Journal of Electronic Imaging

JElectronicImaging.org

Feature and spatial relationship coding capsule network

Tenghao Han
Rencheng Sun
Fengjing Shao
Yi Sui

SPIE. IS&T imaging.org

Tenghao Han, Rencheng Sun, Fengjing Shao, Yi Sui, "Feature and spatial relationship coding capsule network," *J. Electron. Imaging* **29**(2), 023004 (2020), doi: 10.1117/1.JEI.29.2.023004

Feature and spatial relationship coding capsule network

Tenghao Han, Rencheng Sun,* Fengjing Shao, and Yi Sui
Qingdao University, Department of Computer Science and Technology, Qingdao, China

Abstract. A capsule network encodes entity features into a capsule and maps a spatial relationship from the local feature to the overall feature by dynamic routing. This structure allows the capsule network to fully capture feature information but inevitably leads to a lack of spatial relationship guidance, sensitivity to noise features, and easy susceptibility to falling into local optimization. Therefore, we propose a novel capsule network based on feature and spatial relationship coding (FSc-CapsNet). Feature and spatial relationship extractors are introduced to capture features and spatial relationships, respectively. The feature extractor abstracts feature information from bottom to top, while attenuating interference from noise features, and the spatial relationship extractor provides spatial relationship guidance from top to bottom. Then, instead of dynamic routing, a feature and spatial relationship encoder is proposed to find the optimal combination of features and spatial relationships. The encoder abandons the idea of iterative optimization but adds the optimization process to the backpropagation. The experimental results show that, compared with the capsule network and its multiple derivatives, the proposed FSc-CapsNet achieves significantly better performance on both the Fashion-MNIST and CIFAR-10 datasets. In addition, compared with some mainstream deep learning frameworks, FSc-CapsNet performs quite competitively on Fashion-MNIST. [DOI: 10.1117/1.JEI.29.2.023004]

Keywords: deep learning; neural network; capsule network; image classification; three-dimensional convolution; spatial relationship coding.

Paper 190849 received Sep. 20, 2019; accepted for publication Feb. 14, 2020; published online Mar. 6, 2020.

1 Introduction

Traditional convolutional neural networks (CNNs)[1] have obvious limitations for exploring spatial relationships. The general method for classifying images of the same type taken from different angles is to train multiple neurons to process features and then add a top-level detection neuron to detect the classification results. This approach tends to remember the dataset rather than summarizing the solution, and it requires large amounts of training data to cover different variants and avoid overfitting. This characteristic also makes CNNs very vulnerable when dealing with tasks based on moved, rotated, or resized samples.

Unlike CNNs, capsule networks (CapsuleNet)[2] use capsules[3] to capture a series of features and their variants. In the capsule network, higher-layer capsules are used to capture the overall features, such as "face" or "car," while the lower-layer capsules are used to capture local entity features such as "nose," "mouth," or "wheels," leading to a completely different approach than a convolutional network when abstracting the overall feature from local features. However, this is not enough. A complete identification process requires both bottom–up feature abstraction and top–down spatial relationship guidance. The capsule network defines a transformation matrix between adjacent capsule layers to implement feature abstraction. Then, dynamic routing is used to find the optimal combination of features to activate the next layer of capsules. However, the entire process is still solely a bottom–up process. The definition of the transformation matrix is settled in the lower-layer capsule, and feature information flows from lower-layer capsules to higher-layer capsules. To achieve a more natural recognition, a top–down stream of spatial relationship information is also needed.

*Address all correspondence to Rencheng Sun, E-mail: qdsunstar@163.com

As an example, suppose we are doing a jigsaw puzzle in which a dog's picture is split and the pieces are mixed up. The process of restoring the original image from the fragments is generally similar: first, we obtain a piece of the picture—maybe the dog's leg or perhaps its mouth. Obviously, these fragments are far from sufficient to restore the picture: we do not know where they should be placed. Thus, naturally, in the second step, we begin to imagine where the dog's legs should be and how far they are from its mouth. Step by step, we restore the pieces to form the dog. This results from spatial relationship guidance, which links the fragments together to form a whole. If we do not know the spatial information of the image, the possibility of restoring the dog is almost zero. That is why we need to improve the capsule network. More accurately, naturally, the identification process requires the guidance of spatial relationships, and the capsule network is not good at this.

In view of the fact that the capsule network uses a single transformation matrix, it is sensitive to noise features, which causes the model to perform poorly when processing images with complex spatial relationships. At the same time, the optimization idea of the iterative loop in the dynamic routing algorithm also slows down the model's convergence speed. For the purpose of making up for these shortcomings as much as possible, we propose a new capsule network called a feature and spatial relationship coding capsule network (FSc-CapsNet). Unlike the original capsule network, we propose a new feature and spatial relationship coding structure for FSc-CapsNet to replace the feature extraction process and dynamic routing. The feature and spatial relationship coding structure consists of three parts: feature and spatial relationship extraction, fusion, and encoding. The feature and spatial relationship extraction part includes two processes. The first is the bottom–up feature extraction. FSc-CapsNet uses the feature extractor to extract the decisive features required for classification from the original features. The second is a top–down spatial relationship extraction. Unlike standard CapsuleNet, we capture the spatial relationships in images through a spatial relationship extractor based on the higher-layer capsules.

FSc-CapsNet extracts the decisive features of the original features using a feature extractor. These features have a large influence on the classification result, and they will greatly affect the classification result for the current entity. The extraction weakens the influence of noise features on the classification results. Noise features can be considered as features that have a negative effect or may even interfere with the classification results. After feature extraction, FSc-CapsNet strengthens the positive impact of the decisive features and weakens the negative impact of noise features on the classification results. It is easier to obtain the correct classification result using only decisive features than using all of the original features.

The spatial relationship extractor captures the spatial relationships that exist between different features belonging to the same entity. Usually, a fixed spatial relationship exists between different parts belonging to the same entity, while the spatial relationships between different entities are different. Under the guidance of different spatial relationships, combinations of the same features will result in different classification results. Consider the handwritten digit 6 as an example: all handwritten digits 6 should include a circle and a line. Regardless of the position or shape of the handwritten digit 6, the circles and lines follow the spatial pattern representative of the digit 6. When a circle and a line exist but they are combined using the spatial pattern of the number 9, we know that the handwritten digit is a 9 and not a 6. Examples of handwritten numbers 6 and 9 are shown in Fig. 1. We use a spatial relationship extractor to capture the spatial

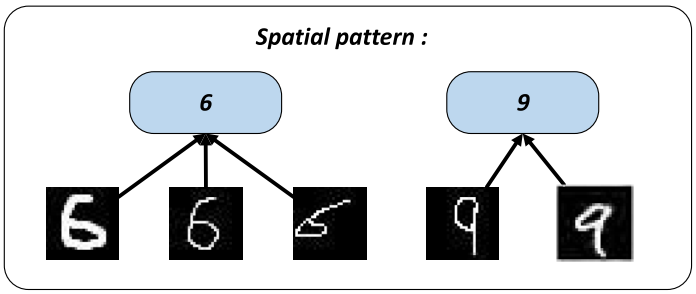

Fig. 1 Two different spatial relationship patterns: numbers 6 and 9.

relationships that correspond to the classification target. The spatial relationships differ depending on the classification target, and they affect the combinations of lower-layer capsules but are not affected by them. Therefore, the process of spatial relationship extraction can be considered a top–down process.

In the capsule network, dynamic routing is used to find the optimal combination of the lower-layer capsules to activate the higher-layer capsules, which is the classification result. In dynamic routing, the capsules in the same layer are combined according to their contribution weights to activate a capsule in a higher layer during iterations. Iterative loops slow down network convergence and at the same time make the optimization process easily fall into a local optimum.

In FSc-CapsNet, the feature extractors map the feature vector into a feature matrix when extracting features. Subsequently, FSc-CapsNet fuses the spatial relationships in another dimension based on the feature matrix, forming a three-dimensional (3-D) information cube. Considering the complexity of finding the optimal combination of information cubes, it is not appropriate to continue using dynamic routing. Instead, we use a 3-D convolutional kernel[4] to construct a feature and spatial relational encoder that simultaneously finds the optimal combination of features and spatial relationships. The 3-D convolutional kernel can model two different dimensions of information at the same time. The feature and spatial relationship encoder does not need to find the optimal combination through iteration. Instead, it performs the optimization process during backpropagation. The encoder can adaptively find the global optimal combination during the network training process. This combination method does not require manual evaluation indicators, and it achieves a more direct and accurate search for the optimal combination of features and spatial relationships. In addition, FSc-CapsNet establishes a reconstruction unit via a deconvolutional layer to explore the possibility of using the deconvolutional layer[5] instead of a fully connected layer as the reconstruction unit. The deconvolutional layer can be regarded as a pooling process from low to high resolution through its unique upsampling operation. The upsampling process gives it stronger antinoise and pooling abilities.

The FSc-CapsNet established a new feature and spatial relationship coding structure. The feature and spatial relationship extractor extracts the decisive features and spatial relationships from the original features, respectively. The feature and spatial relationship encoder replaces the dynamic routing with iterative optimization using the 3-D convolution kernel to simultaneously find the optimal combination of decisive features and spatial relationships. The structure introduces top–down spatial information guidance for the capsule network and incorporates the feature combination optimization process into the backpropagation. Compared with the original capsule network and its derivatives, FSc-CapsNet achieves significantly better performance on both the Fashion-MNIST and CIFAR-10 datasets. In addition, compared with other mainstream deep learning frameworks, FSc-CapsNet achieves a competitive performance on Fashion-MNIST. We made three improvements compared with the standard capsule network:

1. We propose a feature extractor and a spatial relationship extractor that extract features and spatial relationships, respectively. A top–down stream of spatial relationships is introduced into the capsule network, and the interference from noise features is attenuated.
2. We propose a feature and spatial relationship encoder that replaces dynamic routing and adds the process of finding an optimal combination to the backpropagation, and the training process becomes end to end.
3. The deconvolution layer replaces the fully connected layer as the base layer of the reconstruction unit with its powerful pooling and antinoise abilities.

2 Related Work

In recent years, many researchers have introduced spatial relations in the CNN classification process. Spatial transformer networks (STNs)[6] transform input patterns and even feature maps through a series of affine transformation operations and add point of view invariance to the standard CNN input without requiring any extra training supervision or modification. STNs have achieved state-of-the-art performance on several benchmarks. Deformable convolutional networks[7] further expand the STNs and perform different feature map, perform spatial transformations at different locations, and introduce two new modules to enhance the transformation

modeling capacity of CNNs: deformable convolution and region of interest pooling. Experiments have shown that this method has good results on complex object detection visual tasks and semantic segmentation. Harmonic networks (H-Nets)[8] implement a rotating equivalent feature map using a circular harmonic filter, and they use complex returns to obtain the maximum response and direction. H-Nets use a rich, parameter-efficient, and computationally efficient representation to introduce rotation invariance in deep feature map coding. At the same time, their layer versatility is sufficient for use in conjunction with the latest architectures and technologies. CapsuleNet[2] explore new ways to encode feature information based on a method for rewriting the underlying unit to introduce rotation invariance in the image classification process. Compared with H-Nets, CapsuleNet perform better and require fewer parameters.

Subsequently, explorations and innovations to the capsule network structure have emerged one after another. The proposed generative adversarial capsule network (CapsuleGAN)[9] uses a framework that employs CapsuleNet instead of standard CNNs to explore the fusion of CapsuleNet and semisupervised models. CapsuleGAN is superior to a convolutional generative adversarial network at modeling the image data distribution on the MNIST and CIFAR-10 datasets. A deep inception capsule network for gamma-turn prediction[10] was also proposed, and its performance was significantly better. This paper also verifies that even with a relatively small number of input samples, CapsuleNet's capsules are extremely effective at extracting advanced features for use in classification tasks. Generalized CapsuleNet [11] added a routing process to the optimization process and automatically set the number of routing networks. However, the paper reports that the routing procedures in CapsuleNet are not sufficiently well integrated into the entire training process because the optimal number of routing processes must be found manually. To overcome this problem, they embedded the routing process into the optimization process along with all of the other parameters in the neural network, making the coupling coefficients in the routing process fully trainable. This approach coincides with our encoder design idea. Furthermore, we abandon the idea of route iteration and use the 3-D convolutional kernel to find the optimal combination. MS-CapsNet[12] proposed a multiscale capsule network that is more robust and efficient for feature representation. The authors proposed a multiscale feature extraction method to expand the ability of the capsule network to recognize images. In contrast, we further segment the original features and map the feature vectors to the feature matrices.

TextCaps[13] introduced a technique for generating new training samples from existing samples that simulates the actual changes in human handwriting input by adding random noise to the corresponding instantiation parameters. This strategy is useful in character recognition for localized languages that lack large amounts of labeled training data. The stacked capsule autoencoder (SCAE)[14] is an unsupervised capsule network that defines a new representation learning method in which any encoder can learn the viewpoint representation by inferring the local parts and their poses and identifying the object to which these parts belong. In a comparison of many unsupervised models, SCAE achieved a significant improvement. It also tries to find a fixed relationship between the part and the whole, and the difference from our approach is that an unsupervised approach is implemented by the authors of SCAE.

CapsuleNet also have fairly good performance when dealing with real-world problems or with realistic image datasets. A novel method for traffic sign detection using CapsuleNet[15] achieved an accuracy of 97.6% on the German Traffic Sign Recognition Benchmark dataset. The work investigating CapsuleNet with dynamic routing[16] explored the effects of CapsuleNet for text classification and proposed three strategies for stabilizing dynamic routing processes to mitigate the interference of certain noise capsules. A series of experiments were performed on six text classification benchmarks using a capsule network. The experiment showed that the capsule network achieves a significant improvement when transferring a single label to a multilabel text classification via a strong baseline method.

3 Feature and Spatial Relationship Coding Capsule Network

In CapsuleNet, a capsule is defined as a vector with both direction and length, where the direction represents the feature of the entity and the length represents the probability of the entity's

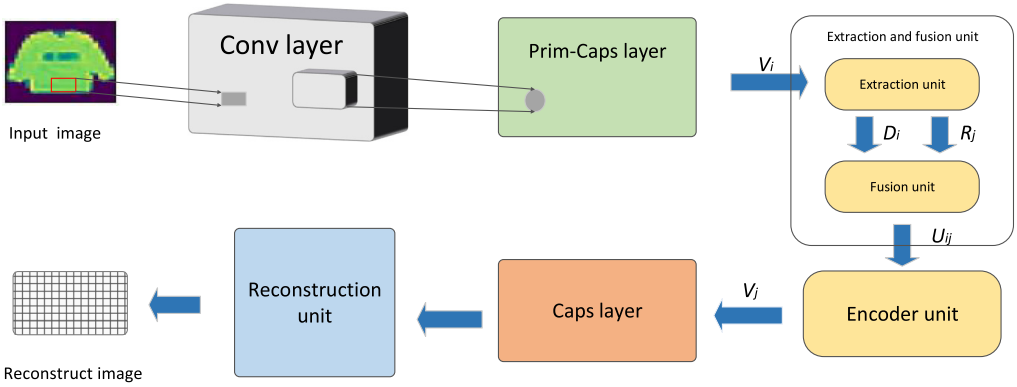

Fig. 2 The FSc-CapsNet consists of three layers and a reconstruction unit.

existence. After applying "Squashing" nonlinear activation as Eq. (1), the activation probability of the capsule, that is, its length, is compressed to (0, 1):

$$\text{Squashing}: \mathbf{V_i} = \frac{\|\mathbf{S_i}\|^2}{1+\|\mathbf{S_i}\|^2} \times \frac{\mathbf{S_i}}{\|\mathbf{S_i}\|}, \tag{1}$$

where $\mathbf{S_i}$ is the input vector and $\mathbf{V_i}$ is the output vector of the capsule i.

The capsule network defines a transformation matrix W_{ij} to implement the feature abstraction process, where i is any lower-layer capsule and j is any higher-layer capsule. The features mentioned in this paper generally refer to the vector-type output of a capsule. A capsule with a higher dimension can be regarded as carrying more feature information. Lower-layer capsules represent local features, while higher-layer capsules represent global features, in other words, the classification result. In a capsule network, the capsules in the Prim-Caps layer are called primary capsules, while the Caps layer consists of the ultimate capsules. The output $\mathbf{V_i}$ of capsule i in the Prim-Caps layer can be thought of as the original feature, and the output $\mathbf{V_j}$ of ultimate capsule j represents the classification result.

In FSc-CapsNet, we propose a new feature and spatial relationship coding structure that includes two stages: an extraction and fusion unit and an encoder unit. The structure of the FSc-CapsNet is shown in Fig. 2, which consists of four parts: a standard convolutional layer ("Conv layer"), a "Prim-Caps layer," a "Caps layer," and a deconvolutional reconstruction unit. The extraction unit, the fusion unit, and the encoder unit are applied between the Prim-Caps layer and the Caps layer. Here $\mathbf{V_i}$ refers to the output of capsule i in the Prim-Caps layer, and $\mathbf{V_j}$ refers to the output of ultimate capsule j, which represents the classification result.

As shown in Fig. 2, we perform feature and spatial relationship extraction unit instead of the transformation matrix W_{ij} to abstract the decisive feature and spatial relationships. The decisive feature D_i exists in the feature matrix, but the spatial relationship R_j exists between the feature matrices. The U_{ij} is the output of a preliminary combination of the decisive features and spatial relationships by fusion unit. It contains both the features most beneficial to the classification results and the guidance of the spatial relationships from the classification results. It is more advantageous to use U_{ij} as the basic unit for activating the higher-layer capsules than to use the original features. Finally, the encoder performs the final combination of the fusion feature U_{ij}, finds the optimal combination, and activates the higher-layer capsule j to obtain the classification results $\mathbf{V_j}$.

3.1 *Extraction and Fusion Unit*

As shown in Fig. 3, FSc-CapsNet separately defines the feature extractor FE_i and the spatial relationship extractor SE_j. The FE_i is used to extract the decisive feature D_i of the entity, and the SE_j is used to find the spatial relationship R_j from the classification results. For any primary capsule i, there is one and only one corresponding FE_i, and for any ultimate capsule j, there is

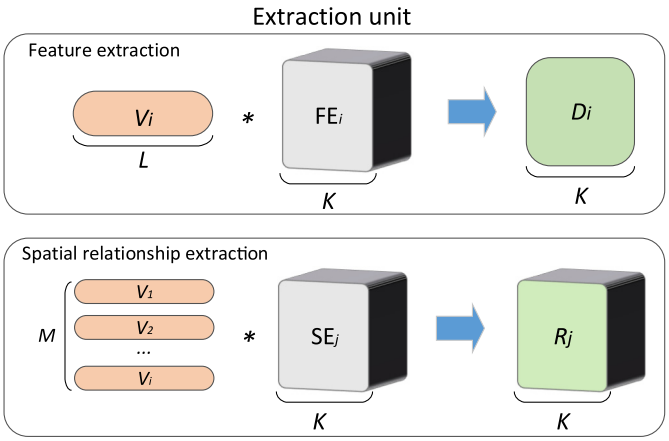

Fig. 3 The process of extracting decisive features D_i and spatial relationships R_j from the original feature $\mathbf{V_i}$ by the feature extractor FE_i and the spatial relationship extractor SE_j.

one and only one corresponding SE_j. The process of extracting D_i from $\mathbf{V_i}$ by FE_i should not be affected by the classification result. Because different classification results are activated by combinations consisting of the same set of decisive features, the classification results do not affect the feature information carried by the decisive features. They only affect the combination of decisive features. Different classification results have different spatial relationships, and different spatial relationships can combine the same decisive features into different classification results.

In general, a fixed spatial relationship exists between the different parts belonging to the same entity, while the spatial relationships between different entities are often different. We define a spatial relationship extractor SE_j to capture this spatial relationship between a series of original features that are subordinate to a given classification result. It can also detect differences between features that are not part of the same classification result. The spatial relationship R_j reflects the contribution of the decisive feature D_i to the classification result $\mathbf{V_j}$.

As shown in Fig. 3, the feature extractor and spatial relationship extractor are defined as a 3-D matrix—one dimension higher than the transform matrix W_{ij}—used to expand the feature vector into a feature matrix. In a capsule network, it is generally believed that the higher the dimension of the capsule, the more information it carries. Thus, a feature matrix carries more information than does a feature vector. To some extent, expanding the feature vector to a higher dimension also achieves the selection and combination of feature information. Assume the number of primary capsules in the Prim-Cap layer is M, and the number of ultimate capsules in the Cap layer is N. Each primary capsule has L dimensions, and the dimension of the ultimate capsule is H. The size of the feature extractor FE_i is defined as $[L, K, K]$, and the size of spatial relationship extractor SE_j is also defined as $[L, K, K]$. There are a total of M feature extractors and N spatial relationship extractors. Here K is a constant between (L, H), D_i is calculated by Eq. (2), and R_j is calculated by Eq. (3).

$$D_i = \mathbf{V_i} * \mathrm{FE}_i, \tag{2}$$

$$R_j = \mathbf{V_i} * \mathrm{SE}_j. \tag{3}$$

As mentioned above, R_j represents a fixed spatial relationship between a series of features subordinate to a given classification result. It also reveals the differences between features that are not part of the same classification result. Therefore, the spatial relationship R_j can be used to measure the contribution of decisive features to the classification results. The process of fusing decisive features with spatial relationships to obtain fusion features is shown in Fig. 4. We linearly superimpose the decisive feature D_i with the spatial relationships R_j to obtain the fusion feature U_{ij}, as in Eq. (4). The fusion feature U_{ij} contains the features that are most beneficial to the classification results, as well as guidance from the spatial relationships of the classification results.

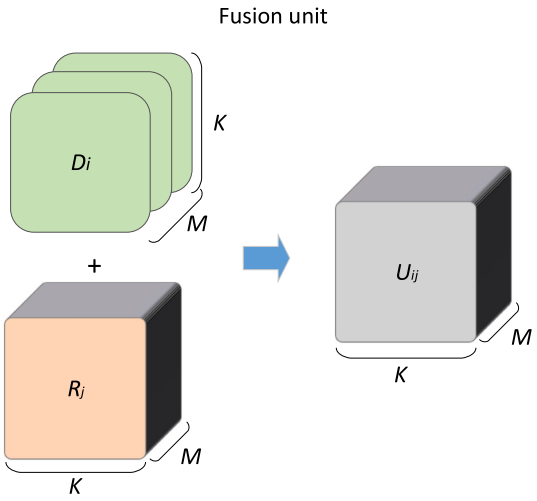

Fig. 4 The decisive feature D_i linearly superimpose with the spatial relationships R_j to obtain the fusion feature U_{ij}.

$$U_{ij} = D_i + R_j. \tag{4}$$

3.2 *Encoder Unit*

In the capsule network, dynamic routing is used to find the optimal combination of the primary capsules to activate the ultimate capsules. The capsule network needs to enhance the contribution of the features that are beneficial to the classification results and weaken those that are harmful. Therefore, the contribution weight C_{ij} is introduced as an evaluation indicator to measure the contributions of features to the classification results. The C_{ij} is calculated by the coupling coefficient B_{ij} between the predicted feature $\hat{U}_{j|i}$ and $\mathbf{V_j}$, as shown in Eqs. (5) and (6).

$$B_{ij} = B_{ij} + \hat{U}_{j|i} \cdot \mathbf{V_j}, \tag{5}$$

$$C_{ij} = \mathrm{Softmax}(B_{ij}). \tag{6}$$

The dynamic routing process is shown in Fig. 5. In iteration, the primary capsule i combines with each other capsule to activate the ultimate capsule j according to its C_{ij}. The classification result $\mathbf{V_j}$ of this iteration participates in the next iteration as input to update C_{ij}. After several iterations, the optimal feature combination is obtained, where the optimal feature combination is the optimal solution of C_{ij}. For feature vectors, dynamic routing is a suitable way to find the optimal combination of features, but it is not suitable for FSc-CapsNet. In addition, the initial value of C_{ij} will greatly influence the search process of the optimal solution, and the iterative optimization process independent of backpropagation can easily fall into a local optimal solution.

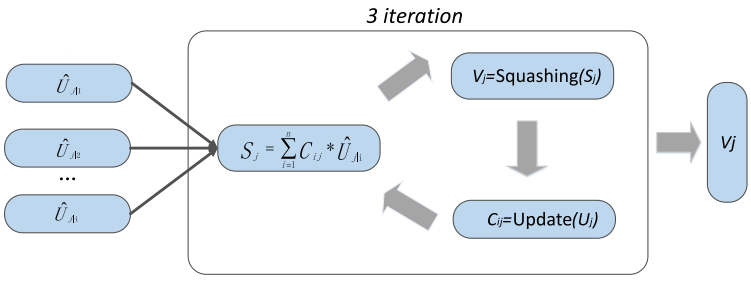

Fig. 5 In the dynamic routing, the contribution weight C_{ij} is updated simultaneously.

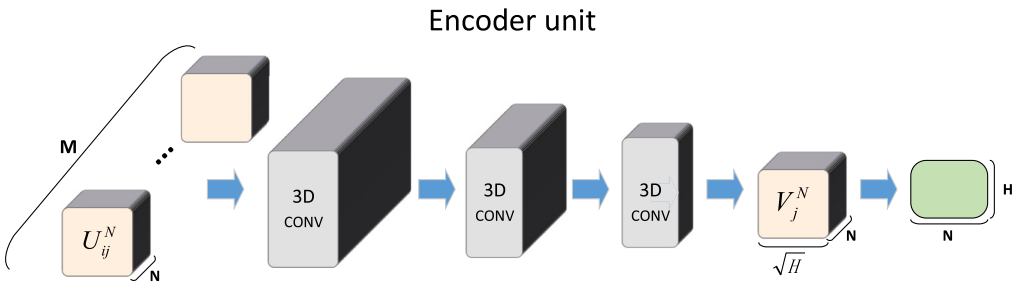

Fig. 6 The feature and spatial relationship encoder consists of three 3-D convolutional layers, and the fusion feature U_{ij} is abstracted into ultimate capsules by 3-D convolutional kernel. Here M is the number of primary capsules and the channels of the encoder and N is the number of ultimate capsules and the depth of encoder. Finally, the encoder resizes the output V_j^N to $[N, H, 1]$, where H is the dimension of an ultimate capsule.

In FSc-CapsNet, the decisive features and spatial relationships are fused into an information cube, which is called the fusion feature U_{ij}. We propose a feature and spatial relationship encoder to adaptively find the optimal combination of fusion features, as shown in Fig. 6. The 3-D convolutional kernel is used as a basic unit to construct three convolutional layers, forming the feature and spatial relationship encoder. The encoder takes the fusion feature U_{ij} as input, the number of primary capsules as the number of 3-D convolutional channels, and the number of ultimate capsules as the depth of the 3-D convolutional kernel. As the 3-D convolutional encoding progresses, the feature matrix in the depth range is continuously abstracted into smaller scales by the 3-D convolutional kernel, which can be considered as a combination of the fusion features. At the same time, the feature matrix between the different channels is combined into a new feature matrix, which can be considered as a combination of the spatial relationships between the fusion features. Finally, the encoder adaptively finds the optimal solution that takes both directions into consideration.

The feature matrix in the depth-N range is defined as U_{ij}^N and is input to the three 3-D convolutional layers to obtain the output of the ultimate capsules. The output of the ultimate capsules in the depth-N range is defined as V_j^N. The value v_{ij}^{xyz} at position (x, y, z) on the j'th feature matrix in the i'th 3-D convolutional layer is given in Eq. (7). Finally, $\mathbf{V_j}$ is resized to $[H, 1]$ and activated by the Squashing nonlinear activation as Eq. (1).

$$v_{ij}^{xyz} = \mathrm{ReLU}\left[\sum_m \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{t=0}^{T_i-1} W_{ijm}^{pqt} v_{(i-1)m}^{(x+p)(y+q)(z+t)} + b_{ij}\right], \tag{7}$$

where T_i is the depth of the 3-D convolutional kernel and W_{ijm}^{pqt} is the (p, q, t)'th value of the 3-D kernel connected to the m'th feature matrix in the previous layer. Inside the encoder, the activation function of the convolutional layer is the rectified linear unit (ReLU[17]).

In the general deep learning model, a 3-D convolutional kernel is generally used to learn the spatiotemporal features. It can simultaneously model time and space features to establish a connection between the two. The feature abstraction process of two adjacent 3-D convolutional layers is shown in Fig. 7. The 3-D convolutional kernel, simultaneously along with the depth and channel direction, could abstract the feature matrix I_{1-i}^b of the b layer from the I_{1-i}^a in a layer. To demonstrate the abstraction of features better, we simplified the representation of the feature map, focusing only on the encoding process corresponding to one higher level capsule, so I_i^a is defined as the output of the a'th layer 3-D convolution layer feature map. Generally speaking, U_{ij} in Fig. 6 can be regarded as the input of the first 3-D convolutional layer, which is defined as I_i^0, and $\mathbf{V_j}$ in Fig. 6 is the output of the third 3-D convolutional layer, which is I_i^3.

Figure 8 shows the bottom–up feature abstraction process and the process of capturing the spatial relationships in a top–down manner in FSc-CapsNet. In the extraction unit, the two streams of information are independent. The fusion unit fuses the two streams to obtain the

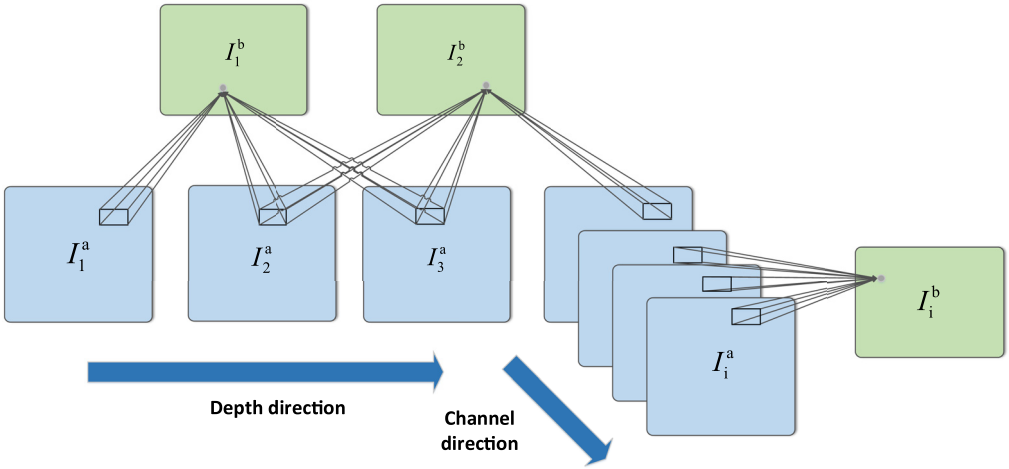

Fig. 7 The feature abstraction process of two adjacent 3-D convolutional layers.

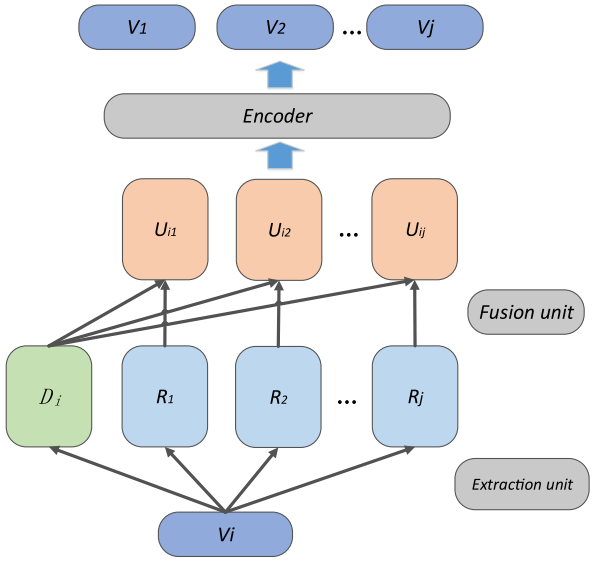

Fig. 8 The process of feature and spatial relationship coding, including feature and spatial information extraction, fusion, and coding unit.

fusion feature. The encoder finds the optimal fusion feature combination through the 3-D convolutional kernel while considering both the fusion features and the spatial relationships.

3.3 *Deconvolutional Reconstruction Unit*

As shown in Fig. 9, we set the deconvolutional reconstruction unit to reconstruct an image from the ultimate capsules. After the output $\mathbf{V_j}$ of the ultimate capsule j passes through the mask layer, the feature information carried by $\mathbf{V_j}$ is decoded using three deconvolutional layers to obtain a reconstructed image. The reconstructed image can assist in training. We use the Euclidean distance between the reconstructed image and the original image as the reconstructed loss. In addition, the reconstructed loss after zooming in 0.0005 times is added to the overall loss during training as in Eq. (8). The deconvolutional reconstruction structure is shown in Fig. 9.

$$L_j = T_j * \max(0, m^+ - \|U_j\|)^2 + \lambda(1 - T_j) * \max(0, \|U_j\| - m^-)^2 + 0.0005 * \sum_i^n (O_{ij} - R_{ij})^2, \tag{8}$$

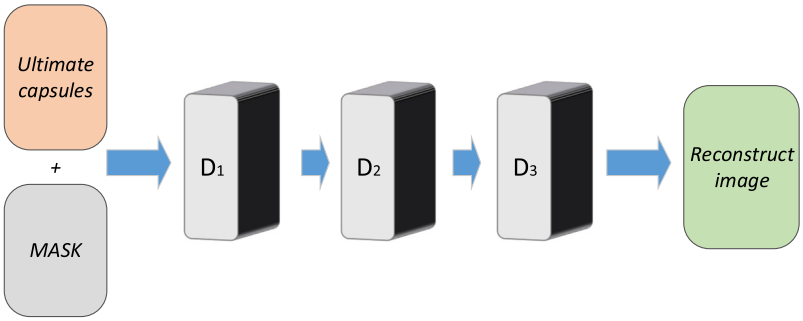

Fig. 9 The architecture of deconvolutional reconstruction unit.

where T_j stands for the label during training, $\lambda = 0.5$, $m^+ = 0.9$, and $m^- = 0.1$. Here O_{ij} and R_{ij} represent the pixel intensities of the original and reconstructed images, respectively.

4 Experiments

To evaluate the performance of FSc-CapsNet, we conducted an experiment on the Fashion-MNIST[18] and CIFAR-10 datasets.[19] Compared with the standard MNIST, the samples in the Fashion-MNIST and CIFAR-10 datasets are both more diverse and more realistic. The "components," for example, the body part of the top, the sleeve, and the collar, shared between the different sample types are clearer, and these parts are common to various categories, such as coats and T-shirts. The accuracy achieved on this type of dataset can demonstrate the advantages of FSc-CapsNet's use of spatial relationships to address classification tasks. Fashion-MNIST is an even slightly greater classification challenge than is MNIST.

The Fashion-MNIST dataset contains 10 categories and 70,000 samples. The size, format, and training/testing sets of Fashion-MNIST are identical to those of MNIST. Each image is gray-scale with a size of [28, 28]. The training set size is 60,000, and the testing set consists of 10,000 images.

4.1 *Model Structure and Parameter Settings*

The structure of the FSc-CapsNet is shown in Fig. 2, which consists of four parts: Conv layer, Prim-Caps layer, Caps layer, and deconvolutional reconstruction unit. The FSc-CapsNet's parameter settings are shown in Table 1, including Conv layer, Prim-Caps layer, Caps layer, the encoder unit, and deconvolutional reconstruction unit, and the table shows the specific

Table 1 The parameter settings of FSc-CapsNet.

Layer	Input channel	Kernel size	Stride	Depth	Output channel
Conv layer	1	7×7	2	—	256
Prim-Caps layer	256	5×5	1	—	$32 \times 6 \times 6$
3-D Conv-1 layer	$32 \times 6 \times 6$	3×3	2	10	32
3-D Conv-2 layer	32	3×3	2	10	8
3-D Conv-3 layer	8	3×3	2	10	1
Caps layer	1	—	—	—	10
DeConv-1 layer	1	3×3	1	—	5
DeConv-2 layer	5	3×3	2	—	3
DeConv-3 layer	3	4×4	2	—	1

settings of the six parameters, including input channel, kernel size, stride, depth, and output channel.

In FSc-CapsNet, Conv layer is a standard two-dimensional (2-D) convolutional layer with ReLU activation. Prim-Caps layer converts the feature map abstracted by Conv layer into 1152 ($32 \times 6 \times 6$) channels primary capsule, and each primary capsule is eight-dimensional (8-D). Caps layer consists of 10 ultimate capsules, and each ultimate capsule is 16-dimensional (16-D). The encoder unit used for the spatial relation coding consists of three 3-D convolutional layers. And the deconvolutional reconstruction unit consists of three deconvolutional layers. Before reconstructing the image, the 16-D feature vector in the ultimate capsule is converted into a [4, 4, 1] feature map, which is used as the input to the deconvolutional layer.

4.1.1 *Extraction and fusion unit*

The extraction and fusion unit are applied between the Prim-Caps and Caps layers. In the extraction unit, the feature extractor and the spatial relationship extractor are both set to [8, 10, 10]. They transform the 8-D feature vector into a feature matrix of [10, 10].

The loss function used in FSc-CapsNet is shown in Eq. (8). We use Adam gradient descent optimization algorithm[20] during training, and the learning rate starts from 0.01. The entire model is built on PyTorch. The batch size is set to 128, and the hyperparameters for Adam are set to their defaults.

5 Results

The standard capsule network is used as a baseline in this paper along with several of its derivatives. The standard capsule network consists of a four-layer structure: a Conv layer, a Prim-Caps layer, a Caps layer, and a reconstruction unit. The parameter settings in the first three layers are the same as those in FSc-CapsNet, and dynamic routing is applied between the Prim-Caps layer and the Caps layer. The final reconstructed structure consists of 512, 1024, and 784 channels in three fully connected layers. We also selected two derivatives of the standard capsule network: MS-CapsNet[12] and TextCaps.[13] Both of these derivatives were applied to the Fashion-MNIST and CIFAR-10 datasets, and the main improvements from both variants also lie in their feature extraction processes. In terms of accuracy, both of these models exceed that of the standard capsule network.

We evaluated our experiments on the Fashion-MNIST and CIFAR-10 datasets. Table 2 shows a comparison of between our experimental group and some baseline. The baseline group includes four CapsuleNet: standard capsule network, MS-CapsNet, MS-CapsNet with Dropout, and TextCaps. The experimental group has four FSc-CapsNet models: standard FSc-CapsNet, FSc-CapsNet with a fully connected reconstruction unit, FSc-CapsNet with a large-scale feature matrix, and FSc-CapsNet with more larger-scale feature matrices. Insofar as was possible, we used the same parameter settings for all of the CapsuleNet to ensure a fair comparison.

Table 2 shows the average accuracy and parameter sizes of the baseline and FSc-CapsNet methods on both Fashion-MNIST and CIFAR-10. The comparison of FSc-CapsNet with several baselines shows that, clearly, FSc-CapsNet models achieve significantly better performances than do the capsule network and its various derivatives. On the Fashion-MNIST dataset, the average accuracy of the standard capsule network is 92.28%, ∼1.41% lower than average accuracy of the standard FSc-CapsNet. After replacing the reconstruction unit with fully connected layers, the average accuracy of FSc-CapsNet was reduced to 93.58%—which was still 1.3% higher than the accuracy of standard capsule network. After extending the feature matrix from [10, 10] to [16, 16], the average accuracy of FSc-CapsNet rose to 1.56% higher than the standard capsule network, reaching 93.84%. We found that, compared with the variants of the capsule network, FSc-CapsNet achieves higher accuracy on Fashion-MNIST than that of MS-CapsNet or MS-CapsNet with Dropout. The performance of FSc-CapsNet is close to that of TextCaps. The difference between the two is ∼0.02%. Moreover, when the extension of the feature matrix is added, FSc-CapsNet performs better than TextCaps.

It is worth mentioning that we attribute the accuracy performance advantage of FSc-CapsNet to the use of spatial relationships for two reasons. First, after the performance of

Table 2 The comparison of average accuracy and parameter sizes of the baseline and FSc-CapsNet methods on both Fashion-MNIST and CIFAR-10 (M is for millions).

	MNIST (%)	Fashion-MNIST (%)	Par (M)	CIFAR-10 (%)	Par (M)
CapsuleNet	99.56 ± 0.031	92.28 ± 0.22	5.4	72.56 ± 0.42	6.2
MS-CapsNet	99.58 ± 0.011	93.10 ± 0.45	10.8	75.1 ± 0.37	11.2
MS-CapsNet + Dropout[a]	99.58 ± 0.015	93.40 ± 0.38	10.8	75.7 ± 0.55	11.2
TextCaps	99.62 ± 0.018	93.71 ± 0.44	17.4	78.69 ± 0.27	18.2
FSc-CapsNet	**99.61 ± 0.014**	**93.69 ± 0.28**	6.8	**79.10 ± 0.33**	9.6
FSc-CapsNet + FC[b]	**99.59 ± 0.028**	**93.58 ± 0.32**	7.5	**78.73 ± 0.19**	11.3
FSc-CapsNet + LM[c]	**99.62 ± 0.011**	**93.84 ± 0.16**	12.3	**79.64 ± 0.21**	18.7
FSc-CapsNet + MLM[d]	**99.62 ± 0.015**	**93.82 ± 0.21**	22.1	**79.88 ± 0.15**	34.0

[a]MS-CapsNet + Dropout is a derivative of MS-CapsNet.
[b]FSc-CapsNet + FC refers to FSc-CapsNet with the fully connected layer.
[c]FSc-CapsNet + LM is a derivative that extends the dimension of the feature matrix to [16, 16].
[d]FSc-CapsNet + MLM is a derivative that extends the dimension of the feature matrix to [20, 20].
Note: Bold values indicate results that are better than capsule network.

the capsule network reaches a high enough level, the increase in the parameter scale cannot get a better performance. We have repeatedly tested some CapsuleNet with larger parameter scales, and none of them can achieve better performance. Second, compared with the capsule network, the improvement of the FSc-CapsNet parameter is mainly concentrated in the 3-D convolution layer of the encoder structure, but in the feature abstraction part that exists in both models, the parameters of FSc-CapsNet are even less than the capsule network. In FSc-CapsNet, the parameters in the 3-D convolution layer are mainly used to find the optimal combination of spatial relationships, that is, the introduction of spatial relationships has led to an increase in the number of parameters in this part. Therefore, the fundamental reason why FSc-CapsNet can achieve better performance is always the use of spatial relationships, rather than simply the increase in the parameter.

To better compare FSc-CapsNet and the capsule network in the use of spatial relationships, we selected some samples and have shown their reconstructed images obtained by the two models in Fig. 10. As shown in Fig. 10, there are eight original samples in the upper part of the subgraph, and the bottom half is the reconstructed images. Compared with the capsule network, the reconstructed image of FSc-CapsNet is closer to the original image, and the boundaries of the entity in the images are clearer. After comparing the reconstructed image with the original image one by one, it was found that there were no missing parts in the reconstructed image.

Among the four variants of FSc-CapsNet, we can see that proper feature matrix selection leads FSc-CapsNet to achieve higher accuracy. Compared with that of the standard FSc-CapsNet, the accuracy of FSc-CapsNet with the feature matrix extension increases by 0.15%. That is, a [16, 16] feature matrix on the Fashion-MNIST dataset achieves better performance

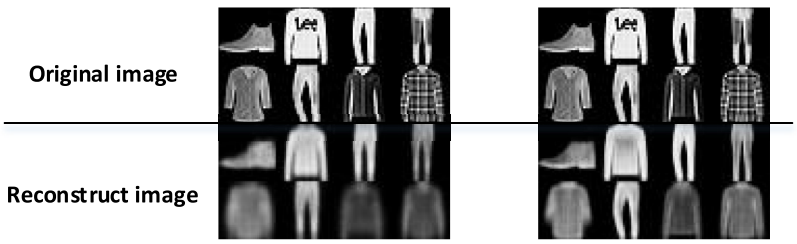

Fig. 10 The reconstructed images of some samples obtained by (a) the capsule network and (b) FSc-CapsNet.

than a [10, 10] feature matrix, but continuing to extend the feature matrix to [20, 20] did not yield a significant performance increase. In fact, a slight drop occurred. We believe this result is because the samples in Fashion-MNIST are grayscale. Thus, the feature information carried by each sample is insufficient to support such a large feature matrix, which ultimately leads to overfitting.

After comparing the performances of the standard FSc-CapsNet and FSc-CapsNet with a fully connected reconstruction unit, we found that this version of the deconvolution reconstruction unit performs sufficiently well to be able to replace the fully connected layer. The standard FSc-CapsNet achieves an accuracy score of 0.11% higher than the FSc-CapsNet with a fully connected reconstruction unit. Moreover, in terms of parameter size, the former is reduced by ∼10%, compared with the latter. We believe this occurs because the deconvolutional layer upsamples the output of the encoder, further weakening the interference of noise features. The deconvolutional layer can be thought of as a pooling layer added to the encoder, which enhances the generalizability of the encoder and reduces the possibility of overfitting. Therefore, the deconvolutional layer is more suitable for FSc-CapsNet than is the fully connected layer.

We also applied our FSc-CapsNet to the CIFAR-10 dataset. CIFAR-10 is a color-image dataset more similar to natural objects. Consequently, each CIFAR-10 sample is a three-channel RGB image with a size of 32×32. Compared with handwritten characters, CIFAR-10 contains real objects in the real world. Not only do the samples include large amounts of noise, but the spatial relations and features of the objects are not the same. All of these factors make CIFAR-10 difficult to classify.

As shown in Table 2, FSc-CapsNet also achieved a significantly better performance on the CIFAR-10 dataset than did the standard capsule network. The highest average accuracy of the standard capsule network and its variants is 78.69%. The average accuracy achieved by FSc-CapsNet is 79.73%—1.04% higher than TextCaps and 7.17% higher than the standard capsule network. FSc-CapsNet with larger-scale feature matrices can even achieve an accuracy of 79.64%. FSc-CapsNet has a greater accuracy advantage on the CIFAR-10 dataset than on the Fashion-MNIST dataset. We believe this is due to the filtering of noise features during feature extraction and the guidance of spatial relationships. The former reduces noise feature interference, while the latter optimizes the feature combination. In addition, after analyzing the variants of FSc-CapsNet, we found that FSc-CapsNet achieved a higher accuracy improvement on CIFAR-10 than on Fashion-MNIST after extending the feature matrix to [20, 20]. After extending the feature matrix from [16, 16] to [20, 20], the accuracy achieved by FSc-CapsNet was reduced by 0.02% on Fashion-MNIST, while on CIFAR-10, the accuracy was improved by 0.24%. We believe this is because the samples in CIFAR-10 carry more information that can support a larger feature matrix without overfitting. Therefore, a larger feature matrix leads to better accuracy performance.

As shown in Table 3, we also tested three mainstream deep learning architectures, VGG-16,[21] GoogLeNet,[22] and ResNet-18,[23] and compared their best accuracies with the derivatives of the

Table 3 The accuracy performance and parameter numbers of some mainstream models.

	Fashion-MNIST (%)	CIFAR-10(%)
CapsuleNet	92.50	72.98
MS-CapsNet	93.55	75.47
TextCaps	94.15	78.96
FSc-CapsNet + MLM	**94.01**	**80.03**
VGG-16	93.50	93.40
GoogLeNet	93.70	92.07
ResNet-18	**94.90**	**93.82**

standard capsule network on the Fashion-MNIST and CIFAR-10 datasets. FSc-CapsNet achieves competitive performance on Fashion-MNIST. FSc-CapsNet scores lower than only TextCaps and ResNet-18 and higher than VGG-16 and GoogLeNet. On CIFAR-10, the accuracy of FSc-CapsNet surpasses that of TextCaps, which has the highest accuracy of all of the capsule network variants. In contrast to the mainstream deep learning frameworks, the accuracy of FSc-CapsNet cannot be called excellent.

In fact, the performance of all CapsuleNet and their variants in the CIFAR-10 dataset is far inferior to that of mainstream CNNs (for details, refer to Table 1 in Sec. 5). There are two main reasons for the poor performance of the capsule network on the CIFAR-10 dataset. On the one hand, the capsule network structure is sensitive to noise features, and the RGB image has many more noise features than the grayscale image. On the other hand, the capsule network has no pooling layer, which makes it difficult for the capsule network to process input images with complex spatial relationships. FSc-CapsNet reduces the impact of noise features on classification results through feature and spatial relationship extractors. In addition, the feature and spatial relationship encoder and deconvolution reconstruction unit play the role of a pooling layer to a certain extent, further weakening the influence of noise features. From Table 1, it can be clearly seen that the performance of FSc-CapsNet compared with that of the capsule network has been greatly improved on the CIFAR-10 dataset. In other words, the above modifications improve the robust to noise features of FSc-CapsNet.

However, it must be acknowledged that, although improved in various ways, FSc-CapsNet still follows the basic framework of the capsule network and has not been able to completely eliminate the influence of noise features. This has led to the performance of FSc-CapsNet in the CIFAR-10 dataset that is still not as good as the mainstream CNNs.

5.1 *Computational Cost and Convergence Speed*

We define three different computational costs to compare the differences between FSc-CapsNet and capsule network in detail, including absolute computational cost, relative computational cost, and effective computational cost.

Absolute computational cost refers to the ratio of the calculation time to the number of iterations during training. Relative calculation cost refers to the calculation time required to reach a given accuracy. Effective calculation cost refers to the calculation time required for the model to reach the best accuracy. On the same computing platform, we believe that the training time can reflect the computational cost of the model during training.

On Fashion-MNIST, the difference between FSc-CapsNet and the capsule network in three different computational costs is recorded in Table 4. After repeated analysis and comparison, we found that in terms of absolute computational cost, compared with the capsule network, FSc-CapsNet takes approximately three times the cost to iterate once. We believe that the 3-D convolution layer has caused a significant increase in computational cost. At present, it is inevitable that 3-D convolution is more computationally expensive than 2-D convolution.

Although FSc-CapsNet is much higher than the capsule network in terms of absolute computational cost, it can be clearly found that there is no particularly large gap after comparing the relative calculation costs and even the FSc-CapsNet is slightly lower than the capsule network.

Table 4 The difference between FSc-CapsNet and capsule network in three kinds of computational costs on Fashion-MNIST.

			RC[c] (ks)			
	AC[a] (ms)	EC[b] (ks)	90.0%	91.0%	92.0%	93.0%
CapsuleNet	0.85	8.10	0.76	3.06	6.65	—
FSc-CapsNet	2.35	17.3	1.51	3.02	5.66	9.85

[a]AC refers to absolute computational cost.
[b]EC refers to effective computational cost.
[c]RC refers to relative computational cost

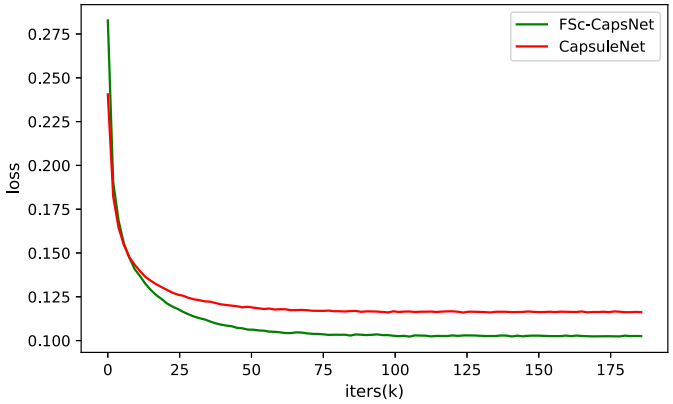

Fig. 11 The loss function changes of FSc-CapsNet and capsule network with the number of iterations.

The faster convergence speed of FSc-CapsNet is the main reason for the above differences. We show the changes in loss function of FSc-CapsNet and the capsule network with the number of iterations on the Fashion-MNIST and CIFAR-10 dataset in Fig. 11. Obviously, FSc-CapsNet can achieve better accuracy performance with fewer iterations. In other words, its recognition efficiency is higher, which leads to a lower level of relative computational cost and effective computational cost. Table 4 also shows the effective computational cost of the two models. FSc-CapsNet requires approximately 2.1 times the calculation time of the capsule network. Given that FSc-CapsNet can achieve an accuracy performance of ∼1% higher than the capsule network, we believe that the computational cost investment is acceptable.

In addition, we conducted an experiment on the complete SVHN dataset.[24] Using this dataset, the input size is extended to [32, 32], and the number of channels is expanded to three. The best accuracy achieved by FSc-CapsNet is 93.47%, which is significantly better than the standard capsule network (by 2.96%), which achieved only 90.51%.

6 Conclusions

In this paper, we propose a network called FSc-CapsNet. Compared with the original capsule network and its multiple derivatives, the experimental results show that FSc-CapsNet achieves significantly better accuracy on both Fashion-MNIST and CIFAR-10 datasets. In addition, compared with some mainstream deep learning frameworks, FSc-CapsNet obtains a very competitive performance on Fashion-MNIST.

The feature and spatial relationship coding structure defines the feature and spatial relationship extractor and encoder. The spatial relationship extractor maps feature vectors to feature matrices and introduces top–down spatial relationship information flow into the capsule network, and the separate extraction unit reduces the influence of noise features on the classification results. The feature and spatial relationship encoder replaces the dynamic routing and synchronously implements abstraction and combination of feature and spatial relationships through the 3-D convolutional kernel, which put the optimization process of feature combination into the backpropagation. The deconvolution layer adds pooling to the encoder, further weakening the negative impacts of noise features and reducing the possibility of overfitting.

Although the performance of FSc-CapsNet on CIFAR-10 is better than that of CapsuleNet and their multiple derivatives, there is still a gap compared with the accuracy of mainstream CNNs. The introduction of 3-D convolutional layers inevitably increases the computational cost in training. Thus, a better and more efficient encoder structure needs to be explored in ongoing research.

Acknowledgments

The authors declare that they have no conflicts of interest to this work.

References

1. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.(NIPs)*, pp. 1097–1105 (2012).
2. S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.(NIPS)*, pp. 3856–3866 (2017).
3. G. E. Hinton, A. Krizhevsky, and S. Wang, "Transforming auto-encoders," *Lect. Notes Comput. Sci.* **6791**, 44–51 (2011).
4. J. Shuiwang, Y. Ming, and Y. Kai, "3D convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(1), 221–231 (2013).
5. M. D. Zeiler et al., "Deconvolutional networks," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recognit.*, pp. 2528–2535 (2010).
6. M. Jaderberg et al., "Spatial transformer networks," in *Adv. Neural Inf. Process. Syst. 28* (2015).
7. J. Dai et al., "Deformable convolutional networks," in *IEEE Int. Conf. Comput. Vision (ICCV)*, pp. 764–773 (2017).
8. D. Worrall et al., "Harmonic networks: deep translation and rotation equivariance," in *IEEE Conf. Comput. Vision and Pattern Recognit.* (2017).
9. A. Jaiswal et al., "CapsuleGAN: generative adversarial capsule network," in *European Conf. Comput. Vision (ECCV) Workshops*, pp. 112–117 (2018).
10. C. Fang, Y. Shang, and D. Xu, "Improving protein gamma-turn prediction using inception capsule networks," *Sci. Rep.* **8**, 15741 (2018).
11. Z. Chen and D. Crandall, "Generalized capsule networks with trainable routing procedure," *Comput. Res. Repository (CoRR)*, arXiv:1808.08692 (2018).
12. X. Canqun et al., "MS-CapsNet: a novel multi-scale capsule network," *IEEE Signal Process. Lett.* **25**, 1850–1854 (2018).
13. V. Jayasundara et al., "TextCaps: handwritten character recognition with very small datasets," in *IEEE Winter Conf. Appl. Comput. Vision*, pp. 254–262 (2019).
14. H. Wallach et al., "Stacked capsule autoencoders," in *Int. Conf. Neural Inf. Process. Syst.(NIPS)*, pp. 15512–15522 (2019).
15. D. Amara, "Novel deep learning model for traffic sign detection using capsule networks," *Int. J. Pure Appl. Math.* **118**, 4543–4548 (2018).
16. W. Zhao et al., "Investigating capsule networks with dynamic routing for text classification," *Comput. Res. Repository (CoRR)* (2018).
17. X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. and Stat.*, Vol. 15, pp. 315–323 (2011).
18. H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *Comput. Res. Repository (CoRR)* (2017).
19. A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep. 1, Computer Science Department, University of Toronto (2009).
20. D. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *Int. Conf. Learn. Represent.* (2014).
21. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Comput. Res. Repository (CoRR)* (2014).
22. C. Szegedy et al., "Going deeper with convolutions," *Comput. Res. Repository (CoRR)* (2014).
23. K. He et al., "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 770–778 (2016).
24. Y. Netzer et al., "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop Deep Learn. and Unsupervised Feature Learn.* (2011).

Tenghao Han received his BE degree in computer science and technology from Qingdao University, Qingdao, Shandong, in 2017. Currently, he is pursuing his ME degree in computer science and technology at Qingdao University.

Rencheng Sun received his BS degree in mathematics education in 2003, his MS degree in computer software theory in 2006 and his PhD in system theory from Qingdao University in 2010.

Fengjing Shao received her bachelor's degree in computer science from Shandong University in 1982. In 1988, she received her master's in business administration from Shiga University. She received her doctorate in technology from Osaka University in 1991.

Yi Sui received her BS degree in computer science technology in 2006, her MS degree in computer software theory in 2009, and her PhD in system theory from Qingdao University in 2012.