Received: 29 December 2020 | Revised: 13 July 2021 | Accepted: 1 August 2021

IET Computer Vision

DOI: 10.1049/cvi2.12068

ORIGINAL RESEARCH PAPER

A dynamic routing CapsNet based on increment prototype clustering for overcoming catastrophic forgetting

Meng Wang[1,2] | Zhengbing Guo[1,2] | Huafeng Li[1]

[1]Information Engineering and Automation, Kunming University of Science and Technology, Kunming, Yunnan, China

[2]Yunnan Key Laboratory of Artificial Intelligence, Kunming University of Science and Technology, Kunming, Yunnan, China

Correspondence

Zhengbing Guo, Information Engineering and Automation, Kunming University of Science and Technology, Kunming, Yunnan 650500, China.
Email: cn_gzb@126.com

Funding information

National Natural Science Foundation of China, Grant/Award Number: 6206204

Abstract

In continual learning, previously learnt knowledge tends to be overlapped by the subsequent training tasks. This bottleneck, known as catastrophic forgetting, has recently been relieved between vision tasks involving pixel shuffles etc. Nevertheless, the challenge lies in the continuous classification of the sequential sets discriminated by global transformations, such as excessively spatial rotations. Aim at this, a novel strategy of dynamic memory routing is proposed to dominate the forward paths of capsule network (CapsNet) according to the current input sets. To recall previous knowledge, a binary routing table is maintained among these sequential tasks. Then, an increment procedure of competitive prototype clustering is integrated to update the routing of the current task. Moreover, a sparsity measurement is employed to decouple the salient routing among the different learnt tasks. The experimental results demonstrate the superiority of the proposed memory network over the state–of–the–art approaches by the recalling evaluations on extended sets of Cifar–100, CelebA and Tiny ImageNet etc.

KEYWORDS

capsule network, catastrophic forgetting, continual learning, dynamic routing, prototype clustering

1 | INTRODUCTION

Intelligent creatures can learn and memorise required knowledge throughout their lives. This ability, referred to as continual learning, is the key to achieving general artificial intelligence. In continual learning, the previously learnt knowledge tends to be erased by the overlapping of subsequent training tasks, which is known as catastrophic forgetting (CF) [1]. Over the years, various approaches have been proposed to relieve this problem. These existing solutions can be briefly summarised into three categories including self–refreshing memory, stochastic gradient descent (SGD) and restraint approaches [2]. Recently, despite that the utilisation of deep neural networks with large–scale trainable parameters has made great progress in the CF problem, there are still bottlenecks in dealing with the continuous discrimination of practical sequential sets involving global transformations [3].

In the early days, researchers generally applied self–refreshing memory schemes through hybrid training to constantly stimulate neurons with information related to historical data to form an activated competition, so as to minimise the impact on the important parameters related to historical knowledge while learning new knowledge [4]. Alternatively, another solution is to use an ensemble of neural networks. When a new task arrives, this ensemble strategy sets up a new network branch and shares the representations between the tasks [5, 6]. Similarly, replay methods address the imbalance between previous and new data by shared classifiers [7–9] or generative procedures [10, 11] to replay previous data alleviate the problem. These approaches often improve the performance of the whole tasks and are practically used in continual learning. However, they have a complexity limitation, especially in inferences, since the number of networks increases with the number of new tasks that need to be learnt [12].

Another solution focusses on using implicit distributed storage of information in the typical learning with stochastic gradient descent (SGD) [13, 14]. These approaches adopt the idea of dropout or maxout to distributively store the information for each task by utilising the large capacity of deep neural networks [15]. Unfortunately, most studies following this solution had limited success and failed to preserve performance on the old task when an extreme change to the

environment occurred [16, 17]. Alternatively, Fernando et al. [18] proposed PathNet, which extends the idea of the ensemble approach for parameter reuse within a single network. The authors in [19] propose a random path selection algorithm, RPS–Net, that progressively chooses optimal paths for the new tasks while encouraging parameter sharing. This model alleviates the complexity issue of the ensemble approaches in continual learning with a plausible way.

Later studies suggest that the CF problem can be effectively relieved by restraining and pruning the connections of deep networks [20–22]. The genetic algorithms are first introduced to select more adaptive network connections according to different task sets [18]. To restrain forgetting the learnt knowledge, the previous model is re–utilised to predict the new data, after which a virtual training set is formulated for a task transferring learning [23]. Recently, the weight regularisation is integrated into the iterative optimisation to adapt new tasks by transferring the old network configurations [24, 25]. Also, the evolutionary theory of networks has been further inspirited [26] in terms of the efficient transferring strategy of elastic weight consolidation (EWC) [27, 28]. Learning without forgetting (LWF) [29] utilises the knowledge distillation loss to preserve the performance of previous tasks. In addition, there are studies that continually train the neural network by using the sequential Bayesian estimation [30, 31], which introduces a constraint in the loss function that directs plasticity to the weight that contributes the most to the previous tasks. CLAW [32] is proposed in terms of the probabilistic modelling and variation inference. Sparse–based approaches often embed a hidden layer of sparse coding to improve CF [33], nevertheless the obtained sparsity might impair the learning generalisation on new tasks. TFW [34] adopts activation masks on each layer to prevent both catastrophic forgetting and negative transfer. In Ref. [23, 35], a context–dependent gating signal is designed, such that only sparse and mostly non–overlapping patterns of units are active for each task. Generally, the above solutions can relieve CF on some simple scenes, such as the pixel shuffled Mixed National Institute of Standards and Technology database (MNIST) and Cifar–100, but these strategies become inefficient when there involves complex transfers on larger sets as CelebA and ImageNet. One reason is the inefficient representation between and within complex tasks, thus the spatial scales of inputs are also limited. Moreover, though these researches attempt to maintain a balance between the simplicity and plasticity of the original networks during continual learning, this balance is hard to preserve once the task sequence becomes longer or more difficult.

Recently, Hinton et al. proposed a capsule network (CapsNet) [36, 37] as an alternative architecture of CNNs. For the existing CNNs, one limitation lies is that the lack of local invariance features is easy to cause the extreme variations of global discriminating outputs [38]. In contrast, the extended capsule vectors can represent more salient spatial characters, such as the direction, the scale and other object attributions [39]. To illuminate this, we visualise the actual instances of different architectures. The class activation map [40] (CAM) of CNN and CapsNet are shown in Figure 1.

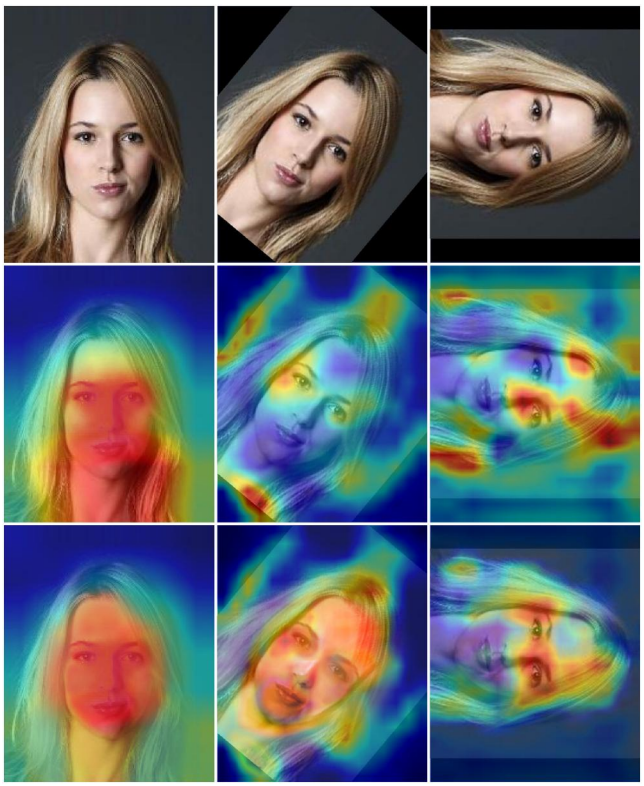


FIGURE 1 The class activation map (CAM) of CNN and CapsNet. From top to bottom are original pictures, CAM of CNN and CAM of CapsNet

In addition, CapsNet extends the original architecture adaptability to varying task scenes by providing powerful analysis capabilities for global object characters, such as axis rotations. Inspired by this concept, the present study takes CapsNet as the basic framework to explore the new solutions for continual learning and then introduces a dynamic routing activated by different task inputs, as well as a learning module of task discrimination based on prototype clustering to alleviate catastrophic forgetting. The accuracy and recalling evaluations of related state–of–the–art approaches are discussed on sequential task sets with varying spatial rotations including Cifar–100, CelebA and ImageNet etc. The main contributions of this study are summarised as follows.

1) An extend framework based on CapsNet is proposed to overcome the catastrophic forgetting on continual learning. Overall, this architecture has both adaption and retention on sequential image sets involving global transformations, such as excessively spatial rotations.
2) A dynamic memory routing is formulated to dominate the feedforward connections of CapsNet according to the input task sets. Unlike the other capsule routing algorithms, an independent routing table is maintained among all the sequential tasks and updated with each training batch.
3) An increment prototype clustering for the discrimination of sequential tasks is integrated into the proposed dynamic

routing. Different from the existing memory networks, the current task sets are analysed and discriminated by learning the prototypes of each task set on–line.

The rest of this study is organised as follows. Section 2 briefly reviews related work, and the proposed approach is detailed in the next section. Section 4 demonstrates the comparison results as well as their analysis. The main conclusions are presented in Section 5.

2 | RELATED WORK

2.1 | Model adaption on continual learning

To relieve catastrophic forgetting, the discriminative models should update the weights according to the current dataset and simultaneously retain the previously learnt knowledge. This dynamic memory mechanism can be achieved by compressed and incremental procedures [30]. The compressed approaches usually adopt weight restrain and network pruning to minimise the configuration overlapping among different tasks [41, 42]. Another solution known as PathNet [18] adopts evolutionary strategy to select and activate neurons on forward paths according to the current inputs. The disadvantage might lie in that the network scale will still increase as the learnt tasks are accumulated. One solution is to reduce the numerical precision required for storing the network weights [43, 44], and the other is to reduce the network complexity by trimming the trivial network weights [45, 46]. In addition, sparse regularisation has been introduced to improve the network compression [47–49]. In contrast, incremental approaches mainly utilise the previous model to learn new tasks without impacting the learnt knowledge. The research in Ref. [50] is a typical example of using an incremental approach to continually gain new knowledge, and LWF belongs to this strategy [29]. Compared with the compression methods, the incremental operations reduce the weight overlapping between training sets. Nevertheless, as the number of tasks increases, the previously learnt knowledge still tends to be confused by the subsequent tasks, since the shared transferring module has limited capability for task discrimination. It implies that some architectures of task analysis should be integrated into continual learning so as to strike a balance between the plasticity and stability of the network by more dynamical structures.

2.2 | Routing strategies of capsule network

On vision tasks, although the architecture of deep convolution has many successful implementations due to its powerful non–linear representations, its disadvantages are the inability to align spatial features between each layer and the lack of global invariant representations [36]. Thus, Hinton et al. suggested using an extending convolutional architecture as the capsules network (CapsNet) yielding to dynamic routing and vectored representations. This architecture has been successfully applied to various complex vision scenes [51]. Later, there are several routing strategies as routing by agreement [37] has been proposed to dynamically update the connections between capsules. These routing algorithms of CapsNet differ from other neural networks in many significant aspects [42]. First, the capsule routing supports the perceptual transmission learning of object attributions, which has been verified as an effective strategy with higher generalisation and fewer parameters [39, 52]. Moreover, the capsule routing is a definable field of the unsupervised clustering, which is based on the natural divergences between input samples. In this study, the Hinton's CapsNet is utilised as the basic architecture for the continual learning tasks.

2.3 | Prototype clustering approaches

The supervised discriminative learning cannot naturally discover and represent the inter–structures of data samples. Thus, the evolutionary prototype algorithm (EKP) (zheng et al., 2010) is proposed to achieve unsupervised representation learning with global search capability. To further discover and generalise the knowledge contained in small batches [53, 54], Snell et al. [50] proposed a prototype network that performs clustering iterations to learn an embedded mapping from input samples to a prototype space. By using the Euclidean distance as a similarity metric, the training procedures makes the embedded sample vectors as close as possible to the prototype of their own category and as far as possible to the prototypes of the other category. According to the few–shots model in Ref. [50], let $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ be a small support set including N labelled samples with each $x_i \in \mathbb{R}^D$ and the corresponding label $y_i \in \{1, \ldots, k\}$, then d_k denotes the set of samples labelled with k. The prototype network maps the sample set to the M–dimensional space through an embedding function $f_\theta : \mathbb{R}^D \to \mathbb{R}^M$ and calculates a prototype vector c_k for each category

$$c_k = \frac{1}{|d_k|} \sum_{(x_i, y_i) \in d_k} f_\theta(x_i). \tag{1}$$

Then, the prototype network adopts a softmax function to regularise the distances from the network output f_θ to each prototype c_k

$$p_\theta(y = k|x) = \frac{\exp(-d(f_\theta(x), c_k))}{\sum_{k'} \exp\left(-d(f_\theta(x), c_{k'})\right)}. \tag{2}$$

This prototype network is updated by minimising the negative log–probability $J_\theta = log_{p_\theta}(y = k|x)$ according to the label k. On the learning iterations, a subset of classes is randomly selected from the training set, then a subset of examples within each class is chosen as the support set, and a subset of the remainder is formed as query points.

3 | THE PROPOSED DYNAMIC ROUTING NETWORK

3.1 | Paradigm of sequential learning

Let an annotated set $d^t = \{(x_1, y_1), (x_2, y_2) \ldots, (x_m, y_m)\}$ be utilised for training single classification task, then a sequence $D^t = (d^1, d^2 \ldots, d^t)$ can be ordered with the task index t for sequential learning. In this order, each task set d^t is fed into a model f_θ with trainable parameters θ to optimise a determinant map $f_\theta : x_i \to y_i$, where $(x_i, y_i) \in D^t$. On the sequential learning, the previous parameters learnt by $(x_i', y_i') \in D^{t'}$ with $t' = 1, \ldots t1$ are easily covered by the current updating gradients denoted as $\nabla\theta = \nabla_\theta l(f_\theta(x_j^t), y_j^t)$, where $(x_i^t, y_i^t) \in d^t$. It implies that catastrophic forgetting is inevitable as the previous loss $l(f_\theta(x_j^{t'}), y_j^{t'})$ increases during the sequential training. To overcome this problem, the parameters should be extended to a configuration set $\mathcal{N} = (\theta, \delta)$, where δ denotes the network structure of f. Then the map $f_{\mathcal{N}^t} : x_i \to y_i$ for the single task $(x_i, y_i) \in d^t$ corresponds to the configuration subset $\mathcal{N}^t = (\theta, \delta^t)$. Therefore, the whole network should be trained to decide both the shared weights θ and the task–aligned structure δ^t, which will be detailed in the following sections.

3.2 | Dynamic memory routing on CapsNet

For complex vision applications, the capsule network (CapsNet) provides more integral representations of spatial attributes by employing both vectored capsules and the routing between them. The capsule routing can be dynamically formulated by activating the important capsules aligned to different task sets. Therefore, this study focusses on the capsule routing for building up the memory mechanism on sequential learning. The basic routing is applied on the capsule layer l yielding a map $f_l : u_i \to v_j$ with $v_j = \rho_s(\sum_i c_{ij}(w_{ij}u_i))$, where the projected capsule $\hat{u}_{j|i} = w_{ij}u_i$, and the routing weights $c_{ij} = e^{b_{ij}} / \sum_k e^{b_{ik}}$, determine the connection intensity between the capsules in terms of the consistency b_{ij}. In addition, the squashing non–linear activation $\rho_s(x)$ and the margin loss L_{margin} are manually selected.

On this basis, the routing weights can be denoted by a matrix $C^t = [c_{ij}]_{m\times n}$ $(i = 1, 2, \ldots m; j = 1, 2, \ldots n)$ for the single task set d^t. This routing matrix is applied to determine which projected capsules $\hat{U} = [\hat{u}_{j|i}]_{m\times n\times q}$ are activated to be fed forward by the element multiplication as

$$\begin{aligned} V &= \rho_s(C^t \cdot WU) \\ &= \rho_s(C^t \cdot \hat{U}), \end{aligned} \tag{3}$$

where the matrices are organised as $U = [u_i]_{m\times p}$, $V = [v_j]_{n\times q}$ and $W = [w_{ij}]_{m\times n\times p\times q}$ with the capsule dimensions p and q, respectively. This basic routing architecture is then expanded to deal with sequential sets for multiple tasks. Thus, a memory tensor called multi–task dynamic routing table (MDRT) illuminated as $P = (C^1, C^2, \ldots, C^t)$ is formulated by all the routing matrices of the single task C^t as shown in Figure 2(b). On the sequential learning, the task routing C^t should be selected from the table P by a selection operation $P(t) = C^t$. The mechanism of multi–task routing can be indicated with

$$V = \rho_s(P(t) \cdot WU), \tag{4}$$

where the capsule routing C^t can be dynamically selected in terms of the current tasks index t. In this study, this architecture is called the dynamic memory routing network (DMRN) as shown in Figure 2(a).

Assuming that each of the projected capsules $\hat{u}_{j|i}$ is weighted by the task routing C^t, then the routed capsules exist as $\tilde{U} = C^t \cdot \hat{U}$. It implies that this capsule routing tends to feedforward the more discriminative capsules in $\hat{U}$ for the current task t. Inspirited by it, the routing matrix C^t can be considered as a clustering centre of these projected capsules $\hat{U}$ depending on the current input. Then, the dynamic selection of task routing is achieved by comparing the distances between $\hat{U}$ and all the clustering centre, namely P. In order to ignore the influence of element amplitudes between them, a task label t is then predicted by minimising the cosine distance

$$\begin{aligned} \hat{t} &= \min_t d_{\text{cosine}}(C^t, \bar{U}) \\ &= \min_t \left(1 \frac{<\overrightarrow{C}^t, \overrightarrow{\bar{U}}>}{\|\overrightarrow{C}^t\| \|\overrightarrow{\bar{U}}\|}\right), \end{aligned} \tag{5}$$

where $\bar{U}$ is the compressed tensor of $\hat{U}$ by averaging on the capsule dimension, and $\overrightarrow{(.)}$ denotes the vectorisation with matrix flattening. Then the routing equation can be rewritten as

$$V = \rho_s(P(\min_t d_{\text{cosine}}(C^t, \bar{U})) \cdot \hat{U}). \tag{6}$$

In terms of this equation, the current task routing also can be selected by a multiplication on channel dimension $P(\hat{t}) = P \cdot \hat{\mathbf{t}}$ with one–hot prediction $\hat{\mathbf{t}} = onehot(\hat{t})$. Moreover, the capsule routing P should perform as a kind of hard weighting mechanism in order to separate from the continual weighting W; thus all the elements in $C^t \in P$ are fed forward using the binary approximation as applied in the binary neural network (BNN) [55]. That means the forward path of single task routing should be a binary mask with the elements $\tilde{c}_{ij}^t \in (0, 1)$, obtained by an activation operation $\tilde{C}^t = \rho_r(C^t)$ with a binary gate function ρ_r. Otherwise, in the back–propagation phase, this non–differentiable operation should be dropped to ensure the updating of continual gradients. Thus, the forward paths of capsules denoted by $V^t = \rho_s(\tilde{C}^t \cdot WU)$ can also provide gradient updating for all the differentiable variables as

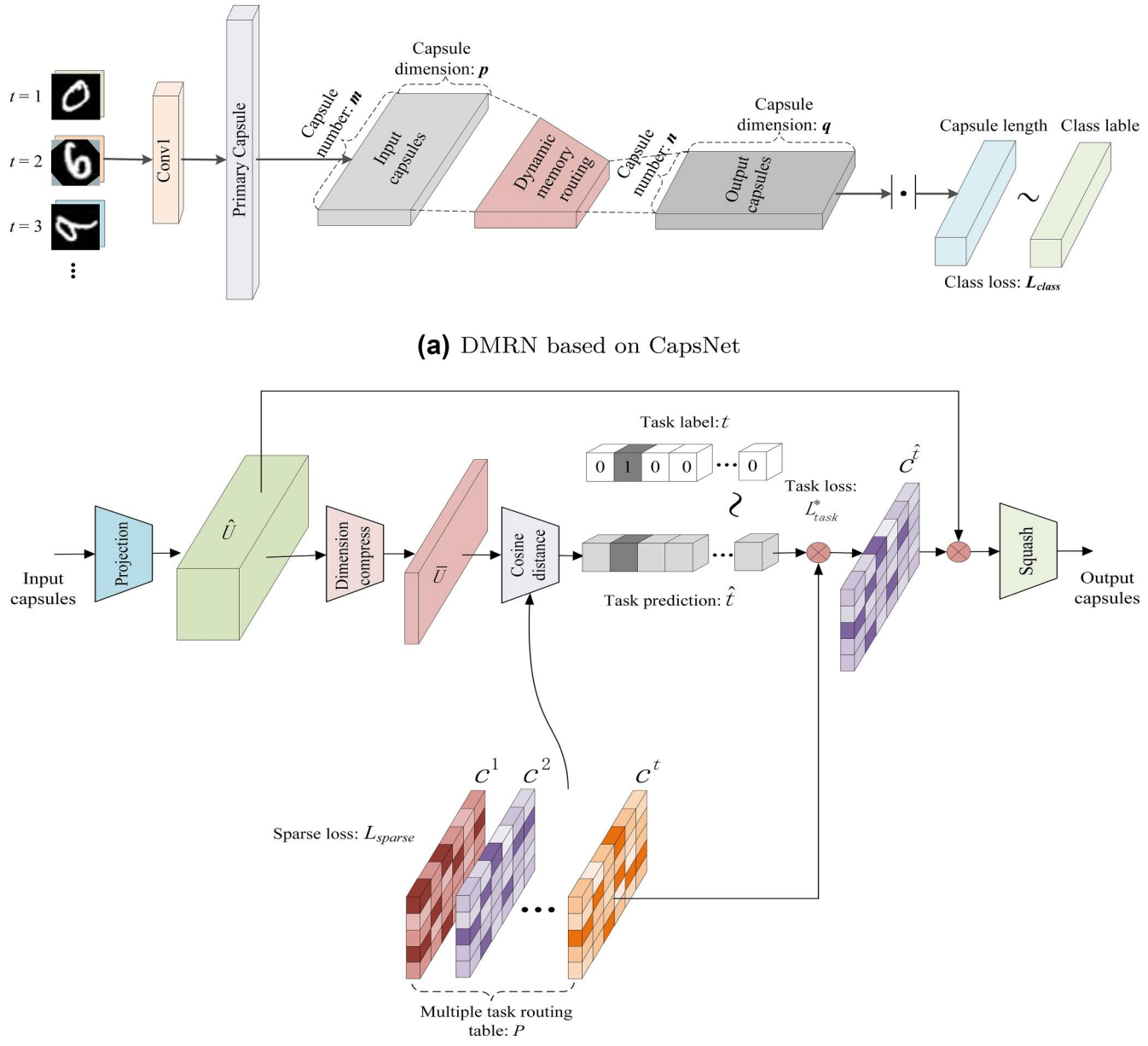


FIGURE 2 The proposed dynamic memory routing network (DMRN). (a): Overall block diagram of DMRN based on CapsNet. (b): Detailed flow chart of the dynamic memory routing using the multi–task dynamic routing table (MDRT)

$$\begin{cases} \nabla w_{ij} = \widetilde{c}_{ij}^{t} \cdot e_{ij} \cdot u_i \\ \nabla \widetilde{c}_{ij}^{t} = e_{ij} \cdot w_{ij} \cdot u_i \\ \nabla u_i = \sum_j \widetilde{c}_{ij}^{t} \cdot e_{ij} \cdot w_{ij}, \end{cases} \tag{7}$$

where e_{ij} is the error value back–propagated from the higher layers. It implies that if the specific capsule routing is inactivated when $\widetilde{c}_{ij}^{t} = 0$, the related gradient updating tends to be restrained. The above formulations provide a harmonious updating strategy of the whole network weights excluding the intrinsic MDRT, which is unfolded in the next section.

3.3 | Increment prototype clustering

To achieve sequential learning, the key mechanism is to analyse the input task sets on–line, then based on it to automatically switch one task routing to another as a retrospective of the previous memory. In this study, the routing table P is thus

dynamically updated by a prototype clustering in an increment manner. The task routing $P(t) = C^t$ is intrinsically the clustering prototypes according to the projected capsules $\bar{U}$ as illuminated in Figure 3. Thus, the proposed increment prototype clustering (IPC) first calculates the minimum cosine distances between them to estimate the task index $\hat{t} = \min_t d_{\text{cosine}}(C^t, \bar{U})$ within the currently learnt task range, after which the prototype P is updated by

$$P(\hat{t})^s = \begin{cases} P(\hat{t})^{s1} + \Delta s \cdot (\bar{U} P(\hat{t})^{s1}), & if \quad \hat{t} = t \\ P(\hat{t})^{s1} \Delta s \cdot (\bar{U} P(\hat{t})^{s1}), & if \quad \hat{t} \neq t, \end{cases} \tag{8}$$

where s is the updating step with a step length $\Delta s \in (0, 1)$, and this formula is the standard prototype clustering as described in Ref. [56]. Note that if the task prediction $\hat{t}$ is correct, the prototype $P(\hat{t}) = C^{\hat{t}}$ is updated by the equation in the first line, otherwise the mislabelled prototype is penalised by the second line, which is uniformly applied during the whole learning iterations. Furthermore, these above prototype updating equations can be uniformly integrated into the whole feedforward of CapsNet through a softmax operation

$$p(\hat{t} = t|U) = \frac{\exp(d(\bar{U}, P(t)))}{\sum_{t'} \exp\big(d(\bar{U}, P(t'))\big)}. \tag{9}$$

This equation is aligned to the current clustering number t increasing during the whole sequential task learning, which implies that this softmax operation can perform a prototype clustering in an increment manner. It means that the previous tasks t' are shared by their learnt knowledge to discriminate the current task t. Then, the updating loss of the routing table P can be indicated in terms of the negative log–probability

$$\begin{aligned} L_{\text{task}} &= \log p(\hat{t} = t|\bar{U}) \\ &= d(\bar{U}, P(t)) \sum_{t'} d(\bar{U}, P(t')), \end{aligned} \tag{10}$$

which is practically equivalent to the former prototype updating equations as shown in the second line. According to the memory networks, the local gradient restrain is important to maintain the previously learnt knowledge. The learning strategy should tend to focus on the current task routing. Thus, the one–hot task label $\mathbf{t} = onehot(t)$ is applied as a restrained mask of outputs for the training phase, and the masked task loss can be expressed as

$$L^*_{\text{task}} = \log p(\mathbf{t} \cdot \hat{\mathbf{t}} = \mathbf{t}|\bar{U}) \tag{11}$$

where $\hat{\mathbf{t}}$ is one–hot outputs of the task predictions sharing the same dimension with $\mathbf{t}$. Therefore, only the dynamic routing aligned to the output neurons of the reference label can be updated through back–propagation. In addition, the activated elements in MDRT should be sparse to build salient and discriminative routing paths for each sequential task. Thus, the intrinsic MDRT is restricted by norm–$L1$ loss

$$L_{\text{sparse}} = \sum_t \| \overrightarrow{P}(t) \|_1 \tag{12}$$

for each element in the learnt routing table $P(t)$. Finally, the total routing loss of the capsule layer can be given by

$$L_{\text{route}} = L^*_{\text{task}} + \lambda \cdot L_{\text{sparse}}. \tag{13}$$

where $\lambda \in [0, 1]$ is the sparse coefficient, and this layered routing loss can be integrally calculated with the final CapsNet loss L_{margin} with the default margin formula. Theoretically, it should be pointed out that the one–hot discriminate function is discontinued. For this reason, it will restrain the gradient back–propagation from the higher layers, and the prototype updating is thus relatively independent of the basic CapsNet.

3.4 | Implementation of the proposed network

The proposed DMRN can be implemented by a uniform continual learning framework as presented in Algorithm 1. In

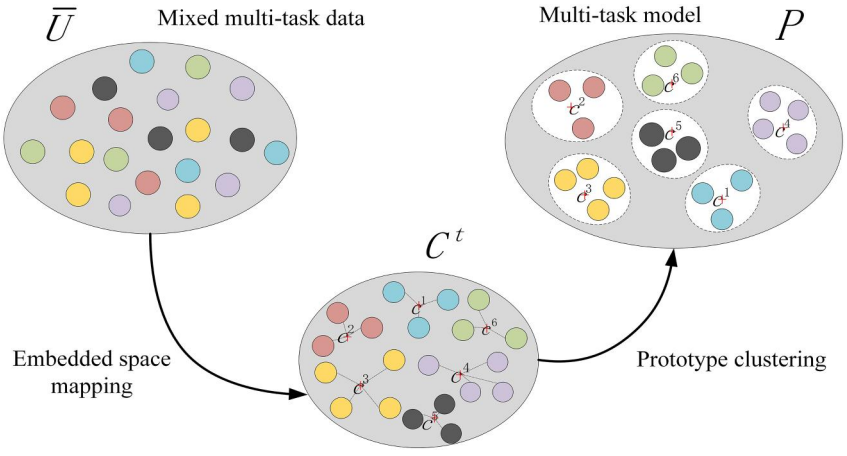


FIGURE 3 The example diagram of the prototype clustering on multiple task sets

addition, though the activated task routes might be completely decoupled by the sparse loss, the MDRT $P(t)$ can also share a part of capsules as co–encoding between these tasks, which would compress the total informative entropy of network structures. In this study, we then attempt to adopt a hard threshold ε^n to regulate the compression rate among different task routes as

$$c_{ij}^t = \rho_{\text{relu}}(c_{ij}^t \varepsilon_{ij}^n) \tag{14}$$

where the activation function $\rho_{\text{relu}}(x) = \max(x, 0)$ and $\varepsilon_{ij}^n = \text{sort}(\{c_{ij}^1, c_{ij}^2, \ldots c_{ij}^t\}, n)$ define the n^{th} element of the former set in descending order, thus n is the number that one capsule can be simultaneously activated among different tasks. In this study, n is uniformly assigned to 1 to apply a strong mutex restrain.

Algorithm 1 Dynamic memory routing algorithm

Input:
The output capsules U of the previous layer;
Output:
The input capsules V of the next layer.
feedforward:
1: Project the input capsules by $\hat{U} = WU$;
2: predict the task label by $\hat{t} = \min_t d_{\text{cosine}}(C^t, \bar{U})$;
3: calculate the output capsules $V = \rho_s(P(\hat{t}) \cdot \hat{U})$;
4: output V to the next layer.
loss calculation:
5: calculate the loss of classification $L_{\text{task}}^* = \log p(\mathbf{t} \cdot \hat{\mathbf{t}} = \mathbf{t} | \bar{U})$;
6: calculate the loss of sparse routing table $L_{\text{sparse}} = \| \overrightarrow{P}(t) \|_1$;
7: obtain the total loss by $L_{\text{route}} = L_{\text{task}}^* + \lambda \cdot L_{\text{sparse}}$
backward propagation:
8: update the projection weights by $\nabla w_{ij} = \widetilde{c}_{ij}^t \cdot e_{ij} \cdot u_i$;
9: update the current routing table by $\nabla \widetilde{c}_{ij}^t = e_{ij} \cdot w_{ij} \cdot u_i$;
10: update the error of input capsules by $\nabla u_i = \sum_j \widetilde{c}_{ij}^t \cdot e_{ij} \cdot w_{ij}$.

To our knowledge, the present research studies related to memory and forgetting have attracted more attentions on differential neural gates. For instance, the existing LSTM/RNNs have similar memory gates that are dominated by both the current inputs and the last states as

$$y^t, \xi^t = f(x^t, \xi^{t1}) \tag{15}$$

where both the state ξ^t and the output y^t should be updated in each step. By contrast, the proposed DMRN has no temporal state that is coupled with the sequence model. Furthermore, the prototype routing $P(t)$ can recall the feedforward paths learnt previously, and also this operation is independent of the temporal sequence. Then the proposed architecture can be expressed as

$$y^t = f(x^t, \theta, \delta(t)) \tag{16}$$

which only depends on the input x^t and the model structure $\delta(t)$ aligned to the attribution of the current input set d^t. Therefore, the proposed architecture is a fully feedforward network that spontaneously achieves route selection according to the inputs of sequential learning. In addition, the network structures $\delta(t)$ can be considered as the prior condition to replace the initialised states ξ^{t1}, which also implies that the mechanism of dynamic memory routing is decoupled on the temporal domain.

4 | EXPERIMENTS AND ANALYSIS

4.1 | Experimental settings

In order to evaluate the performance of different models in overcoming catastrophic forgetting, extensive experiments are performed on datasets including MNIST (LeCun et al., 1998), Cifar–100 (Krizhevsky and Hinton, 2009), SVHN (Yuval Netze et al., 2011), Fashion–MNIST (Xiao et al., 2017), CelebA (Liu et al., 2016) and Tiny ImageNet (J. Deng et al., 2009). In order to formulate different sequential tasks, each dataset is sequentially extended by data enhancements such as axis rotations, as shown in Figure 4. In the experiments, various datasets are extended.

For online sequential learning, which is more difficult than other existing reports on overcoming catastrophic forgetting. In details, CIFAR–100 is composed of 60k 32×32 RGB images of 100 classes, with 600 images per class. Each class has 500 images for training and 100 images for testing. MNIST consists of 28×28 grey–scale images of handwriting, and Fashion–MNIST comprises grey–scale images of the same size. SVHN includes digits cropped 32×32 colour images from house number scene, and CelebA involves numerous face images with 218×178 pixels and each face aligned with 40 attributions as multi–annotations. Tiny ImageNet is a $64 \times 64 \times 3$ resized version of ImageNet with 200 classes. The sequential learning on the sets involving globally spatial transformations is a challenge at present. Thus, each image in these sets is rotated with different angles of 18° to formulate the sequential task sets. We extensively compare the proposed technique with existing state–of–the–art methods for overcoming CF. These default set baselines include EWC [27], HAT [23], LWF [29], IMM (mode) [30], less–forgetting learning (LFL) [31] and PathNet [18] as well as some traditional methods, including standard SGD with dropout operation [14], CLAW [32], TFW [34], iCaRL [9] and RPS–Net [19].

In comparison experiments, employing an AlexNet architecture (Krizhevsky et al., 2012) with 3 convolutional layers of

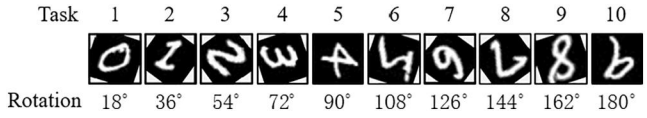


FIGURE 4 Schematic of the split Mixed National Institute of Standards and Technology database (MNIST) task protocol. Among them, each task includes all types of pictures, that is, the numbers 0–9. And the processing of other datasets is the same as MNIST

64, 128, and 256 filters with 4 × 4, 3 × 3, and 2 × 2 kernel sizes, respectively, plus two fully connected layers of 2048 units each. In this study, a CapsNet–based architecture containing two convolutional layers and one fully connected layer is implemented [39]. The first layer (Conv1) has 256 channels, 9 × 9 convolution kernels with single stride and ReLU activation. This layer converts pixel intensities to the activities of local features that are then used as inputs to the next layer. The second layer (Primary Capsules) is a convolutional capsule layer with 32 channels of convolutional 8D capsules (i.e. each primary capsule contains 8 convolutional units with a 9 × 9 kernel). Each primary capsule output yields the outputs of all Conv1 units whose receptive fields overlap with the location of the centre of the capsule. In total, primary capsules have 32 × 6 × 6 capsule outputs (each output is an 8D vector) and each capsule in the 6 × 6 grid shares its weight with the other. The final layer (Digit Caps) has one 16–D capsule per digit class and each of these capsules receives input from all the capsules in the layer below. On this basis, the prototype clustering sub–network is integrated with the dynamic memory routing based on sparse restrain, so that the parameters can be updated online after each optimisation iteration.

The memory capability can be measured by training the model on a new task set and then using the previously learnt sets to test the recalling accuracy. To achieve the sequential analysis, we resize images in CelebA to 109 × 89, while maintaining the other datasets with the original size. For all the evaluating tasks, the unified learning rate and the sparse loss coefficient are fixed at 0.001 and 0.2, respectively, and each batch size is set to 48. All the methods share the same task order, data split rate, batch shuffle operation and weight initialisation. The other network configurations are selected manually. In different tasks, the optimal number of channels for each task is empirically chosen and maintained, and then the optimal capsule dimension is determined through multiple test runs.

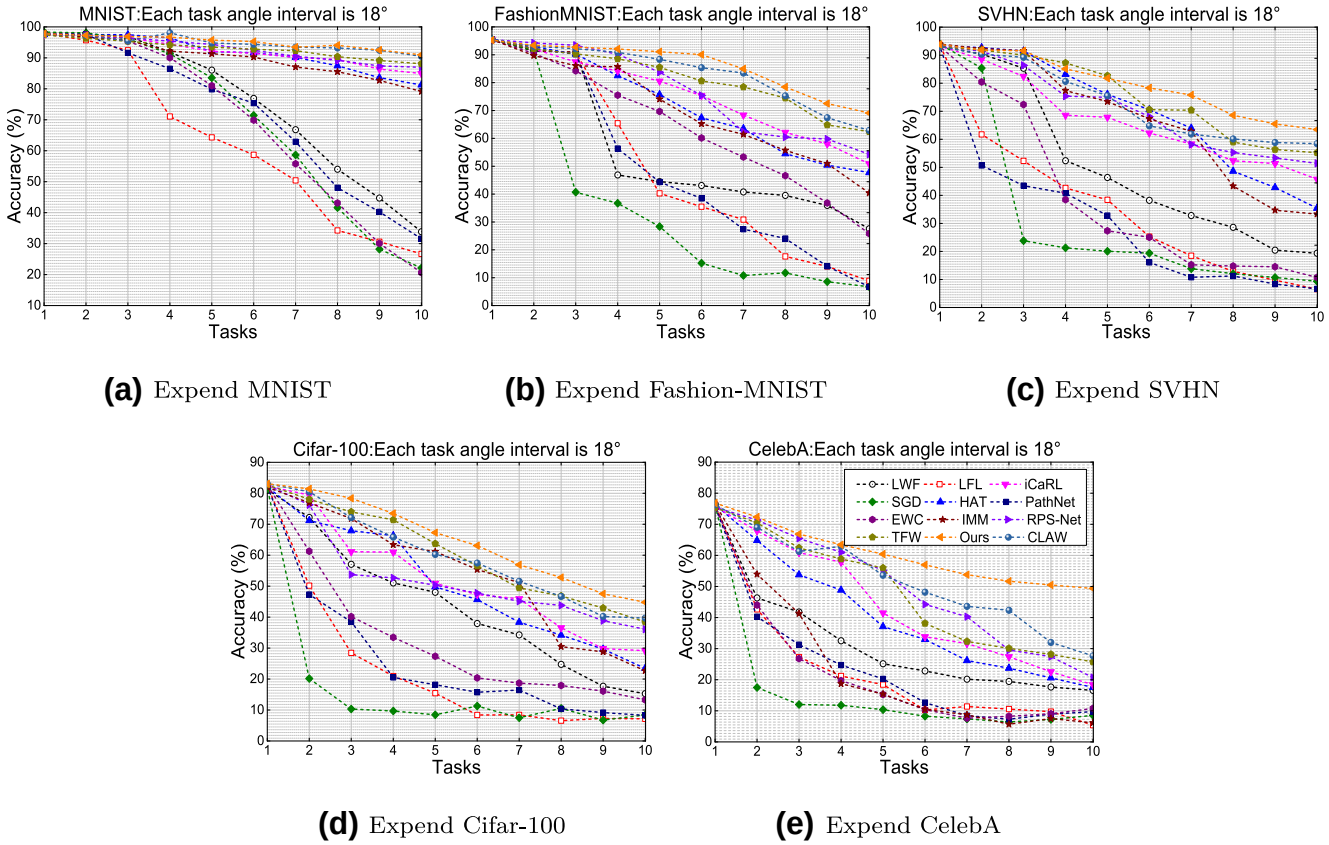
4.2 | Results and comparisons

The purpose of this experiment is to assess the impact of learning previous tasks on the current task. In other words, we want to evaluate whether an algorithm avoids the CF problem, by evaluating the relative performance achieved on a unique task after learning a varying number of previous tasks. For the sake of fairness, the task after training will not be retrained. The experimental dataset was expanded to 10 tasks with the same rotation angle interval. Among them, a single task revolves 18° clockwise or anticlockwise with the image as the centre. In other words, the rotation angle of the first task is 18° and the rotation angle of the 10th task is 180°.

We extensively compare the proposed technique with existing state–of–the–art methods for overcoming CF in Figure 5. The results indicate the superior performance of the proposed method in all settings. For the case of expend CelebA in Table 1, we outperform CLAW by an absolute margin of 21.69%. Compared with the second best method, our approach achieves a relative gain of 8.29% and 6.36%, respectively for the case of expend SVHN and expend CIFAR–100 dataset. In the expend MNIST, our performance is only 0.09% below the CLAW approach. Table 1 compares the accuracy of the 10th task (A_10) of different methods on different datasets. The results show that LWF does well when learning each new task with the help of the representation of the previous tasks. However, as more tasks are included, the older tasks start forgetting more. IMM (mode) has the opposite effect, it focuses on intransigence and tries to keep the knowledge of the older tasks, running out of capacity for the newer tasks. This allows for the approach to not forget much and even has a small backward transfer, but at the cost of performing worse with newer tasks. EWC has one of the worse performances, possibly due to the difficulty of having a good approximation of the FIM when there are so many classes per task. TFW has a good overall performance with non–forgetting, and relies on the amount of capacity of the network more than the other approaches. Probabilistic modelling and variational inference can perform well on specific tasks, but when the task becomes complex, ClAW performance will degrade due to network redundancy. LFL has certain advantages in the first tasks. As the number of tasks increases, it becomes difficult to select hyper–parameters, and the robustness decreases, thus the memory cannot be preserved well. Soft constraint methods such as SGD have similar problems as well.

In order to further verify the performance differences of each model on the conventional dataset and the rotating dataset, we conducted experiments on Tiny ImageNet. Table 2 and Table 3 compare different methods on the conventional dataset and the rotating dataset. The results show that the existing methods perform well in conventional multi–task incremental learning, but CF still exists after image rotation. Hat and RPS–Net are not forgotten in the classes randomly split with no rotation, and the average test accuracy reaches 64.01% and 65.37%. But in the classes randomly split with rotation, their accuracy is reduced to 24.94% and 32.34%. After the task image is rotated, the features become blurred, if the network is not robust to this change, and the identification will become difficult.

In addition, the rotation task will occupy more memory of the network, and the network structure is easier to be redundant. It is not difficult to find out from the comparison between Figure 6 and Figure 7, and the accuracy of new tasks of EWC, LFL and LWF is always lower than that of old tasks. Capsule network just solves this problem. This may benefit from the fact that the proposed routing table generated by prototype clustering involves fewer activated overlapping neurons, so the increase in the number of tasks does not easily affect the network weights learnt in a relatively long period.



(a) Expend MNIST **(b)** Expend Fashion-MNIST **(c)** Expend SVHN

(d) Expend Cifar-100 **(e)** Expend CelebA

FIGURE 5 The accuracy of different methods on different datasets is compared, and one task is learnt incrementally at a time

TABLE 1 Average test accuracy (accuracy of the first task after learning all tasks.) on different datasets' task protocol

Methods	MNIST (A_{10})	Fashion–MNIST (A_{10})	SVHN(A_{10})	Cifar–100 (A_{10})	CelebA (A_{10})
LWF	33.84 (±0.27)	27.57 (±0.67)	19.36 (±1.63)	15.29 (±1.26)	16.57 (±2.87)
LFL	26.7 (±0.45)	8.9 (±2.29)	6.54 (±1.85)	7.16 (±2.63)	5.38 (±3.51)
HAT	81.26 (±0.02)	47.69 (±0.09)	35.42 (±0.84)	23.45 (±1.95)	17.55 (±2.39)
iCaRL	84.93 (±0.01)	50.86 (±0.27)	45.74 (±0.57)	29.18 (±0.89)	18.37 (±0.94)
SGD	22.13 (±0.51)	6.69 (±0.60)	9.37 (±1.35)	8.39 (±1.10)	8.53 (±5.37)
PathNet	31.57 (±0.32)	6.72 (±2.51)	6.63 (±1.75)	8.28 (±2.13)	9.73 (±3.41)
RPS–Net	86.97 (±0.05)	54.29 (±0.02)	51.42 (±1.26)	36.07 (±1.61)	20.69 (±2.16)
EWC	20.65 (±0.20)	25.83 (±0.36)	10.73 (±0.55)	13.29 (±1.82)	10.78 (±3.22)
IMM	79.15 (±0.03)	40.38 (±0.17)	33.31 (±1.005)	22.68 (±1.65)	6.14 (±3.08)
TFW	88.07 (±0.34)	62.39 (±0.04)	55.19 (±0.19)	38.43 (±1.38)	25.73 (±1.57)
CLAW	90.47 (±0.02)	62.83 (±0.02)	58.42 (±0.13)	39.67 (±0.65)	27.68 (±0.97)
Ours	90.38 (±0.01)	69.1 (±0.03)	63.48 (±0.12)	44.79 (±0.67)	49.37 (±0.97)

Note: Each experiment was performed 10 times with different random seeds, and the mean (SEM) over these runs is reported.

Abbreviations: EWC, elastic weight consolidation; LFL, less–forgetting learning; LWF, Learning without forgetting; MNIST, Mixed National Institute of Standards and Technology database; SGD, stochastic gradient descent.

The vector feature of the image is represented by the capsule, which can effectively extract the feature of the rotation image. Dynamic memory routing updates each training batch in time to improve memory efficiency and reduce memory consumption. From Figure 7, it is not difficult to find that our model can achieve the maximum accuracy under the condition that each task has 50 training batches.

In general, CapsNet is helpful to improve the recognition rate of rotating images as shown in Figure 1. However, the improvement of memory performance benefits from the

TABLE 2 Comparison with the state–of–the–art

Tiny ImageNet – classes randomly split with no rotation											
APPROACH	**Task1 (0°) (1–50)**	**Task2 (0°) (51–100)**	**Task3 (0°) (101–150)**	**Task4 (0°) (151–200)**	**Task5 (0°) (201–250)**	**Task6 (0°) (251–300)**	**Task7 (0°) (301–350)**	**Task8 (0°) (351–400)**	**Task9 (0°) (401–450)**	**Task10 (0°) (451–500)**	**Avg. All**
LWF	22.32 (–46.13)	36.38 (–22.36)	42.45 (–23.33)	45.89 (–17.86)	50.38 (–17.44)	52.87 (–11.50)	55.38 (–7.51)	56.43 (–8.85)	60.43 (–2.36)	61.45 (0.00)	48.4
LFL	27.43 (–40.83)	32.58 (–35.68)	33.48 (–34.57)	35.48 (–31.91)	36.37 (–31.05)	40.53 (–27.64)	43.57 (–23.27)	45.69 (–21.39)	51.89 (–15.53)	67.68 (0.00)	41.47
HAT	57.38 (0.00)	63.72 (0.00)	63.85 (0.00)	65.83 (0.00)	68.27 (0.00)	65.17 (0.00)	62.14 (0.00)	62.96 (0.00)	66.48 (0.00)	64.29 (0.00)	64.01
iCaRL	67.64 (+0.25)	60.03 (0.00)	59.17 (0.00)	58.49 (0.00)	60.37 (0.00)	62.38 (0.00)	65.47 (0.00)	64.78 (0.00)	60.09 (0.00)	67.19 (0.00)	62.56
SGD	20.65 (–43.94)	28.67 (–28.69)	32.45 (–30.32)	30.17 (–34.11)	32.17 (–31.61)	33.24 (–28.55)	35.79 (–26.75)	37.69 (–25.11)	55.32 (–8.13)	60.89 (0.00)	36.7
PathNet	24.37 (–41.04)	27.29 (–38.12)	31.08 (–34.24)	35.49 (–29.98)	31.28 (–33.11)	36.27 (–29.02)	38.49 (–26.93)	42.38 (–22.83)	58.08 (–7.17)	65.24 (0.00)	38.99
RPS–Net	68.18 (–0.01)	64.28 (0.00)	62.07 (0.00)	66.29 (0.00)	62.07 (0.00)	67.35 (0.00)	65.43 (0.00)	66.42 (0.00)	65.35 (0.00)	66.28 (0.00)	65.37
EWC	28.69 (–39.47)	27.34 (–40.79)	32.13 (–36.32)	37.08 (–31.39)	40.84 (–27.64)	43.57 (–24.75)	46.23 (–21.96)	52.37 (–15.80)	60.29 (–7.90)	57.37 (0.00)	42.59
IMM	65.34 (0.00)	55.48 (–2.16)	46.38 (–0.31)	38.5 (–1.53)	35.29 (–1.08)	36.19 (–1.34)	33.17 (–0.91)	30.04 (–0.65)	27.92 (–0.15)	27.45 (0.00)	39.57
TFW	63.6 (0.00)	60.57 (0.00)	57.69 (0.00)	55.48 (0.00)	56.37 (0.00)	52.78 (0.00)	50.29 (0.00)	48.39 (0.00)	46.27 (0.00)	44.39 (0.00)	53.59
CLAW	66.35 (+0.06)	64.38 (0.00)	65.28 (+0.15)	63.78 (0.00)	65.17 (0.00)	64.38 (0.00)	63.79 (0.00)	60.28 (0.00)	62.17 (0.00)	64.22 (0.00)	63.98
Ours	68.42 (0.00)	65.75 (0.00)	66.27 (0.00)	66.02 (0.00)	62.54 (0.00)	66.91 (+0.01)	65.69 (0.00)	64.47 (0.00)	66.32 (0.00)	65.38 (0.00)	65.77

Note: Tiny ImageNet with no rotation on different models from scratch. Accuracy of each task after learning all tasks. The number between brackets indicates forgetting. Classes are randomly split and fixed for all approaches.

Abbreviations: EWC, elastic weight consolidation; LFL, less–forgetting learning; LWF, Learning without forgetting; SGD, stochastic gradient descent.

TABLE 3 Comparison with the state–of–the–art approach

Tiny ImageNet – classes randomly split with rotation											
APPROACH	**Task1 (18°)**	**Task2 (36°)**	**Task3 (54°)**	**Task4 (72°)**	**Task5 (90°)**	**Task6 (108°)**	**Task7 (126°)**	**Task8 (144°)**	**Task9 (162°)**	**Task10 (180°)**	**Avg.**
	(1–50)	(51–100)	(101–150)	(151–200)	(201–250)	(251–300)	(301–350)	(351–400)	(401–450)	(451–500)	All
LWF	11.08 (–52.88)	5.08 (–45.60)	6.25 (–34.78)	7.69 (–24.8)	5.08 (–19.71)	4.09 (–10.2)	5.59 (–4.97)	3.96 (–3.47)	3.19 (–2.23)	6.74 (0.00)	5.88
LFL	8.46 (–52.17)	7.94 (–27.64)	6.38 (–19.95)	7.22 (–12.16)	3.27 (–14.09)	4.27 (–6.39)	6.73 (–1.51)	3.22 (–2.2)	4.26 (–1.21)	4.67 (0.00)	5.64
HAT	24.39 (–35.58)	26.79 (–31.28)	31.48 (–22.79)	32.74 (–17.88)	27.5 (–19.16)	26.84 (–14.38)	15.73 (–19.76)	23.18 (–4.51)	20.96 (–0.82)	19.82 (0.00)	24.94
iCARL	33.68 (–29.11)	28.05 (–29.43)	34.23 (–21.14)	37.32 (–15.04)	35.11 (–13.82)	31.66 (–10.72)	30.29 (–7.63)	21.74 (–10.74)	26.07 (–1.26)	20.57 (0.00)	29.87
SGD	6.83 (–55.24)	28.67 (–28.69)	27.35 (–23.08)	30.17 (–13.69)	7.33 (–22.95)	10.37 (–17.46)	11.08 (–10.5)	7.65 (–6.24)	3.27 (–3.78)	3.2 (0.00)	13.59
PathNet	7.42 (–54.45)	7.25 (–36.61)	31.08 (–34.24)	32.66 (–22.16)	27.09 (–21.98)	23.81 (–17.85)	15.44 (–20.8)	11.15 (–15.26)	14.36 (–9.89)	17.69 (0.00)	18.79
RPS–Net	34.65 (–28.83)	35.04 (–22.65)	37.62 (–15.85)	36.88 (–15.91)	34.2 (–16.16)	34.07 (–11.32)	28.63 (–12.06)	29.21 (–5.94)	27.89 (–2.47)	25.18 (0.00)	32.34
EWC	6.37 (–57.08)	6.13 (–49.29)	7.33 (–40.05)	6.67 (–33.84)	3.27 (–31.95)	4.32 (–26.93)	4.79 (–20.03)	4.39 (–14.03)	8.78 (–4.51)	5.53 (0.00)	5.76
IMM	20.19 (–40.89)	26.39 (–29.84)	33.19 (–19.2)	31.36 (–17.33)	22.43 (–17.79)	24.98 (–10.44)	20.69 (–9.52)	25.47 (–3.17)	17.36 (–3.3)	15.09 (0.00)	23.72
TFW	40.19 (–23.38)	41.38 (–18.00)	42.03 (–12.66)	40.26 (–12.53)	31.09 (–17.6)	36.49 (–9.05)	33.07 (–9.66)	32.69 (–5)	30.69 (–4.78)	31.47 (0.00)	35.94
CLAW	46.69 (–17.23)	47.85 (–14.52)	49.32 (–11.07)	43.71 (–12.73)	44.99 (–9.83)	40.69 (–10.18)	40.69 (–5.29)	35.2 (–6.46)	39.82 (–0.21)	34.29 (0.00)	42.33
Ours	53.07 (–12.25)	52.69 (–10.88)	52.45 (–10.04)	48.29 (–12.79)	54.96 (–4.36)	52.06 (–5.9)	50.83 (–4.65)	48.98 (–3)	46.93 (–0.7)	45.66 (0.00)	50.59

Note: Tiny ImageNet with rotation on different models from scratch. Accuracy of each task after learning all tasks. The number between brackets indicates forgetting. Classes are randomly split and fixed for all approaches.

Abbreviations: CLAW, continual learning with adaptive weights; EWC, elastic weight consolidation; HAT, hard attention to the task; IMM, incremental moment matching; LFL, less–forgetting learning; LWF, Learning without forgetting; SGD, stochastic gradient descent; TFW, Ternary feature masks without any forgetting.

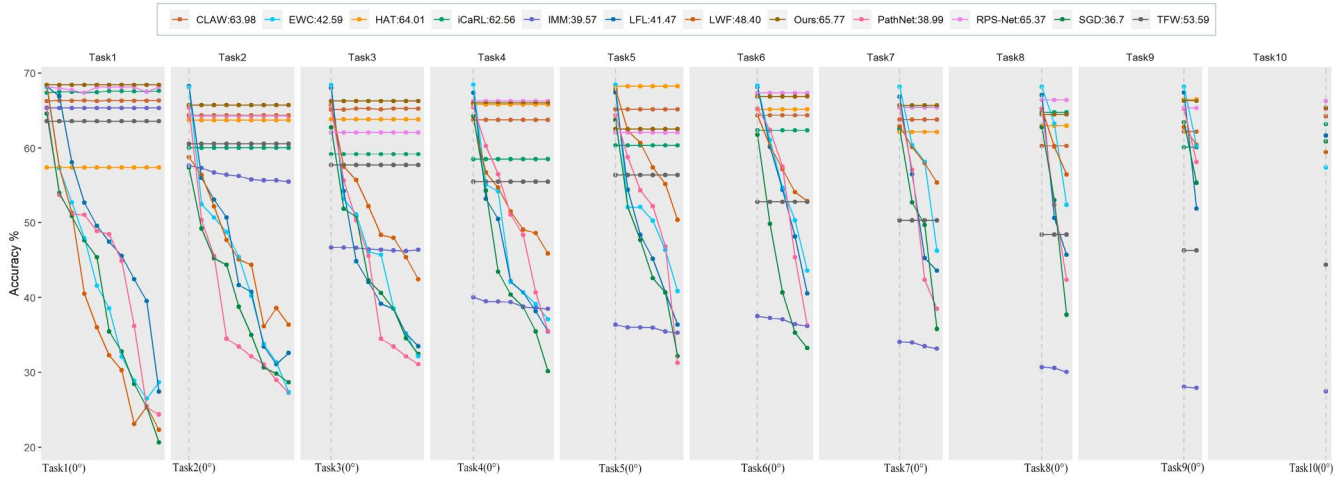


FIGURE 6 Per task accuracy for Tiny ImageNet–classes randomly split with no rotation from scratch

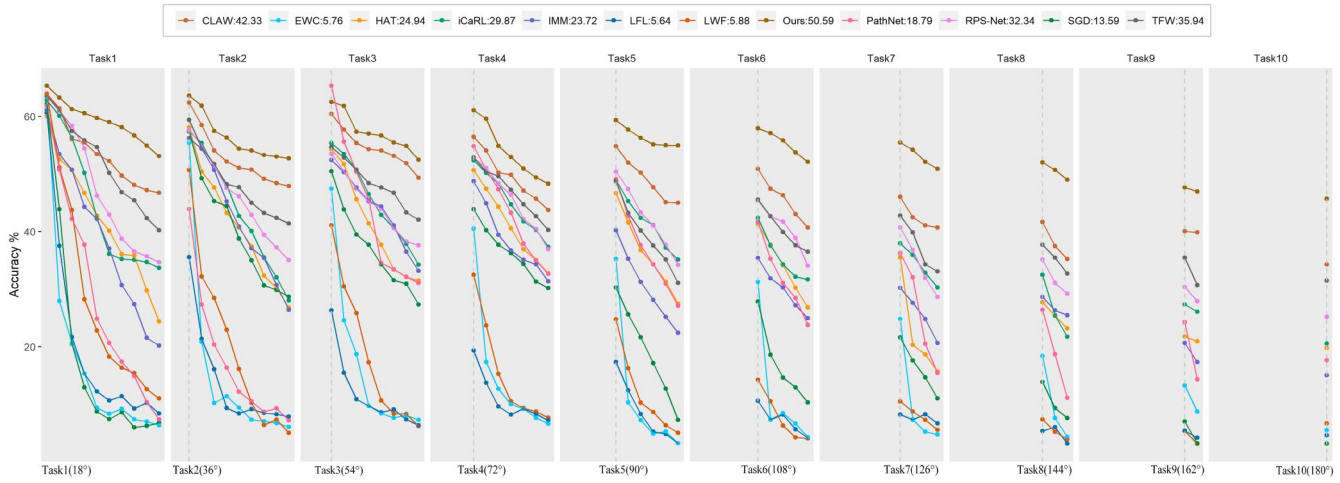


FIGURE 7 Per task accuracy for Tiny ImageNet–classes randomly split with rotation from scratch

capsule architecture as well as the dynamic memory routing to rotating image sequences. It can be seen from Table 2 that our average accuracy is slightly higher than that of the comparison methods on the non–rotating task set. In Table 3, in terms of the memory difference from the last task to the first task, RPS–Net is 9.47%, CLW is 12.4%, and our method is 7.41%. Overall, CapsNet can only improve the accuracy of initial recognition, and the memory preservation and weight pruning still need to be dominated by the proposed dynamic memory routing.

4.3 | Quantitative analysis of parameters

As mentioned above, it is expected that after learning multiple tasks, the network can remember the knowledge about previous tasks as much as possible. In practical procedures, it is found that the parameter of the capsule number for each pixel in convolutional feature maps per task (CNPT) greatly impacts the knowledge of previous tasks. Therefore, in order to determine the suitable scale of capsule networks, these effects on different CNPTs are quantitatively analysed. The specific impacts of this parameter are discussed through additional experiments. Based on these additional configurations, an average of 4 reference points are selected in the training iterations of each task to evaluate the accuracy of the recall, as shown in Figure 8.

To accomplish the analysis, the MNIST set is extended to 5 task sets with rotation operations of interval 30°. Each task has a training set of 30,000 and a test set of 3000. The parameter of CNPT is set to 3, 9, 16, 22, 32, and 48, respectively before recording the test accuracy of the first task. It can be seen in Figure 8 that 16 is the most suitable parameter, and after learning the fifth task, the network has the most knowledge to remember the first task. In addition, when the parameter is too small, after the fifth task is trained, more previous knowledge is forgotten. When the parameter is too large, however, the accuracy will increase gradually despite that the accuracy is not high at the beginning of the training. Besides, large task channels will waste a lot of resources during training.

Therefore, this parameter of task channels should be practically fixed for each sequential task.

Furthermore, the capsule dimension has a direct influence on retaining the previously learnt knowledge. Similarly, the parameter of capsule dimension is discussed. Under the same data set, CNPT is kept as the best parameter, and the capsule dimension parameters are changed for comparison experiments. On the task configurations, the network performance is the best when the capsule dimension is 4, as shown in Figure 9. The parameter is too small (e.g. 2–D) to provide the required network capacity for the task sequences, thus the network cannot be fully trained and the test accuracy of the first task will quickly decline. In case of a too large parameter (e.g. 32–D and 48–D), although the first two tasks are slightly benefited, the small redundancy will result in a degraded performance when $t > 3$. After numerical testing implementations, it is found that the relative optimal parameters for each selected reference dataset are different. Furthermore, the dimension of the capsules determines the representation capability of spatial attributions. In summary, the optimal task channels corresponding to MNIST, Fashion–MNIST, Cifar–100 and CelebA are, respectively, 16, 16, 22, 32 and 48 and the optimal capsule dimensions are, respectively, 4, 8, 8, 16 and 22. Overall, the task channels and capsule dimensions are mainly determined by the dataset. They are also slightly affected by the rotation angle, but this effect is small enough to be ignored.

5 | CONCLUSION

This study presents an extended CapsNet that uses an increment prototype clustering to yield dynamic memory routing. It has been proved to be capable of retaining the previous knowledge while learning new task sets. In the experiments, the testing image sets are enhanced by axis rotations. Through a series of experiments, the effectiveness of the proposed network in overcoming catastrophic forgetting is verified and compared with some state–of–the–art approaches. The main conclusions are drawn as follows: First, the capsule–based memory architecture has the adaptability to global axis rotation and performs much better than the CNN–based approaches. Second, the prototype clustering–based dynamic memory routing can be successfully applied in sequential classification tasks so that the recalling performance is improved. In addition, the sparse routing table can efficiently reduce weight overlapping, especially when the number of

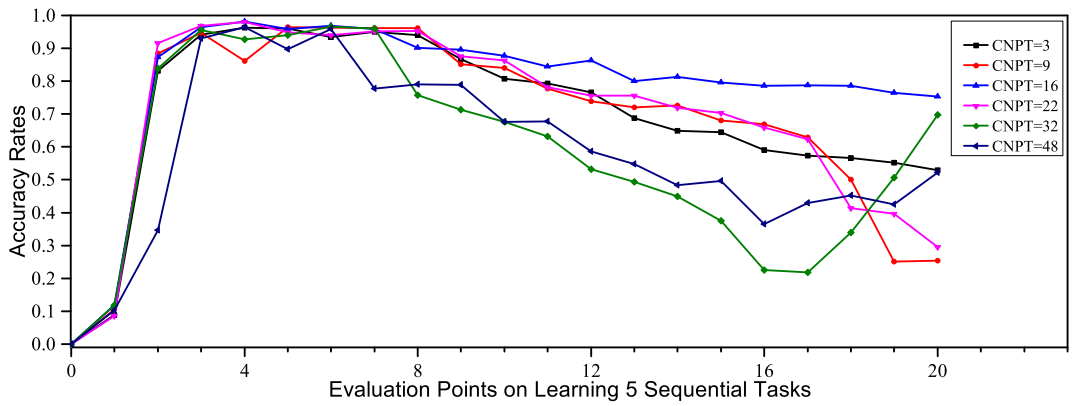


FIGURE 8 The quantitative effect of convolutional feature maps per task (CNPTs) is measured by the accuracy on the first task set when the training iterations are finished at each following evaluation point

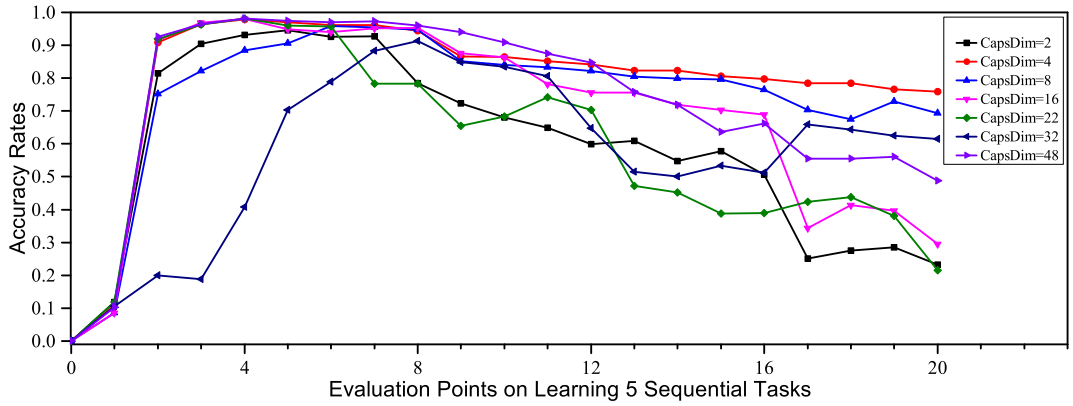


FIGURE 9 The quantitative effect of capsule dimensions is measured by the accuracy on the first task set when the training iterations are completed at each subsequent evaluation point

tasks increases. For challenging cases such as object deformations and occlusions, the enhancement of the dynamic model and efficiency needs further study.

ACKNOWLEDGEMENT

This research was funded by National Natural Science Foundation of China (no. 6206204).

DATA AVAILABILITY STATEMENT

Our experiments were conducted on public data sets, including MNIST, Fashion–MNIST, SVN, Cifar–100, CelebA and Tiny ImageNet.

ORCID

Zhengbing Guo https://orcid.org/0000-0002-1049-8565

REFERENCES

1. Robins, A.: Catastrophic forgetting, rehearsal and pseudorehearsal. Connect Sci. 7(2), 123–146. (1995)
2. Kemker, R., et al.: Measuring catastrophic forgetting in neural networks. In: Thirty–second AAAI conference on artificial intelligence (2018)
3. Parisi, G.I., et al.: Continual lifelong learning with neural networks: A review. Neural Network. 113 (2019)
4. Ans, B., Rousset, S.: Neural networks with a self–refreshing memory: Knowledge transfer in sequential learning tasks without catastrophic forgetting. Connect Sci. 12(1), 1–19 (2000)
5. Lee, S.–W., et al.: Dual–Memory Deep Learning Architectures for Lifelong Learning of Everyday Human Behaviors. IJCAI. pp. 1669–1675. (2016)
6. Rusu, A.A., et al.: Progressive Neural Networks. arXiv (2016)
7. Castro, F.M., et al.: End–to–end incremental learning. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 233–248. (2018)
8. Hou, S., et al.: Learning a unified classifier incrementally via rebalancing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 831–839. (2019)
9. Rebuffi, S.–A., et al.: iCaRl: Incremental classifier and representation learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2001–2010. (2017)
10. Shin, H., et al.: Continual learning with deep generative replay. In: Advances in Neural Information Processing Systems, pp. 2990–2999. (2017)
11. van de Ven, G.M., Tolias, A.S.: Generative Replay with Feedback Connections as a General Strategy for Continual Learning. arXiv preprint arXiv:1809, p. 10635 (2018)
12. Wu, Y., et al.: Large scale incremental learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 374–382. (2019)
13. Goodfellow, I.J., et al.: An Empirical Investigation of Catastrophic Forgetting in Gradient–Based Neural Networks. arXiv preprint arXiv:1312.6211 (2013)
14. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680. (2014)
15. Srivastava, N., Salakhutdinov, R.R.: Discriminative transfer learning with tree–based priors. In: Advances in Neural Information Processing Systems, pp. 2094–2102. (2013)
16. Fuangkhon, P., Tanprasert, T.: An incremental learning algorithm for supervised neural network with contour preserving classification. In: 2009 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2, pp. 740–743. IEEE (2009)
17. Ren, J., et al.: Machine learning scan and application in susy. Tech. Rep. 8 (2017). https://arxiv.org/abs/1708.06615
18. Fernando, C., et al.: Pathnet: Evolution Channels Gradient Descent in Super Neural Networks. arXiv preprint arXiv:1701.08734 (2017)
19. Rajasegaran, J., et al.: Random path selection for continual learning. In: Advances in Neural Information Processing Systems, pp. 12669–12679. (2019)
20. Peng, J., et al.: Overcoming Catastrophic Forgetting by Soft Parameter Pruning. arXiv preprint arXiv:1812.01640 (2018)
21. Mallya, A., Lazebnik, S.: Packnet: Adding multiple tasks to a single network by iterative pruning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7765–7773. (2018)
22. Liu, Z., et al.: Metapruning: Meta learning for automatic neural network channel pruning. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3296–3305. (2019)
23. Serrà, J., et al.: Overcoming Catastrophic Forgetting with Hard Attention: To the Task. arXiv preprint arXiv:1801.01423 (2018)
24. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: Proceedings of the 34th International conference on Machine learning–Volume 70. JMLR, pp. 3987–3995. (2017)
25. Liu, X., et al.: Rotate your networks: Better weight consolidation and less catastrophic forgetting. In: 2018 24th International Conference on Pattern Recognition (ICPR). IEEE, pp. 2262–2268. (2018)
26. Early, J.: Reducing Catastrophic Forgetting when Evolving Neural Networks. arXiv preprint arXiv:190403178 (2019)
27. Kirkpatrick, J., et al.: Overcoming catastrophic forgetting in neural networks. Proc. Natl. Acad. Sci. Unit States Am. 114(13), 3521–3526 (2017)
28. Chaudhry, A., et al.: Efficient Lifelong Learning with a–Gem. arXiv preprint arXiv:1812.00420 (2018)
29. Li, Z., Hoiem, D.: Learning without forgetting. In: Ieee Transactions on Pattern Analysis and Machine Intelligence. 40(12), 2935–2947 (2017)
30. Lee, S.–W., et al.: Overcoming catastrophic forgetting by incremental moment matching. In: Advances in Neural Information Processing Systems, pp. 4652–4662. (2017)
31. Jung, H., et al.: Less–Forgetting Learning in Deep Neural Networks. arXiv preprint arXiv:1607.00122 (2016)
32. Adel, T., Zhao, H., Turner, R.E.: Continual Learning with Adaptive Weights (Claw). arXiv preprint arXiv:1911.09514 (2019)
33. Coop, R., Mishtal, A., Arel, I.: Ensemble learning in fixed expansion layer networks for mitigating catastrophic forgetting. IEEE Transactions on Neural Networks And Learning Systems. 24(10), 1623–1634 (2013)
34. Masana, M., Tuytelaars, T., van de Weijer, J.: Ternary feature masks: Continual learning without any forgetting. arXiv preprint arXiv:2001.08714 (2020)
35. Masse, N.Y., Grant, G.D., Freedman, D.J.: Alleviating catastrophic forgetting using context–dependent gating and synaptic stabilization. Proc. Natl. Acad. Sci. Unit States Am. 115(44), E10467–E10475 (2018)
36. Krizhevsky, A., Hinton, G.: Learning Multiple Layers of Features from Tiny Images, vol.7 (2009)
37. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Advances in Neural Information Processing Systems, pp. 3856–3866. (2017)
38. Xi, E., Bing, S., Jin, Y.: Capsule Network Performance on Complex Data. arXiv preprint arXiv:1712.03480 (2017)
39. Hinton, G.E., Sabour, S., Frosst, N.: Matrix capsules with EM routing (2018)
40. Wang, Z., Yan, W., Oates, T.: Time series classification from scratch with deep neural networks: A strong baseline. In: 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1578–1585. (2017)
41. Hubara, I., et al.: Binarized neural networks. In: Advances in Neural Information Processing Systems, pp. 4107–4115. (2016)
42. Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto–encoders. In: International Conference on Artificial Neural Networks, pp. 44–51. Springer (2011)
43. Gupta, S., et al.: Deep learning with limited numerical precision. In: International Conference on Machine Learning, pp. 1737–1746. (2015)
44. Han, S., Mao, H., Dally, W.J.: Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. arXiv preprint arXiv:1510.00149 (2015)

45. Molchanov, P., et al.: Pruning Convolutional Neural Networks for Resource Efficient Transfer Learning. arXiv preprint arXiv:1611.06440 3 (2016)
46. Mittal, D., et al.: Studying the plasticity in deep convolutional neural networks using random pruning. Mach. Vis. Appl. 30(2), 203–216 (2019)
47. Ergün, E., Treyin, B.U.: Continual learning with sparse progressive neural networks. In: The 28th IEEE Conference on Signal Processing and Communications Applications (2020)
48. Li, H., et al.: Noise–Robust Image Fusion with Low–Rank Sparse Decomposition Guided by External Patch Prior. Information Sciences (2020)
49. Li, H., et al.: Joint medical image fusion, denoising and enhancement via discriminative low–rank sparse dictionaries learning. Pattern Recogn. 79, 130–146 (2018)
50. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few–shot learning. In: Advances in Neural Information Processing Systems, pp. 4077–4087. (2017)
51. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. 24(5), 603–619 (2002)
52. Zhao, W., et al.: Investigating Capsule Networks with Dynamic Routing for Text Classification. arXiv preprint arXiv:180400538 (2018)
53. Wang, Y., et al.: Generalizing from a Few Examples: A Survey on Few–Shot Learning. arXiv: 190405046 (2019)
54. Li, H., et al.: Discriminative dictionary learning–based multiple component decomposition for detail–preserving noisy image fusion. IEEE Trans. Instrum. Meas. 69(4), 1082–1102 (2019)
55. Courbariaux, M., et al.: Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained To+ 1 or–1. arXiv preprint arXiv:1602.02830 (2016)
56. Ji, J., et al.: A fuzzy k–prototype clustering algorithm for mixed numeric and categorical data. Knowl. Base. Syst. 30, 129–135 (2012)

SUPPORTING INFORMATION

Additional supporting information may be found in the online version of the article at the publisher's website.

How to cite this article: Wang, M., Guo, Z., Li, H.: A dynamic routing CapsNet based on increment prototype clustering for overcoming catastrophic forgetting. IET Comput. Vis. 16(1), 83–97 (2022). https://doi.org/10.1049/cvi2.12068