CARNet: Densely Connected Capsules with Capsule-Wise Attention Routing

Zhi-Xuan Yu, Ye He, Chao Zhu(✉), Shu Tian, and Xu-Cheng Yin

School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, People's Republic of China
zhixuanyu@xs.ustb.edu.cn, YeHe_USTB@outlook.com,
{chaozhu,shutian,xuchengyin}@ustb.edu.cn

Abstract. Convolutional neural networks (CNNs) have been proven to be effective for image recognition, which plays an important role in cyber security. In this paper, we focus on a promising neural network, capsule network, which aims at correcting the deficiencies of CNNs. Routing procedure between capsules, which serves as a key component in capsule networks, computes coupling coefficients with complicated steps iteratively. However, the expensive computational cost poses a bottleneck for extending capsule networks deeper and wider to approach higher performance on complex data. To address this limitation, we propose a novel routing algorithm named *capsule-wise attention routing* based on attention mechanism. With a successful reduction of computational cost in the routing procedure, we construct a deep capsule network architecture named *CARNet*. Our CARNets are proven experimentally to outperform other state-of-the-art capsule networks on SVHN and CIFAR-10 benchmarks while reducing the amount of parameters by 62% at most.

Keywords: Cyber security · Image recognition · Capsule network · Attention mechanism

1 Introduction

Cyber security becomes more and more important nowadays as cyberspace infiltrates into our life rapidly. Every day billions of images containing massive information are generated and transmitted in cyberspace, which poses great challenges to the security of cyberspace. It is necessary for us to search for more efficient and robust methods of image recognition to retrieve useful information from images automatically.

Over the last decade, convolutional neural networks (CNNs) have been widely used in various challenging tasks in computer vision for its remarkable learning capacity. CNNs share weights across positions on the image to achieve translation invariance, which is reasonable but not robust enough when dealing with complex transformations caused by viewpoint changes or part deformation. To correct these deficiencies, Hinton et al. [5] proposed a concept of capsule, which

H. Ning (Ed.): CyberDI 2019/CyberLife 2019, CCIS 1137, pp. 309–320, 2019.
https://doi.org/10.1007/978-981-15-1922-2_22

aims to learn features equivariant to transformations resulted from viewpoint changes. Built upon capsules, capsule networks [6,11] had achieved state-of-the-art performance on MNIST and smallNORB benchmarks.

However, there is still a large room for capsule networks to approach state-of-the-art performance on natural image datasets. CapsNets [6,11] comprise much fewer layers than current well-performed models such as ResNet [4] and DenseNet [7], which contain hundreds of layers. It has been empirically proven that neural networks with deeper and wider architecture are more capable of learning complex hierarchies inside visual entities. Naturally it is worthwhile to explore a deeper capsule network architecture for enhancing its performance on complex data. As discussed in [10], simply stacking up fully-connected capsule layers towards a deep architecture will lead to some undesired problems like expensive computational cost and gradient vanishing. To address these limitations, we start our work with an analysis of the routing algorithm which involves complicated computations.

Routing algorithm in the standard CapsNets routes capsules from low level to high level according to coupling coefficients, which are computed with multiple iterative steps. From another aspect, the routing procedure can be explained as a parallel attention mechanism. So we formulate the computation process as a regression of multiple attention maps and implement the computation of coupling coefficients by two fully-connected layers. Capsules at one position are taken as input to the two-layer subnetwork to output coupling coefficients. Weights in the fully-connected layers are shared across different positions, so capsules at different positions can be routed according to the same criterion. As a result, the routing procedure is feasible to be accomplished in one stage and performed with much less computational cost. We name this novel routing algorithm as *Capsule-wise Attention Routing*, since our motivation comes from attention mechanism. Besides the change in computing coupling coefficients, another modification is the adoption of 2D convolution with larger kernel to transform capsules from low level to high level. Since the original matrix multiplication is equivalent to 1×1 convolution, the modification can be regarded as an enlargement of the convolutional kernel. To prevent the amount of parameters increasing proportionally to the size of convolutional kernel, we implement the convolutions with the idea from depthwise separable convolutions [2].

As the computational cost in routing gets successfully reduced, we are able to build up a deeper capsule network with more capsule layers. We name our model as *CARNet* after the routing algorithm we proposed previously. In addition, we set skip connections between different levels of capsules to help transport gradient flow into low layers during training. With the skip connections, capsules at low level can be routed directly to the top capsules and involved in the final inference.

The rest of our paper is organized as follows. In Sect. 2, we review the related work on capsule networks. In Sect. 3 we introduce how capsule-wise attention routing works and elaborate the architecture of CARNet. In Sect. 4, we evaluate capsule-wise attention routing and CARNets on four object recognition

benchmarks including MNIST, Fashion-MNIST, SVHN and CIFAR-10. Finally we summarize our work and discuss possible future work in Sect. 5.

2 Related Work

Capsule is a neural unit aiming to learn viewpoint-equivariant instantiation parameters and viewpoint-invariant activation probability of some visual entity in images. In dynamic capsules [11], a capsule is organized as a vector called activation vector. Entries of the vector are explained as multiple implicitly defined instantiation parameters of the visual entity, while the length of the vector is the probability indicating the presence. In EM capsules [6], instantiation parameters and activation probability were separately represented by a 4×4 pose matrix and a logistic unit. The complicated internal structure determines that capsule has more complicated intra-computation and inter-computation than single neuron.

Routing procedure happens between adjacent capsule layers. Each capsule in the lower layer will first make predictions for capsules in the higher layer respectively. If two lower capsules make similar predictions for one higher capsule, they are supposed to be routed to that capsule. For every higher capsule, it will receive a cluster of predictions from capsules below, and aggregate them as output. With the routing-by-agreement mechanism, CapsNet not only achieved state-of-the-art performance on MNIST and smallNORB benchmarks but also showed out superiority in distinguishing overlapping digits [11] and resisting white box adversarial attacks [6].

Capsule networks have been further explored in the literature. Wang and Liu [12] formulated dynamic routing as an optimization problem of minimizing clustering loss with a KL regularization term and modified the routing procedure with motivation from solving a clustering object function. Lenssen et al. [9] proposed group equivariant capsule networks with provable equivariance and invariance properties. Zhang et al. [15] proposed two fast routing methods based on kernel weighted density estimation. These works improved the routing algorithm from different aspects but the networks were still relatively shallow.

Concurrent with our work, Rajasegaran et al. [10] proposed a deep capsule network architecture named DeepCaps with a similar motivation to ours. DeepCaps includes 17 capsule layers and impressively outperforms the state-of-the-art capsule networks on Fashion-MNIST, SVHN and CIFAR-10 with much less parameters than the original CapsNet. DeepCaps maintains the framework of dynamic routing and adopts 3D convolution to implement the transformation in routing. Weights among input capsules are shared so that the computational cost can be reduced. DeepCaps also proposes a class-independent reconstruction network at the top of network. Different from DeepCaps, we propose a novel routing algorithm in which the coupling coefficients are computed by a two-layer subnetwork and the transformation is performed by 2D convolution. Besides, our model uses skip connections to connect capsules at different levels in a different way from DeepCaps. And the reconstruction network is not considered for regularization. Performance of DeepCaps and our proposed approach will be compared in Sect. 4.

3 CARNet

In this section, the details of the proposed capsule-wise attention routing algorithm and the architecture of CARNet are presented.

3.1 Capsule-Wise Attention Routing

Consider an intermediate capsule layer that processes N_l input capsules and outputs N_{l+1} capsules. We denote the i-th capsule in layer l at some position by $\mathbf{u}_i^{(x,y)} \in \mathbb{R}^{a_l}$, where (x, y) is the coordinate of capsule and a_l is the dimension of activation vector.

First, capsules at one position are concatenated as a vector $\mathbf{u}^{(x,y)}$. Since the computation of coupling coefficients across positions are performed in the same way, we omit the coordinate for clarity below. Then we pass $\mathbf{u}$ to two cascaded fully-connected layers to regress the log prior probabilities $\mathbf{h} \in \mathbb{R}^{N_l \times N_{l+1}}$. We choose vector $\mathbf{u}$ as the input of the subnetwork because $\mathbf{u}$ is supposed to aggregate all the semantic information of its local receptive field at current layer, which can help generate more proper coupling coefficients. The computation of $\mathbf{h}$ is written as:

$$\mathbf{h} = \mathbf{W}_2 \cdot g\left(\mathbf{W}_1 \mathbf{u} + \mathbf{b}_1\right) + \mathbf{b}_2, \tag{3.1}$$

where $g(\cdot)$ is the ReLU function. We reorganize vector $\mathbf{h}$ into an $N_l \times N_{l+1}$ matrix and subsequently feed it to the softmax function to get coupling coefficients $\mathbf{c}$.

$$\hat{\mathbf{u}}_{i|j} = c_{ij} \mathbf{u}_i, \qquad c_{ij} = \frac{\exp\left(h_{ij}\right)}{\sum_k \exp\left(h_{ik}\right)}, \tag{3.2}$$

where c_{ij} is the coupling coefficient of capsule i and capsule j, $\hat{\mathbf{u}}_{i|j}$ is a weighted activation vector passed from capsule i to capsule j. Each capsule in the higher layer will receive N_L weighted activation vectors from the lower layer and then transform them into the higher capsule space. Dynamic routing implements the transformation from low level to high level by matrix multiplication (1×1 convolution), which makes no use of features in the neighbourhood. Here we adopt 2D convolution with larger receptive field to perform the transformation.

In details, for capsule j, all the weighted capsules $\{\hat{\mathbf{u}}_{i|j} \mid 1 \leq i \leq N_l\}$ are concatenated to be $\hat{\mathbf{u}}_j$, which is followed by a Conv-BN-ReLU block to generate output capsule $\mathbf{v}_j$. Parameters in the blocks are not shared among higher capsules, so these parallel blocks can learn part-whole relationship independently of each others.

These convolutions are performed parallelly in capsules, which are equivalent to group convolutions. So we implement the parallel convolutions with inspiration from depthwise separable convolution [2], which splits the original convolutional operation into a depthwise convolution and a pointwise convolution. First, we concatenate the weighted capsules and perform depthwise convolution on it. Second, we separate the tensor back to the form of capsules and perform pointwise matrix multiplication. When the receptive field is 1×1, we would omit the

Algorithm 1. Capsule-wise attention routing algorithm.

Input: The set of capsules in layer l, $\mathbf{U} = \{\mathbf{u}_i^{(x,y)} \mid \mathbf{u}_i^{(x,y)} \in \mathbb{R}^{a_l}, 1 \leq i \leq N_l, 1 \leq x \leq W_l, 1 \leq y \leq H_l\}$, where N_l is the number of capsules, W_l and H_l are the width and height of the feature map.

procedure ROUTING($\mathbf{U}$)
  for every position (x, y):
    $\mathbf{u} \leftarrow [\mathbf{u}_1; \mathbf{u}_2; \ldots; \mathbf{u}_{N_l}]$;
    $h \leftarrow \mathbf{W}_2 \cdot g(\mathbf{W}_1\mathbf{u} + \mathbf{b}_1) + \mathbf{b}_2$;
    for every capsule i in layer l:
      for every capsule j in layer $(l+1)$:
        $c_{ij} = \exp(h_{ij}) / \sum_k \exp(h_{ik})$
        $\hat{\mathbf{u}}_{i|j} \leftarrow c_{ij}\mathbf{u}_i$;
    $\hat{\mathbf{u}}_j \leftarrow \left[\hat{\mathbf{u}}_{1|j}; \hat{\mathbf{u}}_{2|j}; \ldots; \hat{\mathbf{u}}_{N_l|j}\right]$;
  for every capsule j in layer $(l+1)$:
    $\hat{\mathbf{U}}_j \leftarrow \{\hat{\mathbf{u}}_j^{(x,y)} \mid 1 \leq x \leq W_l, 1 \leq y \leq H_l\}$;
    $\mathbf{V}_j \leftarrow \text{Conv-BN-ReLU}_j(\hat{\mathbf{U}}_j)$;
  return $\mathbf{V} = \{\mathbf{V}_j\}$

first step and perform the second step only, which is equivalent to the transformation in dynamic routing. By this method, we avoid using convolution for each capsule tensor iteratively and take advantage of the speed-up of convolution. In addition, for kernel size $k > 1$, our implementation would reduce the amount of parameters used for transformation by

$$\Delta N_{param} = k^2 N_l a_l + N_l a_l N_{l+1} a_{l+1} - k^2 N_l a_l N_{l+1} a_{l+1}. \tag{3.3}$$

Since $k^2 \ll N_l a_l$ for every layer l in practice, so the reduction rate of parameters is nearly $1/k^2$, which is considerable even when $k = 3$.

Activation probability of a capsule still depends on the length of the activation vector as in [11]. But we don't squeeze the length of vector into $[0, 1]$ at the end of routing by the squashing function. We only compute the activation probability by function

$$P(\mathbf{v}) = \frac{\|\mathbf{v}\|^2}{1 + \|\mathbf{v}\|^2} \tag{3.4}$$

if the probability is needed. We skip the vector-squashing operation in the intermediate capsules and choose ReLU as the activation function to prevent gradient vanishing.

Capsule-wise attention routing computes the coupling coefficients by a two-layer subnetwork, turning the mechanism behind from routing-by-agreement to routing-by-learning. In this way, we avoid computing coupling coefficients iteratively and reduce the cost. Besides, since the computation of coupling coefficients and the low-to-high transformation can both be implemented by convolutional operations, the speed of inference can be accelerated with GPUs. Thanks to the reduction of computational cost in each capsule layer, we can cascade more capsule layers to attain a higher learning capacity.

3.2 CARNet Architecture

The architecture of our proposed CARNet is shown in Table 1. Similar to the standard capsule networks, our deep capsule network starts with several convolutional layers, which extract low-level features from the original image. Then the feature map is reorganized into the form of capsule tensor and passed through cascaded capsule layers. At the top of the network, we compute the prediction probability of each category based on the corresponding capsule. The details of CARNet are demonstrated as follows.

Table 1. CARNet architecture for SVHN and CIFAR-10. Note that "conv" in the table refers to Conv-BN-ReLU block and "CAR" is short for "capsule-wise attention routing". All the convolutional layers are performed with padding except the ones with superscript "$*$". Layers bracketed together comprise a capsule block.

Stage	Output Size	N_{channel}	N_{capsule}	N_{atom}	Layer
Convolution	32×32	128	–	–	conv(5×5, stride 1)
	16×16	256	–	–	conv(3×3, stride 2)
	16×16	256	–	–	conv(3×3, stride 1) $\times 3$
Primary Capsules	16×16	–	32	8	reshape
Capsule−1.x	4×4	–	16	8	$\begin{bmatrix} \text{CAR}(3 \times 3, \text{stride } 1) \\ \text{CAR}(3 \times 3, \text{stride } 2) \\ \text{CAR}(3 \times 3, \text{stride } 1) \end{bmatrix} \times 2$
Capsule−2.x	2×2	–	16	8	$\begin{bmatrix} \text{CAR}(1 \times 1, \text{stride } 1) \\ \text{CAR}(3 \times 3, \text{stride } 1)^* \\ \text{CAR}(1 \times 1, \text{stride } 1) \end{bmatrix}$
Final capsules	1×1	–	10	16	CAR(2×2, stride 1)*
Probability computing	1×1	–	10	–	$P(\mathbf{v}) = \frac{\|\mathbf{v}\|^2}{1+\|\mathbf{v}\|^2}$

Low-Level Feature Extraction. CapsNet proposed in [11] uses a single convolutional layer with a relatively large receptive field to extract low-level features from the image. The convolution is performed without padding, so a large receptive field can help scale down the size of feature map and further reduce the computational cost in the subsequent capsule layers. While in CARNet, to reap the benefit of deeper networks, we replace the single convolutional layer by multiple cascaded convolutional layers with smaller receptive field. And we also use convolutions with zero padding to keep the size of some feature maps fixed.

Skip Connections. We combine three cascaded capsule layers as a single capsule block and set short paths to connect capsule blocks at different levels. The aim of short paths is to downsample lower capsules to make their size consistent with higher ones and then merge them together.

Let us denote the input and output of the n-th capsule block by $\mathbf{U}_n$ and $\mathbf{V}_n$. Due to the convolutional operations in the capsule block, $\mathbf{V}_n$ would get a smaller size than $\mathbf{U}_n$. We use a 1×1 pooling with the same stride and padding (as the convolutional layer) to downsample the input $\mathbf{U}_n$. So the receptive field of the downsampled tensors $\mathbf{U}'_n$ are center-aligned to the receptive field of $\mathbf{V}_n$ at every position. Subsequently $\mathbf{U}'_n$ and $\mathbf{V}_n$ are concatenated and fed to the next capsule block, i.e. $\mathbf{U}_{n+1} = [\mathbf{U}'_n; \mathbf{V}_n]$. In this way, lower capsules with the same receptive field centers are preserved and delivered to any higher capsule blocks by the skip connections, which means capsules at all levels would make contribution to the final result of classification. In other words, every capsule block is allowed to receive all the outputs of its preceding capsule blocks to generate its own output (Fig. 1).

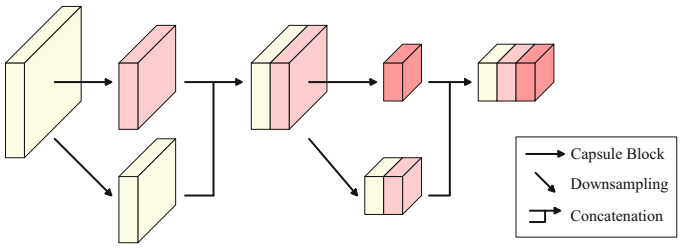

Fig. 1. Short paths connecting capsules in different levels.

Implementation Details. At the bottom of CARNet, we set five convolutional layers to extract features with 256 channels from input image. Each convolutional layer is followed by a BN layer and an activation function ReLU. Then we split up the tensor into 32 tensors with 8 channels, termed primary capsules. Primary capsules are subsequently fed to three cascaded capsule blocks. Every capsule layers in the blocks output $8D$-capsules of 16 types. Skip connections merge capsules from preceding capsule blocks and transport them to the next capsule block. Finally, the primary capsules and capsules from three capsule blocks would be merged and fed to the final capsule layer to generate 10 $16D$ category-specified capsules, from which we compute the recognition probability by Eq. 3.4.

3.3 Loss Function

The loss function we adopted is margin loss proposed in [11], defined as

$$L = \max(0, m^+ - \|\mathbf{v}_t\|)^2 + \lambda \sum_{i \neq t} \max(0, \|\mathbf{v}_i\| - m^-)^2, \tag{3.5}$$

where t is index of the correct category. We use $m^+ = 0.95$ and $m^- = 0.05$ as the upper bound for the correct category and lower bound for the wrong category. The weight for losses from wrong categories λ is set as 0.5 for the whole training procedure.

4 Experiments

We empirically evaluated our capsule-wise attention routing and CARNet on four object recognition benchmarks including MNIST, Fashion MNIST, SVHN and CIFAR-10. Each of them collects images from 10 categories. We implemented CARNet in TensorFlow [1] framework and trained our models on GTX 1080 Ti GPUs. We used Adam optimizer [8] for the training and set the initial learning rate as 0.001, which would get reduced by 0.5 every 20,000 steps.

4.1 Datasets

MNIST and Fashion-MNIST. MNIST is a dataset of handwritten digit images while Fashion-MNIST is a dataset of fashion product images. MNIST and Fashion-MNIST provide images of the same amount (60,000 images for training and 10,000 for test) and in the same format (28×28 greyscale image). All images are captured in white background. For every training image, we randomly shifted it in every directions by up to 2 pixels. No preprocessing was performed for the test images.

SVHN. The Street View House Numbers dataset provides 32×32 RGB images containing numbers in natural scene. A large number of images are available for training (604,388) and test (26,032). Our training and test were performed without data preprocessing and data augmentation.

CIFAR-10. CIFAR-10 dataset is also a natural image dataset. The objects in images come from objects in our daily life. 50,000 training images and 10,000 test images are provided. The images are also 32×32 color images. During training, we perform random shift, random horizontal flipping and random adjustment of brightness and contrast as the data preprocessing. Both the training images and the test images are normalized to have zero mean and unit standard variance before they were fed to the network.

4.2 Capsule-Wise Attention Routing in CapsNet

To evaluate the effectiveness of our novel routing algorithm, we first designed an experiment to compare capsule-wise attention routing with dynamic routing in a shallow capsule network.

We trained a wider version of CapsNet[1], in which the number of intermediate capsules increased to 32 and the dimensions of activation vectors increased to 8 and 16 in primary capsule layer and final capsule layer. Then we got another model by replacing the dynamic routing procedure with two layers of capsule-wise attention routing in the final capsule layer. Note that we don't set reconstruction networks for both of them. Details of the networks are depicted in Table 2.

As shown in Table 3, CapsNet with capsule-wise attention routing consumes only half of parameters in its couterpart but achieves higher accuracies on both SVHN and CIFAR-10. The replacement of routing procedure also helps to speed up the training of CapsNet by about 50%.

Table 2. Architectures of two capsule networks. "CapsNet*" represents CapsNet with capsule-wise attention routing. Number n in "dynamic routing$\times n$" indicates the iteration times in dynamic routing.

Stage	N_{channel}	N_{capsule}	N_{atom}	Layer	
				CapsNet	CapsNet*
Convolution	64	–	–	conv(9×9, stride 1)	
Primary capsules	256	–	–	conv(9×9, stride 2)	
	–	32	8	reshape & vector squashing	
Final capsules	–	10	16	transformation	CAR(5×5, stride 1)
				dynamic routing$\times 3$	CAR(4×4, stride 1)
Probability computing	–	10	–	$P(\mathbf{v}) = \|\mathbf{v}\|$	$P(\mathbf{v}) = \frac{\|\mathbf{v}\|^2}{1+\|\mathbf{v}\|^2}$

Table 3. Accuracies (%) of CapsNet and CapsNet* on SVHN and CIFAR-10.

Model	Param.	FPS	SVHN	CIFAR-10
CapsNet	3.96M	1.12K	95.82	81.80
CapsNet*	1.94M	1.68K	**96.83**	**82.56**

4.3 Performance of CARNet

We trained our CARNet on four benchmarks and compared the performance with proposed capsule networks. We also evaluated the effect of skip connections

[1] The CapsNet we trained is wider than CapsNet for SVHN [11], which consists of a convolutional layer with 64 channels, a primary capsule layer with 16 $6D$-capsules and a final capsule layer with 10 $8D$-capsules.

in our model. In Table 4 we list out the error rates achieved by our models, CapsNet, DeepCaps and variants of ResNet and DenseNet. All the results are achieved by single model.

As capsule network goes deeper, the performance gets improved accordingly especially on natural image datasets. CARNet without skip connections leads the performance of capsule networks on SVHN and CIFAR-10 and performs close to the state-of-the-art capsule networks on MNIST and Fashion-MNIST. While with skip connections, the performance can be further improved on four benchmarks. Our best model achieves an accuracy of 97.72% on SVHN and 92.46% on CIFAR-10 that surpass DeepCaps by 0.56% and 1.45% respectively.

Besides, CARNets also show out a more efficient capability of utilizing parameters than the existing capsule networks. CARNet consumes 3.96M parameters, which can be cut down to 2.73M when the skip connections are removed. The amount of trainable parameters in CARNet is less than CapsNet for MNIST (8.2M) or DeepCaps for CIFAR-10 (7.22M), and much less than other well-performed CNN-based models listed in Table 4. The efficiency of utilizing parameters comes from the use of convolutions in the transformation step in capsule-wise attention routing, which allows capsules to leverage local features when learning part-whole relationship inside the visual entity. On the other hand, the increment in the amount of parameters is controlled within an acceptable limit thanks to the implementation based on depthwise separable convolutions.

Table 4. Accuracies (%) on MNIST, Fashion-MNIST, SVHN and CIFAR-10 datasets. "SC" is short for "skip connections". Results in **bold** are the best in the domain of capsule networks.

Model	Param.	MNIST	F-MNIST	SVHN	CIFAR-10
ResNet v2 [5]	10.2M	–	–	–	95.38
ResNeXt [13]	68.1M	–	–	–	96.42
DenseNet [7]	27.2M	–	95.40	98.41	96.26
Wide ResNet [14]	36.5M	–	95.90	–	95.83
WRN + Random Erasing [16]	36.5M	–	96.35	–	96.92
CapsNet [11]	8.2M	**99.75**	93.62	95.70	–
FREM [15]	8M	99.62	93.80	–	85.70
HitNet [3]	–	99.68	92.30	94.50	73.30
DeepCaps [10]	7.22M	99.72	**94.46**	97.16	91.01
CARNet w/o SC (Ours)	2.73M	99.72	94.30	97.61	91.88
CARNet with SC (Ours)	3.96M	99.74	**94.46**	**97.72**	**92.46**

5 Conclusion

In this paper, we proposed a novel routing algorithm, capsule-wise attention routing, which uses a two-layer subnetwork to regress coupling coefficients as

multiple attention maps. We adopted 2D convolution to replace the linear transformation so that local features can be utilized in transforming capsules from low level to high level. In addition, we formulated the parallel transformation among capsules as group convolutions and implemented it with the inspiration from depthwise separable convolutions. The new implementation was consistent with the original transformation in dynamic routing and was proven to help utilize parameters more efficiently.

Based on capsule-wise attention routing, we further proposed a deep capsule network called CARNet. We stacked multiple capsule layers in our model and set skip connections to densely connect different levels of capsules. CARNet achieved state-of-the-art performance on MNIST and Fashion-MNIST and outperformed the state-of-the-art capsule network on SVHN and CIFAR-10, which are both datasets containing natural images. It is an inspiring step of capsule networks to approach the state-of-the-art performance on natural image datasets, but the performance gap still exists between capsule network based models and the state-of-the-art CNN models. In the future, we plan to explore more efficient routing algorithm and make further attempt to deepen the capsule network architecture for better performance on complex data.

Acknowledgements. This work was supported by National Natural Science Foundation of China under Grant 61703039 and Beijing Natural Science Foundation under Grant 4174095.

References

1. Abadi, M., et al.: Tensorflow: large-scale machine learning on heterogeneous distributed systems. CoRR abs/1603.04467 (2016)
2. Chollet, F.: Xception: deep learning with depthwise separable convolutions. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017, pp. 1800–1807 (2017)
3. Deliège, A., Cioppa, A., Droogenbroeck, M.V.: HitNet: a neural network with capsules embedded in a hit-or-miss layer, extended with hybrid data augmentation and ghost capsules. CoRR abs/1806.06519 (2018)
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016, pp. 770–778 (2016)
5. Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. In: Honkela, T., Duch, W., Girolami, M., Kaski, S. (eds.) ICANN 2011. LNCS, vol. 6791, pp. 44–51. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21735-7_6
6. Hinton, G.E., Sabour, S., Frosst, N.: Matrix capsules with EM routing. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018, Conference Track Proceedings (2018)
7. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017, pp. 2261–2269 (2017)

8. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015, Conference Track Proceedings (2015)
9. Lenssen, J.E., Fey, M., Libuschewski, P.: Group equivariant capsule networks. In: Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, Montréal, Canada, 3–8 December 2018, pp. 8858–8867 (2018)
10. Rajasegaran, J., Jayasundara, V., Jayasekara, S., Jayasekara, H., Seneviratne, S., Rodrigo, R.: DeepCaps: going deeper with capsule networks. CoRR abs/1904.09546 (2019)
11. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017, pp. 3859–3869 (2017)
12. Wang, D., Liu, Q.: An optimization view on dynamic routing between capsules. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018, Workshop Track Proceedings (2018)
13. Xie, S., Girshick, R.B., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017, pp. 5987–5995 (2017)
14. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, 19–22 September 2016 (2016)
15. Zhang, S., Zhou, Q., Wu, X.: Fast dynamic routing based on weighted kernel density estimation. In: Lu, H. (ed.) ISAIR 2018. SCI, vol. 810, pp. 301–309. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-04946-1_30
16. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. CoRR abs/1708.04896 (2017)