layer of the “Convs” model. “Conv-Res2Net-SE” represents the model that uses the SE block after each Res2Net block of the “Convs-Res2Net” model. RS-CapsNet represents the model which is obtained by applying the method of linear combination between capsules and the method of using the intermediate capsules and primary capsules to construct the classification capsules together to the “Convs-Res2Net-SE” model.

| Model                  | CIFAR10 Accuracy | Reconstruction |
|------------------------|------------------|----------------|
| Convs                  | 86.76%           | No             |
| Convs-Res2Net          | 88.90%           | No             |
| Convs-Res2Net-SE       | 89.25%           | No             |
| RS-CapsNet             | 89.81%           | No             |

From the results, we can know that Res2Net block, SE block, the method of linear combination between capsules, and the idea that firstly construct the intermediate capsules that represent the large parts of the detected object, then use the intermediate capsules and the primary capsules to construct the classification capsules together, all of these methods can improve the performance of the original CapsNet.

The method of linear combination between capsules is applied to the capsules at the same pixel location, then we perform a squashing function on the capsules that are constructed by the linear combination method. The reasons for using squashing function are that it cannot only shrink the length of capsules to [0, 1) and keep the direction of capsules constant, but also provide more nonlinearity for the entire network. Table 2 shows the comparison of network performance after applying the squashing function or not to the capsules that are constructed by the method of linear combination between capsules. From the results, we can know that the method of linear combination between capsules followed by a squashing function can make the network performance better.
TABLE 2. Comparison of test accuracy after applying squashing function or not to the capsules that are constructed by the method of linear combination between capsules.

| CIFAR10 |  |  |
|---------|---|---|
| Model   | Accuracy | Squashing function |
| RS-CapsNet | 89.13% | No |
| RS-CapsNet | **89.81%** | Yes |

TABLE 3. Comparison of test accuracy for the model with different number of intermediate capsules produced by each partial feature maps.

| CIFAR10 |  |  |
|---------|---|---|
| Model   | N₃ | Accuracy |
| RS-CapsNet | 5 | 89.50% |
| RS-CapsNet | 8 | 89.64% |
| RS-CapsNet | 10 | **89.81%** |
| RS-CapsNet | 12 | 89.73% |
| RS-CapsNet | 16 | 89.83% |

2) SELECTION OF HYPERPARAMETERS

When we use the method of using the partial feature maps to construct the intermediate capsules that represent the large parts of the detected object, it will involve two hyperparameters, the one is the number (N₃) of the intermediate capsules that are produced by each partial feature maps, and the other is the dimension (D₂) of each intermediate capsule.

For the hyperparameter N₃, we think it should not be too big. Different from the capsules which are directly reshaped from the convolutional feature maps, these intermediate capsules are produced by the routing operation, so they can represent the detected object better and have less redundant information. At the same time, the hyperparameter N₃ shouldn’t be too small, because the classification capsules are calculated by the weighted sum of all prediction capsules of the lower-level capsules, if N₃ is too small, the effect of the intermediate capsules that are produced by the partial feature maps will not be obvious. We test the performance of RS-CapsNet when N₃ = 5, 8, 10, 12, 16 respectively, and the results are shown in table 3. From the results, we can know that the performance of RS-CapsNet with N₃ = 10 is comparable to that with N₃ = 16 and it is better than others. However, there are more capsules and more parameters in RS-CapsNet with N₃ = 16 than that with N₃ = 10. Thus, we use N₃ = 10.

TABLE 4. Comparison of test accuracy for the model with different dimension of intermediate capsules.

| CIFAR10 |  |  |
|---------|---|---|
| Model   | D₂ | Accuracy |
| RS-CapsNet | 8 | **89.81%** |
| RS-CapsNet | 12 | 89.44% |

For the hyperparameter D₂, we test D₂ = 8 and D₂ = 12 respectively, the results are shown in table 4. From the results, we can know the performance of RS-CapsNet with D₂ = 8 is better than that with D₂ = 12. Though the dimension of the primary capsules is 8, they are directly reshaped from the feature maps instead of routing operation, so the primary capsules have much redundant information. For the capsules that are produced by the routing operation, they can represent the detected object better and have little redundant information. When D₂ is 12, routing operation will involve more training parameters. Thus, we use D₂ = 8.

Fig. 7 and Fig. 8 show the effects of different N₃ and D₂ values on the loss variation during the RS-CapsNet training, respectively. In both figures, before the first decline in the learning rate, the curves of loss variation are almost same during the network training process. However, in the later training process, the effects of different N₃ and D₂ values on the loss variation become obvious. After the first decline in the learning rate, the rate of loss variation of RS-CapsNet with N₃ = 10 or D₂ = 8 is bigger than that of other networks. When the networks get convergence, the loss of RS-CapsNet with N₃ = 10 or D₂ = 8 is smaller than other networks.
3) COMPARISON AND ANALYSIS OF PERFORMANCE

Fig. 9 shows the curves of the loss variation of RS-CapsNet and Baseline-CapsNet during the training. Because RS-CapsNet uses the convolutional layers of extracting multi-scale features and improved routing process, the initial loss of network training is bigger than that of the Baseline-CapsNet. However, the convergence rate of RS-CapsNet is faster than that of the Baseline-CapsNet, and the number of epoch required for getting convergence is less than that of Baseline-CapsNet. When two networks all get convergence, the loss of RS-CapsNet is smaller than that of Baseline-CapsNet.

![FIGURE 9. The curves of loss variation for two models on CIFAR10 dataset with iteration increasing. RS-CapsNet has faster convergence rate, fewer training epochs, and smaller loss.](image)

### TABLE 5.
Comparison of test accuracy, number of training parameters, and training time of an epoch on CIFAR10 dataset. (M-millions, s/E-seconds/Epoch, ens-ensemble, Recon-Reconstruction)

| CIFAR10          | Model       | Params (M) | Time(s/E) | Accuracy (%) | Recon   |
|------------------|-------------|------------|-----------|--------------|---------|
| RS-CapsNet       | 2.73M       | 360        | 90.15%    |              |         |
| Baseline CapsNet | 11.98M      | 470        | 79.24%    |              | No      |
| Sabour et al. [10] | 14.36M     | -          | 89.40%    | Yes          |         |
| HitNet [25]      | 8.89M       | -          | 73.30%    | Yes          |         |
| CV-CapsNet++ [27] | 2.69M       | 940        | 86.70%    | No           |         |
| MS-CapsNet [24]  | 11.20M      | 1650       | 75.70%    | Yes          |         |
| MLCN [26]        | 14.25M      | 1370       | 75.18%    | Yes          |         |
| DeeperCaps [38]  | 5.81M       | -          | 81.29%    | Yes          |         |
| RS-CapsNet       | 5.01M       | 570        | 89.81%    | No           |         |
| RS-CapsNet (7-ens) | 5.01M     | 570        | **91.32%** | No           |         |

CIFAR10 is a relatively complex dataset, each image in the CIFAR10 dataset has not only rich features, but also backgrounds and a lot of noises. Table 5 shows the comparison of the test accuracy, the number of training parameters, and the training time of an epoch. The performance of RS-CapsNet is better than that of all the baselines on the CIFAR10 dataset. When we only use a single model, there is a 0.41% improvement compared with Sabour’s CapsNet [10]. If we use the ensemble learning method, there is a 1.88% improvement. For the networks trained on CIFAR10, RS-CapsNet has only 5.01 million parameters, while Sabour’s CapsNet has 14.36 million parameters, so we achieve the reduction of parameter number by 65.11%. And we can achieve 89.81% on CIFAR10 with a single network, but Sabour’s CapsNet can only achieve 89.40% with ensemble learning of 7 models.

The SVHN dataset is not as complex as CIFAR10 dataset, but the images in SVHN dataset are RGB images and it can be seen as similar in flavor to MNIST (e.g., the images are small cropped digits). Table 6 shows the results of the comparison. We can achieve 96.50% on SVHN with a single network, there is a 0.8% improvement compared to Sabour’s CapsNet. And there is a 1.38% improvement with the ensemble learning of 7 models.

### TABLE 6.
Comparison of test accuracy, number of training parameters, and training time of an epoch on SVHN dataset. (M-millions, s/E-seconds/Epoch, ens-ensemble, Recon-Reconstruction)

| SVHN          | Model        | Params (M) | Time(s/E) | Accuracy (%) | Recon   |
|---------------|--------------|------------|-----------|--------------|---------|
| RS-CNN        | 2.73M        | 360        | 95.56%    |              |         |
| Baseline CapsNet | 6.68M       | 660        | 95.11%    | No           |         |
| Sabour et al. [10] | 14.36M     | -          | 95.70%    | Yes          |         |
| HitNet [25]   | 8.89M        | -          | 94.5%     | Yes          |         |
| RS-CapsNet    | 5.01M        | 956        | 96.50%    | No           |         |
| RS-CapsNet (7-ens) | 5.01M     | 956        | **97.08%** | No           |         |

The FashionMNIST is a relatively simple dataset, because all the images have been normalized to be the image with single channel and they do not have backgrounds. Because the images have no background, so we do not use the SE block when we use FashionMNIST dataset. Table 7 shows the comparison of the performance between RS-CapsNet and some baselines. From the results, the performance of RS-CapsNet is comparable to that of the baseline on the dataset without background.

### TABLE 7.
Comparison of test accuracy, number of training parameters, and training time of an epoch on FashionMNIST dataset. (M-millions, s/E-seconds/Epoch, ens-ensemble, Recon-Reconstruction)

| FashionMNIST | Model        | Params (M) | Time(s/E) | Accuracy (%) | Recon   |
|--------------|--------------|------------|-----------|--------------|---------|
| RS-CNN       | 2.73M        | 340        | 93.54%    |              |         |
| Baseline CapsNet | 6.63M       | 380        | 91.64%    | No           |         |
| HitNet [25]  | 8.89M        | -          | 92.30%    | Yes          |         |
| CV-CapsNet++ [27] | 2.50M       | 900        | **94.40%** | No           |         |
| MS-CapsNet [24] | 10.8M       | 2160       | 92.70%    | Yes          |         |
| MLCN [26]    | 10.63M       | 1200       | 92.63%    | Yes          |         |
| RS-CapsNet   | 5.00M        | 540        | 93.51%    | No           |         |
| RS-CapsNet (7-ens) | 5.00M     | 540        | **94.08%** | No           |         |

In addition to three commonly used datasets CIFAR10, SVHN, and FashionMNIST for testing the performance of the Capsule Network algorithm, we also perform experiments on the CIFAR100 dataset. Compared with CIFAR10 dataset...
TABLE 8. Comparison of test accuracy, number of training parameters, and training time of an epoch on CIFAR100 dataset. (M-millions, s/E-seconds/Epoch, Recon-Reconstruction)

| Model               | Params | Time(s/E) | Accuracy  | Recon |
|---------------------|--------|-----------|-----------|-------|
| RS-CNN              | 2.77M  | 350       | 61.14%    | -     |
| Baseline CapsNet    | 23.78M | 1550      | 55.75%    | No    |
| RS-CapsNet          | 16.80M | 1200      | 64.14%    | No    |

with 10 classes, 60000 images, CIFAR100 dataset with 100 classes, 60000 images, is more complex. Table 8 shows the performance comparison between RS-CapsNet and all baselines. Compared with Baseline-CapsNet, the performance of RS-CapsNet gets a great improvement. Meanwhile, RS-CapsNet has fewer parameters and needs less time to train for an epoch.

4) ROBUSTNESS TO AFFINE TRANSFORMATION

In order to test the robustness of RS-CapsNet to affine transformation, we firstly train the RS-CapsNet on a padded MNIST dataset, in which each image is simply centered in the 40 × 40 image by adding 6 edge background pixels around the original 28 × 28 image. We then test the trained RS-CapsNet on the AffNIST dataset. Table 9 shows the comparison of the robustness of different networks. The experimental results show that the robustness of RS-CapsNet to affine transformation is better than that of RS-CNN and all other baseline Capsule Networks. On one hand, it means that RS-CapsNet can provide translation equivariance as original CapsNet [10]. On the other hand, it proves that RS-CapsNet can provide better translation equivariance than other baseline Capsule Networks.

TABLE 9. Comparison of robustness to affine transformation. (Recon-Reconstruction)

| Model            | Accuracy | Recon |
|------------------|----------|-------|
| RS-CNN           | 75.03%   | -     |
| BaselineCapsNet  | 75.78%   | No    |
| Sabour et al. [10] | 79%      | Yes   |
| Lennsen et al. [39] | 89.10%   | Yes   |
| L-CapsNet [40]   | 87.80%   | No    |
| RS-CapsNet       | 90.31%   | No    |

5) DISCUSSIONS

RS-CNN is a network architecture using the convolutional layer of RS-CapsNet, global average pooling, classification layer, and cross-entropy loss function. The performance of RS-CapsNet is comparable with RS-CNN on CIFAR 10 dataset, and better than RS-CNN on other datasets. In particular, RS-CapsNet can provide better translation equivariance. Baseline-CapsNet is similar to the original CapsNet [10], except that it does not have reconstruction sub-network, so its network architecture is very simple. Compared with Baseline-CapsNet, RS-CapsNet uses the convolutional layers to extract multi-scale features and improved routing process. Therefore, the whole network architecture is a bit more complicated than Baseline-CapsNet, and it will take more time to train for an epoch. Nevertheless, RS-CapsNet has fewer parameters, fewer training epochs, and better classification performance. Cv-CapsNet++, MS-CapsNet, and MLCN all use multi-lane-like architecture, different lanes are responsible for different dimensions of classification capsules, so these three networks take fewer epochs to get convergence. However, the training time of an epoch on CIFAR10 dataset by using Cv-CapsNet+++, MS-CapsNet, and MLCN is 1.64 times, 2.89 times, and 2.40 times as long as that of RS-CapsNet respectively. The training time of an epoch on FashionMNIST dataset by using Cv-CapsNet+++, MS-CapsNet, and MLCN is 1.66 times, 4.00 times, and 2.22 times as long as that of RS-CapsNet respectively. The number of parameters of RS-CapsNet is reduces by 55.26% and 64.84% than that of MS-CapsNet and MLCN respectively. Thus, compared with RS-CapsNet, all three networks need more time to train for an epoch, and the last two networks have more parameters. In addition, HitNet and DeeperCaps have 43.64% and 13.76% more parameters than RS-CapsNet respectively. Except for the performance of Cv-CapsNet++ on the FashionMNIST, the performance of RS-CapsNet is better than all other Capsule Networks.

All the experimental results show that RS-CapsNet has a better performance. On the image dataset in which the images have the backgrounds, the performance of our model is better than that of the all baseline Capsule Networks. On the image dataset in which the images have no backgrounds, the performance of our model is comparable to that of the baselines. Our model not only has better performance, fewer training parameters, less time for an epoch, but also provides better translation equivariance.

V. CONCLUSION

In this paper, we propose an improved Capsule Network called RS-CapsNet, which can extract multi-scale features and highlight useful features. Meanwhile, small convolutional kernels are used to reduce the number of the parameters, which is 65.11% less than the original CapsNet. Moreover, the method of linear combination between capsules is employed to fuse the features of capsules to enhance the representation ability of capsules for detected object and reduce the number of the capsules. In addition, RS-CapsNet constructs the classification capsules by using the intermediate capsules representing the large parts of the detected object and the primary capsules representing the small parts of the detected object so that different types of the capsules can complement each other with the feature information to better construct the classification capsules. All of the experimental results show that RS-CapsNet achieves better performance than Sabour’s CapsNet and some other baselines on
CIFAR10, CIFAR100, SVHN, FashionMNIST, and AffiNIST datasets. As a future work, we plan to further improve the dynamic routing algorithm of RS-CapsNet.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Proc. Neural Inf. Process. Syst., 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, arXiv:1409.1556. [Online]. Available: http://arxiv.org/abs/1409.1556

[3] 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.

[4] 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.

[5] X. Wang, F. Lee, and Q. Chen, “Similarity-preserving hashing based on deep convolutional networks for large-scale image retrieval,” J. Vis. Commun. Image Represent., vol. 61, pp. 260–271, May 2019.

[6] Z. Qin, Z. Li, Z. Zhang, Y. Bao, G. Yu, Y. Peng, and J. Sun, “ThunderNet: Towards real-time generic object detection on mobile devices,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 6718–6727.

[7] 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.

[8] L. Xie, F. Lee, L. Liu, Z. Yin, and Q. Chen, “Hierarchical coding of convolutional features for scene recognition,” IEEE Trans. Multimedia, vol. 22, no. 5, pp. 1182–1192, May 2020.

[9] G. Hinton, S. Sabour, and N. Frosst, “Matrix capsules with EM routing,” in Proc. 6th Int. Conf. Learn. Represent., May 2018, pp. 1–15.

[10] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic routing between capsules,” in Proc. Int. Conf. Neural Inf. Process. Syst., Nov. 2017, pp. 3856–3866.

[11] S. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. H. S. Torr, “Res2Net: A new multi-scale backbone architecture,” IEEE Trans. Pattern Anal. Mach. Intell., early access, Aug. 30, 2019, doi: 10.1109/TPAMI.2019.2938758.

[12] J. Hu, L. Shen, and G. Sun, “Squeeze- and-excitation networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Jun. 2018, pp. 7132–7141.

[13] G. E. Hinton, A. Krizhevsky, and S. D. Wang, “Transforming auto-encoders,” in Proc. Int. Conf. Artif. Neural Netw., 2011, pp. 44–51.

[14] A. Shahroudnejad, P. Afshar, K. N. Plataniotis, and A. Mohammadi, “Improved explainability of capsule networks: Relevance path by agreement,” in Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP), Dec. 2018, pp. 1850–1854.

[15] A. Deliege, A. Cioppa, and M. Van Droogenbroeck, “Hitnet: A neural network with capsules embedded in a hit-or-miss layer, extended with hybrid data augmentation and ghost capsules,” 2018, arXiv:1806.06519. [Online]. Available: https://arxiv.org/abs/1806.06519

[16] V. M. D. Rosario, E. Borin, and M. Breternitz, “The multi-lane capsule network,” IEEE Signal Process. Lett., vol. 26, no. 7, pp. 1006–1010, Jul. 2019.

[17] X. Cheng, J. He, J. He, and H. Xu, “Cv-CapsNet: Complex-valued capsule network,” IEEE Access, vol. 7, pp. 85492–85499, 2019.

[18] B. Gagana, H. A. U. Athri, and S. Natarajan, “Activation function optimizations for capsule networks,” in Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI), Sep. 2018, pp. 1172–1178.

[19] E. Tobing, A. Murtaza, K. Han, and M. Y. Yi, “EP-CapsNet: Extending capsule network with inception module for electrophoresis binary classification,” in Proc. IEEE 18th Int. Conf. Biomed. Bioeng. (BIBE), Oct. 2018, pp. 327–333.

[20] S. Wang, G. Liu, Z. Li, S. Xuan, C. Yan, and C. Jiang, “Credit card fraud detection using capsule network,” in Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC), Oct. 2018, pp. 3679–3684.

[21] K. Lei, Q. Fu, and Y. Liang, “Multi-task learning with capsule networks,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2019, pp. 1–8.

[22] A. Lin, J. Li, and Z. Ma, “On learning and learned data representation by capsule networks,” IEEE Access, vol. 7, pp. 50808–50822, 2019.

[23] R. Saqur and S. Vivona, “Capsgen: Using dynamic routing for generative adversarial networks,” in Proc. Sci. Inf. Conf., 2019, pp. 511–525.

[24] J. Yin, S. Li, H. Zhu, and X. Luo, “Hyperspectral image classification using CapsNet with well-initialized shallow layers,” IEEE Geosci. Remote Sens. Lett., vol. 16, no. 7, pp. 1095–1099, Jul. 2019.

[25] A. Mobiny and H. Van Nguyen, “Fast capsnet for lung cancer screening,” in Proc. Int. Conf. Med. Image Comput. Comput.-Assis. Intervent., 2018, pp. 741–749.

[26] P. Afshar, K. N. Plataniotis, and A. Mohammadi, “Capsule networks for brain tumor classification based on MRI images and coarse tumor boundaries,” in Proc. ICASSP-IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), May 2019, pp. 1368–1372.

[27] E. Xi, S. Bing, and Y. Jin, “Capsule network performance on complex data,” 2017, arXiv:1712.03480. [Online]. Available: http://arxiv.org/abs/1712.03480

[28] Y. Xiong, G. Su, S. Ye, Y. Sun, and Y. Sun, “Deeper capsule network for complex data,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2019, pp. 1–8.

[29] J. E. Lenssen, M. Fey, and P. Libuschewski, “Group equivariant capsule networks,” in Proc. Neural Inf. Process. Syst., 2018, pp. 8844–8853.

[30] T. Jeong, Y. Lee, and H. Kim, “Ladder capsule network,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 3071–3079.

[31] A. Ahmad, B. Kakillioglu, and S. Velipasalar, “3D capsule networks for object classification from 3D model data,” in Proc. 52nd Asilomar Conf. Signals, Syst., Comput., Oct. 2018, pp. 2225–2229.

[32] C. Xiang, L. Zhang, Y. Tang, W. Zou, and C. Xu, “MS-CapsNet: A novel multi-scale capsule network,” IEEE Signal Process. Lett., vol. 25, no. 12, pp. 1850–1854, Dec. 2018.

[33] S. Yang, received the B.S. degree in automation from the Changzhou Institute of Technology, Changzhou, China, in 2018. He is currently pursuing the M.S. degree in control theory and control engineering with the University of Shanghai for Science and Technology, Shanghai, China. His research interests include computer vision and deep learning.

[34] FEIFEI LEE (Member, IEEE) received the Ph.D. degree in electronic engineering from Tohoku University, Japan, in 2007. She is currently a Professor with the University of Shanghai for Science and Technology. Her research interests include pattern recognition, video indexing, and image processing.
RAN MIAO received the Bachelor of Engineering degree from Xi’an Polytechnic University. He is currently pursuing the master degree with the School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology. His research interests include machine learning and scene recognition task specifically.

JIAWEI CAI received the B.S. degree in detection guidance and control technology from Shenyang Aerospace University, Shenyang, China, in 2018. He is currently pursuing the M.S. degree in control science and control engineering with the University of Shanghai for Science and Technology, Shanghai. His research interests include computer vision and deep learning.

LU CHEN received the B.S. degree from the Industrial and Commercial College, Anhui University of Technology, Maanshan, China, in 2016. He is currently pursuing the M.S. degree in control science and engineering with the University of Shanghai for Science and Technology, Shanghai, China. His research interests include computer vision and deep learning.

WEI YAO received the B.S. degree in automation from the Qilu Institute of Technology, Jinan, China, in 2018. She is currently pursuing the M.S. degree in control science and engineering with the University of Shanghai for Science and Technology, Shanghai, China. Her research interests include computer vision and deep learning.

KOJI KOTANI (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electronic engineering from Tohoku University, Japan, in 1988, 1990, and 1993, respectively. He is currently a Professor with the Department of Intelligent Mechatronics, Akita Prefectural University. He is engaged in the research and development of high performance devices/circuits as well as intelligent electronic systems. He is a member of the Institute of Electronics, Information and Communication Engineers of Japan.

QIU CHEN (Member, IEEE) received the Ph.D. degree in electronic engineering from Tohoku University, Japan, in 2004. Since then, he has been an Assistant Professor and an Associate Professor with Tohoku University. He is currently a Professor with Kogakuin University. His research interests include pattern recognition, computer vision, information retrieval and their applications. He serves on the editorial boards of several journals, as well as committees for a number of international conferences.

* * *