Multi-task continuous learning model

Zhengbing Guo\textsuperscript{1,2}, Meng Wang\textsuperscript{2}

\textsuperscript{1} Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, 650500, China

\textsuperscript{2} Yunnan Key Laboratory of Artificial Intelligence, Kunming University of Science and Technology, Kunming, 650500, China

*Corresponding author’s e-mail: wangmeng@kmust.edu.cn

Abstract. In continual learning, previous learned knowledge tends to be overlapped by the subsequent training tasks. This bottleneck, known as catastrophic forgetting (CF), has recently been relieved between vision tasks involving simple tasks. Nevertheless, the challenge lies in the continuous classification of the sequential sets discriminated by global transformations, such as excessively spatial rotations. Aiming 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 current task. Moreover, a sparsity measurement is employed to decouple the salient routing among the different learned tasks. The experimental results demonstrate the superiority of the proposed memory network over the state-of-art approaches by the recalling evaluations on extended sets of SVHN, Cifar-100 and CelebA etc.

1. Introduction

Intelligent creatures can learn and memorize 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 learned knowledge tends to be erased by the overlapping of subsequent training tasks, which is known as catastrophic forgetting\textsuperscript{[1]} (CF). Over the years, various approaches have been proposed to relieve this problem. These existing solutions can be briefly summarized into three categories including self-refreshing memory, stochastic gradient descent (SGD) and restraint approaches\textsuperscript{[2]}.

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 minimize the impact on the important parameters related to historical knowledge while learning new knowledge\textsuperscript{[3]}. 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\textsuperscript{[4,5]}. These approaches often improve the performance of the whole tasks and is practically used in continual learning. However, they has a complexity limitation, especially in inferences, since the number of networks increases with the number of new tasks that need to be learned\textsuperscript{[6]}

Another solution focuses on using implicit distributed storage of information in the typical learning with stochastic gradient descent\textsuperscript{[7]} (SGD). These approaches adopt the idea of dropout or maxout to distributively store the information for each task by utilizing the large capacity of deep neural networks\textsuperscript{[8]}. 


Evidence ofammable
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\cite{9,10}.

Later studies suggest that the CF problem can be effectively relieved by restraining and pruning the connections of deep networks\cite{11,12,13}. Recently, the weight regularization is integrated into the iterative optimization to adapt new tasks by transferring the old network configurations\cite{14}. Also, the evolutionary theory of networks has been further inspired\cite{15} in terms of the efficient transferring strategy of elastic weight consolidation\cite{16,17} (EWC). Learning without forgetting\cite{18} (LWF) utilizes the knowledge distillation loss to preserve the performance of previous tasks. In addition, there are studies that continually train the neural network by using sequential Bayesian estimation\cite{19}, which introduces a constraint in the loss function that directs plasticity to the weight that contributes the most to the previous tasks. CLAW\cite{20} is proposed in terms of the probabilistic modelling and variation inference. Sparse-based approaches often embeds a hidden layer of sparse coding to improve CF\cite{21}, nevertheless the obtained sparsity might impair the learning generalization on new tasks. TFM\cite{22} adopts activation masks on each layer to prevent both catastrophic forgetting and negative transfer. 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.

In this paper, we provide a new solution for continuous multi task learning. We extend CapsNet\cite{23} to overcome the catastrophic forgetting on continual learning. Overall, this model has good adaptability for multi-task continuous learning problems with global transformations (such as excessively spatial rotations). 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. 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 analyzed and discriminated by learning the prototypes of each task set onlin.

2. The proposed Model of DMRN

2.1 Paradigm of sequential learning

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

2.2 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 paper focuses 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 \( f_i : u_i \rightarrow v_j \) with \( v_j = \rho_s(\sum_i (w_{ij}u_i)) \), where the projected capsule \( \hat{u}_{jj} = w_ju_i \), and the routing weights \( c_{ij} = e^{h_i} / \sum_k e^{h_k} \) which 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 \( L_{\text{margin}} \) are manually selected.

On this basis, the routing weights can be denoted by a matrix \( C = [e_{ij}]_{mn} \) for the single task set \( d^t \). This routing matrix is applied to determine which projected capsules \( \hat{U} = [\hat{u}_{ij}]_{mn \times p \times q} \) are activated to be fed forward by the element multiplication as

\[
V = \rho_s(C' \cdot WU) = \rho_s(C' \cdot \hat{U})
\]

where the matrixes is organized as \( U = [u_{ij}]_{nxp} \), \( V = [v_{ij}]_{nxq} \) and \( W = [w_{ij}]_{mn \times p \times q} \) 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 \( P = (C^1, C^2, ..., C^t) \) is formulated by all the routing matrixes of single task \( C^t \) as shown in Figure 1(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 multi-task routing can be indicated with

\[
V = \rho_s(P(t) \cdot WU)
\]

where the capsule routing \( C^t \) can be dynamically selected in terms of the current tasks index \( t \). In this paper, this architecture is called dynamic memory routing network (DMRN) as shown in Figure 1(a).
(b): Detailed flow chart of the dynamic memory routing using the multi-task dynamic routing table (MDRT).

Figure 1. DMRN based on CapsNet

Assuming that each the projected capsules \( \hat{u}_j \) are weighted by the task routing \( \tau_C \), then there exists routed capsules as \( \hat{U} = \tau_C \cdot \hat{U} \). It implies that this capsule routing tends to feedforward the more discriminative capsules in \( \hat{U} \) for the current task \( t \). Inspired by it, the routing matrix \( \tau_C \) can be considered as a clustering center of these projected capsules \( \hat{U} \) depending the current input. Then, the dynamic selection of task routing is achieved by comparing the distances between \( \hat{U} \) and all the clustering centers, namely \( P \). In order to ignore the influence of element amplitudes between them, a task label \( t \) is then predicted by minimizing cosine distance

\[
\hat{t} = \min_t d_{\text{cosine}}(C^t, \hat{U}) \\
= \min_t (1 - \frac{\langle C^t, \hat{U} \rangle}{\|\hat{C}^t\| \|\hat{U}\|})
\]

(3)

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

\[
V = \rho_s(P(\min_t d_{\text{cosine}}(C^t, \hat{U})) \cdot \hat{U})
\]

(4)

In terms of this equation, the current task routing also can be selected by a multiplication on channel dimension \( P(\hat{t}) = P \cdot \hat{t} \) with the one-hot prediction \( t = \text{onehot}(\hat{t}) \). 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) \cite{courbariaux2016binarized}. That means the forward path of single
task routing should be a binary mask with the elements \( \tilde{c}_{ij} \in (0,1) \), which is obtained by an activation operation \( \tilde{C} = \rho_t(C') \) with a binary gate function \( \rho_t \). 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 \( V^t = \rho_s(\tilde{C}' \cdot WU) \) can also provide gradient updating for all the differentiable variables as

\[
\begin{align*}
\nabla w_{ij} &= \bar{e}_{ij} \cdot e_{ij} \cdot u_i \\
\nabla \bar{e}_{ij} &= e_{ij} \cdot w_{ij} \cdot u_i \\
\nabla u_i &= \sum_j \bar{e}_{ij} \cdot e_{ij} \cdot w_{ij}
\end{align*}
\]

where \( e_{ij} \) is the error value back-propagated from the higher layers. It implies that if the specific capsule routing is inactivated when \( \tilde{c}_{ij} = 0 \), 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.

2.3 Increment prototype clustering

To achieve sequential learning, the key mechanism is to analysis the input task sets on-line, then based on it to automatically switch one task routing to another as a retrospective of previous memory. In this paper, the routing table \( P \) is thus dynamically updated by a prototype clustering in an increment manner. The task routing \( P(t) = C^i \) is intrinsically the clustering prototypes according to the projected capsules \( \bar{U} \) as illuminated in Figure 2.

![Figure 2. The example diagram of the prototype clustering on multiple task sets](image)

Thus, the proposed increment prototype clustering (IPC) first calculates the minimum cosine distances between them to estimate task index \( \hat{t} = \min, d_{\text{cosine}}(C^i, \bar{U}) \) within the current learned task range, then the prototype \( P \) is updated by

\[
P(\hat{t})^\gamma = \begin{cases} P(\hat{t})^{\gamma-1} + \Delta s \cdot (\bar{U} - P(\hat{t})^{\gamma-1}), & \text{if } \hat{t} = t \\ P(\hat{t})^{\gamma-1} - \Delta s \cdot (\bar{U} - P(\hat{t})^{\gamma-1}), & \text{if } \hat{t} \neq t \end{cases}
\]

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. Note that if the task prediction \( \hat{t} \) is correct, the prototype \( P(\hat{t}) = C^i \).
is updated by the equation in the first line, otherwise the mislabeled prototype is penalized by the second line, which are uniformly applied during the whole learning iterations. Furthermore, these above prototype updating equations can be uniformly integrated in the whole feedforward of CapsNet through a softmax operation

$$p(\hat{t} = t | U) = \frac{\exp(-d(\hat{U}, P(t)))}{\sum_{i} \exp(-d(\hat{U}, P(i)))}$$  

(7)

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 increment manner. It means that the previous tasks $t'$ are shared their learned knowledge to discriminate the current task $t$. Then, the updating loss of routing table $P$ can be indicated in terms of the negative log-probability

$$L_{\text{task}} = -\log p(\hat{t} = t | U) = d(\hat{U}, P(t)) - \sum_{i} d(\hat{U}, P(i'))$$

(8)

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 previous learned knowledge. The learning strategy should tend to focus on the current task routing. Thus, the one-hot task label $t = \text{onehot}(t)$ is applied as a restrained mask of outputs for the training phase, and the masked task loss can be expressed as

$$L'_{\text{task}} = -\log p(\hat{t} = t | U)$$

(9)

where $\hat{t}$ is one-hot outputs of the task predictions sharing the same dimension with $t$. 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-$L_1$ loss

$$L_{\text{sparse}} = \sum_{i} || \hat{P}(t) ||_1$$

(10)

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

$$L_{\text{route}} = L'_{\text{task}} + \lambda \cdot L_{\text{sparse}}$$

(11)

where $\lambda \in [0,1]$ is the sparse coefficient, and this layered routing loss can be integrally calculated with the final CapsNet loss $L_{\text{margin}}$ 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. Results and discussion

3.1 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 enhancement such as axis rotations, as shown in Figure 3.
Figure 3. Schematic of the split 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.

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 $32 \times 32$ 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 $28 \times 28$ gray-scale images of handwriting, and Fashion-MNIST comprises gray-scale images of the same size. SVHN includes digits cropped $32 \times 32$ color images from house numbers scene, and CelebA involves numerous face images with $218 \times 178$ 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 of the classes. The sequential learning on the sets involving globally spatial transformations are a challenge at present. Thus, each image in these sets is rotated with different angles of $18^\circ$ 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, HAT\cite{24}, LWF, IMM\cite{19}(mode), less-forgetting learning\cite{25}(LFL) and PathNet\cite{26}, as well as some traditional methods, including standard SGD with dropout operation\cite{7}, CLAW, TFM\cite{22}, iCaRL\cite{27} and RPS-Net\cite{28}.

In this paper, a CapsNet-based architecture containing two convolutional layers and one fully connected layer is implemented \cite{hinton2018matrix}. The first layer (Conv1) has 256 channels, $9 \times 9$ 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 8-D capsules (i.e., each primary capsule contains 8 convolutional units with a $9 \times 9$ kernel). Each primary capsule output yields the outputs of all Conv1 units whose receptive fields overlap with the location of the center of the capsule. In total, Primary Capsules has $32 \times 6 \times 6$ capsule outputs (each output is an 8-D vector) and each capsule in the $6 \times 6$ grid shares their weights with each 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 optimization iteration.

The memory capability can be measured by training model on a new task set and then using the previously learned sets to test the recalling accuracy. To achieve the sequential analysis, we resize images in CelebA to $109 \times 89$, while maintain 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 initialization. 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.

3.2 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 re trained. The experimental data set was expanded to 10 tasks with the same rotation angle interval. Among them, a single task revolves $18^\circ$ clockwise or anticlockwise with the image as the center. In other words, the rotation angle of the first task is $18^\circ$ and
the rotation angle of the tenth task is $180^\circ$.

Figure 4. The accuracy of different methods on different datasets is compared, and one task is learned incrementally at a time.

We extensively compare the proposed technique with existing state-of-the-art methods for overcoming CF in Figure 4. The results indicate superior performance of the proposed method in all settings. For the case of expend CelebA in Table 1 and Figure 4(e), 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 are shown in Figure 4 (c) and (d). In the expend MNIST, Figure 4 (a), our performance is only 0.09% below the CLAW approach. Table 1 compares the accuracy of the tenth task \((A^{10})\) of different methods on different dataset. 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 have 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 is so many classes per task. TFM has a good overall performance with non-forgetting, and rely 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.

### Table 1. Average test accuracy (accuracy of the first task after learning all tasks.) on different datasets

| Methods  | MNIST \((A^{10})\) | Fashion-MNIST \((A^{10})\) | SVHN \((A^{10})\) | CIFAR-100 \((A^{10})\) | CelebA \((A^{10})\) |
|----------|-----------------|-----------------|----------------|-----------------|----------------|
| LWF      | 33.84\(\pm 0.3\) | 27.57\(\pm 0.7\) | 19.36\(\pm 1.6\) | 15.29\(\pm 1.3\) | 16.57\(\pm 2.9\) |
| LFL      | 26.7\(\pm 0.5\)    | 8.9\(\pm 2.3\)     | 6.54\(\pm 1.9\)     | 7.16\(\pm 2.6\)     | 5.38\(\pm 3.5\)     |
| HAT      | 81.26\(\pm 0.0\)    | 47.69\(\pm 0.1\)    | 35.42\(\pm 0.8\)    | 23.45\(\pm 2.0\)    | 17.55\(\pm 2.4\)    |
| iCaRL    | 84.93\(\pm 0.0\)    | 50.86\(\pm 0.3\)    | 45.74\(\pm 0.6\)    | 29.18\(\pm 0.9\)    | 18.37\(\pm 0.9\)    |
| SGD      | 22.13\(\pm 0.5\)    | 6.69\(\pm 0.6\)     | 9.37\(\pm 1.4\)     | 8.39\(\pm 1.1\)     | 8.53\(\pm 5.4\)     |
| PathNet  | 31.57\(\pm 0.3\)    | 6.72\(\pm 2.5\)     | 6.63\(\pm 1.8\)     | 8.28\(\pm 2.1\)     | 9.73\(\pm 3.4\)     |
| RPS-Net  | 86.97\(\pm 0.0\)    | 54.29\(\pm 0.0\)    | 51.42\(\pm 1.3\)    | 36.07\(\pm 1.6\)    | 20.69\(\pm 2.2\)    |
| EWC      | 20.65\(\pm 0.2\)    | 25.83\(\pm 0.4\)    | 10.37\(\pm 0.6\)    | 13.29\(\pm 1.8\)    | 10.78\(\pm 3.2\)    |
| IMM      | 79.15\(\pm 0.0\)    | 40.38\(\pm 0.2\)    | 33.31\(\pm 1\)      | 22.68\(\pm 1.7\)    | 6.14\(\pm 3.1\)     |
| TFW      | 88.07\(\pm 0.3\)    | 62.39\(\pm 0.0\)    | 55.19\(\pm 0.2\)    | 38.43\(\pm 1.4\)    | 25.73\(\pm 1.6\)    |
| CLAE     | 90.47\(\pm 0.0\)    | 62.83\(\pm 0.0\)    | 58.42\(\pm 0.2\)    | 39.67\(\pm 0.7\)    | 27.68\(\pm 0.9\)    |
| Ours     | 90.38\(\pm 0.0\)    | 69.1\(\pm 0.0\)     | 63.48\(\pm 0.1\)    | 44.79\(\pm 0.7\)    | 49.37\(\pm 0.8\)    |

### 4. Conclusion

This paper presents an extended CapsNet which 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 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 then the number of tasks increases. For challenging cases such as object deformations and occlusions, the enhancement of dynamic model needs further study.
Acknowledgments

Acknowledgements I would like to express my gratitude to all those who helped me during the writing of this thesis. Acknowledgements This research was funded by National Natural Science Foundation of China (61563025, 61562053, 61762056), Yunnan Science and Technology Department of Science and Technology Project (2016FB109, 2017FB094) and Scientific Research Foundation of Yunnan Education Department (2017ZZX149).

References:
[1] Robins, A. Catastrophic forgetting, rehearsal and pseudorehearsal. Connection Science, 7:123–146, 1995.
[2] Kemker R, McClure M, Abitino A, et al. Measuring catastrophic forgetting in neural networks[C]//Proceedings of the AAAI Conference on Artificial Intelligence, 2018, 32(1).
[3] Ans B, Rousset S. Neural networks with a self-refreshing memory: knowledge transfer in sequential learning tasks without catastrophic forgetting[J]. Connection science, 2000, 12(1): 1-19.
[4] Lee S W, Lee C Y, Kwak D H, et al. Dual-Memory Deep Learning Architectures for Lifelong Learning of Everyday Human Behaviors[C]//IJCAI. 2016: 1669-1675.
[5] Rusu A A, Rabinowitz N C, Desjardins G, et al. Progressive neural networks[J]. arXiv preprint arXiv:1606.04671, 2016.
[6] Wu Y, Chen Y, Wang L, et al. Large scale incremental learning[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 374-382.
[7] Goodfellow I J, Mirza M, Xiao D, et al. An empirical investigation of catastrophic forgetting in gradient-based neural networks[J]. arXiv preprint arXiv:1312.6211, 2013.
[8] Srivastava N, Salakhutdinov R. Discriminative Transfer Learning with Tree-based Priors[C]//NIPS. 2013, 3(4): 8.
[9] Fuangkhon P, Tanprasert T. An incremental learning algorithm for supervised neural network with contour preserving classification[C]//2009 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology. IEEE, 2009, 2: 740-743.
[10] Ren J, Wu L, Yang J M, et al. Machine Learning Scan and Application in SUSY[J]. arXiv preprint arXiv:1708.06615.
[11] Peng J, Hao J, Li Z, et al. Overcoming Catastrophic Forgetting by Soft Parameter Pruning[J]. arXiv preprint arXiv:1812.01640, 2018.
[12] Mallya A, Lazebnik S. Packnet: Adding multiple tasks to a single network by iterative pruning[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 7765-7773.
[13] Liu Z, Mu H, Zhang X, et al. Metapruning: Meta learning for automatic neural network channel pruning[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 3296-3305.
[14] Liu X, Masana M, Herranz L, et al. Rotate your networks: Better weight consolidation and less catastrophic forgetting[C]//2018 24th International Conference on Pattern Recognition (ICPR). IEEE, 2018: 2262-2268.
[15] Early J. Reducing catastrophic forgetting when evolving neural networks[J]. arXiv preprint arXiv:1904.03178, 2019.
[16] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. Proc. of the National Academy of Sciences of the USA, 114(13):3521–3526, 2017.
[17] Chaudhry A, Ranzato M A, Rohrbach M, et al. Efficient lifelong learning with a-gem[J]. arXiv preprint arXiv:1812.00420, 2018.
[18] Li Z, Hoiem D. Learning without forgetting[J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 40(12): 2935-2947.
[19] Lee S W, Kim J H, Jun J, et al. Overcoming catastrophic forgetting by incremental moment matching[J]. arXiv preprint arXiv:1703.08475, 2017.
[20] Adel T, Zhao H, Turner R E. Continual learning with adaptive weights (claw)[J]. arXiv preprint arXiv:1911.09514, 2019.
[21] Coop R, Mishtal A, Arel I. Ensemble learning in fixed expansion layer networks for mitigating catastrophic forgetting[J]. IEEE transactions on neural networks and learning systems, 2013, 24(10): 1623-1634.
[22] Masana M, Tuytelaars T, van de Weijer J. Ternary feature masks: continual learning without any forgetting[J]. arXiv preprint arXiv:2001.08714, 2020.
[23] Sabour S, Frosst N, Hinton G E. Dynamic routing between capsules[J]. arXiv preprint arXiv:1710.09829, 2017.
[24] Serra J, Suris D, Miron M, et al. Overcoming catastrophic forgetting with hard attention to the task[C]//International Conference on Machine Learning. PMLR, 2018: 4548-4557.
[25] Jung H, Ju J, Jung M, et al. Less-forgetting learning in deep neural networks[J]. arXiv preprint arXiv:1607.00122, 2016.
[26] Fernando C, Banarse D, Blundell C, et al. Pathnet: Evolution channels gradient descent in super neural networks[J]. arXiv preprint arXiv:1701.08734, 2017.
[27] Rebuffi S A, Kolesnikov A, Sperl G, et al. icarl: Incremental classifier and representation learning[C]//Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2017: 2001-2010.
[28] Rajasegaran J, Hayat M, Khan S H, et al. Random path selection for continual learning[J]. Advances in Neural Information Processing Systems, 2019, 32: 12669-12679.