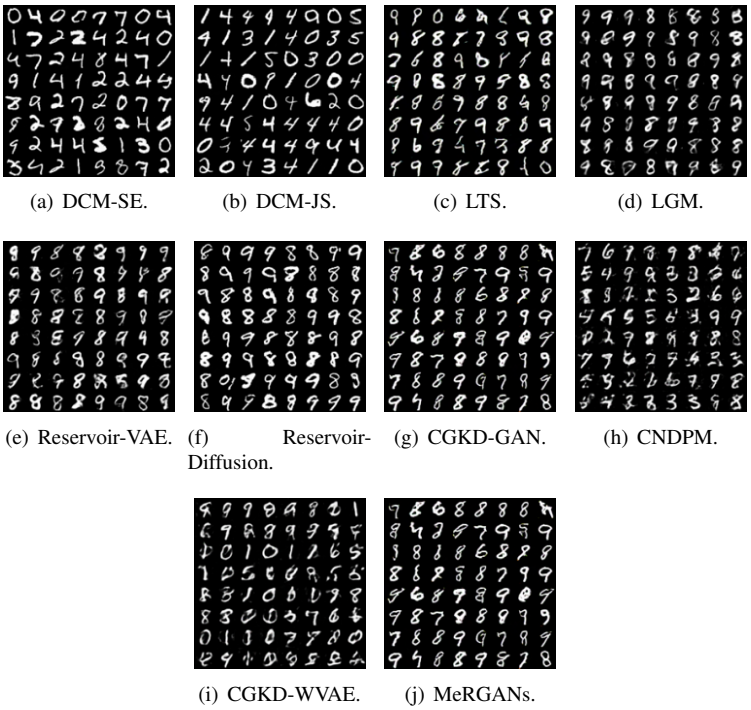
Appendix for Online Task-Free Continual Generative and Discriminative Learning via Dynamic Cluster Memory

March 29, 2024

Contents

A Additional information for the related work



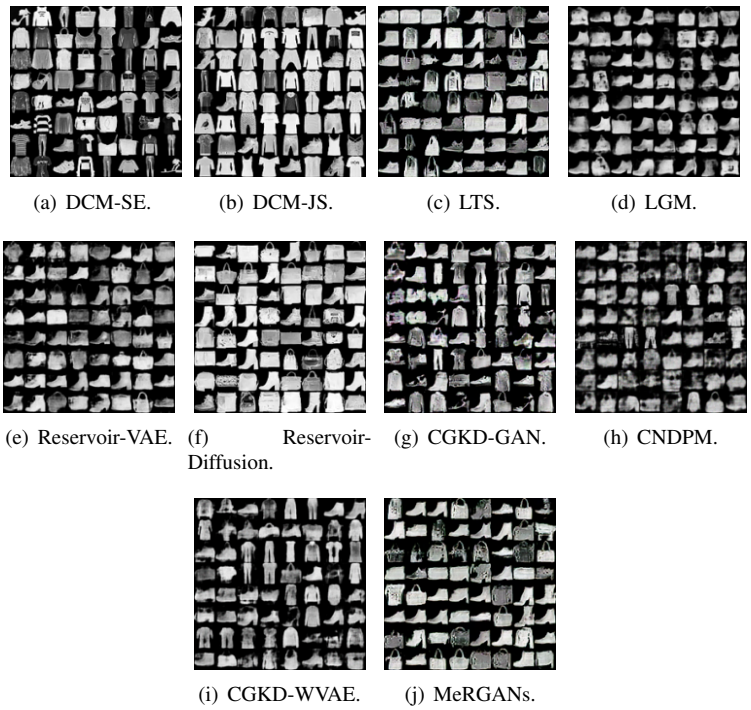
(a) DCM-SE. (b) DCM-JS. (c) LTS. (d) LGM.

(e) Reservoir-VAE. (f) Reservoir-Diffusion. (g) CGKD-GAN. (h) CNDPM.

(i) CGKD-WVAE. (j) MeRGANs.

Figure 1: The generation results of various models on Split MNIST.

Denoising Diffusion Probabilistic Models (DDPMs) (18) is a recent popular generative model, which has achieved excellent performances in image synthesis applications (7; 13; 19). Different from other kinds of generative technologies such as GANs and VAEs, which have a fast generation process in which an image can be directly yield from a noise vector, the DDPM generative processing involves a considerable number of optimization iterations, resulting in a heavy computational requirement. This weakness inspires many attempts to develop several solutions to accelerate the generation process of the DDPM. These works usually focus on performing the diffusion process in a low-dimensional latent space (16) or shortening the reverse diffusion steps (3; 10; 17; 19; 21; 24). The proposed Dynamic Cluster Memory (DCM) can also be used in these improved DDPMs to enable them for task-free continual learning, which will be investigated in our future work.



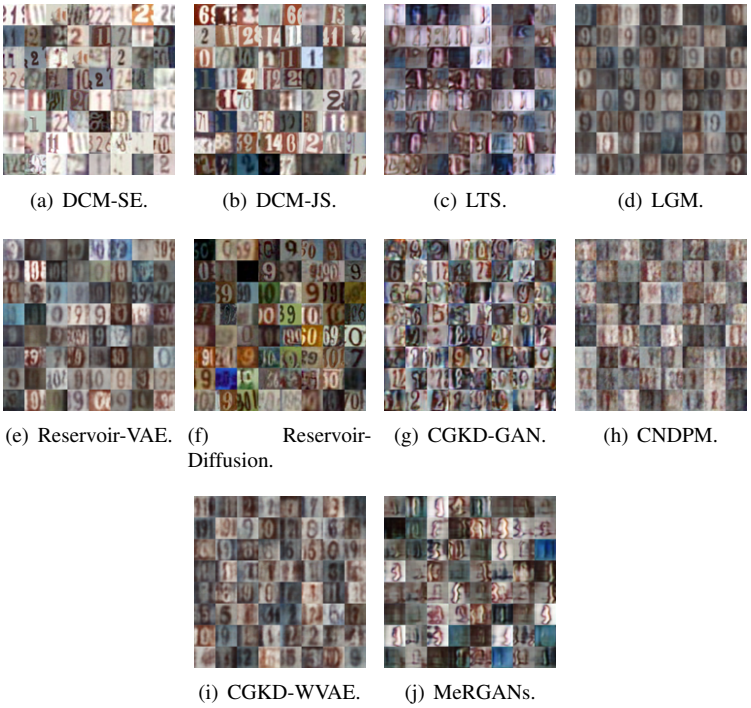
(a) DCM-SE. (b) DCM-JS. (c) LTS. (d) LGM.

(e) Reservoir-VAE. (f) Reservoir-Diffusion. (g) CGKD-GAN. (h) CNDPM.

(i) CGKD-WVAE. (j) MeRGANs.

Figure 2: The generation results of various models on Split Fashion.

Although the DDPM models have achieved remarkable results in various applications, their performance is relying on learning from large-scale datasets. When training a DDPM model in continual learning, it will quickly forget how to generate past images. One attempt to apply the DDPM model in continual learning was proposed in (4), which employs the DDPM model as a generative replay network. A similar idea was proposed in (8), which employs the diffusion model for distillation and replay. However, these approaches require huge computing costs for the generative replay process. In addition, these approaches are designed for task-aware continual learning, which cannot be used in a more realistic setting such as the Online Task-Free Continual Learning (OTFCL) (1)). In contrast, the proposed DCM can train a DDPM model in an efficient way under OTFCL without requiring the sampling process of the DDPM and supervised signals.



(a) DCM-SE. (b) DCM-JS. (c) LTS. (d) LGM.

(e) Reservoir-VAE. (f) Reservoir-Diffusion. (g) CGKD-GAN. (h) CNDPM.

(i) CGKD-WVAE. (j) MeRGANs.

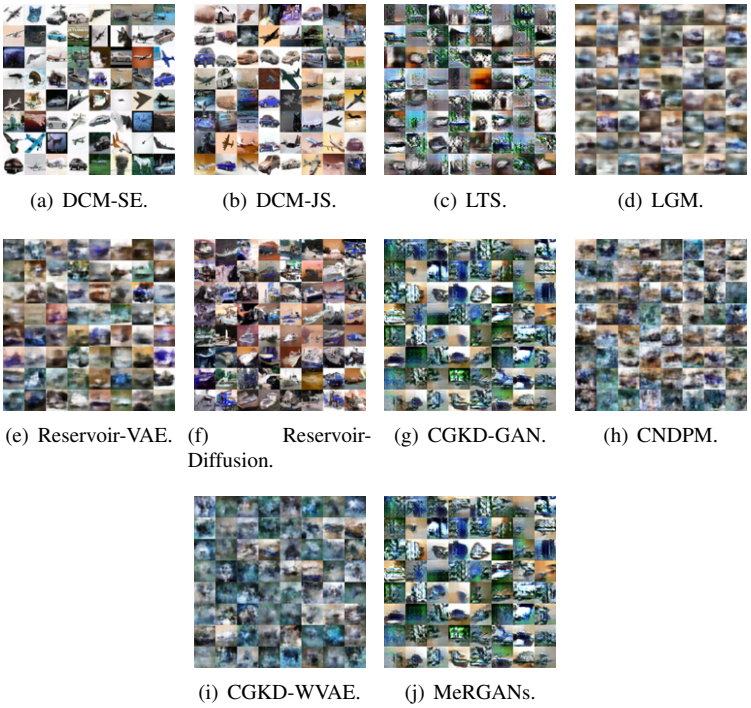
Figure 3: The generation results of various models on Split SVHN.

B Additional information for experiment settings

In this section, we provide detailed experiment information, including the model's hyperparameters, datasets and network architecture.

B.1 The hyperparameter configuration

The hyperparameter configuration and GPU hardware. In the experiments, we adopt Adam (9) optimization algorithm with a learning rate of 0.0001 for training all models. In addition, we use a Tesla V100 GPU in the experiments while using the operating system (Ubuntu 18.04.5). For the proposed DCM-JS, the threshold (λ in Eq.(10) of the paper) for Split MNIST, Split Fashion, Split SVHN and Split CIFAR10 is 40. We also adopt the threshold $\lambda = 20$ for the DCM-JS for other datasets. For the proposed DCM-JS, the threshold λ for all datasets is 2000.



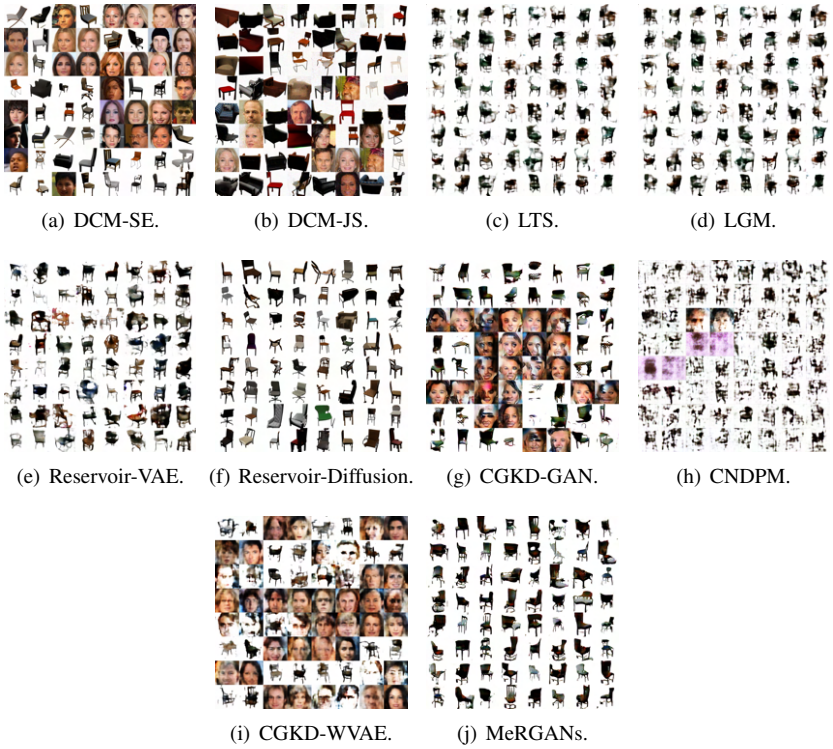
(a) DCM-SE. (b) DCM-JS. (c) LTS. (d) LGM.

(e) Reservoir-VAE. (f) Reservoir-Diffusion. (g) CGKD-GAN. (h) CNDPM.

(i) CGKD-WVAE. (j) MeRGANs.

Figure 4: The generation results of various models on Split CIFAR10.

Network architectures. In the experiments, we employ the neural network from (15), which is a U-net architecture for implementing the diffusion model. We describe the neural network information of other methods in the following. We implement the inference mode lof the VAE framework by using a convolutional network consisting of five layers with 32, 64, 128, 256 and 512 units. We also add two separate fully connected layers where each layer has 128 units on the bottom of the inference model. We implement the decoder of the VAE framework by using a convolutional network consisting of six layers with 512, 256, 128, 64, 32 and 3 units. For the GAN-based framework, we implement the generator using using a ResNet architecture (5) consisting of four residual blocks. Each block has three convolutional layers with 256, 256 and 256 units. We also implement the GAN's discriminator by using a ResNet architecture consisting of four residual blocks. The final layer of the discriminator is implemented using a fully connected layer that outputs a single value.More detailed information about neu-



(a) DCM-SE. (b) DCM-JS. (c) LTS. (d) LGM.

(e) Reservoir-VAE. (f) Reservoir-Diffusion. (g) CGKD-GAN. (h) CNDPM.

(i) CGKD-WVAE. (j) MeRGANs.

Figure 5: The generation results of various models on CelebA to Chair.

ral networks is provided in the source code from https://github.com/dtuzi123/DCM.

B.2 Evaluation and dataset setting

Evaluation : In the image generation under unsupervised learning, we employ the FID score (6) to evaluate the distance between 5,000 generated images and 5,000 real samples. In the classification task under supervised learning, we use the average classification accuracy as the performance criterion.

Datasets. In the following, we introduce the detailed information for the datasets used in the experiments.

Split MNIST. MNIST (12) is a digit dataset comprising 60,000 training and 10,000 testing samples, respectively. We divide MNIST into five parts as in (2), as Split MNIST, where each part contains samples from two different classes, .

(a) DCM-SE. (b) DCM-JS. (c) LTS. (d) LGM.

(e) Reservoir-VAE. (f) Reservoir-Diffusion. (g) CGKD-GAN. (h) CNDPM.

(i) CGKD-WVAE. (j) MeRGANs.

Figure 6: The generation results of various models on Split ImageNet.

Split Fashion. Fashion (22) is a dataset which consists of a training set of 60,000 samples and a testing set of 10,000 examples. We divide Fashion into five parts as in (2), where each part contains samples from two different classes, namely Split Fashion.

Split CIFAR10, CIFAR10 (11) is a nature image datasets that consists of 60,000 and 10,000 training and testing samples, respectively. We split CIFAR10 into five parts, as in (2), as Split CIFAR10, where each part contains samples from two different classes.

Split SVHN. SVHN (14) is a real-world image dataset that consists of 73,257 digit training and 26,032 digit testing samples, respectively. Let us to split SVHN into five parts according to (2), as Split SVHN, where each part contains samples from two different classes.

(a) DCM-SE. (b) DCM-JS. (c) LTS. (d) LGM.

(e) Reservoir-VAE. (f) Reservoir-Diffusion. (g) CGKD-GAN. (h) CNDPM.

(i) CGKD-WVAE. (j) MeRGANs.

Figure 7: The generation results of various models on CelebA to CACD.

C Additional experiment results and ablation studies

In this section, we provide more ablation results to analyze the proposed DGM.

C.1 Visual results

In this section, we provide additional visual results. The generation results of various models on Split MNIST, Split Fashion, Split SVHN and Split CIFAR10 are shown in Fig. 1, Fig. 2, Fig. 3 and Fig. 4, respectively. The visual generation results of various models on complex datasets are shown in Fig. 5, Fig. 6, Fig. 7 and Fig. 8, respectively.

(a) DCM-SE. (b) DCM-JS. (c) LTS. (d) LGM.

(e) Reservoir-VAE. (f) Reservoir-Diffusion. (g) CGKD-GAN. (h) CNDPM.

(i) CGKD-WVAE. (j) MeRGANs.

Figure 8: The generation results of various models on CelebA to ImageNet.

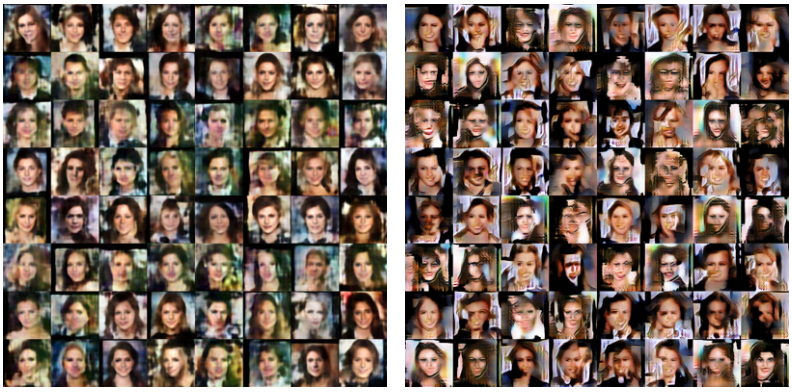
C.2 The memory size

In this section, we report the number of memorized samples for each approach. The memory size of various models on Split MNIST, Split Fashion, Split SVHN and Split CIFAR10 is reported in Table 1, which shows that the proposed DCM requires fewer memorized samples than other models. We also provide the memory size of various models trained on complex datasets in Table 2.

Datasets	DCM-SE	DCM-JS	LTS	LGM	R-VAE	R-DDPM	CGKD-GAN	CNDPM	CGKD-WAE	CGKD-VAE
Split MNIST	1408	1536	2000	2000	2000	2000	2000	2000	2000	2000
Split Fashion	1664	1664	2000	2000	2000	2000	2000	2000	2000	2000
Split SVHN	1536	1664	2000	2000	2000	2000	2000	2000	2000	2000
Split CIFAR10	1792	1792	2000	2000	2000	2000	2000	2000	2000	2000

Table 1: The number of stored samples of various models for the class-incremental learning paradigm, achieved by various models.

(a) DCM-SE. (b) DCM-JS.



(c) CGKD-WAE. (d) CGKD-GAN.

Figure 9: The generation results of various models on CACD ($128 \times 128 \times 3$).

C.3 The dynamic expansion process

In this section, we investigate the dynamic memory allocation process using different expansion threshold configurations. We train DCM-SE and DCM-JS on Split MNIST with different expansion thresholds. During the training process, we record the distribution (task) shift and the number of memory clusters changes at each training time, and we plot the results in Fig. 15 and Fig. 16, respectively. We can see that a small threshold λ encourages the proposed DCM-SE to frequently build new memory clusters. A similar result can be observed in Fig. 16 for the proposed DCM-JS. In addition, Both DCM-JS and DCM-SE can add new memory clusters to adapt to the data dis-

(a) DCM-SE. (b) DCM-JS.

(c) CGKD-WAE. (d) CGKD-GAN.

Figure 10: The generation results of various models on FFHQ ($128 \times 128 \times 3$).

tribution shift when giving an appropriate threshold. Furthermore, the best threshold configuration for DCM-JS and DCM-SE is 2200 and 50, respectively,

C.4 Dynamic expansion signals

In this section, we investigate how the proposed approach provides signals for memory expansion during the training. We train DCM-SE and DCM-JS with different λ thresholds on Split MNIST, and we record the signal evaluated by the left-hand side of Eq. (10) of the paper at each training time. We plot the expansion signals, when varying λ for DCM-SE and DCM-JS in Fig. 17 and Fig. 18, respectively. We can observe

(a) DCM-SE. (b) DCM-JS.

(c) CGKD-WAE. (d) CGKD-GAN.

Figure 11: The generation results of various models on CelebA-HQ ($128 \times 128 \times 3$).

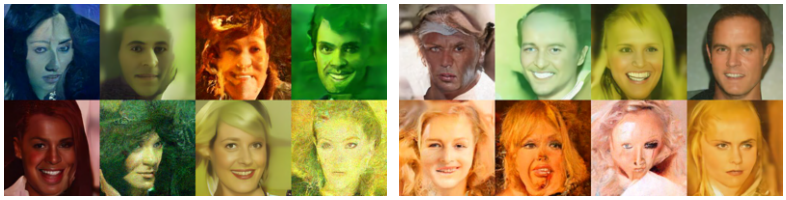
several peaks indicating expansion signals for the memory indicating when the data distribution changes at certain training times. These results show that both square loss and JS divergence can provide suitable signals for memory expansion.

C.5 The performance of the proposed model when changing λ.

In this section, we investigate the relationship between model performance and memory size when changing λ. We train DCM-SE and DCM-JS on Split MNIST with different thresholds. Once the training is finished, we provide the performance (FID) on testing samples and the number of memory clusters in Fig. 19. We can observe that

(a) DCM-SE. (b) DCM-JS.

Figure 12: The image generation results of various models on CACD with $256 \times 256 \times 3$.

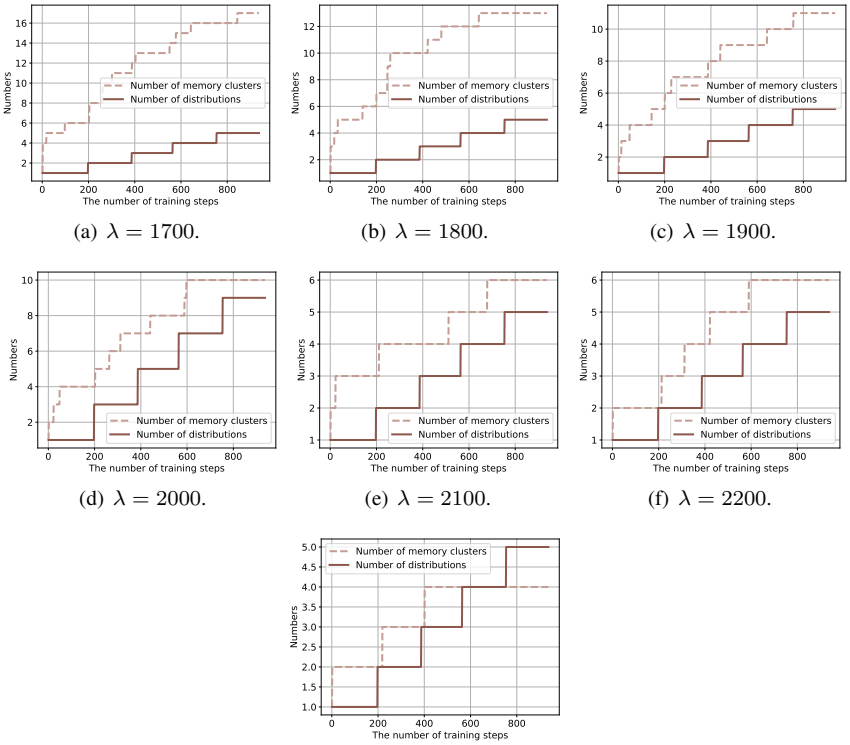


(a) DCM-SE. (b) DCM-JS.

Figure 13: The image generation results of various models on CelebAHQ with $256 \times 256 \times 3$.

(a) DCM-SE. (b) DCM-JS.

Figure 14: The image generation results of various models on FFHQ with $256 \times 256 \times 3$.



(a) $\lambda = 1700$. (b) $\lambda = 1800$. (c) $\lambda = 1900$.

(d) $\lambda = 2000$. (e) $\lambda = 2100$. (f) $\lambda = 2200$.

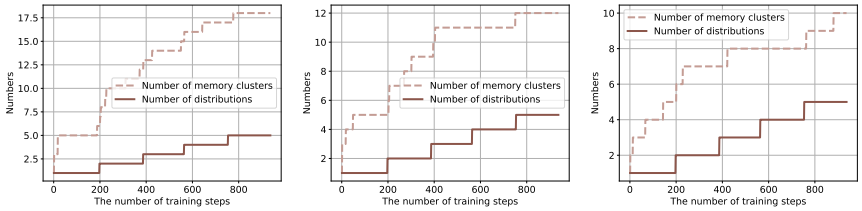
(g) $\lambda = 2300$.

Figure 15: The number of memory clusters and distributions when changing λ in DCM-SE.

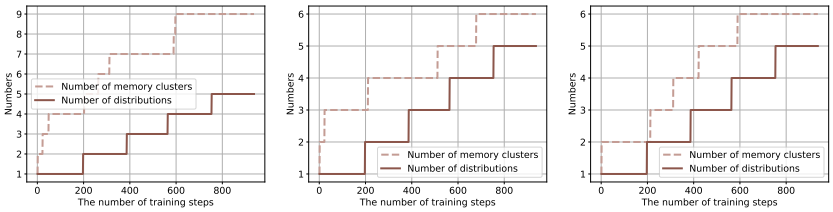
increasing the memory size does not lead to a significant improvement in the model performance. In addition, both DCM-SE and DCM-JS can still achieve good performance by using only five memory clusters.

C.6 The time required for the memorisation operations

Since our proposed memory system is a non-parametric approach, we can easily investigate how much time is required for the memory optimization during the training. We learn DCM-SE and DCM-JS on Split MNIST, Split Fashion, Split SVHN and Split CIFAR10, respectively, in which we only perform the memory optimization and do not train the model. We provide the memory optimization time (seconds) in Fig. 20. The empirical results show that the JS-based memory approach requires more operation



(a) $\lambda = 40$. (b) $\lambda = 42$. (c) $\lambda = 44$.



(d) $\lambda = 46$. (e) $\lambda = 48$. (f) $\lambda = 50$.

Figure 16: The number of memory clusters and distributions when changing λ in DCM-JS.

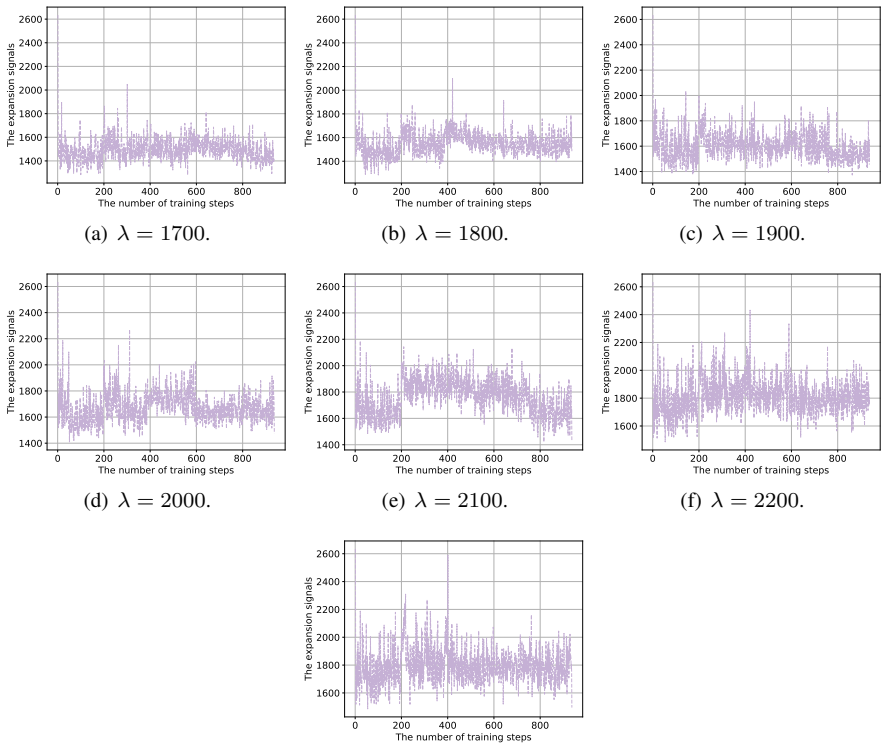
Datasets	DCM-SE	DCM-JS	LTS	LGM	R-VAE	R-DDPM	CGKD-GAN	CNDPM	CGKD-WAE	CGKD-VAE
CelebA-3DChair	1920	1920	2000	2000	2000	2000	2000	2000	2000	2000
CelebA-CACD	1920	1920	2000	2000	2000	2000	2000	2000	2000	2000
CelebA-ImageNet	1920	1920	2000	2000	2000	2000	2000	2000	2000	2000
Split MINIImageNet	1920	1920	2000	2000	2000	2000	2000	2000	2000	2000

Table 2: The number of stored samples of various models for the domain-incremental learning paradigm, achieved by various models.

times than The QL-based method. However, both DCM-SE and DCM-JS are efficient memory approaches.

C.7 The maximum number of samples in the memory cluster

In this section, we investigate whether the number of samples for each memory cluster can significantly affect the memory expansion process and model performance. We train both DCM-QL and DCM-JS on Split MNIST using different π configurations (the maximum number of samples for each memory cluster), where we set $\lambda = 2000$ and $\lambda = 42$ for DCM-QL and DCM-JS, respectively. We provide the results in Fig. 21, which shows that changing the memory cluster size does not lead to a significant change in the model performance and the number of memory clusters.



(a) $\lambda = 1700$. (b) $\lambda = 1800$. (c) $\lambda = 1900$.

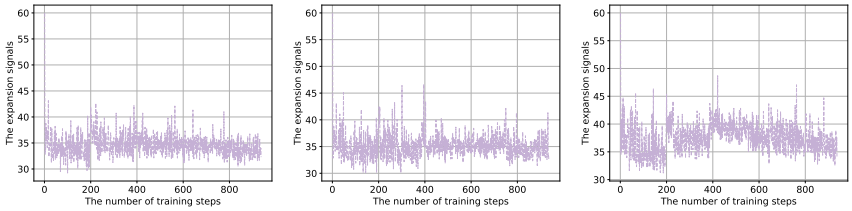
(d) $\lambda = 2000$. (e) $\lambda = 2100$. (f) $\lambda = 2200$.

(g) $\lambda = 2300$.

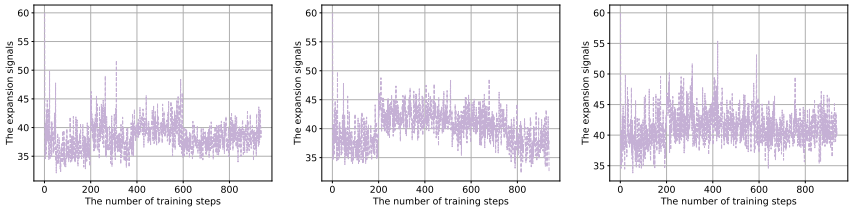
Figure 17: The memory expansion signals when changing λ in DCM-SE.

C.8 Training other types of generative models using DCM

Since the proposed DCM does not interact with the model's optimization during the training, we can directly employ the proposed DCM to train different generative models under OTFCL without modifications. In this section, we train the VAE and GAN using the proposed DCM-SE on Split MNIST, respectively, called DCM-SE-VAE and DCM-SE-GAN. We provide the results in Tab. 3, which show that the proposed DCM-SE can also train other models well while training the diffusion model can achieve the best performance.



(a) $\lambda = 40$. (b) $\lambda = 42$. (c) $\lambda = 44$.



(d) $\lambda = 46$. (e) $\lambda = 48$. (f) $\lambda = 50$.

Figure 18: The memory expansion signals when changing λ in DCM-JS.

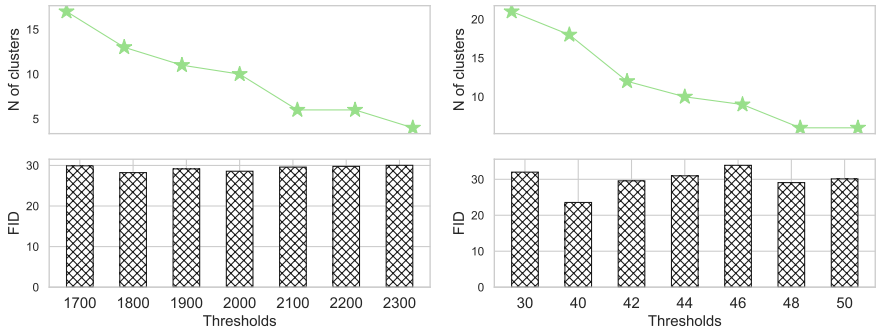
Datasets	DCM-SE	DCM-JS	DCM-SE-VAE	DCM-SE-GAN
Split MNIST	28.57	30.63	45.23	38.76

Table 3: Image generation performance using the Fréchet Inception Distance score (FID) for class-incremental learning.

C.9 Additional information for the classification task

Following from (23), we design a dynamic expansion model in which each component has a VAE model $\mathcal{A}_i = \{p_{\phi_i}(\mathbf{x} \mid \mathbf{z}), q_{\varphi_i}(\mathbf{z} \mid \mathbf{x}) \mid\mid p(\mathbf{z})\}$ and a classifier $f_{\theta_i}(\mathbf{x})$, where i represents the component index. The primary role of the VAE model is to check the network expansion during the training and component selection at the testing phase. Then we design a simple but effective dynamic expansion criterion at t_i as :

$$\min\{f_{\mathrm{MMD}}(\mathbf{Z}'_j, \mathbf{Z}_i) \mid j = 1, \cdots, k-1\} > \lambda^{\mathbf{expansion}}, \tag{1}$$



(a) DCM-SE. (b) DCM-JS.

Figure 19: The number of memory clusters and the model performance when changing λ.

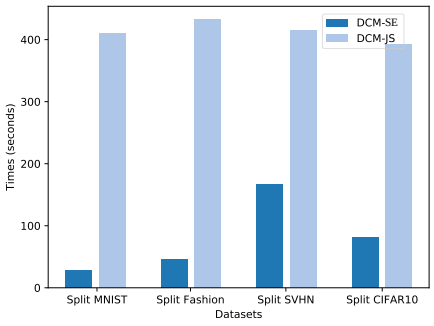
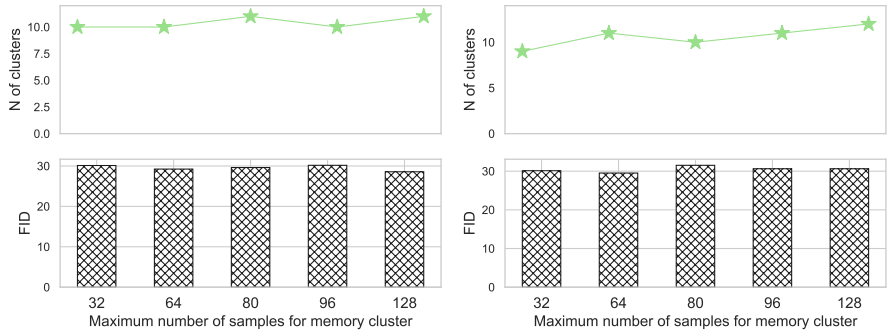


Figure 20: The time required for the memory optimization during the training.

where $f_{\mathbf{MMD}}(\cdot,\cdot)$ is the Maximum Mean Discrepancy (MMD) (20) and $\lambda^{\text{expansion}}$ is a threshold used to control the model expansion. $f_{\text{MMD}}(\cdot,\cdot)$ is defined as :

$$\begin{aligned} f_{\text{MMD}}(\mathbf{Z}'_j, \mathbf{Z}_i) = \frac{1}{b(b-1)} \sum\nolimits_{a \neq g}^{b} \Big\{ & f(\mathbf{z}_a^i, \mathbf{z}_g^i) \\ & + f(\mathbf{z'}_a^j, \mathbf{x'}_g^j) - f(\mathbf{z}_a^i, \mathbf{z'}_g^j) \\ & - f(\mathbf{z'}_g^j, \mathbf{z}_a^i) \Big\}, \end{aligned} \tag{2}$$

where $\mathbf{Z}'_j = \{\mathbf{z'}_1^j, \cdots, \mathbf{z'}_b^j\}$ are the latent variables generated using the VAE model of the j-th component, where $b = 10$ is the batch size. $\mathbf{Z}_i = \{\mathbf{z}_1^i, \cdots, \mathbf{z}_b^i\}$ are the latent variables generated using the encoder $q_{\varphi_k}(\mathbf{z} \,|\, \mathbf{x})$ by the current component (the k-th component) that receives the data batch $\mathbf{X}_i$ at t_i.



(a) DCM-SE. (b) DCM-JS.

Figure 21: The model performance and the number of memory clusters when changing the maximum number of samples for each memory cluster.

C.10 The selection process of λ

Since the proposed dynamic expansion mechanism defined by Eq. (10) of the paper, only accesses the training samples provided at each learning time, we can calculate and observe the expansion signal (left-hand-side of Eq. (10) of the paper) at each learning time. We consider a suitable hyperparameter range ($[1700, 2300]$ and $[30, 50]$) for DCM-SE and DCM-JS according to the expansion signals, respectively. The expansion signal results are provided in **Appendix-C4**. Then the optimal hyperparameter is searched within the hyperparameter range while a small validation set (200 data samples) is used to evaluate the performance.

References

[1] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proc. of IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 11254–11263, 2019. 3

[2] Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8250–8259, 2021. 6, 7

[3] Kamil Deja, Anna Kuzina, Tomasz Trzcinski, and Jakub Mikolaj Tomczak. On an-

alyzing generative and denoising capabilities of diffusion-based deep generative models. In *Advances in Neural Information Processing Systems (NeurIPS), arXiv preprint arXiv:2206.00070*, 2022. 2

[4] Rui Gao and Weiwei Liu. DDGR: continual learning with deep diffusion-based generative replay. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10744–10763. PMLR, 2023. 3

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 5

[6] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local Nash equilibrium. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pages 6626–6637, 2017. 6

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:6840–6851, 2020. 2

[8] Quentin Jodelet, Xin Liu, Yin Jun Phua, and Tsuyoshi Murata. Class-incremental learning using diffusion model for distillation and replay. In *Proc. of the IEEE/CVF International Conference on Computer Vision*, pages 3425–3433, 2023. 3

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. Int. Conf. on Learning Representations (ICLR), arXiv preprint arXiv:1412.6980*, 2015. 4

[10] Zhifeng Kong and Wei Ping. On fast sampling of diffusion probabilistic models. *arXiv preprint arXiv:2106.00132*, 2021. 2

[11] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Univ. of Toronto, 2009. 7

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998. 6

[13] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021. 2

[14] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 7

[15] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proc. of the International Conference on Machine Learning (ICML), vol. PMLR 139*, pages 8162–8171, 2021. 5

[16] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 2

[17] Robin San-Roman, Eliya Nachmani, and Lior Wolf. Noise estimation for generative diffusion models. *arXiv preprint arXiv:2104.02600*, 2021. 2

[18] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, pages 2256–2265. PMLR 37, 2015. 2

[19] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR) arXiv preprint arXiv:2010.02502*, 2021. 2

[20] Ilya O Tolstikhin, Bharath K Sriperumbudur, and Bernhard Schölkopf. Minimax estimation of maximum mean discrepancy with radial kernels. *Advances in Neural Information Processing Systems*, 29:1930–1938, 2016. 18

[21] Zhendong Wang, Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Diffusion-gan: Training gans with diffusion. *arXiv preprint arXiv:2206.02262*, 2022. 2

[22] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 7

[23] Fei Ye and Adrian G. Bors. Continual variational autoencoder learning via online cooperative memorization. In *Proc. European Conference on Computer Vision (ECCV), vol. LNCS 13683*, pages 531–549, 2022. 17

[24] Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Truncated diffusion probabilistic models and diffusion-based adversarial auto-encoders. In *International Conference on Learning Representations (ICLR), arXiv preprint arXiv:2202.09671*, 2023. 2