Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Continual compression model for online continual learning

Fei Ye[a], Adrian G. Bors[b,*]

[a] School of Information Software Engineering, University of Electronic Science and Technology of China, Chengdu, China
[b] Department of Computer Science, University of York, York YO10 5GH, UK

ARTICLE INFO

Keywords:
Continual learning
Dynamic expansion model
Task-Free Continual Learning
Component pruning

ABSTRACT

Task-Free Continual Learning (TFCL) presents a notably demanding but realistic ongoing learning concept, aiming to address catastrophic forgetting in sequential learning systems. In this paper, we tackle catastrophic forgetting by introducing an innovative dynamic expansion framework designed to adaptively enhance the model's capacity for novel data learning while also remembering the information learnt in the past, by using a minimal-size processing architecture. Our proposed framework incorporates three key mechanisms to mitigate model' forgetting: (1) by employing a Maximum Mean Discrepancy (MMD)-based expansion mechanism that assesses the disparity between previously acquired knowledge and that from the new training data, serving as a signal for the model's architecture expansion; (2) a component discarding mechanism that eliminates components characterized by redundant information, thereby optimizing the model size while fostering knowledge diversity; (3) a novel training sample selection strategy that leads to the diversity of the training data for each task. We conduct a series of TFCL experiments that demonstrate the superiority of the proposed framework over all baselines while utilizing fewer components than alternative dynamic expansion models. The results on the Split Mini ImageNet dataset, after splitting the original dataset into multiple tasks, are improved by more than 2% when compared to the closest baseline.

1. Introduction

Continually learning new concepts represents an innate ability of living beings that allows them to learn new skills and adapt to changing environments. Continual learning is required for real-life online learning systems such as those used for autonomous driving vehicles, robots or for streaming services. However, most existing deep learning models fail to achieve continual learning since they would rewrite their previously learnt parameters with new values to adapt when learning the new tasks. As a result, the previously learnt information of previous tasks is lost, leading to performance degeneration [1]. Such performance degeneration is referred to in machine learning as catastrophic forgetting [2] .

The current methods for addressing continual learning can be summarized into three categories: memory-based approaches [3,4], regularization-based methods [5,6] and dynamic expansion models (DEMs) [7,8]. The memory-based methods [4] use a memory buffer to store a few past samples and replay them during the new task learning to deal with forgetting. Another type of the memory-based models trains a deep generative model such as a Variational Autoencoder (VAE) [9] or a Generative Adversarial Network (GAN) [10] to learn and then generate past samples which are combined with new data samples to relieve network forgetting [11]. Such an approach can provide infinite generative replay data samples, which overcome the limitation of a fixed-size memory buffer system. Regularization-based methods [5] usually consider additional terms to the loss function for controlling forgetting. A combined replay and regularization approach is the teacher-student framework where the teacher module is frozen after the training, while the student module is the currently learning model. This approach aims to align the output of the teacher and student modules for the given new data samples by using a penalty term in the primary objective function [6]. Different from the memory-based and regularization-based methods, the dynamic expansion models can automatically add new hidden layers and nodes into a unified network architecture when learning new tasks [8]. The primary advantage of using the dynamic expansion model is that it does not suffer from forgetting problems and can deal with learning long task sequences.

Most existing continual learning studies only consider a simple assumption where the task information and corresponding indices are given during the training. However, this learning scenario is not realistic in most real-time applications, such as when continuously streaming online incoming data. In this paper, we address a more challenging learning scenario called the Task-Free Continual Learning (TFCL) [12], which trains a model on a data stream without accessing any task

* Corresponding author.
E-mail addresses: feiye@uestc.edu.cn (F. Ye), adrian.bors@york.ac.uk (A.G. Bors).

https://doi.org/10.1016/j.asoc.2024.112427
Received 21 May 2024; Received in revised form 4 October 2024; Accepted 23 October 2024
Available online 12 November 2024

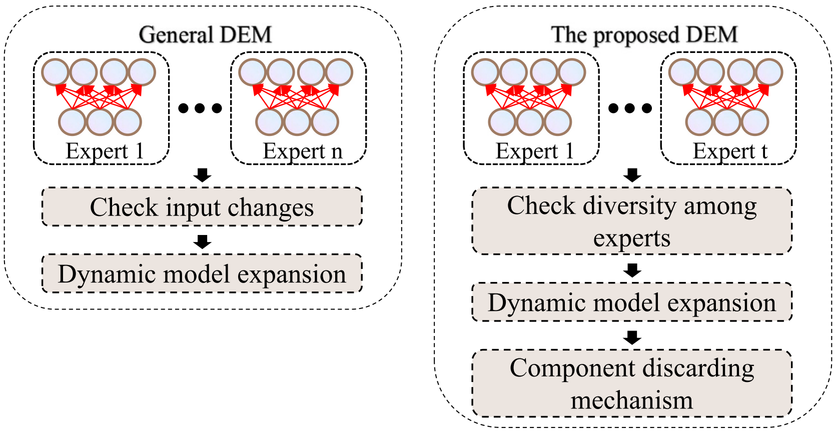


Fig. 1. Comparison between a general DEM and the proposed DEM. The general DEM framework checks the input data shifts, using them as expansion signals while the proposed DEM framework evaluates the knowledge diversity among experts as the expansion signal. In addition, the proposed DEM framework enables a component discarding mechanism to reduce its complexity, optimizing the model.

information or boundaries. A popular and efficient approach for implementing TFCL is by considering a memory buffer for storing some of the past training data [13–15]. When the model proceeds to a new training step, the memorized samples are replayed from the memory buffer to update the model, thus relieving forgetting. However, due to the absence of task information, memory-based approaches require enabling a sample selection approach, aiming to store a diversity of data that can represent all data modalities [13]. In this paper we propose a novel Dynamic Expansion Model (DEM) to address network forgetting in TFCL driven by the following motivations:

- Since memory-based methods usually have only a limited memory capacity, it is intractable to consider such methods for learning infinite data streams. Instead, we employ a new dynamic expansion model (DEM) to deal with the TFCL challenges.
- When compared to the static models, DEM provides a better generalization performance while also being able to preserve the knowledge of much more information from the past learning.

One of the key mechanisms for DEM is the dynamic expansion process that controls and regularizes the network size [16]. Using a Neural Dirichlet process-based dynamic expansion mechanism was proposed in [16], which adds new components automatically when detecting novel information. Another DEM approach from [3] considers evaluating the sample log-likelihood as the signal for the network expansion. However, these two methods [3,16] only consider expanding the network architecture when the input distributions changes, eventually leading to components learning similar knowledge. We provide a comparison between the general DEM and the proposed DEM framework in Fig. 1. An optimal DEM should have a compact network architecture while achieving optimal performance where each expert/component should learn and capture different information. However, most existing DEM methods only consider addressing forgetting problems while ignoring how to compress the model without sacrificing much performance, resulting in non-optimal network architectures.

Aiming to address the limitations of prior works, this study proposes the Continual Compression Model for Online Continual Learning (CCM-OCL) that simultaneously addresses catastrophic forgetting and model compression for DEMs through two goals: (1) Firstly, CCM-OCL dynamically expands its network architecture to learn new information while freezing all previously learnt components to avoid forgetting; (2) Secondly, CCM-OCL optimizes its model size by automatically removing those components deemed redundant.

The contributions of the paper are summarized in the following:

- A novel dynamic expansion mechanism enabled by the Maximum Mean Discrepancy (MMD) measure. The proposed dynamic expansion mechanism dynamically adds new components to the model without accessing any task information.
- A novel approach enables the diversity-aware sample selection which ensures the preservation of a diversity of knowledge in the memory buffer. This sample selection approach can further promote the discrepancy among components, leading to a more compact model structure.
- A new component discarding mechanism is proposed for selectively removing redundant components, resulting in a compact, yet efficient network architecture without sacrificing performance.
- Through multiple TFCL experiments we compare the proposed CCM-OCL model with several well-known methods. The empirical results on several datasets show that the proposed method leads to a compact DEM achieving better performance than other methods.

2. Related works

In this section we provide a comprehensive literature review for continual learning. Continual learning methods can be divided into three categories: memory-based, regularization-based and dynamic expansion models. A succinct categorization of the main approaches is provided in Table 1. In the following we discuss some of the more important continual learning approaches.

2.1. Memory-based methods

Memory-based methods utilize a memory buffer to retain critical past data samples, facilitating the learning process and aiming to mitigate network forgetting during continual learning [3,4]. One typical memory-based approach is the Gradient Coreset-based Replay (GCR) [6], which employs a sample selection approach to preserve the data samples approximating the gradient of all seen data using the model parameters updated at the current task learning. Another approach, the Experience Packing Factor (EPF) [20] is used for continual image learning, by storing important patches, for each image being learnt, in the memory buffer. Specifically, EPF employs the saliency maps as the guideline for choosing which image patches to store. The Adaptive Rehearsal [21] is a simple still efficient memory-based approach which employs the Boltzmann sampling to selectively store past data samples into a memory buffer. However, these memory-based methods only consider simple and popular continual learning environments in which the new classes appear during new training phases. The model's performance can be further improved by enabling the memory-based methods with regularization-based mechanisms, by

Table 1
Outline of significant continual learning models.

Methods	Method type	TFCL	Year
MIR [17]	Memory-based methods	Yes	2019
GSS [18]	Memory-based methods	Yes	2019
Rainbow Memory [19]	Memory-based methods	No	2019
CoPE [13]	Memory-based methods	Yes	2021
GMED [14]	Memory-based methods	Yes	2021
GCR [6]	Memory-based methods	No	2022
EPF [20]	Memory-based methods	No	2023
Adaptive Rehearsal [21]	Memory-based methods	No	2024
DCM [22]	Memory-based methods	No	2024
LwF [23]	Regularization-based method	No	2017
EWC [24]	Regularization-based method	No	2017
Online EWC [25]	Regularization-based method	No	2018
VCL [26]	Regularization-based method	No	2018
MAML-Rep [27]	Regularization-based method	No	2019
ACL [28]	Regularization-based method	No	2020
RFG [29]	Regularization-based method	No	2022
A Closer Look [30]	Regularization-based method	No	2023
DER+++refresh [31]	Regularization-based method	No	2024
Progressive Neural Networks (PNN) [32]	Dynamic expansion model	No	2016
AdaNet [8]	Dynamic expansion model	No	2017
Expert Gate [33]	Dynamic expansion model	No	2017
Compacting, Picking, and Growing [7]	Dynamic expansion model	No	2019
CURL [3]	Dynamic expansion model	Yes	2019
CN-DPM [16]	Dynamic expansion model	Yes	2020
Batch Ensemble [34]	Dynamic expansion model	No	2020
Dynamic-CVA [35]	Dynamic expansion model	Yes	2024

penalizing, during the training, changes in some important network weights [5,6], while managing the memory buffer. Despite providing some promising performances, such methods depend on fixed memory capacities, and are not able to store sufficient information for updating the model when learning an infinite number of tasks.

Another direction of research consists of training a generative network, such as a GAN [10] or a VAE [9], aiming to preserve and reproduce previously learnt information [11]. Once the learning of a task is finished, the generator can produce data samples, consistent with the previously learnt information, which is then combined with new samples to relieve forgetting. However, the performance of such approaches is relying on the quality of the generative replay samples. The model's performance will be decreased significantly when the generator cannot produce high-quality generative replay samples. This is usually caused by the unstable training and mode collapse [36], a phenomenon analyzed and discussed in [37]. In addition, these approaches suffer from an ever-increasing computational complexity burden when learning an increasing number of tasks. Another drawback of these approaches is that they are not scalable when learning many tasks [38].

This paper can address the weaknesses of the memory-based methods [4] by developing a novel DEM that adaptively creates new experts to capture underlying data distributions over time. In addition, the proposed approach can be applied to a more realistic continual learning setting in which the data stream consists of infinite samples.

2.2. Regularization-based methods

Regularization-based approaches aim to alleviate catastrophic forgetting by employing an auxiliary loss term that is added into the primary objective function to penalize significant changes in the network's weights when training the model on a new task [23,39–41]. There are two regularization-based approaches including by adding additional penalty terms in the primary objective function as in the Variational Continual Learning (VCL) [26], or by using knowledge distillation [23]. Learning Without Forgetting (LwF) [23], is perhaps the best known continual learning regularization approach, which employs knowledge distillation in order to enable the newly learnt model to remember previously learnt information. The knowledge distillation mechanism trains a dual processing model framework, called teacher-student, using a specific loss function that aims to align the output data between the student and teacher processing modules. Furthermore, by combining parameter regularization and knowledge distillation into a unified framework rehearsal-free continual learning models [30] can achieve good performance in continual learning without requiring to store any past data samples.

Empirical results have shown that the Elastic Weight Consolidation (EWC) [24] can achieve good results in the task-incremental setting in which each new task contains data samples from all classes. However, the performance of the EWC in the class-incremental setting suffers from significant performance degeneration. This issue/problem was addressed by Schwarz et al. [25], which introduces a new approach called Online EWC which updates the Fisher matrix only at the latest learnt task, reducing the model's complexity. The Variational Continual Learning (VCL) [26] uses Bayesian inference to relieve forgetting for both classifiers and VAE models. Several studies have considered adversarial training [28] to regulate the learned representations in order not to be forgotten during the continual learning process. Such approaches can also be implemented using meta-learning [27]. However, one significant weakness is the amount of computational resources required by such approaches for relieving forgetting, especially when the number of tasks increases ceaselessly [42]. Adversarial Continual Learning [28] uses adversarial learning as a regularization approach to learn task-invariant features that capture the shared information among different tasks. Projected Functional Regularization [29] uses self-supervised learning for regulating network optimization. The Dark Experience Replay (DER) [43], matches the network's logits sampled throughout the optimization trajectory promoting consistency with its past. Furthermore, the DER+++refresh [31] builds upon the results of DER [43] by introducing an unlearning procedure that encourages critical network parameters to only change slowly while allowing more significant changes for those parameters not deemed important.

2.3. Dynamic Expansion Models (DEM)

The DEM framework has been used to deal with forgetting in continual learning by relying on its scalability and generalization abilities, [7,8]. The primary idea of the DEM is to dynamically add new hidden units and layers when learning new tasks. Previously learnt information is preserved in the frozen network parameters to prevent forgetting [32], while only the newly created parameters are updated to learn new knowledge. The Progressive Neural Network [32] builds its architecture upon a starting small network, by gradually adding new component nodes and parameters for learning new tasks. Specifically, this approach preserves all previous information in frozen sub-networks and only updates the newly created parameters when training on a new task. The idea of this expansion mechanism is also applied to [33], which introduces a new mixture model, namely the Expert Gate for CL, where each expert is implemented as an autoencoder for learning a new task. When the learning of all tasks is finished, the reconstruction error is used as the expert selection mechanism during the testing phase. Combining network pruning and model expansion mechanisms was proposed in [7], aiming to achieve a compact dynamic expansion model. This approach relies on the task information to pick certain parameters for learning a new task and thus cannot be applied to TFCL. Furthermore, these dynamic expansion models usually have a large model size when considering learning a long sequence of tasks since each expert has its own individual parameters. This issue is solved by the BatchEnsemble [34], which consists of several modules, each used to learn a new task while considering shared parameters among the ensemble members, reducing the whole model's size. More recently, a gating-based Dirichlet process [38] enabled to reuse existing components to learn related new tasks. The DEM-based models enjoy several advantages over static network architectures: (1) They can preserve the best information from the past tasks and thus can provide a better model performance than the static model; (2) They can deal with an infinite number of tasks [38].

However, most existing dynamic expansion frameworks [34,38] need accessing task information to check the model's expansion process and thus cannot address TFCL. In addition, unlike in the approach proposed in this paper, these methods do not use an appropriate sample selection approach when a memory buffer is used during the learning of the expanding model.

2.4. TFCL learning paradigm

Most existing continual learning methods can only be applied to a general setting, where the task information is accessible at all times. However, this is not a realistic setting. Recent studies have given attention to the TFCL setting, where the model is trained on a data stream without accessing any task information or identity. A popular and efficient way to deal with TFCL is called the memory-based approach which uses a memory buffer to preserve some past training samples. During the subsequent learning, samples from the memory buffer are replayed to determine the model to remember past information. However, since the memory buffer cannot store many samples forever, a suitable sample selection approach is required to selectively preserve only certain data over time. The first paper, which considered training a continual learning classifier on the memory buffer for addressing TFCL was explored in [12]. This approach was then extended by the Maximal Interfered Retrieval (MIR) model [17] to train both a VAE and a classifier by using an information retrieval approach which selectively preserves the most distinct data. More recently, the Gradient Sample Selection (GSS) [18] can selectively store critical samples in the memory buffer by using a constrained optimization reduction approach. Furthermore, the sample selection process can be implemented within a *learner-evaluator* framework called the Continual Prototype Evolution (CoPE) [13], which ensures that the number of memorized samples for each task is equal. Another approach to address TFCL modifies the data samples from the memory buffer, by means of the Gradient-based Memory Editing (GMED) algorithm [14], in which the edited samples would increase the loss in the upcoming model updates. Furthermore, managing a memory system without accessing any supervised signals such as the task or class labels is the Dynamic Cluster Memory (DCM) [22], which dynamically builds several new memory clusters to store critical past information. Different from most existing memory-based methods, the DCM memory system can be used to train deep generative models in continual unsupervised learning. Despite performing very well under TFCL, these methods rely on a single memory buffer with a fixed capacity, making it hard to deal with data streams consisting of several different data domains. This inspires several studies to employ dynamic expansion models for TFCL. The first study applying DEM to TFCL, introduced in [3], proposed an unsupervised learning framework, namely the Continual Unsupervised Representation Learning (CURL). This framework dynamically builds new inference models to capture novel latent information while the generator replays past samples to relieve forgetting. A similar DEM approach, called the Continual Neural Dirichlet Process Mixture (CN-DPM) was introduced in [16]. CN-DPM dynamically creates new VAE-based components that are added to a mixture model by means of a neural Dirichlet process. However, the dynamic expansion mechanism in these methods employs either the sample log-likelihood estimation or density estimation, which ignores the diversity of information contained in the model's mixture components when expanding, leading to sub-optimal network architectures. Implementing a mixture system by using a dynamic expansionable memory cluster was proposed for the task-free continual generation and representation learning in the Dynamic Continual Variational Autoencoder (Dynamic-CVA) [35].

Existing dynamic expansion models [3,16,35] usually evaluate the input shift as the expansion signal, which would tend to create more components/experts when similar data samples appear at different training times. In contrast, the approach proposed in this paper implements the model-expansion checking process by evaluating the distance between the knowledge learnt by the current expert and that of each previously learnt expert, which leads to a compact network architecture. Moreover, the reduction in the number of parameters of the dynamic expansion models has not been explored before in the context of TFCL. The approach from this paper introduces a novel component discarding mechanism that can compress the mixture model without losing much performance.

3. Methodology

In this section, we introduce the proposed Continual Compression Model for Online Continual Learning (CCM-OCL), which employs a novel dynamic expansion mechanism using the MMD measure for enabling the DEM for TFCL. We then propose a diversity-aware sample selection approach enabling the current learning component to acquire novel knowledge. We also propose a component discarding strategy aimed at removing components that represent knowledge closely resembling that of the others. We end with a unifying optimization framework for achieving a good performance with a compact model.

3.1. Problem statement

Let $\mathcal{X}$ and $\mathcal{Y}$ be the space of the data samples and their classes, respectively. Let $\mathcal{D}$ be a data stream trained on a set of training steps/times $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$, where n represents the total number of training steps. Under the class-incremental setting, we divide a training dataset $\mathcal{D}_c$ into several parts $\{\mathcal{D}_c^1, \ldots, \mathcal{D}_c^{c'}\}$ where each $\mathcal{D}_c^j$ represents a set of samples, and c' is the total the number of parts. A data stream $\mathcal{D}$ in this setting is then formed as:

$$\mathcal{D} = \{\mathcal{D}_c^1 \cup \mathcal{D}_c^2 \cdots \cup \mathcal{D}_c^{c'}\}\,. \tag{1}$$

At a certain training time $\mathcal{T}_i$, the model can only access a small batch of samples $\{\mathbf{x}_t, \mathbf{y}_t\}_{t=1}^{b}$ drawn from $\mathcal{D}$ while all previously learnt data samples are not available. In addition, we also consider a more challenging learning paradigm, which involves multiple data domains, expressed as:

$$\mathcal{D} = \{\mathcal{D}_c^1 \cup \mathcal{D}_c^2 \cdots \cup \mathcal{D}_c^{c'} \cdots, \mathcal{D}_{c2}^{c'}\}\,, \tag{2}$$

where $c2$ is the index of the training dataset $\mathcal{D}_{c2}$, which is entirely different from $\mathcal{D}_c$. The model will address the class and domain shift

Table 2
The description of notations.

Notations	Descriptions
$\mathcal{T} = \{\mathcal{T}_1, \dots, \mathcal{T}_n\}$	n number of training times for tasks.
$\mathcal{D}$	The data stream.
$\mathcal{L}_C(S_t, \mathcal{M}_i)$	The loss function used to update the classifier.
$\mathcal{L}_V(S_t, \mathcal{M}_i)$	The loss function used to update the VAE model.
$D_{KL}(P' \,\|\, Q')$	The KL divergence.
$\mathcal{L}^e_{\mathrm{M}}(P, Q)$	The unbiased empirical estimate for the MMD criterion.
S_t	The tth component.
$\mathcal{M}_i$	The memory buffer updated at $\mathcal{T}_i$.
$\|\mathcal{M}_i\|$	The number of data samples for the memory buffer $\mathcal{M}_i$.
λ	The expansion threshold for the proposed approach.
$\mathcal{L}_d(\mathbf{x}_s)$	Calculating the discrepancy score for each sample.
$f_{\mathrm{FID}}(\mathbf{D}', \mathbf{D})$	The FID score for two datasets.
$\mathbf{E}$	The relationship matrix among experts.
$\mathcal{L}_{diver}(S_a)$	Calculating the diversity score for an expert/component.
$\mathcal{L}_{ce}(\mathbf{y}'_j, \mathbf{y}_j)$	The cross-entropy loss.

over time when using the setting from Eq. (2). The primary goal of a continual learning model is that of minimizing the loss values on all previous data batches at $\mathcal{T}_i$, expressed as:

$$\min_{\xi}\left\{\sum_{j=1}^{i} \mathcal{L}(\mathbf{Y}_j, f_{\xi}(\mathbf{X}_j))\right\}, \tag{3}$$

where $\mathbf{X}_j$ and $\mathbf{Y}_j$ denote the jth data batch and corresponding class labels at $\mathcal{T}_j, j \le i$. $\mathcal{L}(\cdot,\cdot)$ is a loss function and $f_{\xi}(\cdot)$ is a classifier defined by the parameter set ξ. However, updating the model's parameter set ξ using Eq. (3) in continual learning is impossible because we cannot access all previously seen data batches. Once the whole training procedure is finished, we evaluate the model's performance using all testing datasets. In Table 2 we provide some important notations and abbreviations used in this paper.

3.2. MMD-based expansion mechanism

A straightforward and efficient approach for addressing forgetting in TFCL is by utilizing a defined-capacity memory buffer designed to retain a limited number of previous examples. Such a memory buffer is denoted as $\mathcal{M}_i$, where the subscript i indicates that $\mathcal{M}_i$ was updated at $\mathcal{T}_i$. Let $\mathcal{G} = \{S_1, \dots, S_t\}$ be a DEM and we assume that the mixture model $\mathcal{G}$ is made up of t components, where each component S_t has a classifier C_t and a Variational Autoencoder (VAE) V_t used for the component selection at the inference time, [9]. The classifier C_t is implemented using a fully connected or a convolution network $f_{\xi_t} : \mathcal{X} \to \mathcal{Y}$, where ξ_t is the model's parameter, which is used to predict the class label for each sample. The VAE model V_t consists of two neural networks $f^v_{\omega_t} : \mathcal{X} \to \mathcal{Z}$ and $f^v_{\theta_t} : \mathcal{Z} \to \mathcal{X}$. The former predicts the Gaussian hyper-parameters that are used to form an encoding distribution $q_{\omega_t}(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$, where $\{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$ are inferred by $f^v_{\omega_t}$. Meanwhile, the latter network is used to model a decoding distribution $p_{\theta_t}(\mathbf{x} \mid \mathbf{z})$, assumed to be Gaussian. To ensure that the error backpropagated properly through the encoding–decoding process during the training, we use the reparameterization trick to implement the sampling process of the latent variable $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\tau}$, where $\odot$ is the element-wise product and $\boldsymbol{\tau} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

When optimizing the whole mixture framework during continual learning, one reasonable approach is to update only one component while freezing all previously learnt components. Let $\mathcal{L}_C$ denote the classification loss used to update the classifier C_t and $\mathcal{L}_V$ represents the VAE loss used to update V_t. The current component S_t is trained, using the samples from $\mathcal{M}_i$ at $\mathcal{T}_i$ for updating the loss functions for the component's classifier and VAE, respectively, as:

$$\mathcal{L}_C(S_t, \mathcal{M}_i) \stackrel{\Delta}{=} \frac{1}{|\mathcal{M}_i|}\sum_{j=1}^{|\mathcal{M}_i|} \mathcal{L}_{ce}\big(f_{\xi_t}(\mathbf{x}_j), \mathbf{y}_j\big), \tag{4}$$

$$\mathcal{L}_V(S_t, \mathcal{M}_i) \stackrel{\Delta}{=} \mathbb{E}_{q_{\omega_t}(\mathbf{z}|\mathbf{x})}\left[\log p_{\theta_t}(\mathbf{x}_t \mid \mathbf{z})\right] - D_{KL}\left[q_{\omega_t}(\mathbf{z} \mid \mathbf{x}_t) \;\|\; p(\mathbf{z})\right], \tag{5}$$

where $\mathcal{L}_{ce}(\cdot,\cdot)$ is the cross-entropy loss, defined as:

$$\mathcal{L}_{ce}(\mathbf{y}'_j, \mathbf{y}_j) = \sum_{c=1}^{\hat{K}} y_c \log(y'_c), \tag{6}$$

where $\hat{K}$ is the total number of categories. y_c and y'_c denote the cth dimension of $\mathbf{y}_j$ and $\mathbf{y}'_j$, respectively. We use $|\mathcal{M}_i|$ to denote the number of memorized samples at $\mathcal{T}_i$. $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a pre-defined Gaussian prior, and $D_{KL}(\cdot \;\|\; \cdot)$ is the Kullback–Leibler (KL) divergence, defined as:

$$D_{KL}(P' \;\|\; Q') = \frac{1}{2}\Bigg[\log\frac{|\boldsymbol{\Sigma}_{Q'}|}{|\boldsymbol{\Sigma}_{P'}|} - d^{\mathbf{z}} + (\boldsymbol{\mu}_{P'} - \boldsymbol{\mu}_{Q'})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}_{Q'}(\boldsymbol{\mu}_{P'} - \boldsymbol{\mu}_{Q'}) + \mathrm{tr}(\boldsymbol{\Sigma}^{-1}_{Q'}\boldsymbol{\Sigma}_{P'})\Bigg], \tag{7}$$

where $\boldsymbol{\mu}_{P'}$ and $\boldsymbol{\mu}_{Q'}$ denote the mean vector of the Gaussian distribution P' and Q', respectively. $d^{\mathbf{z}}$ represents the dimension of the latent space. $\boldsymbol{\Sigma}_{Q'}$ and $\boldsymbol{\Sigma}_{P'}$ denote the covariance matrix for Q' and P', respectively. The superscript T and -1 of a matrix denotes the matrix transpose and inverse matrix while $\mathrm{tr}(\cdot)$ denotes the matrix trace.

The Maximum Mean Discrepency (MMD) is a popular measure used to evaluate the distance between two probability density functions. Such a distance is formed on the basis of embedding probabilities in a Reproducing Kernel Hilbert space (RKHS) [44]. Let us define Q and P as two Borel probability measures. We consider $\mathbf{x}$ and $\mathbf{u}$ as random variables over a topological space $\mathcal{X}$. We consider $\{f \in \mathcal{F} | f : \mathcal{X} \to \mathbf{R}\}$ to denote a function, while $\mathcal{F}$ represents a class of functions. We define the MMD between Q and P as [44]:

$$\mathcal{L}_{\mathrm{M}}(P, Q) \stackrel{\Delta}{=} \sup_{f\in\mathcal{F}}\left(\mathbb{E}_{\mathbf{x}\sim P}\left[f(\mathbf{x})\right] - \mathbb{E}_{\mathbf{u}\sim Q}\left[f(\mathbf{u})\right]\right). \tag{8}$$

where sup denotes the least upper bound of a set of numbers. If $P = Q$, we have $\mathcal{L}_{\mathrm{M}}(P, Q) = 0$. The function class $\mathcal{F}$ is considered as a unit ball in an RKHS with a positive definite kernel $k(\mathbf{x}, \mathbf{x}')$. Calculating Eq. (8) is usually computationally intractable. In practice, the MMD is estimated on the embedding space [45], expressed as:

$$\mathcal{L}^2_{\mathrm{M}}(P, Q) = \|\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q\|^2, \tag{9}$$

where $\boldsymbol{\mu}_P$ and $\boldsymbol{\mu}_Q$ denote the mean embedding of P and Q, respectively. $\|\cdot\|^2$ denotes the Euclidean distance. $\boldsymbol{\mu}_P$ is defined as:

$$\boldsymbol{\mu}_P(\mathbf{x}') = \int k(\mathbf{x}, \mathbf{x}')\frac{\partial P(\mathbf{x})}{\partial \mathbf{x}}d\mathbf{x}, \tag{10}$$

where $P(\mathbf{x})$ denotes the probability density function for P. $\boldsymbol{\mu}_P$ also satisfies the following equation:

$$\mathbb{E}[f(\mathbf{x})] = \langle f, \boldsymbol{\mu}_P\rangle_{\mathcal{H}}, \tag{11}$$

where $\langle f\,,\cdot\rangle_{\mathcal{H}}$ denotes the inner product. Since RKHS has the reproducing property $f \in \mathcal{F}, f(\mathbf{x}) = \langle f\,,\, k(\mathbf{x}, \mathbf{x}')\rangle_{\mathcal{H}}$, Eq. (9) can be calculated using the kernel functions:

$$\mathcal{L}^2_{\mathrm{M}}(P, Q) = \mathbb{E}_{\mathbf{x},\mathbf{x}'\sim P}[k(\mathbf{x}, \mathbf{x}')] - 2\mathbb{E}_{\mathbf{x}\sim P,\mathbf{u}\sim Q}[k(\mathbf{x}, \mathbf{u})] + \mathbb{E}_{\mathbf{u},\mathbf{u}'\sim Q}[k(\mathbf{u}, \mathbf{u}')], \tag{12}$$

where $\mathbf{u}'$ and $\mathbf{x}'$ are independent copies of $\mathbf{u}$ and $\mathbf{x}$, respectively. In practice, we employ the same number of samples from P and Q ($N_P = N_Q$), where N_P and N_Q are the total counts of samples for P and Q, respectively. Then Eq. (12) can be estimated using an unbiased empirical estimate, defined as:

$$\mathcal{L}^e_{\mathrm{M}}(P, Q) = \frac{1}{N_P(N_p - 1)}\sum_{i\neq j}^{N_P} h(i, j), \tag{13}$$

where $h(i, j) = k(\mathbf{x}_i, \mathbf{x}_j) + k(\mathbf{u}_i, \mathbf{u}_j) - k(\mathbf{x}_i, \mathbf{u}_j) - k(\mathbf{x}_j, \mathbf{u}_i)$.

In the following, we describe how the MMD measure can be used as the expansion signal for the proposed dynamic expansion mechanism. Assume that the DEM $\mathcal{G} = \{S_1, \dots, S_t\}$ has already learnt t components. During the training, we only update the current component S_t at $\mathcal{T}_i$. The dynamic expansion process is implemented by encouraging the probabilistic diversity between the trained components when adding a new component. We can express this goal as an optimization problem:

$$c^\star = \arg \max_{c=i,i+1,\dots,n} \sum_{j=1}^{t-1} \mathcal{L}_{\mathrm{M}}^{e}\big(\mathrm{P}_{\tilde{\mathbf{z}}_j}, \mathrm{P}_{\mathbf{z}_{\mathcal{M}_c}}\big)\,, \tag{14}$$

where $\mathrm{P}_{\tilde{\mathbf{z}}_j}$ is the distribution of the latent vectors $\{\tilde{\mathbf{z}}_{j,1},\dots,\tilde{\mathbf{z}}_{j,m}\}$. Each latent vector $\tilde{\mathbf{z}}_{j,s}$ is predicted using the inference model with the sample $\tilde{\mathbf{x}}_{j,s}$ produced by V_j. $\mathrm{P}_{\mathbf{z}_{\mathcal{M}_c}}$ is the distribution formed using the latent vectors $\{\mathbf{z}_1,\dots,\mathbf{z}_m\}$. Each latent vector $\mathbf{z}_s$ is predicted based on the stored sample $\mathbf{x}_s$ drawn from the memory $\mathcal{M}_c$ using the inference model of V_t. The right-hand side of Eq. (14) is used to evaluate the distance between the current memory buffer and the knowledge preserved in each previously learnt component. $c^\star$ is the optimal solution that achieves the maximum distance. Eq. (14) can be viewed as a recursive optimization problem if we expand the model $\mathcal{G}$ (S_t is added). However, searching for the optimal solution $c^\star$ in Eq. (14) is impossible since we need to access the data from all training steps, which are not available. Therefore, we propose a new dynamic expansion mechanism for the model in order to be able to evaluate Eq. (14) at $\mathcal{T}_i$, expressed as:

$$\min\Big\{\mathcal{L}_{\mathrm{M}}^{e}\big(\mathrm{P}_{\tilde{\mathbf{z}}_1}, \mathrm{P}_{\mathbf{z}_{\mathcal{M}_i}}\big),\dots,\mathcal{L}_{\mathrm{M}}^{e}\big(\mathrm{P}_{\tilde{\mathbf{z}}_{t-1}}, \mathrm{P}_{\mathbf{z}_{\mathcal{M}_i}}\big)\Big\} \ge \lambda, \tag{15}$$

where λ is a hyper-parameter used to control the model's expansion. If the current memory buffer $\mathcal{M}_i$ stores sufficiently different samples with respect to the already learnt knowledge at $\mathcal{T}_i$ (satisfying Eq. (15)), we freeze the current component S_t and build a new component S_{t+1} at the next training step $\mathcal{T}_{i+1}$. The λ's value influences the model's expansion. A small λ tends to frequently add new components during the training while a large λ has the opposite effect.

The MMD measure has several limitations, which can be summarized into three aspects: (1) The estimation of the MMD measure is imprecise when the target distributions are complex; (2) The choice of different kernel functions $k(\cdot,\cdot)$ can lead to different results for the MMD; (3) The MMD criterion requires considerable computational costs when it is evaluated on the high-dimensional data space. In the experiments we evaluate the MMD on a low-dimensional latent space, which is computationally efficient.

In order to avoid learning from data samples representing similar information, we clear up the memory buffer $\mathcal{M}_i$ after is being used to train each component/expert. If similar data would at different times $\mathcal{T}_i$ be stored in the memory buffer $\mathcal{M}_i$, the components trained on this memory buffer would capture and accumulate similar knowledge with each other. This paper addresses this issue, as described in detail in the next section, by developing a data sample selection approach that encourages the diversity of information stored in the memory buffer.

3.3. Diversity-aware sample selection

Unlike other sample selection approaches, such as [13], which ensures balanced data samples across all tasks, this section introduces a new sample selection approach for $\mathcal{M}_i$ aiming to store novel samples which are then used to train the current learning component. The primary idea of the proposed sample selection is to encourage the current component to learn examples that are different from the information stored by each previously trained component. Before we introduce the proposed approach, we first discuss a straightforward sample selection approach, which is considered as the baseline, called the Sliding Window approach, which always stores incoming samples while removing the earliest stored samples when the memory buffer is full at $\mathcal{T}_i$, as a first-in-first-out (FIFO) approach :

$$\mathcal{M}_i = \mathcal{M}_{i-1} \bigcup \mathcal{D}_i\ ,\ \mathcal{M}_i = \bigcup_{j=b}^{|\mathcal{M}_i|+b} \mathcal{M}_i[j]\,, \tag{16}$$

where $\mathcal{M}_i[j]$ is the jth stored sample of $\mathcal{M}_i$ and b is the batch size, considered as $b = 10$ in the experiments, while $|\mathcal{M}_i|$ is the previous buffer size. However, Eq. (16) only allows the current memory buffer to store newly seen information while ignoring previous data samples and the sample diversity. This can result in $\mathcal{M}_i$ storing many similar data samples. To address this problem, we propose a new sample selection approach to selectively preserve those data samples that are critical for training the model.

The primary idea of the proposed approach is to store those data samples which are different from the information which is already accumulated. Thus, in the first step we evaluate the novelty of incoming samples. Since VAEs have inference mechanisms and can estimate the sample log-likelihood, we use the VAEs associated with the existing CCM-OCL components to evaluate the novelty of incoming samples, [3]. To achieve this goal, we calculate the sum of the negative sample log-likelihood estimated by each previously learnt component as a discrepancy score for each sample to be memorized :

$$\mathcal{L}_d(\mathbf{x}_s) = \frac{1}{t-1}\sum_{j=1}^{t-1}\Big\{-\mathcal{L}_V(S_j,\mathbf{x}_s)\Big\}\,, \tag{17}$$

where $\mathbf{x}_s$ is the sth memorized sample. A small $\mathcal{L}_d(\mathbf{x}_s)$ indicates that the sample $\mathbf{x}_s$ is known by at least one of the previously learnt components and this sample should not be stored in order to be used for training. Then, we selectively store the samples through the proposed Diverse Sampling Selection :

$$\mathbf{x}_s' = \arg\min_{\mathbf{x}_s\in\mathcal{M}_i}\{\mathcal{L}_d(\mathbf{x}_s)\}\,, \tag{18}$$

then we remove the selected sample by considering that $\mathbf{x}_s' \notin \mathcal{M}_i$. We recursively repeat the exclusion of other samples $\mathbf{x}_s'$, considered as redundant, until $|\mathcal{M}_i| \le |\mathcal{M}_i|^{Max}$, where $|\mathcal{M}_i|^{Max}$ is the maximum buffer size.

3.4. The component discarding mechanism

Existing dynamic expansion frameworks ignore the model compression and thus do not result in compact network architectures. In this section, we propose further optimizing the model size by developing a new component discarding mechanism that selectively deletes unimportant components that have learnt similar knowledge. The other advantage of the proposed component discarding mechanism is that of promoting knowledge diversity among the model's components.

For removing those components characterized by representing overlapping information we introduce a graph relation matrix to articulate the knowledge similarity among components, serving as a framework to facilitate the elimination of redundant components. We illustrate the detailed component discarding mechanism in Fig. 2, where we assume that the DEM has already learnt t components. Firstly, we employ the VAE of each component to generate 1,000 samples, representing the previous knowledge learnt by the model, denoted as $\mathrm{P}_{\tilde{x}_j}$ for the jth component. Then we calculate the relationship graph on a pair of components using the Fréchet Inception Distance (FID) score, [46], estimated from the samples generated by the VAEs associated with each component. FID is chosen for two main reasons: (1) The FID score is a symmetric measure and does not require knowing the exact density form of each data distribution; (2) FID was shown as an efficient measure when considered for calculating differences between empirical data distributions, used to evaluate the performance of GAN models on image generation tasks [47]. Let $\mathbf{E} \in I\!R^{t\times t}$ be a relationship matrix whose entry $E_{(c,g)}$ is the edge between a pair of components S_c and S_g, evaluated as $1/f_{\mathrm{FID}}(\mathrm{P}_{\tilde{x}_c},\mathrm{P}_{\tilde{x}_g})$, where $f_{\mathrm{FID}}(\cdot,\cdot)$ is a function that returns the FID score between the data distributions corresponding to two components, defined as [46]:

$$f_{\mathrm{FID}}(\mathbf{D}',\mathbf{D}) = \|\boldsymbol{\mu}_{\mathbf{D}} - \boldsymbol{\mu}_{\mathbf{D}'}\|^2 + \mathrm{tr}(\boldsymbol{\Sigma}_{\mathbf{D}} + \boldsymbol{\Sigma}_{\mathbf{D}}' - 2(\boldsymbol{\Sigma}_{\mathbf{D}}\boldsymbol{\Sigma}_{\mathbf{D}'})^{1/2})\,, \tag{19}$$

where $\mathbf{D}$ and $\mathbf{D}'$ represent two different datasets. $\boldsymbol{\mu}_{\mathbf{D}}, \boldsymbol{\mu}_{\mathbf{D}'}$ and $\boldsymbol{\Sigma}_{\mathbf{D}}, \boldsymbol{\Sigma}_{\mathbf{D}'}$ represent the means and covariance matrices of the variable vectors, $\mathbf{D}$ and $\mathbf{D}'$, respectively. Since the FID score is symmetric, $\mathbf{E}$ is also a

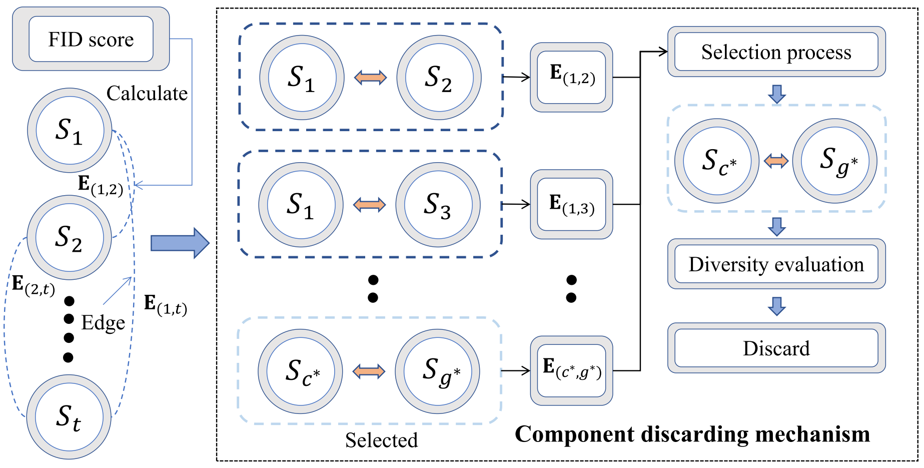


Fig. 2. The proposed mixture component discarding mechanism. We create an adjacency matrix representing a knowledge graph for the model. Each entry of this matrix is the FID score between the distributions of generated data by two VAEs, corresponding to two components. We select a pair of components with the maximal edge value and discard one of them according to a statistical diversity criterion.

symmetric matrix. We then search for a pair of components showing a maximum knowledge similarity by using $\mathbf{E}$:

$$E^\star = \max_{c,g=1,\dots,t} E_{(c,g)},$$
$$c^\star, g^\star = \arg \max_{c,g=1,\dots,t} E_{(c,g)}\,, \tag{20}$$

where $c^\star$ and $g^\star$ are the indices of the selected components. Then we remove one of these components from $\mathcal{G}$ by considering that the deleted component should have a large discrepancy with respect to the other components. To achieve this goal, we define a function that calculates the diversity score for a component:

$$\mathcal{L}_{diver}(S_a) = \frac{1}{t-1}\sum_{j=1}^{t}\frac{1}{E_{(j,a)}}\ ,\ j \neq a\,. \tag{21}$$

We employ Eq. (21) to estimate the diversity score for $S_{c^\star}$ and $S_{g^\star}$, respectively:

$$\tilde{c} = \arg \min_{\hat{c}=\{c^\star,g^\star\}} \left\{\mathcal{L}_{diver}(S_{\hat{c}})\right\}, \tag{22}$$

where $\tilde{c}$ is the index of the final selected component marked to be removed from $\mathcal{G}$. In order to remove more components with overlapping information we continually evaluate Eqs. (20) and (22) until a trade-off is achieved between the number of components and performance.

Another approach consists of removing several components, which are not considered important by employing a threshold λ_2 on the maximum admitted knowledge similarity in the network ;

$$E^\star < \lambda_2, \tag{23}$$

where $E^\star$ is evaluated in Eq. (20) leading to an acceptable number of components.

3.5. The unified optimization framework

In this section, we provide the unified framework which incorporates the sample selection approach, expansion mechanism and discarding mechanism. The overview of the framework is provided in Fig. 3 while the pseudo-code is provided in Algorithm 1, which can be summarized in four steps:

Step 1 (The training process.) In the initial training stage, we build the first expert consisting of a classifier and a VAE model. The proposed dynamic expansion mechanism defined in Eq. (15) requires at least two experts. We freeze the first expert and automatically build the second expert at $\mathcal{T}_{200}$. In the subsequent learning, we assume that the proposed DEM has already learnt t experts. Only the last (current) component S_t is updated on $\mathcal{M}_{i-1}$ and the new training data $\mathbf{X}_i$, at $\mathcal{T}_i$ using the loss functions $\mathcal{L}_C(S_t, \mathcal{M}_i)$ and $\mathcal{L}_V(S_t, \mathcal{M}_i)$. We also update the memory buffer by $\mathcal{M}_i = \mathcal{M}_{i-1} \cup \mathbf{X}_i$.

Step 2 (Check the model expansion.) We perform the dynamic expansion if the memory buffer is full, $|\mathcal{M}_i| \geq |\mathcal{M}_i|^{Max}$. If the criterion from Eq. (15) is satisfied, we freeze the current component S_t and create a new component S_{t+1}, which is added to the model $\mathcal{G}$. We also clean up the memory buffer $\mathcal{M}_i$ to allow the newly created component to learn new data.

Step 3 (Sample selection.) If the memory buffer is not full, $|\mathcal{M}_i| \leq |\mathcal{M}_i|^{Max}$, we perform the sample selection using Eq. (18). We continually store the incoming data samples in the memory buffer as long as it still has the capacity.

Step 4 (Discarding similar components.) Once the whole training procedure is completed, we repeatedly remove from the mixture model $\mathcal{G}$, those components which contain similar information with other components.

4. Experiments

In this section, we provide the results of several experiments to evaluate the effectiveness of the proposed CCM-OCL and compare it with the state-of-the-art. Then we investigate the importance of each component of our methodology by performing a full ablation study. In the following, we provide the detailed experiment settings.

Datasets: Split MNIST splits the original MNIST dataset[1] [48] containing 60k training samples into five subsets according to the category information, as in [13]. Split CIFAR10 divides the original CIFAR10 dataset[2] [49] into five subsets, each consisting of images from two different classes, [13]. Split CIFAR100 splits the original CIFAR100 dataset,[3] consisting of 50,000 training data samples, into 20 subsets where each set contains 2,500 examples from five classes in the given class order, [42]. Similar to other studies we normalize pixel values of all images from $[0, 255]$ to $[-1, -1]$ for each image.

Software/Hardware system: In all experiments, we employ one Tesla V100 GPU for training models. We run the experiments using the operating system (Ubuntu 18.04.5) and use Pytorch library to implement our approach.

Network architecture and hyperparameters: We consider employing the setting from [13], and consider ResNet-18 [50], as the classifier for Split CIFAR10 and Split CIFAR100. For Split MNIST, we employ an

[1] https://yann.lecun.com/exdb/mnist/
[2] https://www.cs.toronto.edu/~kriz/cifar.html
[3] https://www.cs.toronto.edu/~kriz/cifar.html

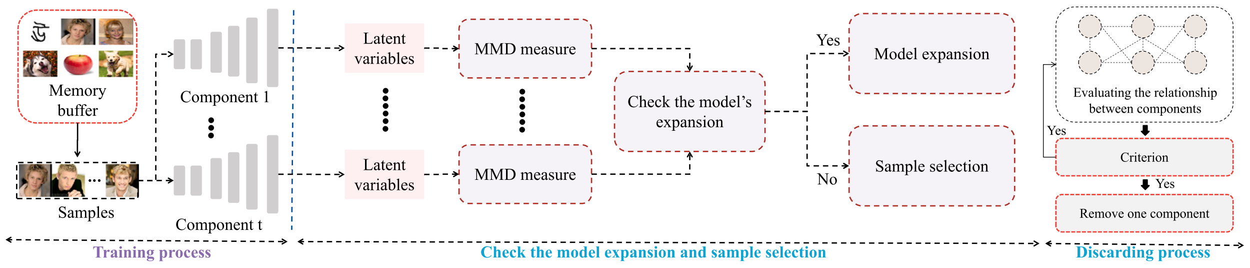


Fig. 3. Overview of the proposed learning framework consisting of four steps. Firstly, we train the current "Component t" on the currently updated memory at $\mathcal{T}_i$. Then we check the model's expansion using Eq. (15) and perform the sample selection using Eq. (18). Finally, when the whole training stage is finished, we discard the redundant components to compress the model. The 'Criterion' denotes the proposed component discarding criterion, defined in Eq. (23).

Algorithm 1: Algorithm for the proposed framework

```
Input: The data stream, the total number of training steps ($N$) ;
Output: The model's parameters ;
for $i < N$ do
    Training process;
    $\{\mathbf{X}_i, \mathbf{Y}_i\} \sim \mathcal{D}$ ;
    Train $S_t$ on $\mathcal{M}_{i-1}$ and $\{\mathbf{X}_i, \mathbf{Y}_i\}$ ;
    $\mathcal{M}_i = \mathcal{M}_{i-1} \cup \{\mathbf{X}_i, \mathbf{Y}_i\}$ ;
    Check the model's expansion ;
    if $|\mathcal{M}_i| = |\mathcal{M}_i|^{Max}$ then
        Estimate the MMD using Eq. (13);
        if $\min\left\{\mathcal{L}^e_{\rm M}(\mathrm{P}_{\tilde{\mathbf{z}}_1}, \mathrm{P}_{\mathbf{z}_{\mathcal{M}_i}}), \cdots, \mathcal{L}^e_{\rm M}(\mathrm{P}_{\tilde{\mathbf{z}}_{t-1}}, \mathrm{P}_{\mathbf{z}_{\mathcal{M}_i}})\right\} \geq \lambda$ then
            Build a new expert $\mathcal{G} = \mathcal{G} \bigcup S_{t+1}$;
            Clear up the memory $\mathcal{M}_i = \{\}$ ;
        end
        else
            Perform the sample selection ;
            $\mathcal{L}_d(\mathbf{x}_s) = \frac{1}{t-1}\sum_{j=1}^{t-1}\left\{-\mathcal{L}_V(S_j, \mathbf{x}_s)\right\}$ ;
            $\mathbf{x}'_s = \arg\min_{\mathbf{x}_s \in \mathcal{M}_i}(\mathcal{L}_d(\mathbf{x}_s)$;
            We repeat recursively the exclusion of $\mathbf{x}'_s$'es until
              $|\mathcal{M}_i| \leq |\mathcal{M}_i|^{Max}$;
        end
    end
    Perform the discard mechanism ;
    Calculate the matrix $\mathbf{E}$ using FID criterion ;
    for $j < (t - n)$, (n is the number of removed components) do
        Search for a pair of components by
          $c^\star, g^\star = \arg\max_{c,g=1,\cdots,t} E_{(c,g)}$ ;
        Diversity evaluation using Eq. (21) ;
        Remove one selected component from $\mathcal{G}$ ;
        Remove all edge values of the deleted component from $\mathbf{E}$ ;
    end
end
```

MLP neural network consisting of two hidden layers of 400 units, [13]. Following from [13], we set the maximum memory size as 2000, 1000 and 5000 data samples for Split MNIST, Split CIFAR10, and Split CIFAR100, respectively. The task information is unavailable and during each training step the model can only see a small batch of training samples, where the batch size is $b = 10$. We consider ten training epochs for each learning step in for all continuous learning models used for the experiments. The proposed framework has two parameters λ and λ_2 for regulating the model optimization. The goal of the hyperparameter λ is to control the network architecture expansion. A small λ tends to allow the framework to create more components over time, resulting in a large model size. In contrast, a large λ tends to create a small number of components, ensuring a compact model structure. The hyperparameter λ_2 is used to regulate the component discarding process. We provide the values for the hyperparameters in Table 3.

Baselines: The proposed CCM-OCL is trained using the MMD-based expansion mechanism as the base model, while when considering the sample selection approach for the memory buffer and the component discarding mechanism, called "CCM-OCL + Discard". We designate the model as "SW + Discard" when adopting a moving-window based sample selection approach for the buffer, when adopting CCM-OCL as the base model. In this study, we compare with two continual learning dynamic expansion baselines: Continual Neural Dirichlet Process Mixture (CN-DPM) [16], and Continual Unsupervised Representation Learning (CURL) [3]. In addition, we also compare the proposed approach with : 'finetune' that learns a classifier using the data stream, Maximal Interfered Retrieval (MIR) [17], Gradient Sample Selection (GSS) [18], Dynamic-Online Cooperative Memorization (OCM), [51], incremental Classifier and Representation Learning (iCARL) [52], Gradient Episodic Memory (GEM) [42], Reservoir [53], Continual Prototype Evolution (CoPE) [13], ER + GMED and ER$_a$ + GMED, [14], where ER is the Experience Replay [54], and ER$_a$ is ER with data augmentation. Dynamic-CVA [35] is a recently proposed dynamic expansion framework, in which each component consists of a hybrid VAEGAN model and a classifier. Specifically, the Dynamic-CVA detects the loss change as the signal for expanding the network architecture.

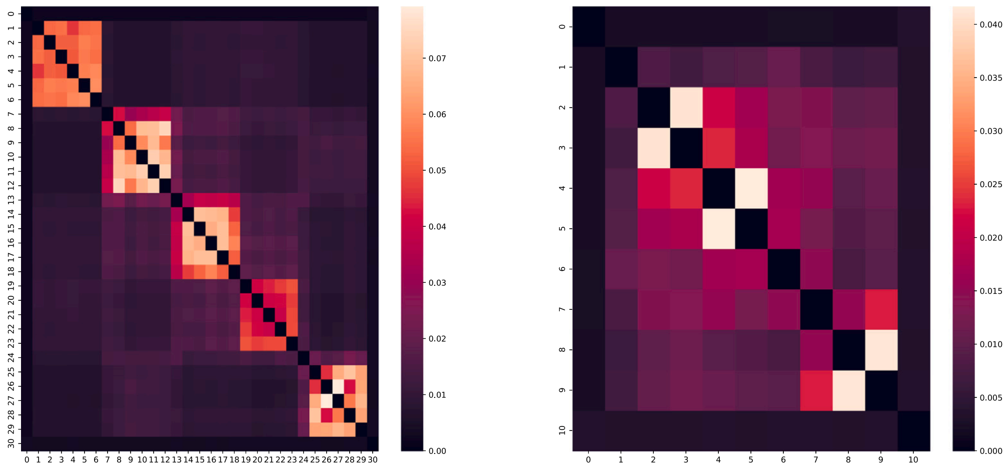
4.1. Continual image classification learning results

Class-Incremental Learning. We examine the performance of the proposed methodology in the class-incremental learning scenario. We follow the standard TFCL experiment setting from [13], in which the model only sees and accesses a small batch of data samples in an online learning manner. The classification results for three datasets, including Split CIFAR10, Split MNIST and Split CIFAR100, are provided in Table 4, where CCM-OCL denotes that the proposed approach employs only the dynamic expansion and sample selection approaches. We also provide the number of components for the proposed model in Table 5. From Table 4, we can observe that the proposed CCM-OCL achieves the best results on these datasets when comparing with other methods. In addition, by employing the component discarding mechanism does not influence the performance too much, according to the results from Table 4, while significantly compressing the model, according to Table 5. The proposed framework outperforms other dynamic expansion models, significantly compacting the number of necessary parameters.

We also investigate how the component relationship matrix guides the component discarding process. We train CCM-OCL including the proposed discarding mechanism under Split MNIST. The relation matrix $\mathbf{E}$ before and after applying the component discarding procedure is shown in Figs. 4-a and 4-b, respectively. The components belonging to the same cluster tend to have a light color indicating a close relationship among these components, and from Fig. 4-a we observe that the learned components from the CCM-OCL are grouped into five clusters. This result is due to the fact that the adjacent components tend to learn similar knowledge with each other. When performing the discarding process, the number of components is reduced to ten as shown in Fig. 4-b. This result shows that the proposed discarding mechanism can efficently remove overlapping knowledge components.

Table 3
Hyperparameter values for the proposed framework. $\mathcal{L}_r$ represents the loss value.

Hyper-par.	Split MNIST	Split CIFAR10	Split CIFAR100	Split MImageNet
λ	0.009	0.04	0.03	0.055
λ_2	0.05	0.03	0.01	0.03
$\mathcal{L}_r$	0.001	0.001	0.001	0.001



(a) The initial relationships. (b) After compressing the model.

Fig. 4. The representation of the graph relationship matrix **E**.

Table 4
Supervised classification performance (accuracy), evaluated from five independent runs for different models.

Methods	Split MNIST	Split CIFAR10	Split CIFAR100
ER + GMED†	82.67 ± 1.90	34.84 ± 2.20	20.93 ± 1.60
MIR*	93.20 ± 0.36	42.80 ± 2.22	20.00 ± 0.57
GEM*	93.25 ± 0.36	24.13 ± 2.46	11.12 ± 2.48
iCARL*	83.95 ± 0.21	37.32 ± 2.66	10.80 ± 0.37
finetune*	19.75 ± 0.05	18.55 ± 0.34	3.53 ± 0.04
reservoir*	92.16 ± 0.75	42.48 ± 3.04	19.57 ± 1.79
CoPE-CE*	91.77 ± 0.87	39.73 ± 2.26	18.33 ± 1.52
GSS*	92.47 ± 0.92	38.45 ± 1.41	13.10 ± 0.94
CoPE*	93.94 ± 0.20	48.92 ± 1.32	21.62 ± 0.69
CURL*	92.59 ± 0.66	–	–
CN-DPM*	93.23 ± 0.09	45.21 ± 0.18	20.10 ± 0.12
ER_a + GMED†	82.21 ± 2.90	47.47 ± 3.20	19.60 ± 1.50
Dynamic-OCM	94.02 ± 0.23	49.16 ± 1.52	21.79 ± 0.68
Dynamic-CVA	95.23 ± 0.05	50.28 ± 1.16	23.58 ± 0.57
CCM-OCL + Discard	96.16 ± 0.11	50.12 ± 0.23	25.24 ± 0.17
SW + Discard	95.93 ± 0.15	49.93 ± 0.23	24.18 ± 0.18
SW	96.81 ± 0.12	50.91 ± 0.25	25.65 ± 0.16
CCM-OCL	**96.95** ± 0.13	**53.71** ± 0.19	**26.03** ± 0.16

* and † represent the results cited from [13], and [14], respectively.

Table 5
The number of components for the proposed model when learning different datasets.

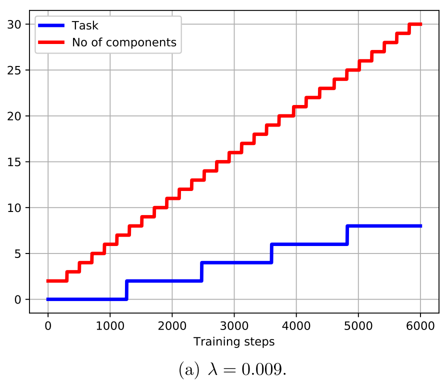
Methods	Split MNIST	Split CIFAR10	Split CIFAR100
CCM-OCL + Discard	10	10	6
SW	30	35	10
CCM-OCL	29	31	9
SW + Discard	10	10	6

In the following, we also investigate the performance achieved by different models when they are trained on a large dataset, by considering MINI-ImageNet dataset,[4] [55], a subset of ImageNet dataset, [56], consisting of 100 categories with 60,000 training samples. Following from [17], we obtain Split MINI-ImageNet after dividing MINI-ImageNet into 20 subsets, with each set consisting of images from five classes. We consider setting the maximum memory size for all methods as 10,000 for Split MINI-ImageNet and using a ResNet-18 network [50] as the classifier. We report the classification results for Split MINI-ImageNet in Table 6, by quoting the results of some related methods from [14]. After the training, the proposed model has ten components and we employ the proposed discarding mechanism, eventually ending with eight components. These results show that the proposed model achieves better results than other methods on this large dataset.

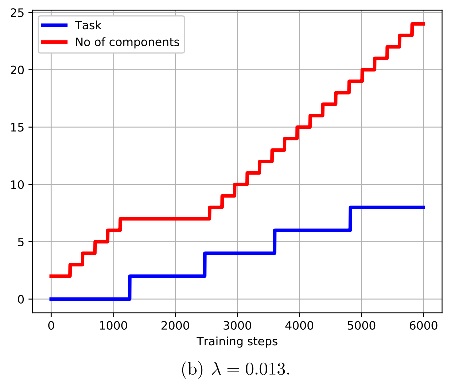
Results when considering fuzzy task boundaries. In a challenging learning environment, the data stream would include mixed data samples from different classes at different times. This setting is called the fuzzy task boundaries [16], in which samples from the next task are introduced and mixed with the training data from the current task after learning half of the current task' data. Specifically, for Split MNIST, the first task contains data samples from categories '0', '1', '2', '3' where the number of samples for the categories '0' and '1' is of 5000 and the number of samples for the categories '2' and '3' is of 2500. The second task contains data samples from the categories '2', '3', '4', '5' where the number of samples for each class is 2500. We perform the same split procedure for the third and fourth tasks while the last task only contains data samples from the categories '8' and '9'. For adapting Split CIFAR10 and Split MImageNet to the fuzzy task boundary setting, we employ the same split procedure. We train the proposed model using this setting for three datasets, including Split MNIST, Split CIFAR10 and Split MImageNet and we present the results in Table 7. We also provide the model complexity, given by the number of components, in Table 8. These results show that the proposed framework provides better performance than other methods under this challenging setting.

Learning multiple domains when considering component reduction. We consider a more complex data stream by combining Split MNIST and Split SVHN, resulting in Split MNIST-SVHN and in the same way we create Split MNIST-CIFAR10, which combines Split MNIST and Split CIFAR10. These data streams involve shifts in both domains and classes over time, representing a more challenging experimental context than other data streams that consider the learning of data

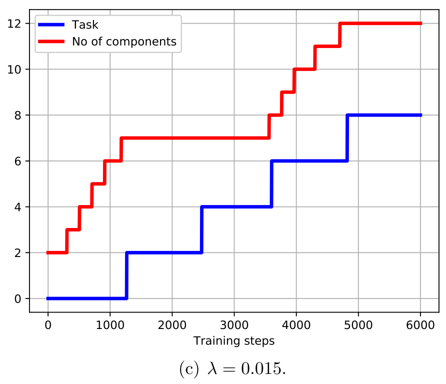
[4] https://github.com/yaoyao-liu/mini-imagenet-tools



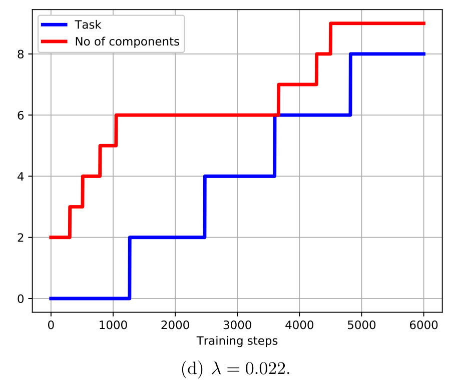
(a) $\lambda = 0.009$.



(b) $\lambda = 0.013$.



(c) $\lambda = 0.015$.



(d) $\lambda = 0.022$.

Fig. 5. The number of components used by the proposed approach when increasing the number of tasks being learnt.

Table 6
Supervised classification performance (accuracy) evaluated after performing 20 independent runs when testing various models on Split MImageNet.

Methods	Split MImageNet
ER + GMED	27.27 ± 1.8
MIR+GMED	26.50 ± 1.3
ER$_a$	25.92 ± 1.2
MIR	25.21 ± 2.2
CCM-OCL + Discard	27.58 ± 2.7
CCM-OCL	**29.63** ± 1.5

Table 7
Supervised classification performance (accuracy) evaluated after performing five independent runs for different models over a data stream with fuzzy task boundaries.

Methods	Split MNIST	Split CIFAR10	Split MImageNet
ER	79.74 ± 4.0	37.15 ± 1.6	26.47 ± 2.3
Vanilla	21.53 ± 0.1	20.69 ± 2.4	3.05 ± 0.6
ER + GMED	82.73 ± 2.6	40.57 ± 1.7	28.20 ± 0.6
MIR	84.80 ± 1.9	38.70 ± 1.7	25.83 ± 1.5
MIR+GMED	86.17 ± 1.7	41.22 ± 1.1	26.86 ± 0.7
CCM-OCL + Discard	95.80 ± 2.1	43.57 ± 1.6	28.27 ± 1.5
CCM-OCL	**96.51** ± 1.8	**44.23** ± 1.4	**29.35** ± 1.2

Table 8
The number of components from the proposed model for Split MNIST, Split CIFAR10 and Split MImageNet when considering fuzzy task boundaries.

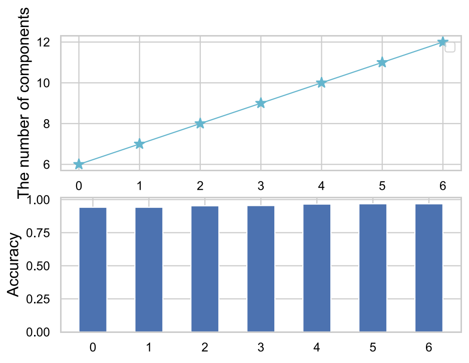
Methods	Split MNIST	Split CIFAR10	Split MImageNet
CCM-OCL + Discard	10	10	6
CCM-OCL	25	32	12

Table 9
Classification results under Split MNIST-SVHN and Split MNIST-CIFAR10. "Parameters" and "No" represents the number of parameters and of components, respectively. "Speed" represents the testing/inference time required for evaluating each testing sample.

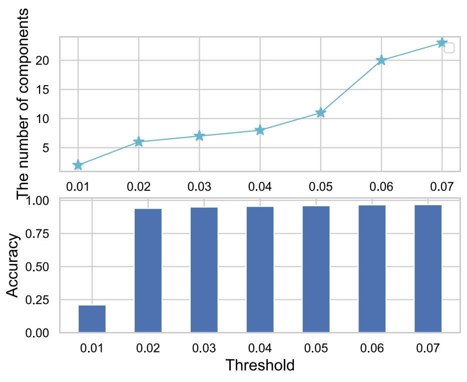
Methods	Split MNIST-SVHN			
	Accuracy	No	Parameters	Speed (s)
CCM-OCL + Discard	63.78	13	126M	0.0048
CCM-OCL	**65.29**	18	175M	0.0064
CCM-OCL + KD	58.82	13	126M	0.0048
CCM-OCL + Pruning	60.63	18	131M	0.0052
CN-DPM	60.26	31	301M	0.0104
Methods	**Split MNIST-CIFAR10**			
CCM-OCL + Discard	62.15	11	106M	0.0032
CCM-OCL	**64.36**	16	155M	0.0050
CCM-OCL + KD	56.64	11	106M	0.0032
CCM-OCL + Pruning	58.97	16	112M	0.0041
CN-DPM	59.92	27	262M	0.0106

from a single domain. We consider various specific architecture reduction continual learning models such as Pruning [57], and Knowledge Distillation (KD) [37]. We also consider an approach similar to the pruning from [57] to reduce the number of mixture components in our framework and call the approach as CCM-OCL + Pruning. KD can be used to transfer knowledge from multiple components to a single one as in [37] and we call this method as CCM-OCL + KD. The classification accuracy of various models on the complex data streams is provided in Table 9. From this table it can be observed that CCM-OCL requires fewer parameters and less inference times than CN-DPM [16] when used in the continual learning of these complex data streams. These results demonstrate that the proposed framework can learn several separate components that capture different information. In contrast, CN-DPM would result in statistically overlapping components following training, eventually also requiring additional computational costs.

Discussion. The class-incremental learning represents an important paradigm assumption within continual learning. Data from each class is provided in a sequential way and it should be learnt without forgetting the information which was learnt before. A robust model under this setting should attain an excellent average classification accuracy upon



(a) Number of remaining components.



(b) Changing λ_2 in (24).

Fig. 6. The classification performance obtained by the proposed model when varying the number of remaining experts.

completing all training phases, as illustrated in Table 4. On the other hand an effective dynamic expansion model must maintain a minimal architecture, defined by a low component count, enabling the seamless deployment on resource-limited machines and devices. The results from Table 5 indicate that the proposed discarding mechanism reduces the number of components from 29 to 10 on the Split MNIST dataset. Such results show substantial model compression without compromising performance. The component discarding mechanism holds numerous potential applications. For instance, when a robot is trained to acquire various skills sequentially, data corresponding for training a certain skill may arise at different times, leading to an increase in redundant components. The proposed mechanism automatically eliminates knowledge-overlapping components, thus enabling the robot to acquire additional skills over time while utilizing a fixed-size storage device.

4.2. Ablation study

In the following we perform an extensive ablation study for evaluating the performance of the proposed approach when using different configurations.

Expansion mechanism: We study the dynamic expansion process by training the proposed framework considering different values for the threshold λ in Eq. (15) and then we examine the change in the number of components for the proposed approach during training. We train CCM-OCL under split MNIST, and record the number of components and the number of tasks seen in each training step. We plot the results in Fig. 5, where we observe that a large threshold λ encourages the proposed model to use fewer components, considering that a single component learns two tasks. A small threshold λ, on the other hand, leads to training more components while multiple components model sub-tasks of a single task.

Component discarding: We study the performance change for the proposed framework when enabling the discarding mechanism while evaluating the performance of the remaining components. We consider to learn the proposed framework CCM-OCL on Split MNIST, by removing the redundant components. Also in Fig. 4-b we visualize the relationship matrix $\mathbf{E}$ among the components after considering the component discarding on the configuration shown in Fig. 4-a. We provide the number of components and the performance in the top and bottom plot, respectively, from Fig. 6-a. These results show that the performance of the model does not suffer from too much degeneration even when the number of components considered is rather low, such as six components in this case.

We also change the threshold λ_2 from Eq. (23), aiming to remove several components, which are not considered that important, during the training, and the results are reported in Fig. 6-b, showing the number of components on top plot and the accuracy in the bottom plot. The proposed framework tends to remove fewer components

Table 10
The classification accuracy calculated by the proposed model on Split MNIST when changing the batch size.

Batch size	10	30	50	100	130	160
CCM-OCL	96.12	95.23	95.89	96.54	98.27	96.23

Table 11
The training time (hours) of various models for the continual supervised learning.

Methods	Split MNIST	Split CIFAR10	Split CIFAR100
Dynamic-CVA	0.69	2.53	3.17
CCM-OCL + Discard	0.52	2.36	2.98

Table 12
Assessing the proposed data selection in the proposed model.

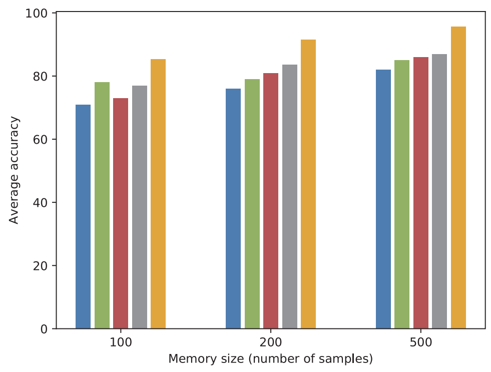
Methods	Split MNIST	Split CIFAR10	Split CIFAR100
MMD-CoPE	95.21	52.75	25.49
MMD-MIR	95.28	52.29	25.52
MMD-reservoir	95.62	52.97	25.38
CCM-OCL	**96.85**	**53.76**	**26.01**

when the threshold λ_2 increases. In contrast, using a very small λ_2 can significantly reduce the model size, while the model's performance would also suffer from significant degeneration.

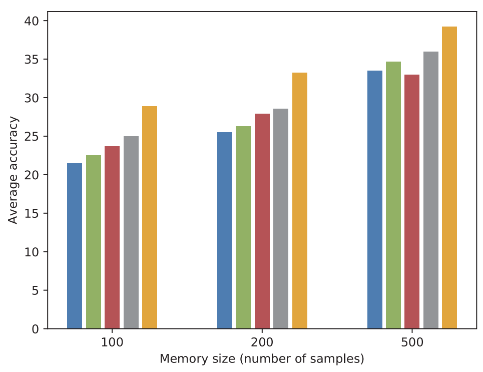
The influence of the batch size on the performance. We investigate whether changing the batch size can significantly affect the performance of the proposed framework. We consider different batch size configurations for training the proposed model on Split MNIST and the results are provided in Table 10. We can observe that the performance of the proposed framework does not change much when using different batch sizes.

Computational complexity. We evaluate the training time for the proposed framework when considering Split MNIST, Split CIFAR10 and Split CIFAR100, and provide the results Table 11. From these results we can observe that the proposed framework requires less computational costs then the state-of-the-art method Dynamic-CVA [35]. The primary reason for these results is because the proposed framework employs an efficient sample selection approach while the dynamic expansion mechanism is based on processing the whole embedding space.

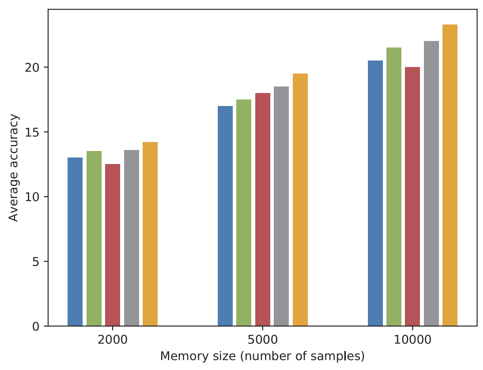
Comparison with other sample selection methods. We compare the proposed data sampling approach with other sample selection methods that are used when considering the MMD approach in CoPE [13], MIR [17] or reservoir, denoting the resulting methods as: MMD-CoPE, MMD-MIR, and MMD-reservoir. The classification accuracy for these approaches when applied on Split MNIST, Split CIFAR10 and Split CIFR100 datasets is reported in Table 12. We observe that the proposed sample selection approach performs better than other sample selection methods on these datasets.



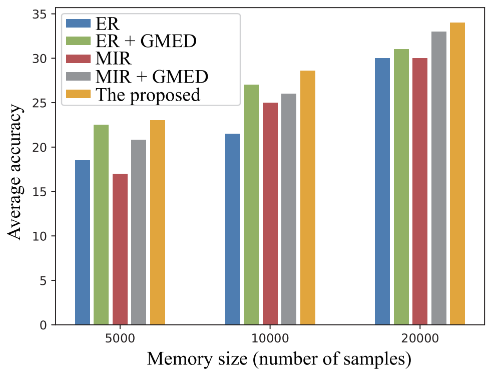
(a) Split MNIST.



(b) Split CIFAR10.



(c) Split CIFAR100.



(d) Split MINI-ImageNet.

Fig. 7. Classification results achieved by different models when changing the memory size.

Changing the memory size. We examine the model's performance on various datasets when varying the memory size. The classification results of the model CM-OCL + Discard when compared with other baselines are reported in Fig. 7, which show that the proposed framework achieves stable performance under different memory configurations.

5. Conclusion and future research

In this paper, we propose a novel dynamic model expansion mechanism for the Task Free Continual Learning (TFCL) scenario. The proposed model employs the Maximum Mean Discrepancy (MMD) measure to evaluate the change in the data distributions over time, providing a better criterion for triggering model expansion when the probabilistic representation of data changes. The proposed approach promotes the learning of distinct knowledge by each expert throughout the entire training process. In addition, we propose a novel sample selection approach for the memory buffer, which selectively preserves critical data samples which are different from the already learnt knowledge by the model. The sample selection approach by encouraging the information diversity in the memory buffer, enables the current component to learn novel information further ensuring the knowledge diversity and discrepancy among the trained components. Furthermore, we propose a component discarding mechanism to remove those components that have learnt similar concepts with each other, significantly reducing the model size and optimizing processing time. The model is extensively tested in the context of TFCL resulting in the state-of-the-art performance for the proposed methodology.

Advantages: One significant advantage of the proposed approach over existing methods is that it introduces a novel dynamic expansion mechanism that evaluates the distance between the knowledge learnt by the current expert and each previously frozen expert from the mixture model. This distance measure is used to regulate the model's expansion process, leading to a compact network architecture. Another advantage of the proposed approach is that it is introducing a novel expert discarding mechanism that automatically removes redundant knowledge-overlapping experts. This mechanism enables deploying the model on resource-constrained platforms, such as drones or robots.

Limitations: One weakness of the proposed approach is the fact that the model's dynamic expansion is sensitive to the selection of the expansion threshold λ which may have to be adapted for different databases. Another weakness is that it suffers from data privacy issues given that past data samples are stored in a memory buffer and then are used each time when checking the model expansion. Such samples may contain confidential information. One way to address this problem would be to store corresponding latent variables instead of real data.

Future Research and Impact: The area of deep learning and artificial intelligence is very dynamic and new processing models are created all the time. The methodology proposed can be extended to other networks with small adjustments to the properties of the networks and models. An important direction of future research consists in designing specialized networks for various applications where continual learning is essential. The proposed dynamic expansion framework can be applied to real-time systems, autonomous driving, robots or drones navigation or by continually assessing the evolution of diseases from medical data. In addition, another benefit of the proposed approach is that it can be applied for training large language models, aiming to reduce computational costs by continually learning new information without the need to retrain on the whole dataset.

CRediT authorship contribution statement

Fei Ye: Writing – original draft, Validation, Software, Methodology, Formal analysis, Conceptualization. **Adrian G. Bors:** Writing – review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] F. Zenke, B. Poole, S. Ganguli, Continual learning through synaptic intelligence, in: Proc. of Int. Conf. on Machine Learning, vol. PLMR 70, 2017, pp. 3987–3995.

[2] G.I. Parisi, R. Kemker, J.L. Part, C. Kanan, S. Wermter, Continual lifelong learning with neural networks: A review, Neural Netw. 113 (2019) 54–71.

[3] Dushyant Rao, Francesco Visin, Andrei A. Rusu, Yee Whye Teh, Razvan Pascanu, Raia Hadsell, Continual unsupervised representation learning, in: Advances in Neural Inf. Proc. Systems, NeurIPS, 2019, pp. 7645–7655.

[4] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. Dokania, P.H.S. Torr, M.'A. Ranzato, On tiny episodic memories in continual learning, 2019, arXiv preprint arXiv:1902.10486.

[5] Yanan Gu, Xu Yang, Kun Wei, Cheng Deng, Not just selection, but exploration: Online class-incremental continual learning via dual view consistency, in: Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2022, pp. 7442–7451.

[6] Rishabh Tiwari, Krishnateja Killamsetty, Rishabh Iyer, Pradeep Shenoy, GCR: Gradient coreset based replay buffer selection for continual learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2022, pp. 99–108.

[7] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, Chu-Song Chen, Compacting, picking and growing for unforgetting continual learning, in: Advances in Neural Information Processing Systems, 2019, pp. 13647–13657.

[8] C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri, S. Yang, AdaNet: Adaptive structural learning of artificial neural networks, ICML, in: Proc. of Int. Conf. on Machine Learning, vol. PMLR 70, 2017, pp. 874–883.

[9] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, 2013, arXiv preprint arXiv:1312.6114.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proc. Advances in Neural Inf. Proc. Systems, NIPS, 2014, pp. 2672–2680.

[11] H. Shin, J.K. Lee, J. Kim, J. Kim, Continual learning with deep generative replay, in: Advances in Neural Inf. Proc. Systems, NIPS, 2017, pp. 2990–2999.

[12] Rahaf Aljundi, Klaas Kelchtermans, Tinne Tuytelaars, Task-free continual learning, in: Proc. of IEEE/CVF Conf. on Computer Vision and Pattern Recognition, 2019, pp. 11254–11263.

[13] Matthias De Lange, Tinne Tuytelaars, Continual prototype evolution: Learning online from non-stationary data streams, in: Proc. of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 8250–8259.

[14] Xisen Jin, Arka Sadhu, Junyi Du, Xiang Ren, Gradient-based editing of memory examples for online task-free continual learning, in: Advances in Neural Information Processing Systems, NeurIPS, 2021, pp. 29193–29205.

[15] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, Jonghyun Choi, Rainbow memory: Continual learning with a memory of diverse samples, in: Proc. of IEEE/CVF Conf. on Computer Vision and Pattern Recognition, CVPR, 2021, pp. 8218–8227.

[16] Soochan Lee, Junsoo Ha, Dongsu Zhang, Gunhee Kim, A neural Dirichlet process mixture model for task-free continual learning, in: Int. Conf. on Learning Representations, ICLR, 2020, arXiv preprint arXiv:2001.00689.

[17] Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, Tinne Tuytelaars, Online continual learning with maximal interfered retrieval, in: Advances in Neural Information Processing Systems, NeurIPS, 2019, pp. 11872–11883.

[18] R. Aljundi, M. Lin, B. Goujaud, Y. Bengio, Gradient based sample selection for online continual learning, in: Advances in Neural Information Processing Systems, NeurIPS, 2019, pp. 11817–11826.

[19] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, Jonghyun Choi, Rainbow memory: Continual learning with a memory of diverse samples, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 8218–8227.

[20] Gobinda Saha, Kaushik Roy, Saliency guided experience packing for replay in continual learning, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2023, pp. 5273–5283.

[21] James Seale Smith, Lazar Valkov, Shaunak Halbe, Vyshnavi Gutta, Rogerio Feris, Zsolt Kira, Leonid Karlinsky, Adaptive memory replay for continual learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 3605–3615.

[22] Fei Ye, Adrian G. Bors, Online task-free continual generative and discriminative learning via dynamic cluster memory, in: Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 26202–26212.

[23] Z. Li, D. Hoiem, Learning without forgetting, IEEE Trans. Pattern Anal. Mach. Intell. 40 (12) (2017) 2935–2947.

[24] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, R. Hadsell, Overcoming catastrophic forgetting in neural networks, Proc. Natl. Acad. Sci. (PNAS) 114 (13) (2017) 3521–3526.

[25] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, Raia Hadsell, Progress & compress: A scalable framework for continual learning, ICML, in: Proc. of Int. Conf. on Machine Learning, vol. PMLR 80, 2018, pp. 4535–4544.

[26] Cuong V Nguyen, Yingzhen Li, Thang D Bui, Richard E Turner, Variational continual learning, in: Proc. of Int. Conf. on Learning Representations, ICLR, 2018, arXiv preprint arXiv:1710.10628.

[27] Khurram Javed, Martha White, Meta-learning representations for continual learning, in: Advances in Neural Information Processing Systems, NeurIPS, 2019, pp. 1820–1830.

[28] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, Marcus Rohrbach, Adversarial continual learning, ECCV, in: Proc. European Conf. on Computer Vision, vol. LNCS 12356, 2020, pp. 386–402.

[29] Alex Gomez-Villa, Bartlomiej Twardowski, Lu Yu, Andrew D Bagdanov, Joost Van de Weijer, Continually learning self-supervised representations with projected functional regularization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 3867–3877.

[30] James Seale Smith, Junjiao Tian, Shaunak Halbe, Yen-Chang Hsu, Zsolt Kira, A closer look at rehearsal-free continual learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2023, pp. 2410–2420.

[31] Zhenyi Wang, Yan Li, Li Shen, Heng Huang, A unified and general framework for continual learning, in: International Conference on Learning Representations, ICLR, 2024, arXiv preprint arXiv:2403.13249.

[32] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, Raia Hadsell, Progressive neural networks, 2016, arXiv preprint arXiv:1606.04671.

[33] R. Aljundi, P. Chakravarty, T. Tuytelaars, Expert gate: Lifelong learning with a network of experts, in: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, CVPR, 2017, pp. 3366–3375.

[34] Yeming Wen, Dustin Tran, Jimmy Ba, BatchEnsemble: an alternative approach to efficient ensemble and lifelong learning, in: Proc. Int. Conf. on Learning Representations, 2020, arXiv preprint arXiv:2002.06715.

[35] Fei Ye, Adrian G. Bors, Task-free continual generation and representation learning via dynamic expansionable memory cluster, in: Proc. of the AAAI Conference on Artificial Intelligence, vol. 38, 2024, pp. 16451–16459.

[36] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, C. Sutton, Veegan: Reducing mode collapse in GANs using implicit variational learning, in: Proc. Advances in Neural Inf. Proc. Systems, NIPS, 2017, pp. 3308–3318.

[37] Fei Ye, Adrian G. Bors, Lifelong teacher-student network learning, IEEE Trans. Pattern Anal. Mach. Intell. 44 (10) (2022) 6280–6296.

[38] Fei Ye, Adrian G. Bors, Lifelong infinite mixture model based on knowledge-driven Dirichlet process, in: Proc. of the IEEE/CVF International Conference on Computer Vision, ICCV, 2021, pp. 10695–10704.

[39] Yujun Shi, Li Yuan, Yunpeng Chen, Jiashi Feng, Continual learning via bit-level information preserving, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 16674–16683.

[40] Jingxin Zhang, Donghua Zhou, Maoyin Chen, Adaptive cointegration analysis and modified RPCA with continual learning ability for monitoring multimode nonstationary processes, IEEE Trans. Cybern. (2022).

[41] W. Chen, Bo. Chen, Y. Liu, X. Cao, A. Zhao, H. Zhang, L. Tian, Max-margin deep diverse latent Dirichlet allocation with continual learning, IEEE Trans. Cybern. 52 (7) (2022) 5639–5653.

[42] David Lopez-Paz, Marc'Aurelio Ranzato, Gradient episodic memory for continual learning, in: Advances in Neural Information Processing Systems, 2017, pp. 6467–6476.

[43] Matteo Buzzega, Angelo Porrello, Davide Abati, Simone Calderara, Dark experience for general continual learning: a strong, simple baseline, in: Advances in Neural Information Processing Systems, NIPS, 2020, pp. 15920–15930.

[44] Ilya O Tolstikhin, Bharath K Sriperumbudur, Bernhard Schölkopf, Minimax estimation of maximum mean discrepancy with radial kernels, Adv. Neural Inf. Process. Syst. 29 (2016) 1930–1938.

[45] Gintare Karolina Dziugaite, Daniel M. Roy, Zoubin Ghahramani, Training generative neural networks via maximum mean discrepancy optimization, 2015, arXiv preprint arXiv:1505.03906.

[46] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter, GANs trained by a two time-scale update rule converge to a local Nash equilibrium, in: Advances in Neural Information Processing Systems, NIPS, 2017, pp. 6626–6637.

[47] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. C. Courville, Improved training of wasserstein GANs, in: Proc. Advances in Neural Inf. Proc. Systems, NIPS, 2017, pp. 5767–5777.

[48] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[49] Alex Krizhevsky, Geoffrey Hinton, Learning Multiple Layers of Features from Tiny Images, Technical Report, Univ. of Toronto, 2009.

[50] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. of IEEE Conf. on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 770–778.

[51] Fei Ye, Adrian G. Bors, Continual variational autoencoder learning via online cooperative memorization, ECCV, in: Proc. European Conference on Computer Vision, vol. LNCS 13683, 2022, pp. 531–549.

[52] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, Christoph H Lampert, iCaRL: Incremental classifier and representation learning, in: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, CVPR, 2017, pp. 2001–2010.

[53] Jeffrey S. Vitter, Random sampling with a reservoir, ACM Trans. Math. Softw. 11 (1) (1985) 37–57.

[54] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, Gregory Wayne, Experience replay for continual learning, NeurIPS, in: Advances in Neural Information Processing Systems, vol. 34, 2019, pp. 348–358.

[55] Ya Le, Xuan Yang, Tiny ImageNet Visual Recognition Challenge, Technical Report, Univ. of Stanford, 2015, pp. 1–6.

[56] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Inf. Proc. Systems, NIPS, 2012, pp. 1097–1105.

[57] Michael Zhu, Suyog Gupta, To prune, or not to prune: exploring the efficacy of pruning for model compression, in: ICLR-Workshop, 2018, arXiv preprint arXiv:1710.01878.