# Capsule Networks With Residual Pose Routing

Yi Liu, De Cheng, Dingwen Zhang, *Member, IEEE*, Shoukun Xu, and Jungong Han, *Senior Member, IEEE*

***Abstract*— Capsule networks (CapsNets) have been known difficult to develop a deeper architecture, which is desirable for high performance in the deep learning era, due to the complex capsule routing algorithms. In this article, we present a simple yet effective capsule routing algorithm, which is presented by a residual pose routing. Specifically, the higher-layer capsule pose is achieved by an identity mapping on the adjacently lower-layer capsule pose. Such simple residual pose routing has two advantages: 1) reducing the routing computation complexity and 2) avoiding gradient vanishing due to its residual learning framework. On top of that, we explicitly reformulate the capsule layers by building a residual pose block. Stacking multiple such blocks results in a deep residual CapsNets (ResCaps) with a ResNet-like architecture. Results on MNIST, AffNIST, SmallNORB, and CIFAR-10/100 show the effectiveness of ResCaps for image classification. Furthermore, we successfully extend our residual pose routing to large-scale real-world applications, including 3-D object reconstruction and classification, and 2-D saliency dense prediction. The source code has been released on https://github.com/liuyi1989/ResCaps.**

***Index Terms*— 3-D point cloud, capsule network (CapsNet), part-whole, residual routing, salient object detection.**

## I. Introduction

CONVOLUTIONAL neural networks (CNNs) have been known as a classic architecture for image recognition due to their high representation power. They can recognize the image by detecting the existence of a specific entity, i.e., invariance. However, an unsophisticated perturbation on the image can fool a well-trained network to fail in recognition [1], [2], [3], [4]. More worryingly, natural and non-adversarial pose changes of familiar objects in the real world are enough to trick deep networks [5], [6].

Instead of pursuing invariance, some researchers advocate the equivariance property in a neural network [7], [8]. A recently developed neural architecture, called capsule networks (CapsNets) [9], [10], was proposed to achieve this argument by encapsulating the poses (instantiation parameters) of an entity in a group of neurons. The routing-by-agreement mechanism in CapsNets can learn the underlying part-whole spatial relationships for further compositional representations of the scene, which can promisingly help to detect properties of many different levels within one network. This is essentially the human vision ability of scene understanding in psychology. However, it is difficult for the traditional CNN models to capture the part-whole hierarchies of the scene. The reason behind can be illustrated as follows. First, traditional CNN models usually infer with fixed parameters, which cannot dynamically allocate a group of neurons to represent a node in a parse tree. In contrast, CapsNets design a dynamic routing algorithm, which can dynamically allocate neurons to represent a small portion of the visual input. Second, the max pooling, a vital component of traditional CNN models, makes neurons in one layer to focus on the most active feature detector in a local pool in the layer below, which results in difficulty in capturing precise spatial relations between entities. Differently, CapsNets discard the operation of max pooling, which ensures our model not throw away information about the precise position of the entity within the region. In light of these two advantages, traditional CNN models are difficult to capture the part-whole hierarchies of the scene, while CapsNets can solve this issue to capture compositional representations of the image for scene understanding.

However, the existing CapsNets have been criticized for not facilitating deep architectures and large-scale real-world applications due to the heavy capsule routing algorithms, e.g., the dynamic routing algorithm [9] and the expectation-maximization (EM) routing algorithm [10]. First, the fully connected routing mechanism produces a large number of transformation matrix parameters, as can be seen in Fig. 1(a). Second, the unsupervised computational routing procedure is computationally expensive. Third, the adopted routing-by-agreement mechanism [9], [10] assumes a cluster distribution of predictions, which may fail when there exist a number of noisy prediction poses or the input data are out of distribution. While some simplified CapsNets [11], [12], [13], [14]

Manuscript received 4 October 2022; revised 31 March 2023, 17 August 2023, and 6 December 2023; accepted 22 December 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62001341, Grant 62293543, Grant 62322605, and Grant U21B2048; in part by the National Natural Science Foundation of Jiangsu Province under Grant BK20221379; in part by the State Key Laboratory of Reliability and Intelligence of Electrical Equipment, Hebei University of Technology under Grant EERI KF2022005; in part by the Changzhou University CNPC-CZU Innovation Alliance under Grant CCIA2023-01; in part by the Changzhou Leading Innovative Talent Introduction and Cultivation Project under Grant 20221460; in part by the Open Fund of the Key Laboratory of Big Data Intelligent Computing for Chongqing University of Posts and Telecommunications under Grant BDIC-2023-A-004; and in part by the Fundamental Research Funds for the Central Universities under Grant ZYTS23173. *(Yi Liu and De Cheng contributed equally to this work.) (Corresponding authors: Dingwen Zhang; Jungong Han.)*

Yi Liu is with the School of Computer Science and Artificial Intelligence, the Aliyun School of Big Data, the School of Software, and CNPC-CZU Innovation Alliance, Changzhou University, Changzhou, Jiangsu 213164, China (e-mail: liuyi0089@gmail.com).

De Cheng is with the School of Telecommunications Engineering, Xidian University, Xi'an, Shaanxi 710071, China (e-mail: dcheng@xidian.edu.cn).

Dingwen Zhang is with the School of Automation, Northwestern Polytechnical University, Xi'an, Shaanxi 710129, China (e-mail: zhangdingwen2006yyy@gmail.com).

Shoukun Xu is with the School of Computer Science and Artificial Intelligence, the Aliyun School of Big Data, and the School of Software, Changzhou University, Changzhou, Jiangsu 213164, China (e-mail: jpuxsk@163.com).

Jungong Han is with the Department of Computer Science, University of Sheffield, S10 2TN Sheffield, U.K. (e-mail: jungonghan77@gmail.com).

Digital Object Identifier 10.1109/TNNLS.2023.3347722

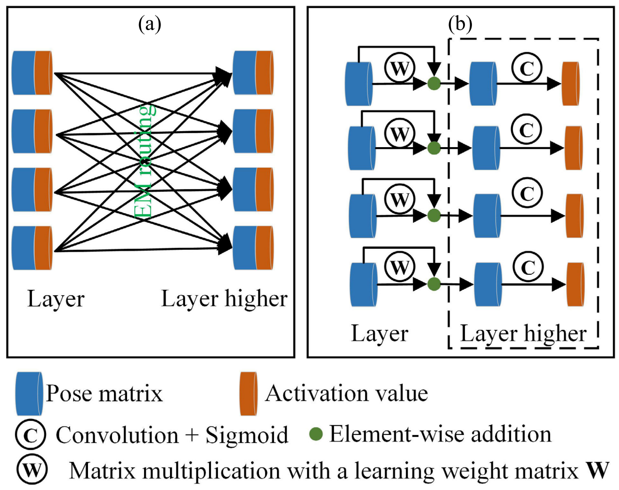


Fig. 1. Residual pose routing versus EM routing. The fully connected EM routing causes large parameters and heavy computation, while our residual pose routing employs the sparsely connected routing, getting parameters reduced, and computation efficient. The pose matrices and activation values of all learned capsules are integrated via convolution and ReLU to achieve the capsule features. (a) EM routing. (b) Residual pose routing.

use auxiliary knowledge like attention, their overall performance is still not satisfactory in terms of a high computation consumption and low recognition accuracy.

From what has been discussed above, the limitation of CapsNets makes a challenge for a deeper CapsNet. First, the existing CapsNets mostly cannot be easily plugged in deep structures owing to their complex routing algorithms. Second, even though the current CapsNets have achieved promising performance on small-scale images and simple tasks, e.g., recognition for digital images and CIFAR, which share the image sizes of $28 \times 28$ and $32 \times 32$, it is a challenge to tackle the large-scale images and complex tasks. This may be because large-scale images and complex tasks contain complicated linear nonseparable problems for the simple shallow CapsNets that include three capsule routing layers. To this end, the deeper architecture of CapsNets is a solution to fit the linear nonseparable problems to solve complex issues. Besides, compared with a shallow network, a deeper architecture will have a powerful feature representation, which helps to capture the semantics in complicated scenes. This is a distinctive characteristic of the deep learning era. Therefore, a deep architecture for CapsNets is an urgent challenge to be solved.

In this article, we propose a simple but powerful capsule routing to implement deep CapsNet. By revisiting the routing-by-agreement procedure in CapsNet [10], we discover that a capsule is able to represent its associated capsules in terms of the pose via a *learned transformation weight matrix*. In this sense, the higher-layer capsule pose can be computed by integrating the lower-level capsule pose and its associates' poses, which is implemented by an identity mapping in Fig. 1(b). The activation of the higher-layer capsule can be implicitly computed from the pose matrix. Such residual pose routing can compute all the higher-layer capsules in a parallel and unidirectional manner, as shown in Fig. 1(b). On top of that, we integrate the learned capsules to achieve the association between each capsule and other capsules in the same layer. Our residual pose routing has three advantages: 1) our sparsely unidirectional connected fashion greatly reduces the network parameters, compared with the reciprocating iterative fully connected pattern; 2) thanks to 1), our model speeds up the inference stage (as will be verified in Section V-B4) and training stage (∼10% training time per epoch compared with the original EM routing algorithm); 3) our routing learns to fit the clustering distribution, which further improves the representation ability for finding the distribution of high-dimension data in complex scenarios, compared to the unsupervised clustering routing in [10]; and 4) our routing is capable of avoiding vanishing gradient, which makes it possible to design a deeper CapsNet architecture.

Inspired by the simplicity and effectiveness of the proposed residual pose routing, we build a *deep* CapsNet architecture, which consists of five capsule routing blocks. Specifically, each of the first four blocks is composed of one Primary Capsule (*PrimaryCaps*) layer and two residual pose routing (*ResP*) layers with the purpose of capsules construction and residual pose routing, respectively. The last block contains one *PriCaps* layer and one residual pose classification (*ResPC*) layer for the purpose of image classification. The downsampling stride of 2 is utilized between two blocks, resulting in a deep ResNet-like architecture. Evaluations on MNIST [15], AffNIST, smallNORB [16], and CIFAR-10/100 [17] show that such deep CapsNet significantly increases the accuracy of image classification.

Thanks to the lightweight of our residual pose routing, we generalize it to the tasks of 3-D reconstruction/classification and 2-D image saliency dense prediction, which are typical large-scale real-world applications. On top of the framework of [18], we incorporate our residual pose routing to explore part-whole relationships for 3-D reconstruction/classification, and get promising gains on ModelNet40 [19]. Besides, on top of the framework of [20], we incorporate our residual pose routing for visual saliency, and prove our algorithm to be simple yet effective compared with the previous part-whole relational saliency methods, particularly in scenarios where lightweight backbone models, e.g., VGG-16, are employed.

The main contributions of this article include the following.

1) A novel residual pose routing algorithm is proposed, which greatly reduces routing parameters and computational complexity.

2) A deep ResNet-like CapsNet architecture thanks to residual pose routing's ability of avoiding gradient vanishing.

3) Successful showcases of our residual pose routing in multiple real-world tasks, such as 3-D reconstruction/classification and 2-D image saliency dense prediction, demonstrate that residual CapsNets (ResCaps) can be well generalized to large-scale real-world applications.

The article is organized as follows. Section II reviews the related work to our model. Section III describes the details of the proposed framework. Section IV illustrates the architecture of our ResCaps network. Section V implements abundant experiments and analysis to study our model. Section VI concludes the article.

## II. Related Work

In this section, we will review the related work to our network, focusing on the capsule routing algorithm, CapsNet architecture, CapsNet-related applications, and residual learning.

### A. Capsule Routing Algorithm

The capsule routing algorithm plays a fundamental role in CapsNet and performs capsules' assignments across adjacent layers. The dynamic routing [9] and EM routing [10] algorithms have been widely recognized for capsules' assignments. The former generates capsules by computing the vector similarity of different low-layer capsules, while the latter computes the capsule pose and activation via executing the EM algorithm on various low-layer capsules. Inspired by their ideas, many variants of capsule routing have been proposed. For example, Hahn et al. [13] introduced a self-routing strategy for capsules' assignments. Ahmed and Torresani [21] utilized a straight-through attentive mechanism for routing coefficients estimation. Ribeiro et al. [22] introduced the uncertainty to the capsule routing to generate global part-whole assignments. Li et al. [23] approximated the routing process by interacting two branches in a supervised manner. Feng et al. [24] proposed a dual-routing capsule graph neural network to solve the problem of few-shot video classification. These methods estimate the capsule routing with some knowledge, e.g., attention. However, they focus less on the lightweight of the capsule routing.

Differently, our residual pose routing 1) generates the capsules in a unidirectional forward manner instead of a recurrent procedure adopted by the dynamic routing and the EM routing algorithms, which can significantly simplify the routing computation; and 2) employs the sparsely connected routing instead of the fully connected routing used by most of the previous capsule routing strategies, which reduces the routing parameters.

### B. CapsNet Architecture

The early CapsNet architecture simply employed a transforming autoencoder [8] to compute the existence probability and spatial location of an entity. Later, they took a milestone step to design the vector CapsNet [9], in which one primary capsule layer was implemented for capsules construction, and one decoder layer for digit reconstruction. They further consolidated it by proposing the Matrix CapsNet [10], which contained one primary capsule layers, two convolutional capsule layers, and one capsule classification layer. Besides, various architectures have been designed for CapsNet. For instance, Lenssen et al. [11] presented a group CapsNet to enhance equivariance properties. Rajasegaran et al. [12] implemented CapsNet with the 3-D convolution. Chen et al. [25] devised a set of optimizable receptors and a transmitter for capsule representation. Vasantharao et al. [26] injected the spatial transformation network into the CapsNet via latent code manipulation. Tao et al. [27] replaced the primary capsule layer of the original CapsNet with an adaptive capsule layer, which preserved the spatial information for each capsule and local relations among capsules. These methods achieve some progress for CapsNets architectures with various operations, e.g., 3-D convolution. However, they still cannot address the demand of the deep CapsNets architecture.

Differently, our ResCaps architecture makes it real for the deep CapsNet, which benefits from our lightweight residual capsule routing algorithm. Specifically, our ResCaps designs a deep ResNet-like architecture composed of five capsule blocks, each of which contains one capsules construction layer and several capsule routing layers.

### C. CapsNet-Related Applications

In light of the excellent property of CapsNet, they have been successfully embedded in many tasks. For instance, Liu et al. [28] employed CapsNets to visual saliency, in which CapsNets were utilized to explore the part-whole relationships in the image to achieve the whole object saliency. Following that, contrast cues derived by CNNs and part-whole relations discovered by CapsNets were integrated to complement each other for better saliency detection [29] and camouflaged detection [30]. Besides, CapsNets endowed the spatial–temporal relationships for regression tracking [31], where spatial CapsNet and temporal CapsNet were designed to encode spatial relationships and temporal relationships, respectively. CapsNets were also successfully embedded in the task of visual question answering with the aim of finding the relevant regions [32] and merging parts with human-prior hierarchies [33]. Garau et al. [34] made use of the viewpoint-equivariance of CapsNet to solve the problem of human pose estimation. Yu et al. [35] discovered face parts with the aid of hierarchical CapsNets. Sun et al. [36] embedded CapsNet for learning canonical pose in 3-D point cloud. Zhao et al. [18] used CapsNet to sparse 3-D point clouds while preserving spatial arrangements of the input data, where the 2-D latent space brought in improvements for several common point cloud-related tasks. Zhuge et al. [20] employed CapsNets to extract part-whole semantics to improve the micro-level integrity for each salient object. Wu et al. [37] devised a user-specific capsule module and a position-aware gating module to capture the sequential patterns at union-level and point-level for the issues of next-item recommendation. Cheng et al. [38] utilized the dynamic routing in the encoder and a static routing in the decoder for zero-shot learning. Wang et al. [39] designed a group CapsNet to segment the hemorrhage region from a non-contract CT scan. Ma and Wu [40] utilized the CapsNet to explore the part-whole relations for regression tracking. Bonheur et al. [41] proposed an only CapsNet for multilabel semantic segmentation.

In this article, we select the object understanding in 3-D point clouds and 2-D image salient object detection to investigate the capacity of our proposed capsule routing. Different from the previous CapsNet-based 3-D object understanding (i.e., PointCaps [18]) and CapsNet-based image salient object detection (i.e., TSPOANet [42]) that adopted the heavy routing [9], [10] for scene parsing, our proposed residual pose routing can not only present lightweight with fewer parameters and simple routing complexity, but also achieve better parsing performance.

### D. Residual Learning

Residual learning was derived from the residual network [43], in which the identity mapping [44] was realized by the residual connection to achieve a deep network while avoiding the gradient vanishing. Thereafter, residual learning has been utilized widely to advance many tasks. For example, Ke et al. [45] designed a side-output residual network to fit the errors between ground-truth and the outputs of the stacked residual units, enforcing the modeling capability to symmetry in complex backgrounds. Tran et al. [46] imputed the missing modality with a cascaded residual autoencoder. Feichtenhofer et al. [47] designed a spatial–temporal residual unit for dynamic scene recognition. Wang et al. [48] used residual skip connection to learn attentions in a Siamese network for online visual tracking.

In view of the power of the residual learning, it has been successfully used for CapsNet. For example, Gugglberger et al. [49] trained a deep CapsNet with a residual connection to integrate the input and its capsule routing out. Ding et al. [50] added a capsule based max-pooling as a skip connection to adaptively choose the routing or max-pooling to reduce computational complexity.

In this work, we employ residual connection in the capsule routing to learn residual part-whole hierarchies in a scene. Different from [49] that used residual connection outside the capsule routing in a straightforward way, we embedded the residual learning inside the capsule routing for pose computation, which learns a residual routing and results in a deep network with avoiding the gradient vanishing problem.

## III. Proposed Residual Pose Routing

### A. Revisiting the Routing-by-Agreement Mechanism in EM Routing

The EM routing algorithm in the matrix CapsNet [10] assumes a Gaussian distribution for capsules. On top of that, the EM algorithm clusters associated capsules together to compose a whole. Based on this routing-by-agreement mechanism in the EM routing algorithm [10], the associated $m^L$ capsules $(\mathbf{Cap}_1^L, \mathbf{Cap}_2^L, \ldots, \mathbf{Cap}_{m^L}^L)$ in layer $L$ will be clustered to compose a whole capsule $\mathbf{Cap}_j^{L+1}$ in layer $(L+1)$, i.e.,

$$\mathbf{Cap}_j^{L+1} = f_{\text{rou}}\left(\mathbf{Cap}_1^L, \mathbf{Cap}_2^L, \ldots, \mathbf{Cap}_{m^L}^L\right) \tag{1}$$

where $f_{\text{rou}}$ represents the routing algorithm, e.g., the EM routing algorithm in [10].

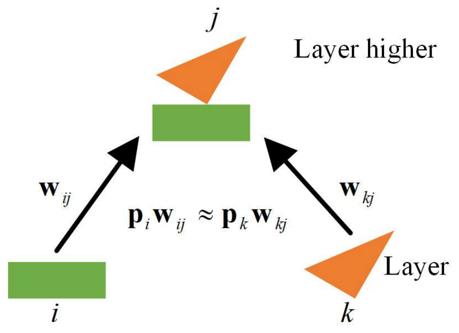
Focusing on (1), the whole $\mathbf{Cap}_j^{L+1}$ can be composed of $m_L$ part capsules. We can imagine that the whole capsule $\mathbf{Cap}_j^{L+1}$ can be composed of $t^L$ sub-whole capsules, i.e.,

$$\mathbf{Cap}_j^{L+1}{=}f_{\text{rou}}\left(\mathbf{SubCap}_1^L, \mathbf{SubCap}_2^L, \ldots, \mathbf{SubCap}_{t^L}^L\right). \tag{2}$$

Similarly, each sub-whole capsule $\mathbf{SubCap}_i^L$ can be composed of several sub-whole capsules further, each of which can be composed of several associated part capsules in layer $L$. For the basic case, each sub-whole consists of two associated part capsules in layer $L$.

Let us discuss the routing-by-agreement for two associated capsules. Suppose capsules $i$[1] and $k$ in one layer and capsule $j$ in the higher layer have pose matrices $\mathbf{p}_i$, $\mathbf{p}_k$, and $\mathbf{p}_j$, respectively, all with the dimension of $4 \times 4$. The finding for the routing-by-agreement mechanism will be elaborated as follows.

[1]In this article, capsule $i$ refers to the $i$th type capsule.



j
Layer higher
$\mathbf{w}_{ij}$
$\mathbf{w}_{kj}$
$\mathbf{p}_i\mathbf{w}_{ij} \approx \mathbf{p}_k\mathbf{w}_{kj}$
Layer
i
k

Fig. 2. Illustration for the routing-by-agreement mechanism. When associated capsule $i$ and capsule $k$ make their familiar whole capsule $j$ in the higher layer, their votes are approximately equal.

*Step 1: Vote computation for the viewpoint invariant relations.*

The viewpoint invariant relations from capsule $i$ to the adjacently higher-layer capsule $j$ can be revealed by vote $\mathbf{v}_{ij}$, which is computed by multiplying the pose matrix $\mathbf{p_i}$ of capsule $i$ with a viewpoint invariant transformation $\mathbf{w}_{ij}$, i.e.,

$$\mathbf{v}_{ij} = \mathbf{p}_i\mathbf{w}_{ij} \tag{3}$$

where $\mathbf{w}_{ij}$ is learned discriminatively through a cost function and the backpropagation. It learns not only what a whole is composed of, but it also makes sure the pose information of the parent capsule matched with its sub-components after some transformation.

*Step 2: Routing by agreement.*

In nature, a whole object, e.g., face, can be composed by two familiar parts, e.g., mouth and nose. In other words, these two parts must share familiar attributes, so they can be composed together to a whole. This nature can be revealed by CapsNets. Specifically, as shown in Fig. 2, given familiar capsules, e.g., capsule $i$ and capsule $k$, an adjacently high-layer capsule $j$ is detected by looking for agreement between their viewpoint invariant relations, i.e., $\mathbf{v}_{ij}$ and $\mathbf{v}_{kj}$. Their agreement in the viewpoint invariant relations can be written as

$$\mathbf{v}_{ij} \approx \mathbf{v}_{kj}. \tag{4}$$

Using (3) and (4) becomes the following formulation:

$$\mathbf{p}_i\mathbf{w}_{ij} \approx \mathbf{p}_k\mathbf{w}_{kj} \tag{5}$$

where $\mathbf{w}_{ij}$ and $\mathbf{w}_{kj}$ are the viewpoint invariant transformation matrices.

In addition, the visualization of the intuition in (5) can be found in Fig. 10 of [28].

### B. Residual Pose Routing

Based on the analysis of the routing-by-agreement mechanism, we will develop a residual pose routing algorithm for capsules assignments. In the following, we will describe the details of the residual pose routing algorithm, including pose matrix computation, activation computation, and capsules integration.

*1) Pose Matrix Computation:* From the routing-by-agreement mechanism in (5), we can draw a conclusion, i.e.,

$$\mathbf{p}_k \approx \mathbf{p}_i \frac{\mathbf{w}_{ij}}{\mathbf{w}_{kj}}. \tag{6}$$

In view of the fact that the parameters $\mathbf{w}_{ij}$ and $\mathbf{w}_{kj}$ are both learnable, we can learn a weight matrix $\mathbf{w}_{ik}$ to substitute for $(\mathbf{w}_{ij}/\mathbf{w}_{kj})$, i.e.,

$$\mathbf{p}_k \approx \mathbf{p}_i \mathbf{w}_{ik}. \tag{7}$$

In (6), we can see that capsule $i$ can compute its associated capsule $k$ via a learned transformation matrix with the dimension of $4 \times 4$. More generally, capsule $k$ is the set of associated versions of capsule $i$, which can be encoded by $\mathbf{w}_{ik}$ in (7) implicitly. On top of that, every capsule in one layer can compute its associated sets of capsules.

Capsule $i$ and its associated capsule set $k$ in layer $L$ can make their familiar whole capsule $j$ in layer $(L+1)$. To achieve this, we choose the simple yet effective vector addition in terms of pose matrices $\mathbf{p}_i^L$ and $\mathbf{p}_k^L$, i.e.,

$$\mathbf{p}_j^{L+1} = \mathbf{p}_i^L + \mathbf{p}_k^L = \mathbf{p}_i^L + \mathbf{p}_i^L \mathbf{w}_{ik}^L = \mathbf{p}_i^L (1+\mathbf{w}_{ik}). \tag{8}$$

In (8), the pose of the familiar capsule $j$ in layer $(L+1)$ can be computed by an identity mapping on the pose of capsule $i$ in layer $L$.

In summary, by revisiting the capsule routing-by-agreement in EM CapsNet [10], we discover that when a capsule $i$ and its associated capsule $k$ vote for a higher-layer capsule, the capsule pose $\mathbf{p}_i$ can represent its associated capsule $\mathbf{p}_k$ via a *learned transformation weight matrix*. Based on this intriguing finding, we draw a further conclusion that $\mathbf{p}_k$ can be generalized to the associated capsules' sets' pose of capsule $i$. In this sense, the higher-layer capsule pose is able to be computed by integrating the lower-level capsule pose and its associates' poses, which can be implemented by an identity mapping.

Suppose there exist $N$ type capsules in layer $L$. By using (8), the pose of capsules in layer $(L+1)$ can be computed by

$$\begin{aligned}\mathbf{p}^{L+1} &= \left[\mathbf{p}_1^{L+1}, \mathbf{p}_2^{L+1}, \dots, \mathbf{p}_N^{L+1}\right] \\ &= \left[\mathbf{p}_1^L(1+\mathbf{w}_1), \mathbf{p}_2^L(1+\mathbf{w}_2), \dots, \mathbf{p}_N^L(1+\mathbf{w}_N)\right]\end{aligned} \tag{9}$$

where $\mathbf{w}_i$ is the learned transformation matrix with the dimension of $4 \times 4$. $[\cdot]$ represents the operation of concatenation.

*a) Primitive understanding for residual pose routing:* Apart from the mathematical formulations of (6) and (7), there is a primitive understanding on our residual pose routing, i.e.,

1) $\mathbf{w}_{ij}/\mathbf{w}_{kj}$ learns the transformation relations between capsule $i/k$ and the higher-layer capsule $j$. In doing (6), the transformation relations between associated capsules $i$ and $k$ must be discovered via the higher-layer capsule $j$.

2) $\mathbf{w}_{ik}$ learns the transformation relations between capsule $i$ and capsule $k$ within one layer. In doing (7), the transformation relations between associated capsules $i$ and $k$ can be discovered directly via a learnable matrix $\mathbf{w}_{ik}$.

3) Comparing 1) and 2), $\mathbf{w}_{ij}/\mathbf{w}_{kj}$ and $\mathbf{w}_{ik}$ both compute the transformation relations between associated capsules $i$ and $k$. It is evident that we can replace $\mathbf{w}_{ij}/\mathbf{w}_{kj}$ with $\mathbf{w}_{ik}$.

*2) Activation Value Computation:* The pose matrices of capsules in layer $(L+1)$ have been achieved from capsules in layer $L$. The activation of each capsule in layer $(L+1)$ can be implicitly encoded by its pose matrix. To this end, a convolution operation is carried out on the pose matrix to encode the implicit knowledge of the pose matrix, which is followed by a Sigmoid function to compute the activation value, i.e.,

$$\mathbf{a}_j^{L+1} = f_{\text{Sig}}\left(f_{\text{Conv}}\left(\mathbf{p}_j^{L+1}\right)\right) \tag{10}$$

where $f_{\text{Sig}}(\cdot)$ and $f_{\text{Conv}}(\cdot)$ represent the Sigmoid function and the convolution operation, respectively.[2] Similar to pose, the activation values of capsules in layer $(L+1)$ can be written as

$$\begin{aligned}\mathbf{a}^{L+1} &= \left[\mathbf{a}_1^{L+1}, \mathbf{a}_2^{L+1}, \dots, \mathbf{a}_N^{L+1}\right] \\ &= \left[f_{\text{Sig}}\left(f_{\text{Conv}}\left(\mathbf{p}_1^{L+1}\right)\right), f_{\text{Sig}}\left(f_{\text{Conv}}\left(\mathbf{p}_2^{L+1}\right)\right), \dots,\right. \\ &\quad \left. f_{\text{Sig}}\left(f_{\text{Conv}}\left(\mathbf{p}_N^{L+1}\right)\right)\right].\end{aligned} \tag{11}$$

## C. Capsules Integration

Each capsule can be achieved by concatenating pose matrix $\mathbf{p}_j^{L+1}$ in (8) and activation value $\mathbf{a}_j^{L+1}$ in (10), i.e.,

$$\mathbf{Cap}_j^{L+1} = \left[\mathbf{p}_j^{L+1}, \mathbf{a}_j^{L+1}\right]. \tag{12}$$

However, the individual computation for the pose of each capsule ignores the associations between it and other capsules in the same layer, which will cause some problems. First, the associate capsules set $\mathbf{p}_k$ of capsule $i$, i.e., $\mathbf{p}_i \times \mathbf{w}_{ik}$, may have redundant knowledge with other capsules in layer $L$. Second, the neglect of the association between capsule $i$ and other capsules in layer $L$ may produce weak whole capsule in $(L+1)$. To solve these problems, we integrate the learned capsules into layer $(L+1)$. We choose a simple convolution operation to encode primitive features from the capsules in layer $(L+1)$, i.e.,

$$\begin{aligned}&\mathbf{CapF}^{L+1} \\ &\quad = f_{\text{ReLU}}\left(f_{\text{Conv}}\left(\left[\mathbf{Cap}_1^{L+1}, \mathbf{Cap}_2^{L+1}, \dots, \mathbf{Cap}_N^{L+1}\right]\right)\right)\end{aligned} \tag{13}$$

where $f_{\text{ReLU}}(\cdot)$ represents the activation function of ReLU. $\mathbf{CapF}^{L+1}$ means the primitive features by integrating all the learned capsules in layer $(L+1)$.

By using (13), the learned capsules are integrated together to achieve more primitive features, which can address the routing redundancy and enhance whole representation to some extent. Algorithm 1 illustrates the procedure of the proposed residual pose routing algorithm.

## D. Deeper Insight Into Residual Pose Routing Algorithm

*1) Gradient Vanishing Avoiding:* Denoting the loss function as $\varepsilon$ from the chain rule of back-propagation [51], we can conclude

$$\begin{aligned}\frac{\partial \varepsilon}{\partial \mathbf{p}^L} &= \frac{\partial \varepsilon}{\partial \mathbf{p}^{L+1}} \frac{\partial \mathbf{p}^{L+1}}{\partial \mathbf{p}^L} \\ &= \frac{\partial \varepsilon}{\partial \mathbf{p}^{L+1}} \frac{\partial\left[\mathbf{p}_1^L(1+\mathbf{w}_1), \mathbf{p}_2^L(1+\mathbf{w}_2), \dots, \mathbf{p}_N^L(1+\mathbf{w}_N)\right]}{\partial\left[\mathbf{p}_1^L, \mathbf{p}_2^L, \dots, \mathbf{p}_N^L\right]}\end{aligned}$$

[2]The weight and bias terms are neglected for simplicity in this article.

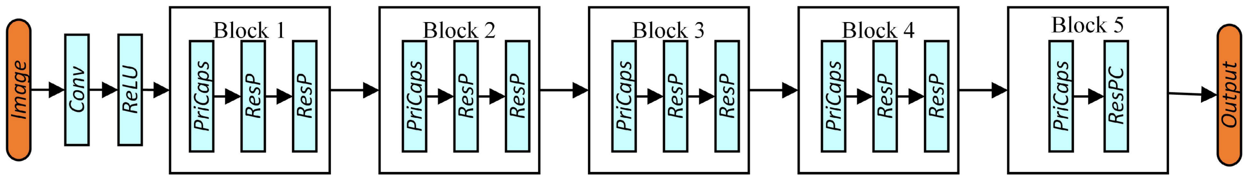


Fig. 3. ResCaps network. The network architecture consists of five blocks. Each of the first four blocks is composed of one *PriCaps* layer and two *ResP* layers. The block five consists of one *PriCaps* layer and one *ResPC* layer.

**Algorithm 1 Residual pose routing algorithm** Returns Pose and Activation of the Capsules in Layer $(L + 1)$ Given the Pose $\mathbf{p}$ of $N$ Type Capsules in Layer $L$ and the Learned Transformation Matrix $\mathbf{w}$ for Each Capsule. $\mathbf{p}_i$ Is the Pose Matrix of Capsule $i$ in Layer $L$

```
procedure RESIDUAL POSE ROUTING ($\mathbf{p}$)
    1. Learn the capsule pose for layer $(L+1)$: $\mathbf{p}_i^{L+1} = \mathbf{p}_i^L(1+\mathbf{w})$;
    2. Learn the capsule activation for layer $(L+1)$: $\mathbf{a}_i^{L+1} = f_{\text{Sig}}(f_{\text{Conv}}(\mathbf{p}_i^{L+1}))$;
    3. Integrate capsules for layer $(L+1)$: $\mathbf{CapF}^{L+1} = f_{ReLU}(f_{\text{Conv}}([[\mathbf{p}_j^{L+1}, \mathbf{a}_j^{L+1}]_1^N]))$.
```

$$= \frac{\partial\varepsilon}{\partial\mathbf{p}^{L+1}}\begin{bmatrix} 1+\mathbf{w}_1 & 0 & \cdots & 0 \\ 0 & 1+\mathbf{w}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1+\mathbf{w}_N \end{bmatrix}. \quad (14)$$

It can be seen in (14) that the gradient of $(\partial\varepsilon/\partial\mathbf{p}^L)$ can be decomposed into two components: 1) $(\partial\varepsilon/\partial\mathbf{p}^{L+1})$ propagates the information directly without concerning any weight layers while focusing on the current-layer pose, which ensures that the information is propagated directly back to any-layer pose; and 2) $(\partial\varepsilon/\partial\mathbf{p}^{L+1})\mathbf{w}_*$ propagates the information through a transformation matrix $\mathbf{w}_*$ to learn the associated versions of each low-layer capsule. Equation (14) guarantees that the gradient for $(\partial\varepsilon/\partial\mathbf{p}^L)$ cannot be vanishing for a mini-batch because the elements of the transformation matrix $\mathbf{w}_*$ cannot be always $-1$ for all samples in a mini-batch.

*2) Complexity Comparison:* We evaluate the complexity[3] of the EM routing [10] and our residual pose routing. For the EM routing [10], the major complexity lies in the matrix multiplication and the EM algorithm, i.e.,

$$O(\text{EMR}) = O\big(B \times H \times W \times N_{\text{low}} \times N_{\text{high}}\big) + O(\text{EM}) \quad (15)$$

where $B$, $H$, and $W$ represent the batch size, height, and width of the input, respectively. $N_{\text{low}}$ and $N_{\text{high}}$ are the capsule-type numbers of adjacent layers. $O(\text{EM})$ represents the complexity of the EM algorithm, which occupies the most cost of the EM routing algorithm [10].

The major complexity of our residual pose routing algorithm lies in the matrix multiplication, i.e.,

$$f^{\text{RPR}} = O(B \times H \times W \times N_{\text{low}}). \quad (16)$$

[3]In this article, we focus on the complexity of the unsupervised computation.

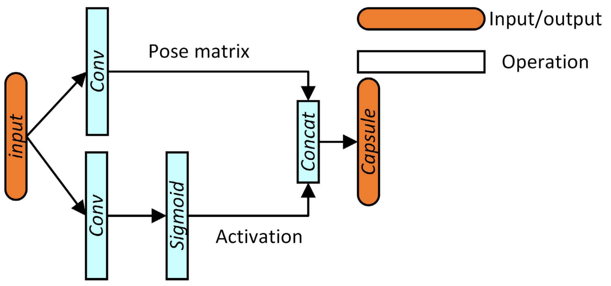


Fig. 4. *PriCaps* layer. Two branches are created to compute the pose matrix and activation from the input image or convolutional features, respectively.

Comparing the complexities of the EM routing and our residual pose routing, we can observe: 1) our residual pose routing gets rid of the EM algorithm, which has significantly reduced the computational costs; and 2) our residual pose routing reduces the computation of matrix multiplication by $\times N_{\text{high}}$ by employing the sparsely connected routing instead of the fully connected routing adopted by the EM routing [10], which additionally leads to a decrease in the routing parameters.

## IV. ResCaps Network

In this section, based on the proposed residual pose routing algorithm, we design a deep CapsNet architecture, named ResCaps.

### *A. Network Architecture*

Fig. 3 illustrates the network architecture of the developed ResCaps consisting of five residual pose routing blocks. Prior to residual pose routing blocks, the operation of Conv+ReLU is used to extract features of the input data. Concretely, each of the shallow four blocks is composed of one primary capsule (*PrimaryCaps*) layer and two residual pose routing (*ResP*) layers. The block 5 consists of one *PriCaps* layer and one *ResPC* layer. In the following, we will describe the details of the network architecture.

The *PriCaps* layer is utilized to generate capsule features from the input image or features. Concretely, two branches are created to compute the pose matrix and activation from the input data, respectively. Specifically, a convolution layer is applied to compute the pose matrix. A convolution and the Sigmoid function are used to compute the activation values. The pose matrix and activation value are concatenated to compose the capsule features. Fig. 4 describes the details of the *PriCaps* layer.

The *ResP* layer employs the residual pose routing algorithm to explore the part-whole relationships in a scene. Fig. 5 illustrates the architecture of the *ResP* layer. Specifically, the

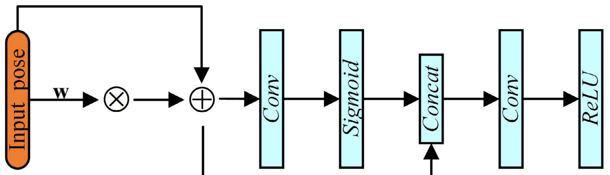


Fig. 5. *ResP* layer. The *ResP* layer is used to embed the residual pose routing algorithm into the network. $\mathbf{w}$ is the learned weight.

pose matrix of the capsule is multiplied with a learnable weight matrix $\mathbf{w}$ with the dimension of $4 \times 4$ to achieve the high-layer capsule pose matrix. On top of that, a convolution and the Sigmoid function are used to generate the higher-layer capsule activate value. A concatenation operation on the pose matrix and the activation value is adopted to obtain the capsule features. At the final stage, a convolution operation and the ReLU activation function are utilized for capsules integration to learn primitive features.

The *ResPC* layer is used for capsules' classification. It has the similar structure with the *ResP* layer except: 1) the output capsule type of the *ResPC* layer is the classification category number and 2) the final convolution and ReLU for capsules integration are not demanded.

*1) Different to RCCaps [49]:* RCCaps [49] simply adds residual connections between capsule layers. In reality, RCCaps [49] indeed has no capsule layers increase, resulting in a fake deep capsule architecture. Differently, our proposed ResCaps has two main highlights: 1) our residual pose routing embeds the residual connection inside the capsule routing derived from the logical analysis, which simplifies the capsule routing with less parameters and light complexity; and 2) our ResCaps assembles several capsule blocks, and each block contains several residual capsule routing layers, which indeed builds a deep capsule architecture.

### B. Loss Function

Given the activation values of the classification capsules, we can train the ResCaps network using the spread loss function in [10]. Concretely, the spread loss is used to maximize the gap between the activation of the target $a_t$ and the activation of the other classes. If the activation of a wrong class $a_i$ is closer to $a_t$ compared to the margin $m$, it will be penalized by the squared distance to the margin. The spread loss function can be formulated as

$$\text{Loss}_i = \max\left(0, m - (a_t - a_i)\right)^2, \text{Loss} = \sum_{i \neq t} \text{Loss}_i. \quad (17)$$

## V. Experiment and Analysis

In this section, we will analyze our ResCaps for image classification, 2-D image saliency, and 3-D object understanding.

### A. Image Classification

In this section, following the previous CapsNet literature, we mostly use the public classification benchmarks for image classification, including MNIST [15], AffNIST [15], SmallNORB [16], and CIFAR-10/100 [17]. We implement the proposed network on Pytorch. For the training procedure, we used the Adam optimizer [52] with the initial learning rate of 0.01. The models were trained on one NAVIDIA 3090Ti GPU with the batch size of 128. The capsule-type numbers of the five blocks are set to [32, 32, 16, 16, $N_C$], where $N_C$ is the category number.

TABLE I
Mean Error (%) and Accuracy (%) on MNIST. "ResCaps" Means Our Entire Framework Consisting of Five Blocks. "-" Represents No Result Released From the Original Paper or Related Papers. The Best Method Is Marked by **Bold**

| Method | Mean error | | Accuracy | |
|---|---|---|---|---|
| | MNIST | AffNIST | MNIST | AffNIST |
| ResCaps | **0.72** | **1.32** | **99.45** | **98.23** |
| Baseline CNN [10] | 0.8 | 14.1 | 99.22 | 66.00 |
| BCN [53] | 2.5 | 8.4 | 97.50 | 91.60 |
| Dynamic [9] | 0.77 | 21 | 99.23 | 81.20 |
| G-Caps [11] | 1.58 | 10.1 | 98.42 | 89.10 |
| EM-Caps [10] | 0.8 | 6.9 | 99.20 | 93.80 |
| Aff-Caps [54] | 0.77 | 6.79 | 99.23 | 93.21 |
| U-Routing [22] | 0.8 | 5.02 | - | - |

TABLE II
Mean Error (%) and Accuracy (%) on SmallNORB. "-" Represents No Result Released From the Original Paper or Related Papers. The Best Method Is Marked by **Bold**

| Method | Mean error | Accuracy |
|---|---|---|
| ResCaps | **0.91** | **99.45** |
| Baseline CNN [10] | 5.2 | - |
| Dynamic [9] | 2.70 | 97.10 |
| FREM [55] | 2.20 | - |
| STAR-Caps [21] | 1.80 | - |
| EM-Caps [10] | 1.80 | 98.00 |
| VB-Routing [56] | 1.60 | - |
| U-Routing [22] | 2.20 | - |

*1) Evaluation on MNIST and AffNIST:* MINIST is a dataset of 60K gray images with the size of $28 \times 28$. Table I lists the performance comparisons of different methods. It can be seen from Table I that our methods outperform the capsule related works as well as CNN networks. Especially, our ResCaps obtains the test error of 0.72% and test accuracy of 99.45%, which are largely better than the other methods.

Besides, we probe into the robustness of our model to affine transformations by using the AffNIST dataset, which is generated by performing 32 random affine transformations on each image of MNIST. Training on the MNIST training set while testing on AffNIST test set containing 320 000 examples are utilized to study the generalization performance. Since AffNIST images are $40 \times 40$, we pad MNIST images for training, i.e., randomly placing the digits on $40 \times 40$ black backgrounds. Data rotation with 30° is additionally used for data augmentation. In Table I, we observe that our models surpass the capsule-related works and CNNs by a large margin.

*2) Evaluation on SmallNORB:* SmallNORB [16] consists of gray-level stereo $96 \times 96$ images of five objects, each of which is given at 18 different azimuths (0–340), nine elevations and six lighting conditions. SmallNORB [16] provides 24 300 training and test set examples. Similar to [10], we standardize the images and resize them to $48 \times 48$. We take $32 \times 32$ random crops for training and center crops at test time. In Table II, our model achieves the test error of 0.91% and test

TABLE III

MEAN ERROR (%) AND ACCURACY (%) ON CIFAR-10. "-" REPRESENTS NO RESULT RELEASED FROM THE ORIGINAL PAPER OR RELATED PAPERS. THE BEST METHOD IS MARKED BY **BOLD**

| Method | Mean error | Accuracy |
|---|---|---|
| ResCaps | **1.14** | **92.38** |
| FlexNet [57] | - | 92.2 |
| Baseline CNN [10] | 19.2 | - |
| Dynamic [9] | 7.86 | 84.50 |
| FREM [55] | 14.3 | - |
| EM-Caps [10] | 11.6 | 82.20 |
| Self-Routing [13] | 7.86 | 92.14 |
| DLME [58] | - | 91.30 |
| MobileNetV2_G32 [59] | - | 90.83 |
| MobileNet_DAIS [60] | - | 91.87 |
| DBN [61] | 17.00 | - |
| Radix_VGG20 [62] | - | 92.2 |

TABLE IV

MEAN ERRORS (%) OF RESCAPS VERSUS RESCAPS-2B. THE BETTER METHOD IS MARKED BY **BOLD**

| Method | MNIST | AffNIST | SmallNORB | CIFAR-10 | CIFAR-100 |
|---|---|---|---|---|---|
| ResCaps | 0.72 | **1.32** | 0.91 | **1.14** | **15.26** |
| ResCaps-2B | **0.54** | 1.86 | **0.64** | 1.14 | 16.05 |

accuracy of 99.45%, which surpass those of the capsule-related works significantly.

*3) Evaluation on CIFAR-10/100:* CIFAR-10 [17] and CIFAR-100 [17] datasets contain images of size 32 × 32 with ten classes and 100 classes, respectively. Each dataset contains 50 000 training images and 10 000 testing images. Table III illustrates the performance of different methods on CIFAR-10 [17]. As shown in Table III, it can be seen that our model achieves the lowest error and highest accuracy, which shows our model surpasses the other methods. Especially, our ResCaps is superior to dynamic CapsNets [9] and EM-Caps [10], which makes known that our model showcases much progress with respect to the CapsNets family.

For CIFAR-100 [17], our ResCaps achieves the test error of 15.26%, which performs better than DensNet [63] (22.27%) and CapsPro [64] (21.93%), i.e., one capsule-related work based on the backbone of ResNet [43].

## B. Ablation Analysis

In this section, we will conduct several experiments to analyze the role of each component in our proposed framework.

*1) Our Residual Pose Routing Versus EM Routing:* From Tables I–III, under the condition of similar architectures, our ResCaps-2B model consistently beats EM-Caps [10] on MNIST, SmallNORB, and CIFAR-10, which demonstrates the superiority of our routing. Besides, in terms of the test time per image in MNIST [15], our ResCaps-2B (1.2 ms) runs faster than EM-Caps (1.9 ms), revealing that our routing method is much simpler than EM-Caps.

*2) Deep Architecture:* To study the effectiveness of our deep architecture, in Table IV, we compare our ResCaps and ResCaps-2B. From the performance comparisons on MNIST, SmallNORB, and CIFAR-10/100, we find that ResCaps performs worse than ResCaps-2B on MNIST and SmallNORB,

TABLE V

MEAN ERROR (%) OF DIFFERENT ROUTING LAYERS IN ONE BLOCK ON SMALNORB. THE BETTER METHOD IS MARKED BY **BOLD**

| Method | SmallNORB |
|---|---|
| ResCaps | **0.91** |
| ResCaps-3L | 1.00 |

TABLE VI

ACCURACY (%) OF DIFFERENT ROUTING LAYERS IN ONE BLOCK ON MNIST AND SMALLNORB. "RESCAPS-2B-3L" EMPLOYS THREE *ResP* LAYERS IN ONE BLOCK UNDER THE SAME SETTING OF "RESCAPS-2B." THE BETTER METHOD IS MARKED BY **BOLD**

| Method | MNIST | SmallNORB |
|---|---|---|
| ResCaps-2B | **99.38** | **92.33** |
| ResCaps-2B-3L | 99.28 | 92.21 |

TABLE VII

MEMORY, PARAMETERS, AND RUNNING TIME OF DIFFERENT MODELS ON CIFAR-10. BESIDES, EVALUATION METRICS ARE LISTED HERE FOR COMPARISON. THE BEST METHOD IS MARKED BY **BOLD**

| | Memory (G) | Parameter (M) | Time (ms) | Error (%) | Accuracy (%) |
|---|---|---|---|---|---|
| ResCaps-2B | **4.8** | 16.4 | **4.2** | **1.14** | **92.38** |
| EM-Caps [10] | 23.70 | **0.02** | 5.3 | 11.6 | 82.20 |
| ResNet-101 [43] | 9.84 | 42.51 | 12.1 | 6.43 | 86.65 |

TABLE VIII

MEAN ERRORS (%) OF RESCAPS VERSUS RCCAPS [49]. THE BETTER METHOD IS MARKED BY **BOLD**

| Method | MNIST | AffNIST | SmallNORB | CIFAR-10 | CIFAR-100 |
|---|---|---|---|---|---|
| ResCaps | **0.72** | **1.32** | **0.91** | **1.14** | **15.26** |
| RCCaps | 1.59 | 2.25 | 1.97 | 2.87 | 34.44 |

which indicates that a shallow CapsNet architecture can achieve promising performance for simple-structure gray-level images. However, ResCaps outperforms ResCaps-2B on AffNIST and CIFAR-10/100, which demonstrates our deep architecture has better robustness to affine transformation on AffNIST and better recognition ability for complex-structure images, e.g., CIFAR-10/100.

*3) Routing Layers:* To study the performance of routing layers in one block, we compare our ResCaps and one modified versions, i.e., ResCaps-3L. Specifically, ResCaps and ResCaps-3L consist of two and three *ResP* layer(s)[4] in one block, respectively. From Table V, we find that our ResCaps consisting of two *ResP* layers can achieve promising performance, compared to ResCaps-3L that employs three *ResP* layers in one block. Besides, similarly in Table VI, under the setting of two blocks, two *ResP* layers perform better than three *ResP* layers. It indicates that two routing layers have an ability of exploring the part-whole relationships in a specific scale of feature maps.

*4) Memory, Running Time, and Parameter Comparison:* Table VII lists the memory, running time, and parameters of different models. First, compared with the CNNs-based model, i.e., ResNet-101, we achieve a lightweight model with fewer parameters and faster running time by a large margin. As well, we also surpass ResNet-101 in terms of error and accuracy.

[4]Considering the memory cost, we only has an ablation study for the three routing layers in one block.

TABLE IX

EVALUATION METRICS INCLUDING $F_\beta$, MAE, AND $E_m$ VALUES OF DIFFERENT METHODS. TOP TWO METHODS ARE MARKED BY RED AND BLUE, RESPECTIVELY. THE DESCRIPTIONS OF ICON-S-RPR AND TSPOANET-RPR CAN BE REFERRED TO THE TEXT

| | **ECSSD [67]** | | | **HKU-IS [68]** | | | **PASCAL-S [69]** | | | **DUT-O [70]** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_\beta$ | MAE | $E_m$ | $F_\beta$ | MAE | $E_m$ | $F_\beta$ | MAE | $E_m$ | $F_\beta$ | MAE | $S_m$ |
| **ICON-S-RPR** | 0.9413 | 0.0236 | 0.9337 | 0.9287 | 0.0216 | 0.9659 | 0.8681 | 0.0473 | 0.8834 | 0.8125 | 0.0427 | 0.8980 |
| **TSPOANet-RPR** | 0.9083 | 0.0425 | 0.9213 | 0.8959 | 0.0341 | 0.9466 | 0.8086 | 0.0723 | 0.8553 | 0.8152 | 0.0429 | 0.8961 |
| **ICON-S [20]** | 0.9408 | 0.0236 | 0.9323 | 0.9280 | 0.0213 | 0.9667 | 0.8652 | 0.0484 | 0.8751 | 0.8111 | 0.0427 | 0.8978 |
| **TSPOANet [42]** | 0.8873 | 0.0515 | 0.9020 | 0.8795 | 0.0391 | 0.9263 | 0.8123 | 0.0749 | 0.8508 | 0.7030 | 0.0638 | 0.8232 |
| **DCR [71]** | 0.9186 | 0.0381 | 0.9202 | 0.9051 | 0.0320 | 0.9476 | 0.8207 | 0.0696 | 0.8535 | 0.7458 | 0.0553 | 0.8539 |
| **DPNet [72]** | 0.9258 | 0.0305 | 0.9264 | 0.9229 | 0.0230 | 0.9627 | 0.8426 | 0.0542 | 0.8674 | 0.7781 | 0.0492 | 0.8764 |
| **SelfReformer [73]** | 0.8982 | 0.0272 | 0.9289 | 0.9142 | 0.0242 | 0.9598 | 0.8160 | 0.0509 | 0.8788 | 0.7890 | 0.0433 | 0.8887 |
| **GPEA [74]** | 0.9292 | 0.0317 | 0.9300 | 0.9187 | 0.0266 | 0.9578 | 0.8450 | 0.0575 | 0.8762 | 0.7597 | 0.0517 | 0.8585 |
| **JointCRF [75]** | 0.8956 | 0.0493 | 0.9152 | 0.8817 | 0.0394 | 0.9384 | 0.7898 | 0.0824 | 0.8368 | 0.7379 | 0.0574 | 0.8571 |
| **ToHR [76]** | 0.9023 | 0.0544 | 0.9171 | 0.8923 | 0.0420 | 0.9357 | 0.8008 | 0.0855 | 0.8465 | 0.7079 | 0.0660 | 0.8411 |
| **BMP [77]** | 0.8682 | 0.0447 | 0.9137 | 0.8705 | 0.0389 | 0.9373 | 0.7578 | 0.0753 | 0.8420 | 0.6917 | 0.0636 | 0.8371 |
| **LFR [78]** | 0.8799 | 0.0525 | 0.9005 | 0.8751 | 0.0396 | 0.9313 | 0.7613 | 0.1066 | 0.7992 | 0.6656 | 0.1030 | 0.7799 |
| **Amulet [79]** | 0.8683 | 0.0589 | 0.9011 | 0.8426 | 0.0501 | 0.9122 | 0.7574 | 0.0997 | 0.8016 | 0.6472 | 0.0976 | 0.7787 |

These observations demonstrate that our model gets superior performance with lower computational complexity.

Second, compared to EM-Caps (23.70 GB GPU memory), our ResCaps-2B that shares the similar architecture with EM-Caps [10] needs (4.80 GB GPU memory) ~20% GPU memory. Besides, our ResCaps-2B is faster than EM-Caps in terms of the running time per image under a similar architecture. However, our model is inferior to EM-Caps [10] in terms of model parameters. The reason behind include: 1) our model has several *PrimaryCaps* layers that will generate much parameters, whereas only one *PrimaryCaps* layer in the entire EM-Caps [10]; and 2) our *ResP* layer has many parameters induced by two convolutions as well as transformation matrices, whereas the main body (*ConvCaps* layer and *ClassCaps* layer) of the EM-Caps [10] just learns transformation matrices. Despite having more parameters, our model gets lower error and higher accuracy, compared to EM-Caps [10].

*5) ResCaps Versus RCCaps [49]:* To explore the superiority of the residual connection in our ResCaps, we compare ResCaps with RCCaps [49] that adds the residual connection between capsule layers, which can be seen in Table VIII. For MNIST, SmallNORB, CIFAR-10, and CIFAR-100, our ResCaps shows a large superiority over RCCaps [49], indicating that the proposed ResCaps has a better classification capacity. Besides, the comparison for AffNIST illustrates that our ResCaps has more robust generalization than RCCaps [49]. These superiority of our ResCaps lie in that our residual connection inside the capsule routing beats that between capsule layers by RCCaps [49].

## C. Two-Dimensional Image Saliency

In view of that the saliency prediction is a fundamental research point in the field of computer vision, we choose the task of salient object detection for the dense prediction evaluation.

To design a deep salient object detection network, following the architecture of [20], we replace the original capsule routing of [20] with our identity mapping routing. We use the cross-entropy loss function and IoU loss function to train the network. The proposed network is implemented in Pytorch. The training dataset of DUTS [65] is chosen as the training dataset with horizontal flipping as the data augmentation technique. The SGD optimizer [66] is used to train our model with an initial learning rate of 5e-2.

*1) Benchmark:* We evaluate the performance of our model on five benchmark datasets, details of which are described as follows.

**ECSSD** [67] contains 1000 images collected from the Internet. These images are with complicated structures.

**HKU-IS** [68] consists of 4447 images with multiple disconnected objects. It is divided into 3000 training images and 1447 test images. We evaluate our methods and other state-of-the-art methods on the test datasets.

**PASCAL-S** [69] includes 850 images describing various scenes.

**DUT-O** [70] has 5168 images with different sizes and complex structures.

*2) Evaluation Criteria:* We evaluate the performance of our model as well as other state-of-the-art methods from both visual and quantitative perspectives. The quantitative metrics include precision recall (PR), F-measure, mean absolute error (MAE), and E-measure. Given a continuous saliency map, a binary mask $B$ is achieved by thresholding. Precision is defined as Precision $= |B \cap G|/|B|$, and recall is defined as Recall $= |B \cap G|/|G|$, where $G$ is the corresponding ground truth. A PR curve is plotted under different thresholds.

F-measure is an overall performance indicator, which is computed by

$$F_\beta = \frac{(1+\gamma^2)\text{Precision} \times \text{Recall}}{\gamma^2\text{Precision} + \text{Recall}}. \tag{18}$$

As suggested in [80], $\gamma^2 = 0.3$.

MAE is defined as

$$\text{MAE} = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} |S(i,j) - G(i,j)| \tag{19}$$

where $W$ and $H$ are the width and height of the image, respectively.

E-measure $(E_m)$ [81] combines local pixel values with the image-level mean value to jointly evaluate the similarity between the prediction and the ground truth.

*3) Performance Comparison With State-of-the-Arts:* Table IX illustrates the quantitative comparison on four

TABLE X

Evaluation Metrics Including $F_\beta$, MAE, and $E_m$ Values of Different Methods. The Best Method Is Marked by **Bold**

| | | ECSSD [67] | | | HKU-IS [68] | | | PASCAL-S [69] | | | DUT-O [70] | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $F_\beta$ | MAE | $E_m$ | $F_\beta$ | MAE | $E_m$ | $F_\beta$ | MAE | $E_m$ | $F_\beta$ | MAE | $S_m$ |
| VGG16 | **TSPOANet-RPR** | **0.9080** | **0.0436** | **0.9224** | **0.8945** | **0.0360** | **0.9438** | **0.8022** | **0.0755** | **0.8512** | **0.7286** | **0.0617** | **0.8505** |
| | **TSPOANet-EM** | 0.8761 | 0.0517 | 0.8964 | 0.8707 | 0.0404 | 0.9316 | 0.7963 | 0.0786 | 0.8348 | 0.6954 | 0.0691 | 0.8278 |
| | **TSPOANet-RPR** | **0.9080** | **0.0436** | **0.9224** | **0.8945** | **0.0360** | **0.9438** | 0.8022 | 0.0755 | **0.8512** | **0.7286** | **0.0617** | **0.8505** |
| | **TSPOANet [42]** | 0.8873 | 0.0515 | 0.9020 | 0.8795 | 0.0391 | 0.9263 | **0.8123** | **0.0749** | 0.8508 | 0.7030 | 0.0638 | 0.8232 |
| | **ICON-V-RPP** | **0.9232** | **0.0335** | **0.9210** | **0.9151** | **0.0276** | **0.9549** | 0.8324 | **0.0663** | 0.8548 | 0.7481 | 0.0654 | **0.8482** |
| | **ICON-V [20]** | 0.9178 | 0.0365 | 0.9193 | 0.9074 | 0.0305 | 0.9163 | **0.8340** | 0.0640 | **0.8612** | **0.7552** | **0.0646** | 0.8331 |
| Swin Transformer | **ICON-S-RPR** | **0.9413** | **0.0236** | **0.9337** | **0.9287** | 0.0216 | 0.9659 | **0.8681** | **0.0473** | **0.8834** | **0.8125** | **0.0427** | **0.8980** |
| | **ICON-S [20]** | 0.9408 | **0.0236** | 0.9323 | 0.9280 | **0.0213** | **0.9667** | 0.8652 | 0.0484 | 0.8751 | 0.8111 | **0.0427** | 0.8978 |

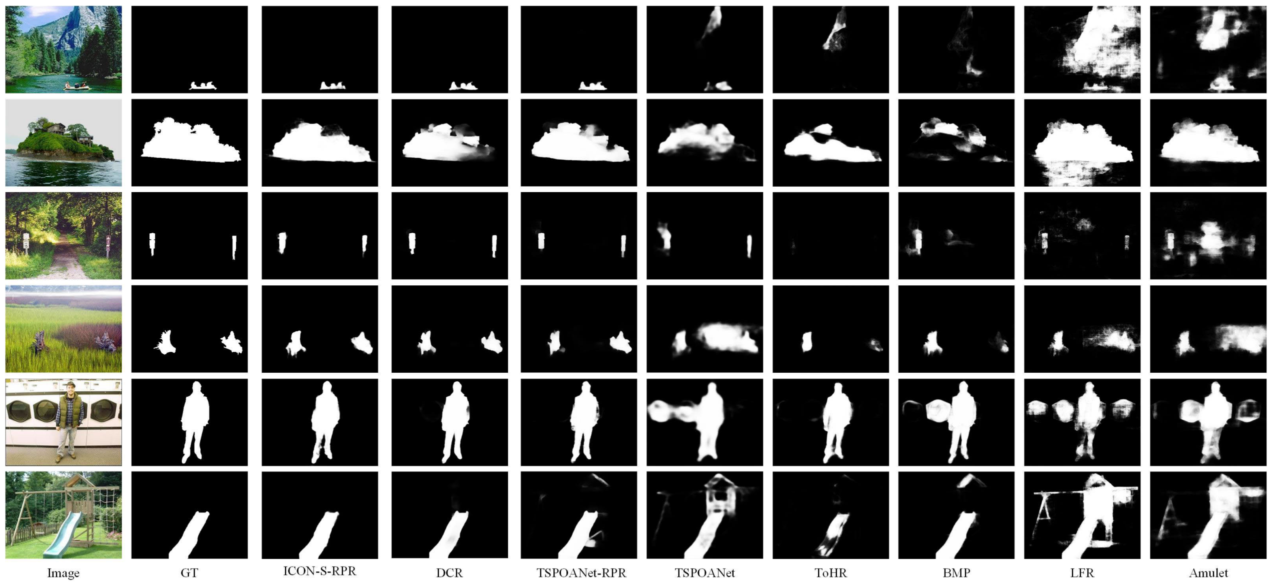


Fig. 6. Image saliency prediction results of some good methods. RPR can detect various-scale salient objects (top two rows), multiple salient objects (middle two rows), and noisy-scene salient objects (bottom two rows), compared to the other methods.

benchmarks with ten state-of-the-art methods, including ICON-S [20], TSPOANet [42], DCR [71], DPNet [72], SelfReformer [73], JointCRF [75], ToHR [76], BMP [77], LFR [78], and Amulet [79]. Specifically in Table IX, ICON-S is one version of ICON [20] with Transformer as the backbone. ICON-S-RPR[5] is implemented by replacing the dynamic routing of ICON-S [20] with our residual pose routing using two blocks. Also, TSPOANet-RPR is achieved by replacing the two-stream capsule routing in TSPOANet [42] with our residual pose routing using two blocks. First in Table IX, ICON-S-RPR is better than the others for most of metrics, which demonstrates the superiority of our residual pose routing for the task of saliency detection. Second, under the same setting, ICON-S-RPR and TSPOANet-RPR perform better than ICON-S [20] and TSPOANet [42] with respect to most of metrics, respectively. These observations indicate our residual pose routing has a great power than the dynamic capsule routing in ICON [20] and the two-stream capsule routing in TSPOANet [42] for salient object detection.

To prove our claim of being simple yet effective model, we list several pairs of models comparison in Table X, including TSPOANet-RPR versus TSPOANet-EM, TSPOANet-RPR versus TSPOANet, ICON-S-RPR versus ICON-S, and ICON-V-RPR versus ICON-V. TSPOANet-RPR and TSPOANet-EM replace the previous capsule routing in TSPOANet [42] with our residual pose routing algorithm and the EM routing algorithm [10], respectively. ICON-S and ICON-V denote the ICON model with backbones of Transformer and VGG16, respectively. ICON-S-RPR and ICON-V-RPR are achieved by substituting our residual pose routing for the original capsule routing in ICON-S and ICON-V, respectively. As illustrated in Table X, ICON-S-RPR performs better than ICON-S with a slight margin, which is because the contribution of the capsule routing algorithm to the performance is inferior to that of the powerful backbone, i.e., swin Transformer. With regard to the ordinary backbone of VGG16, our residual pose routing can achieve a large-margin improvement on various benchmarks, such as $\sim 4\%$ $E_m$ of ICON-V-RPR versus ICON-V on HKU-IS, $\sim 3\%$ $F_\beta$ of TSPOANet-RPR versus TSPOANet-EM on ECSSD and DUT-O, and $\sim 2\%$ $F_\beta$ and $E_m$ of TSPOANet-RPR versus TSPOANet. Such performance margin is essentially significant for the task of salient object detection in the day. Based on the above discussions, our residual pose routing algorithm gets a slight 0.1% improvement when using

[5]It is noted that ICON-S-RPR is re-trained for the optimal performance with the same architecture. Saliency maps of ICON-S-RPR have been released on https://github.com/liuyi1989/ResCaps.

TABLE XI
EVALUATION RECONSTRUCTION QUALITY (‰). THE BEST METHOD IS MARKED BY **BOLD**

| **Method** | OURS | Point-baseline [18] | AtlasNet [82] | Point-Caps [18] |
|---|---|---|---|---|
| **Chamfer Distance** | **1.38** | 1.91 | 1.56 | 1.46 |

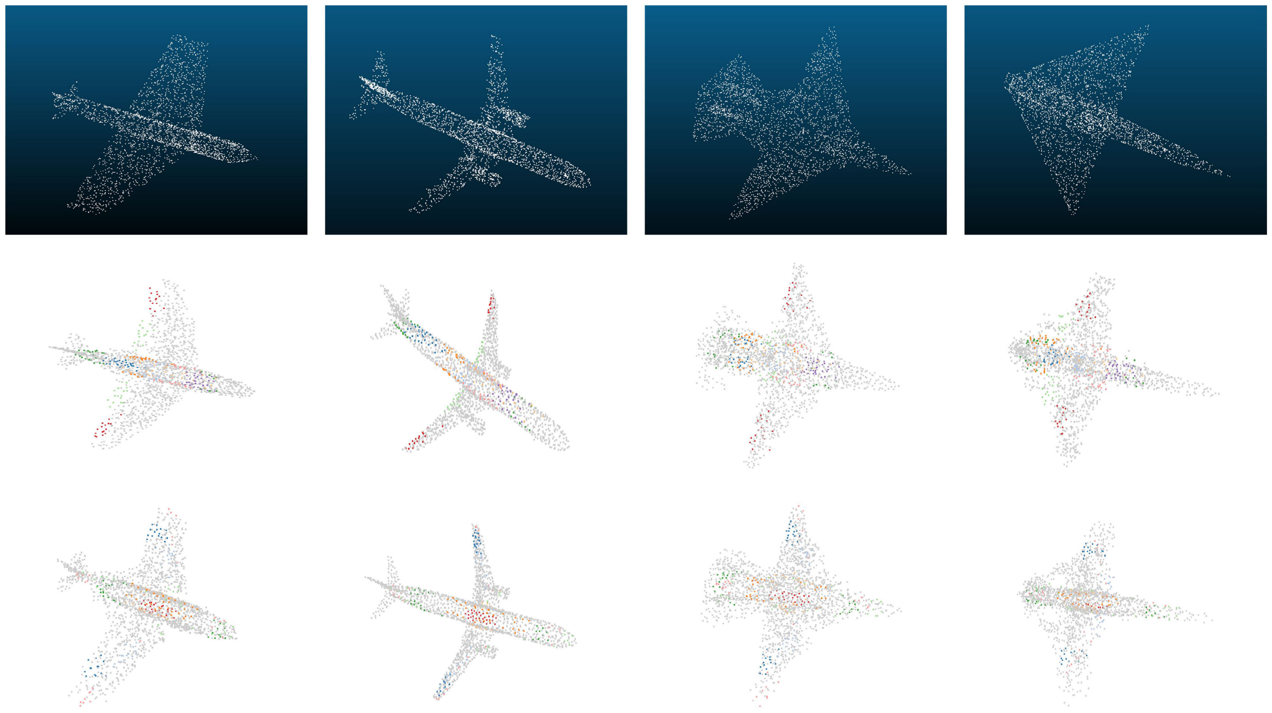


Fig. 7. Three-dimensional object generation. From top to bottom: input, point-caps [18], and our framework.

sophisticated backbones, e.g., swin Transformer. By contrast, our model achieves a 2% $\sim$ 4% improvement on top of the lightweight backbones, e.g., VGG16.

Fig. 6 shows detection results of some good methods. Specifically, as shown in the first two rows of Fig. 6, RPR can detect various-scale salient objects compared to the other methods. As shown in the middle two rows of Fig. 6, RPR can locate and segment multiple salient objects compared with the other methods. As shown in the bottom two rows of Fig. 6, RPR can segment the salient object while suppressing noisy backgrounds, which is challenging for the other methods. These improvements are credited with the part-whole hierarchies learned by the proposed residual pose routing.

### *D. Three-Dimensional Object Understanding*

The simplicity and effectiveness of our residual pose routing make it possible to extend it for large-scale scene understanding. In light of the increasing research trend of 3-D sensing for robotics, autonomous driving, and augmented/mixed reality, we choose 3-D object understanding based on the 3-D point cloud data to evaluate our residual pose routing, including the tasks of 3-D reconstruction and 3-D classification.

*1) Three-Dimensional Reconstruction:* To evaluate our residual pose routing on 3-D reconstruction for point cloud generation, we utilize one block of our ResCaps to substitute for the dynamic capsule routing in Point-Caps [18] for 3-D reconstruction. The input point clouds are aligned to a common reference frame and normalized for training. The Adam optimizer [52] with an initial learning rate of 0.0001 and a batch size of 2 is adopted for training.

We choose the standard Chamfer distance as the reconstruction performance evaluation metric on the ShapeNet Core v2 dataset [83]. For fair comparisons, we use the same training and test splits in AtlasNet [82]. As illustrated in Table XI, our residual pose routing beats the dynamic routing in Point-Caps [82] significantly, specifically improving the Chamfer Distance from 1.46 of Point-Caps [18] to 1.38. Besides, as shown in Fig. 7, it can be seen that our residual pose routing can better reconstruct the input point clouds than Point-Caps [18] that adopts the dynamic routing. Concretely, our residual pose routing reconstructs better object shapes and inner details for better recognition.

*2) Three-Dimensional Classification:* To demonstrate the efficiency of learned reconstruction representation, we evaluate the classification accuracy by performing transfer learning based on the learned latent features. Similar to [84], [85], and [86], we train a linear SVM classifier regress the shape class on ModelNet 40 [19] given the latent features. As shown in Table XII, our accuracy is 89.9% surpassing Point-Caps [18], which demonstrates our residual pose routing can generalize better to new tasks like 3-D reconstruction and classification.

TABLE XII
ACCURACY OF CLASSIFICATION BY TRANSFER LEARNING ON MODELNET 40 [19]. THE BEST METHOD IS MARKED BY **BOLD**

| **Method** | Latent-GAN [85] | FoldingNet [86] | Point-Caps [18] | OURS |
|---|---|---|---|---|
| **Accuracy** | 85.7 | 88.4 | 89.3 | **89.9** |

## VI. CONCLUSION

In this article, we have proposed a simple yet effective residual pose routing algorithm into the CapsNet. Compared with the original EM capsule routing algorithm [10], our residual pose routing computed the capsule pose by an identity mapping learning on the low-layer capsule pose, which helped to avoid gradient vanishing. Different from EM routing [10] that adopted fully connected complex hand-crafted computation, our residual pose routing conducted a sparsely learning framework, which greatly reduced the routing parameters and computation. Inspired by its simplicity and power, our residual pose routing made a deep CapsNet architecture for better image classification. To study the generalization for new tasks of our residual pose routing, it was extended for 2-D image saliency and 3-D reconstruction/classification.

*Future work and limitation.* The model presented in this work has some limitations for future research. First, the robustness for adversarial attacks is an increasing and demanding research point. Therefore, the discussion for adversarial attacks is demanding for our residual pose routing. Second, currently, we evaluate our model for classification on 2-D small-scale images. The generalization on large-scale datasets, e.g., ImageNet [87] and COCO [88], will be a future research.

## REFERENCES

[1] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial Intelligence Safety and Security*. London, U.K.: Chapman & Hall, 2018, pp. 99–112.

[2] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 427–436.

[3] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.

[4] D. Zhang et al., "Onfocus detection: Identifying individual-camera eye contact from unconstrained images," *Sci. China Inf. Sci.*, vol. 65, no. 6, Jun. 2022, Art. no. 160101.

[5] M. A. Alcorn et al., "Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4840–4849.

[6] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "A rotation and a translation suffice: Fooling CNNs with simple transformations," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–21.

[7] T. Cohen and M. Welling, "Group equivariant convolutional networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 2990–2999.

[8] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *Proc. Int. Conf. Artif. Neural Netw.* Berlin Springer, 2011, pp. 44–51.

[9] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–11.

[10] G. E. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with EM routing," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 3856–3866.

[11] J. E. Lenssen, M. Fey, and P. Libuschewski, "Group equivariant capsule networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1–10.

[12] J. Rajasegaran, V. Jayasundara, S. Jayasekara, H. Jayasekara, S. Seneviratne, and R. Rodrigo, "DeepCaps: Going deeper with capsule networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10717–10725.

[13] T. Hahn, M. Pyeon, and G. Kim, "Self-routing capsule networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 7658–7667.

[14] T. Jeong, Y. Lee, and H. Kim, "Ladder capsule network," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 3071–3079.

[15] Y. LeCun. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[16] Y. LeCun, F. Jie Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jan. 2004, pp. 97–104.

[17] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Apr. 2009, pp. 1–60.

[18] Y. Zhao, T. Birdal, H. Deng, and F. Tombari, "3D point capsule networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1009–1018.

[19] Z. Wu et al., "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.

[20] M. Zhuge, D.-P. Fan, N. Liu, D. Zhang, D. Xu, and L. Shao, "Salient object detection via integrity learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3738–3752, Mar. 2023.

[21] K. Ahmed and L. Torresani, "STAR-CAPS: Capsule networks with straight-through attentive routing," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 9101–9110.

[22] F. D. S. Ribeiro, G. Leontidis, and S. D. Kollias, "Introducing routing uncertainty in capsule networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 6490–6502.

[23] H. Li, X. Guo, B. D. Ouyang, and X. Wang, "Neural network encapsulation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 252–267.

[24] Y. Feng, J. Gao, and C. Xu, "Learning dual-routing capsule graph neural network for few-shot video classification," *IEEE Trans. Multimedia*, vol. 35, no. 3, pp. 3738–3752, Mar. 2022.

[25] J. Chen, H. Yu, C. Qian, D. Z. Chen, and J. Wu, "A receptor skeleton for capsule neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 1781–1790.

[26] A. Vasantharao, A. Durai, S. Gabbita, and A. S. Erraguntla, "Spatially transformed capsule networks," in *Proc. 3rd Int. Conf. Pattern Recognit. Mach. Learn. (PRML)*, Jul. 2022, pp. 131–135.

[27] J. Tao, X. Zhang, X. Luo, Y. Wang, C. Song, and Y. Sun, "Adaptive capsule network," *Comput. Vis. Image Understand.*, vol. 218, Apr. 2022, Art. no. 103405.

[28] Y. Liu, D. Zhang, Q. Zhang, and J. Han, "Part-object relational visual saliency," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3688–3704, Jul. 2022.

[29] Q. Zhang, M. Duanmu, Y. Luo, Y. Liu, and J. Han, "Engaging part-whole hierarchies and contrast cues for salient object detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 6, pp. 3644–3658, Jun. 2022.

[30] Y. Liu, D. Zhang, Q. Zhang, and J. Han, "Integrating part-object relationship and contrast for camouflaged object detection," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 5154–5166, 2021.

[31] D. Ma and X. Wu, "CapsuleRRT: Relationships-aware regression tracking via capsules," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10943–10952.

[32] A. U. Khan, H. Kuehne, K. Duarte, C. Gan, N. Lobo, and M. Shah, "Found a reason for me? Weakly-supervised grounded visual question answering using capsules," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 8461–8470.

[33] Q. Cao, W. Wan, K. Wang, X. Liang, and L. Lin, "Linguistically routing capsule network for out-of-distribution visual question answering," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 1594–1603.

[34] N. Garau, N. Bisagno, P. Bródka, and N. Conci, "DECA: Deep viewpoint-equivariant human pose estimation using capsule autoencoders," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 11657–11666.

[35] C. Yu, X. Zhu, X. Zhang, Z. Wang, Z. Zhang, and Z. Lei, "HP-Capsule: Unsupervised face part discovery by hierarchical parsing capsule network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 4022–4031.

[36] W. Sun et al., "Canonical capsules: Self-supervised capsules in canonical pose," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 24993–25005.

[37] B. Wu, X. He, Q. Zhang, M. Wang, and Y. Ye, "GCRec: Graph-augmented capsule network for next-item recommendation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 10164–10177, Apr. 2023.

[38] D. Cheng, G. Wang, B. Wang, Q. Zhang, J. Han, and D. Zhang, "Hybrid routing transformer for zero-shot learning," *Pattern Recognit.*, vol. 137, May 2023, Art. no. 109270.

[39] L. Wang, M. Tang, and X. Hu, "Evaluation of grouped capsule network for intracranial hemorrhage segmentation in CT scans," *Sci. Rep.*, vol. 13, no. 1, p. 3440, Mar. 2023.

[40] D. Ma and X. Wu, "QuadTreeCapsule: QuadTree capsules for deep regression tracking," in *Proc. 30th ACM Int. Conf. Multimedia*, Oct. 2022, pp. 4684–4693.

[41] S. Bonheur, F. Thaler, M. Pienn, H. Olschewski, H. Bischof, and M. Urschler, "OnlyCaps-Net, a capsule only based neural network for 2D and 3D semantic segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput. Assist. Intervent.* Cham, Switzerland: Springer, 2022, pp. 340–349.

[42] Y. Liu, Q. Zhang, D. Zhang, and J. Han, "Employing deep part-object relationships for salient object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1232–1241.

[43] 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.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 630–645.

[45] W. Ke, J. Chen, J. Jiao, G. Zhao, and Q. Ye, "SRN: Side-output residual network for object symmetry detection in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 302–310.

[46] L. Tran, X. Liu, J. Zhou, and R. Jin, "Missing modalities imputation via cascaded residual autoencoder," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4971–4980.

[47] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Temporal residual networks for dynamic scene recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7435–7444.

[48] Q. Wang, Z. Teng, J. Xing, J. Gao, W. Hu, and S. Maybank, "Learning attentions: Residual attentional Siamese network for high performance online visual tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4854–4863.

[49] J. Gugglberger, D. Peer, and A. Rodríguez-Sánchez, "Training deep capsule networks with residual connections," in *Proc. Int. Conf. Artif. Neural Netw.* Cham, Switzerland: Springer, 2021, pp. 541–552.

[50] X. Ding, N. Wang, X. Gao, J. Li, and X. Wang, "Group reconstruction and max-pooling residual capsule network," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 2237–2243.

[51] Y. LeCun et al., "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[53] S. Chang, J. Yang, S. Park, and N. Kwak, "Broadcasting convolutional network for visual relational reasoning," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 754–769.

[54] J. Gu and V. Tresp, "Improving the robustness of capsule networks to image affine transformations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 7283–7291.

[55] S. Zhang, Q. Zhou, and X. Wu, "Fast dynamic routing based on weighted kernel density estimation," in *Proc. Int. Symp. Artif. Intell. Robot.* Cham, Switzerland: Springer, 2018, pp. 301–309.

[56] F. D. S. Ribeiro, G. Leontidis, and S. Kollias, "Capsule routing via variational Bayes," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 3749–3756.

[57] D. W. Romero et al., "FlexConv: Continuous kernel convolutions with differentiable kernel sizes," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–14.

[58] Z. Zang et al., "DLME: Deep local-flatness manifold embedding," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2022, pp. 576–592.

[59] J. Zhong, J. Chen, and A. Mian, "DualConv: Dual convolutional kernels for lightweight deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 9528–9535, Mar. 2023.

[60] Y. Guan et al., "DAIS: Automatic channel pruning via differentiable annealing indicator search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 9847–9858, Apr. 2022.

[61] Y. Ma and D. Klabjan, "Diminishing batch normalization," *IEEE Trans. Neural Netw. Learn. Syst.*, Oct. 2022.

[62] Z. Wang, X. Gu, R. S. M. Goh, J. T. Zhou, and T. Luo, "Efficient spiking neural networks with radix encoding," *IEEE Trans. Neural Netw. Learn. Syst.*, Aug. 2022.

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

[64] L. Zhang, M. Edraki, and G.-J. Qi, "CapProNet: Deep feature learning via orthogonal projections onto capsule subspaces," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–10.

[65] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. S. Torr, and S.-M. Hu, "Global contrast based salient region detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 569–582, Mar. 2015.

[66] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*, 2nd ed. Berlin, Germany: Springer, 2012, pp. 421–436.

[67] Q. Yan, L. Xu, J. Shi, and J. Jia, "Hierarchical saliency detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 1155–1162.

[68] G. Li and Y. Yu, "Visual saliency based on multiscale deep features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5455–5463.

[69] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille, "The secrets of salient object segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 280–287.

[70] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang, "Saliency detection via graph-based manifold ranking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 3166–3173.

[71] Y. Liu, D. Zhang, N. Liu, S. Xu, and J. Han, "Disentangled capsule routing for fast part-object relational saliency," *IEEE Trans. Image Process.*, vol. 31, pp. 6719–6732, 2022.

[72] Z. Wu, S. Li, C. Chen, H. Qin, and A. Hao, "Salient object detection via dynamic scale routing," *IEEE Trans. Image Process.*, vol. 31, pp. 6649–6663, 2022.

[73] Y. Ke Yun and W. Lin, "SelfReformer: Self-refined network with transformer for salient object detection," 2022, *arXiv:2205.11283*.

[74] X. Zhao, H. Liang, and R. Liang, "Position fusing and refining for clear salient object detection," *IEEE Trans. Neural Netw. Learn. Syst.*, Nov. 2022.

[75] Y. Xu et al., "Structured modeling of joint deep feature and prediction refinement for salient object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3788–3797.

[76] Y. Zeng, P. Zhang, Z. Lin, J. Zhang, and H. Lu, "Towards high-resolution salient object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7233–7242.

[77] L. Zhang, J. Dai, H. Lu, Y. He, and G. Wang, "A bi-directional message passing model for salient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1741–1750.

[78] P. Zhang, W. Liu, H. Lu, and C. Shen, "Salient object detection by lossless feature reflection," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 1149–1155.

[79] P. Zhang, D. Wang, H. Lu, H. Wang, and X. Ruan, "Amulet: Aggregating multi-level convolutional features for salient object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 202–211.

[80] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, "Frequency-tuned salient region detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1597–1604.

[81] D.-P. Fan, C. Gong, Y. Cao, B. Ren, M.-M. Cheng, and A. Borji, "Enhanced-alignment measure for binary foreground map evaluation," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 698–704.

[82] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché approach to learning 3D surface generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 216–224.

[83] A. X. Chang et al., "ShapeNet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.

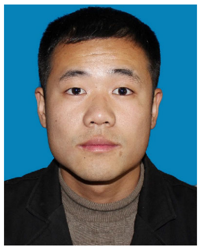
[84] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–9.

[85] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3D point clouds," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 40–49.

[86] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point cloud auto-encoder via deep grid deformation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 206–215.

[87] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[88] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 740–755.



**Yi Liu** received the Ph.D. degree from Xidian University, Xi'an, China, in 2019.

He is currently a Professor with Changzhou University, Changzhou, China. From 2018 to 2019, he was a Visiting Scholar at Lancaster University, Lancaster, U.K. His research interests include machine learning and computer vision, especially on saliency detection, capsule network, 3-D point cloud, and object detection.

**De Cheng** received the B.S. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 2011 and 2017, respectively.

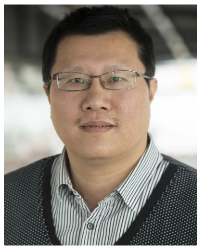
From 2015 to 2017, he was a Visiting Scholar at Carnegie Mellon University, Pittsburgh, PA, USA. He is an Associate Professor with the School of Telecommunications Engineering, Xidian University, Xi'an. His research interests include pattern recognition, machine learning, and multimedia analysis.

**Dingwen Zhang** (Member, IEEE) received the Ph.D. degree from Northwestern Polytechnical University, Xi'an, China, in 2018.

From 2015 to 2017, he was a Visiting Scholar at the Robotic Institute, Carnegie Mellon University, Pittsburgh, PA, USA. He is currently a Professor with the School of Automation, Northwestern Polytechnical University. He is also an Adjunct Researcher at the Hefei Comprehensive National Science Center, Institute of Artificial Intelligence, Hefei, China. His research interests include computer vision and multimedia processing, especially on saliency detection, video object segmentation, temporal action localization, and weakly supervised learning.

**Shoukun Xu** received the Ph.D. degree from the China University of Mining and Technology, Xuzhou, China, in 2001.

He is currently a Professor at Changzhou University, Changzhou, China. He is the Chair of the China Computer Federation Changzhou Branch and a Distinguished Member of the China Computer Federation. His research interests include digital twins, computer vision, and blockchain.



**Jungong Han** (Senior Member, IEEE) is currently the Chair Professor of computer vision at the Department of Computer Science, University of Sheffield, Sheffield, U.K. He also holds a Honorary Professorship at the University of Warwick, Coventry, U.K. His research interests include computer vision, artificial intelligence, and machine learning.

Dr. Han is a fellow of IAPR.