Dynamic Memory-based Continual Learning with Generating and Screening

Siying Tao[1], Jinyang Huang[1], Xiang Zhang[2], Xiao Sun[1,3], and Yu Gu[4] (✉)

[1] School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, China
siyingtao@mail.hfut.edu.cn, {hjy, sunx}@hfut.edu.cn
[2] School of Cybers Science and Technology, University of Science and Technology of China, Hefei, China
zhangxiang@ieee.org
[3] Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, Hefei, China
[4] I$^+$ Lab, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China
yugu.bruce@ieee.org

Abstract. Deep neural networks suffer from catastrophic forgetting when continually learning new tasks. Although simply replaying all previous data alleviates the problem, it requires large memory and even worse, often infeasible in real-world applications where access to past data is limited. Therefore, We propose a two-stage framework that dynamically reproduces data features of previous tasks to reduce catastrophic forgetting. Specifically, at each task step, we use a new memory module to learn the data distribution of the new task and reproduce pseudo-data from previous memory modules to learn together. This enables us to integrate new visual concepts with retaining learned knowledge to achieve a better stability-malleability balance. We introduce an N-step model fusion strategy to accelerate the memorization process of the memory module and a screening strategy to control the quantity and quality of generated data, reducing distribution differences. We experimented on CIFAR-100, MNIST, and SVHN datasets to demonstrate the effectiveness of our method.

Keywords: Continual Learning · Generative replay · Deep learning

1 Introduction

While the majority of deep learning literature mainly focuses on learning models using fixed datasets, real-world data is constantly evolving and generating new classes or domains. This often results in the catastrophic forgetting problem when models are fine-tuned directly with new data, and previous data is not accessible due to concerns such as privacy or device storage limitations. Catastrophic forgetting can result in a loss of previous data distributions and seriously degrade the performance of models to process previous data classes. Continual

learning aims to keep the model learning new tasks without forgetting the knowledge of the old ones, solving the stability-plasticity dilemma [1], which refers to the challenge of achieving a balance between model stability and plasticity in continual learning. In detail, excessive plasticity can lead to a dramatic decline in the model's performance for old tasks, while excessive stability can increase the difficulty for the model to learn new tasks.

There has been a significant effort to tackle catastrophic forgetting in the field of machine learning. Typical methods [7,8], such as knowledge distillation or fixing important parameters in the model, aim to retain previous knowledge by reusing well-trained network components. However, when a large number of new tasks are added to the model, these methods can have limitations. Other approaches [7,9] attempt to store samples from old tasks to inform new ones, but the imbalance between the number of samples and new tasks can lead to a classifier that is biased towards old tasks due to storage capacity limitations. To address this, some methods [2,3] dynamically expand the classification network to accommodate new tasks. However, as the number of tasks increases, the classification network can become increasingly large in parameters and complex in structure, which requires careful pruning as post-processing.

To address the above weaknesses, we propose a two-stage learning framework that decouples the process of remembering old classes and the learning process of new classes to achieve a better stability-malleability trade-off in continuous learning. We nicknamed our framework, DMCL, for Dynamic Memory-based Continual Learning with generating and screening. Within this framework, we design the memory module consisting of a diffusion model [4] and a screening network in pairs that can learn old classes of data distributions and generate pseudo-data with the same distribution. Our main idea is to dynamically generate old class samples by memory module and to learn them together with the new samples, thus allowing both adequate retention of existing knowledge and sufficient flexibility to learn new knowledge. In addition, our framework can be broken down into multiple discrete modules stored on the hard disk and loaded into different memory modules in turn during use. Therefore, the machine memory consumed is very limited, even when there are many tasks.

To achieve this, we propose the N-step model fusion strategy and the screening strategy for the memory module. Specifically, when memorizing the old class data, our N-step model fusion strategy fuses Gaussian transfer kernels from different time steps of the diffusion model in the memory module to speed up convergence. When generating the old class data samples, our screening strategy controls the quantity and quality of generated samples, avoiding class imbalances and reducing distribution differences with actual data.

We validated DMCL on image classification tasks with three commonly used benchmark tests, including the CIFAR-100 [16], MNIST [17], and SVHN [18] datasets. The empirical results and ablation studies show that our method outperforms the existing state-of-the-art methods. In fact, DMCL can be applied to many tasks, not only image classification tasks, as long as the memory module can reliably reproduce the old data distribution.

2 Related Work

2.1 Continual learning

Continual learning is to address the catastrophic forgetting of machine learning on old classes, and common settings include Task-IL, Domain-IL, and Class-IL. In computer vision, most of strategies applied on large-scale datasets use rehearsal learning: a limited amount of the data of old task is kept during training [1]. These data can be either raw pixels or compressed vectors. Others [9,7,11] acquire knowledge of old classes by knowledge distillation and apply some constraints in training new data. These constraints [8] can be directly applied to the model weights, sample intermediate features, and prediction probabilities of the classifier. In addition, there are approaches [5,14] that generate instead of storing old samples through generative models (e.g.,GAN) to alleviate the difficulties of limited storage space for old samples.

2.2 Diffusion model

The diffusion model is a generative model surpassing GAN as the current state-of-the-art proposal in the field of image generation [12]. The model consists of two processes. The first process gradually adds noise to perturb or destroy the data distribution by the forward process, and then the reverse process learns to remove the noise and restore the structure of the original data distribution, resulting in a highly flexible and easily handled generative model. Recently diffusion models have been applied to many works [4] (e.g., image super-resolution, image translation, image segmentation, text-to-image generation, etc.) and showed great potential.

3 Our Approach

In this section, we demonstrate the processing flow of our framework for image classification tasks and the details of the n-step model fusion strategy and the screening strategy.

3.1 Method Overview

We expect DMCL to be capable of classifying an increasing number of classes. We define the image classification tasks to be solved as a sequence of N tasks, represented as $Task = (task_1, task_2, ..., task_n)$. Each task includes different categories of pictures (x_i, y_i) with completely different distributions D. The classification labels y may be the same (for domain incremental learning) or completely different (for class incremental learning). DMCL will learn the $task_n$ data sequentially while retaining the knowledge of previously learned $task_{1,2,...,n-1}$. The main challenge is that DMCL can only temporarily access the data from the current $task_i$, but it must be able to efficiently classify test data from all previous classes $C(task_1, ..., task_i)$.

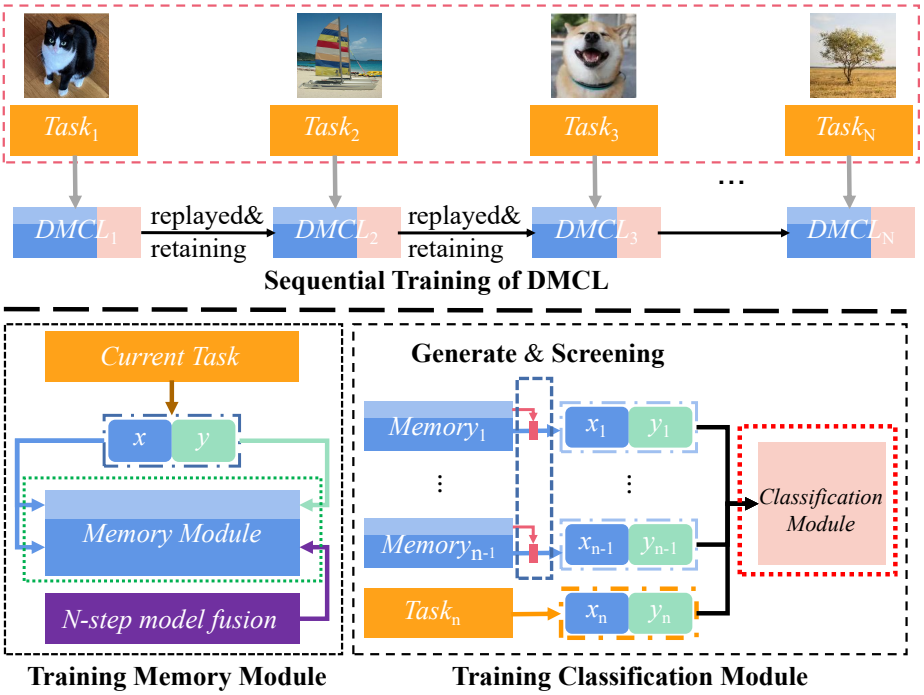

Fig. 1. The process of sequential training of DMCL. In the top subplot, when a new $task_n$ arrives, DMCL first learns the data distribution features of $task_n$ with a memory model, and then replays the old sample data$(x_{1\ldots n-1}, y_{1\ldots n-1})$ from the previous memory modules to retrain the classification module together with the $task_n$ data(x_n, y_n). The left subplot is the current $task_n$ data for the training process of diffusion model and screening network in the memory module, and the N-step model fusion strategy is applied to training diffusion model. The right subplot is a retraining process of the classification module. All memory modules generate samples, select and label pseudo data $(x_{1\ldots n-1}, y_{1\ldots n-1})$ by screening strategy. Then, the pseudo data$(x_{1\ldots n-1}, y_{1\ldots n-1})$ and $task_n$ data (x_n, y_n) work together to retrain the classification module.

To address the forgetfulness of previous tasks, our DMCL learns the sample data for each task by the memory module and re-trains the pseudo-sample data jointly in the new task. The sequence training process of DMCL is at the top of Figure 1. When the $task_n$ arrives, the DMCL first trains the memory module with the current task sample data, and then trains the classification module with the current task sample data and replayed sample data from all the previous memory modules. Formally, the loss function of the i-th classification model is given as

$$L_{train}(\theta_i) = r_i E_{(x,y)\sim C_i}[L(C(x_i, \theta_i), y] + \sum_{t=0}^{i-1} r_t E_{x'\sim M_t}[L(C(x', \theta_i), SN_t(x')] \quad (1)$$

where θ_i are network parameters of the i-th classification module, M_t is the t-th memory module, SN_t is the t-th screening network in the t-th memory module, and r_t is a ratio of mixing data.

3.2 Classification Module

The classification module used in our study is compatible with any classification network, and we chose the Resnet [15] network for our experiments in this paper. When a new task is presented, we first update the number of classification heads in the last layer of the classification network to match the total number of classes, given by $C(task_1, ..., task_{n-1})$. Next, we combine the pictures and labels generated by the memory module, i.e.,$x(Memory_1, ..., Memory_{n-1})$ and $y(Memory_1, ..., Memory_{n-1})$, with the samples of the current $task_n$, in a certain ration depending on the task's importance. As shown in the training classification module of Fig. 1, the diffusion models in the memory module generate samples, and then the screening network then selects and labels the categories through the screening strategy with output pseudo-data $(x_{1...n-1}, y_{1...n-1})$. Finally, the pseudo-data and current task data work together to retain the classification module.

3.3 Memory Module

The memory module is the core part of DMCL and consists of two parts: the diffusion model and the screening network. The diffusion model in our work uses the denoising diffusion probabilistic model [12] (DDPM), which can generate samples with the same distribution as the original data in a limited time by variational inference. The forward chain of DDPM perturbs the data distribution by gradually adding Gaussian noise with a pre-designed schedule until the data distribution converges to a Gaussian distribution. The reverse chain of DDPM starts with the given prior and uses a parameterized Gaussian transition kernel, learning to gradually restore the undisturbed data structure. The noise-adding process q for the forward chain and the noise-removal process p for the reverse chain are defined formally as:

$$q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{2}$$

$$p_\theta(x_t - 1|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \sum_\theta(x_t, t) \tag{3}$$

where the x_t is the sampling noise to image at times step t, and the β_t is a fixed-variance strategy. The $q(x_t|x_0)$ is the process of forward chain, deriving the noise distribution of x_t from x_{t-1}, and I is the constant value. The $p_\theta(x_t - 1|x_t)$ is the process of reverse chain, removing the noise distribution from x_t to $xt - 1$, and the $\mu_\theta(x_t, t)$ is a noise reduction network for predicting the noise at t step.

The screening network in the memory module is a small classification network. Compared with the classification module, the screening network is trained with only one task data, so it has better performance than the classification module in a single task. Like the classification module, the screening network is

compatible with any classification network. The role of the screening network is to remove samples with poor generation quality and match the corresponding pseudo-labels.

On the left of Fig. 1, the training process of the memory module is shown. The diffusion model and the screening network in the memory module are trained simultaneously and independently. The image x and category label y from the current task are used to train the screening network. And for the training of the diffusion model only use the image x of the current task. When training the diffusion model, use the N-step model fusion strategy to speed up its convergence.

The generation and screening processes of the memory module are shown on the right side of Fig. 1. The diffusion model in different memory modules generates various classes of pseudo-sample data. And the corresponding screening network in the memory module selects and marks the generated samples by the screening strategy. In this way, The resulting image $x_{1,\ldots,n-1}$ is closer to the data distribution of the original task and reduces the ambiguity between the different categories $y_{1,\ldots,n-1}$.

3.4 Screening Strategy

The screening strategy is acting when generating samples. Firstly, set a reasonable confidence threshold. After that, the Screening network is used to classify the generated samples and give the confidence level. Samples below the confidence threshold are directly dropped, and the remainders go to the joint training of the next task. Each memory module selects the same number of samples by screening strategy as the number of the new task to decrease the category imbalance problem.

The confidence threshold is finalized by repeated pre-experiments. Using different confidence threshold values, samples are generated by the same memory module and selected using the same screening strategy for training of the classification module. The appropriate confidence threshold value is selected according to the variation in the accuracy of the classification module. The confidence threshold in this paper is 0.98.

Algorithm 1 describes the entire flow of the screening strategy. In the follow-up ablation experiments, we found that the strategy was helpful to improve the classification accuracy, especially when the diffusion model was not fully converged.

3.5 N-step Model Fusion Strategy

The N-step model fusion strategy is applied to the diffusion model training. The core of the diffusion model is the training of a noise-reducing U-Net [12], using a Monte Carlo algorithm. At each training, a time step t is randomly selected from the total time step T. Then the loss is calculated by forward propagation and gradient descent for the U-Net in time step t, with updated parameters of the diffusion model (DM_t). Based on the **Law of Large Numbers** theorem, the U-Net will eventually converge after many iterations.

Algorithm 1 The screening strategy, Pytorch-like

```
#Input:DM(diffusion model),SN( screening network),
#N(number of generation samples),threshold
#Output:X(image generated),Y(label of image)
X = list()       #store images
Y = list()       #store labels
while len(X)<N:
    sample = DM()       #generate sample
    #calculate the label and degree of confidence
    Label, CF = SN(sample)
    if CF >=threshold:
      X.append(sample)       #screening
      Y.append(Label)
return   X,Y
```

Assuming that each gradient descent decreases the loss, the KL scatter between the samples generated by DM_t and the samples from the original task is $KL(D|DM_t)$. By **Jensen's inequality**, it is known that:

$$E(KL(D||DM_t)) \geq KL(D||E(DM_t)) \tag{4}$$

where E is the expectation. If $E(DM_t)$ is used instead of DM_t for the next iteration, a lower KL scatter is obtained. However, the performance of the overhead of computing $E(DM_t)$ is high in practice. As indicated by Algorithm 2, we calculate the mean value of the n-step model fusion in terms to approximate $E(DM_t)$ as follows

$$E(DM_t) \approx \frac{1}{n}\sum_{i=1}^{n} DM_t^i \tag{5}$$

Algorithm 2 N-step model fusion strategy, Pytorch-like

```
#Input:CM(the updated model),now(number of iterations)
#Output:model(model for the next training)
old_models       #temporary strorage models
previous_model       #the model before gradient descent
old_models.append(copy(CM))       #save updated model
#check whether the number of steps has reached N
if check_step(now):
    previous_model=mean(old_models)
return previous_model
```

4 Experiments

4.1 Experiment Setup and Implementation Details

Datasets We experimented with class incremental experiments on the CIFAR-100 [16] dataset, and domain incremental experiments on the MNIST [17] and SVHN [18] datasets. Due to the fact that the MNIST dataset is a 28*28 grayscale image, we transform it into a 32*32 pixel and 3-channel image (keeping with SVHN).

Benchmarks In CIFAR-100, we compare performances on 10 steps (10 new classes per step), 20 steps (5 new classes per step), and 50 steps (2 new classes per step) and report the top-1 accuracy(%) for each step. In both MNIST and SVHN, we conducted domain increment experiments from MNIST to SVHN and from SVHN to MNIST, and report the change in top-1 accuracy(%).

The performance of the classification module directly affects the accuracy of class incremental experiments and domain incremental experiments. Our experimental framework is based on the open-source PYCIL [6] secondary development. We experimented the classification experiments with various specifications of Resnet [15] at CIFAR-100 dataset, and the performance is shown(accuracy of the Resnet18 is 70.55%, Resnet32 is 72.34%, Resnet34 is 74.12%, Resnet50 is 74.56% and Resnet101 is 74.28%). To facilitate comparison with DyTox [2] (Transf. Joint accuracy 76.12%), we chose Resnet34 (Joint accuracy 74.12%) as the benchmark network for the classification module.

We perform repeated pre-experiments on task data of CIFAR-100 to determine the confidence threshold of the screening strategy. As the results are shown in Table 1, We find with a rising threshold, the accuracy rate will start to rise, but will remain the same after reaching a certain value. The experiment was chosen with a threshold value of 0.98.

Table 1. The accuracy of different confidence thresholds on test data.

threshold	0.9	0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99
accuracy	0.801	0.826	0.833	0.847	0.852	0.861	0.872	0.878	0.883	0.882

4.2 Class Incremental Learning

Table 2 shows the results for all approaches on CIFAR-100. The more steps there are, the larger the forgetting is and thus the lower the performances are. These settings are shown in Fig. 2. In the setting, DMCL is close to DER [3] for much fewer parameters(up to 25x less). Critically, DMCL is significantly above other baselines and has better-forgetting resistance for the case of a consistently huge number of tasks: e.g. DMCL is up to +30% in "Last" accuracy in the 20 steps setup.

Table 2. Results on CIFAR-100 averaged over three different class orders. Baseline results come from [2]. The * symbol means that [3] needed setting-sensitive hyperparameters and its reported parameters count was an average over all steps.

	10 steps			20 steps			50 steps		
Methods	**#P**	**Avg**	**Last**	**#P**	**Avg**	**Last**	**#P**	**Avg**	**Last**
Res. Joint	22.45	-	74.12	22.45	-	74.12	22.45	-	74.12
iCaRL[7]	11.22	65.27	50.74	11.22	61.2	43.75	11.22	56.08	35.62
UCIR[8]	11.22	58.66	43.39	11.22	58.17	40.63	11.22	56.86	37.09
BiC[9]	11.22	68.8	53.54	11.22	66.48	47.02	11.22	62.09	41.04
WA[10]	11.22	69.46	53.78	11.22	67.33	47.31	11.22	64.32	42.14
PODNet[11]	11.22	58.03	41.05	11.22	53.97	35.02	11.22	51.19	35.99
RPSNet[13]	56.5	68.6	57.05	-	-	-	-	-	-
DER*[3]	112.27	74.64	64.35	224.55	73.98	62.55	561.39	72.05	59.76
DyTox[2]	10.73	73.66	60.67	10.74	72.27	56.32	10.77	70.2	52.34
DMCL	22.45	71.97	**64.94**	22.45	72.69	**65.08**	22.45	71.03	**60.01**

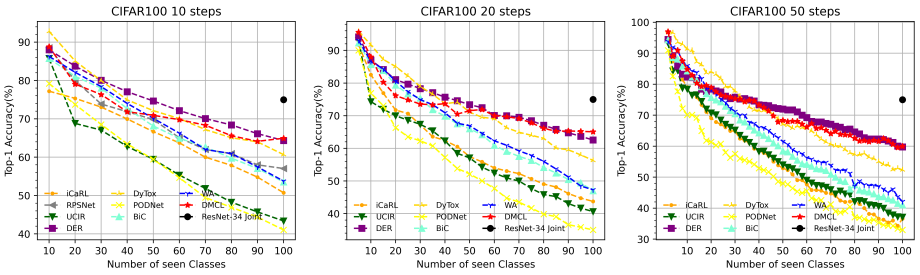

Fig. 2. Performance evolution on CIFAR100. The top-1 accuracy (%) is reported after learning each task. Left is evaluated with 10 steps, middle with 20 steps, and right with 50 steps.

4.3 Domain Incremental Learning

Table 3 shows the results for DMCL and Joint training on MNIST and SVHN. Whether the domain is changed from MNIST to SVHN or vice versa, DMCL achieves the same level of performance as joint training, with 97% average accuracy.

Table 3. The top-1 accuracy results of domain increment experiments on MNIST and SVHN.

	MNIST	SVHN	MNIST to SVHN	SVHN to MNIST
DMCL	99.6	96.56	97.24	96.65
Joint	-	-	96.97	

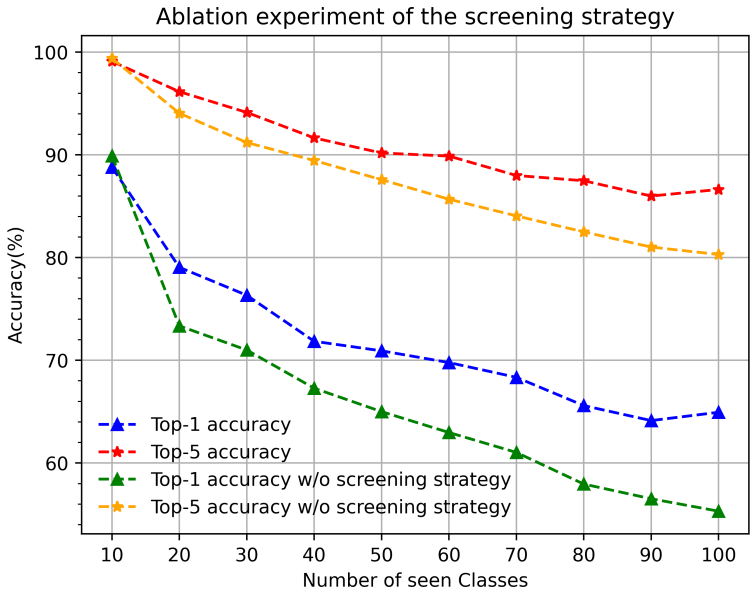

Fig. 3. Performance evaluation of the ablation experiment of screening strategy at CIFAR-100 with 10 steps.

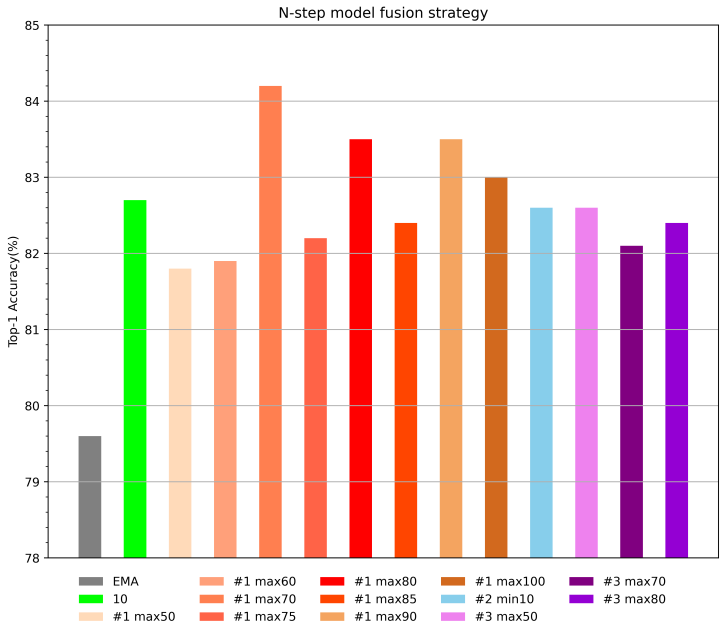

Fig. 4. Performance evaluation of the ablation experiment of the N-step model fusion strategy at CIFAR-100 with the same task. where max or min denotes the maximum or minimum number of fusion steps, and EMA denotes the original exponential moving average of the diffusion model.

4.4 Ablation Experiments

Screening strategy We experimented on CIFAR-100 with a setting of 10 steps. Using the same diffusion model and screening network, the threshold value of the screening strategy is set to 0.98. In Fig. 3, the screening strategy can effectively reduce the difference in the distribution between the generated samples and the original task. In each task, the screening strategy contributed to the quality of sample replication, with a maximum improvement of 9.6% on top1 accuracy and 6.3% on top5 accuracy.

N-step model fusion strategy We conducted experiments on CIFAR100, with the same network structure of diffusion model, the same screening strategy, and the same task data, to compare the memory effect with different fusion steps. Fig. 4 displays the performance comparison of different fusion step strategies, and the baseline strategy is the exponential moving average (EMA) of the diffusion model. We compared the 10 steps fusion and linear adjustment step fusion methods. The linear function is as follows:

$$\#1 : N = \left\lfloor 10 + \frac{steps}{1000} \right\rfloor \quad \#2 : N = \left\lfloor 100 - \frac{steps}{1000} \right\rfloor \quad \#3 : N = \left\lfloor 10 + \frac{steps}{2000} \right\rfloor$$

From Fig. 4, it can be seen that the N-step model fusion strategy has a noticeable performance improvement over the EMA strategy(up to +4.6% in top 1 accuracy).

5 Conclusion

In this paper, we proposed a two-stage framework for reducing catastrophic forgetting in deep neural networks when learning new tasks. At each step, we use a new memory module to memorize the data features of the new task and dynamically generate the pseudo-data of the previous tasks from the previous memory modules. We introduced the N-step model fusion strategy to accelerate the memorization process of the memory module and the screening strategy to select the generated samples and control the quantity and quality of the generated data. Experimental results on CIFAR-100, MNIST, and SVHN datasets show that our method outperforms the state-of-the-art methods in terms of accuracy, which achieves a better stability-malleability balance. Our proposed method provides a practical solution to address catastrophic forgetting in continual Learning, and our approach is easily modularly extended and optimized to suit different demands.

References

1. Grossberg, S. (2013). Adaptive Resonance Theory: How a brain learns to consciously attend, learn, and recognize a changing world. Neural networks, 37, 1-47.https://doi.org/10.1016/j.neunet.2012.09.017.

2. Douillard, A., Ramé, A., Couairon, G., Cord, M. (2022). Dytox: Transformers for continual learning with dynamic token expansion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 9285-9295).
3. Yan, S., Xie, J., He, X. (2021). Der: Dynamically expandable representation for class incremental learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 3014-3023).
4. Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., ... Yang, M. H. (2022). Diffusion models: A comprehensive survey of methods and applications. arXiv preprint arXiv:2209.00796.https://doi.org/10.48550/arXiv.2209.00796
5. Shin, H., Lee, J. K., Kim, J., Kim, J. (2017). Continual learning with deep generative replay. Advances in neural information processing systems, 30.
6. Zhou, D. W., Wang, F. Y., Ye, H. J., Zhan, D. C. (2021). Pycil: A python toolbox for class-incremental learning. arXiv preprint arXiv:2112.12533.https://doi.org/10.1007/s11432-022-3600-y
7. Rebuffi, S. A., Kolesnikov, A., Sperl, G., Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 2001-2010).
8. Hou, S., Pan, X., Loy, C. C., Wang, Z., Lin, D. (2019). Learning a unified classifier incrementally via rebalancing. In Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (pp. 831-839).
9. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., & Fu, Y. (2019). Large scale incremental learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 374-382).
10. Zhao, B., Xiao, X., Gan, G., Zhang, B., Xia, S. T. (2020). Maintaining discrimination and fairness in class incremental learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 13208-13217).
11. Douillard, A., Cord, M., Ollion, C., Robert, T., Valle, E. (2020). PODNet: Pooled Outputs Distillation for Small-Tasks Incremental Learning. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, JM. (eds) Computer Vision – ECCV 2020. ECCV 2020. Lecture Notes in Computer Science(), vol 12365. Springer, Cham. https://doi.org/10.1007/978-3-030-58565-5_6
12. Ho, J., Jain, A., Abbeel, P. (2020). Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems, 33, 6840-6851.
13. Rajasegaran, J., Hayat, M., Khan, S., Khan, F. S., Shao, L., Yang, M. H. (2019). An adaptive random path selection approach for incremental learning. arXiv preprint arXiv:1906.01120.
14. T. Lesort, H. Caselles-Dupré, M. Garcia-Ortiz, A. Stoian and D. Filliat, "Generative Models from the perspective of Continual Learning," 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 2019, pp. 1-8, https://doi.org/10.1109/IJCNN.2019.8851986.
15. He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
16. Krizhevsky, A., Hinton, G. (2009). Learning multiple layers of features from tiny images.
17. Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, https://doi.org/10.1109/5.726791.
18. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.