Training Deep Generative Models Via Lifelong Learning

Fei Ye

Doctor of Philosophy

University of York

Department of Computer Science,

September 2022

Abstract

Lifelong learning represents an essential function of an artificial intelligence system, which can continually acquire and learn novel knowledge without forgetting it. Lately, deep learning has brought new possibilities for the development of artificial intelligence, including artificial lifelong learning systems. However, most existing lifelong learning systems are limited to the classification task, and lifelong generative modelling remains a new stage. In this PhD thesis, our research goal mainly focuses on training deep generative models in the context of lifelong learning. The advantage of our research topic over general continual learning is that we can implement many downstream tasks within a unified framework, including classification, image generation, image interpolation, and disentangled representation learning. Firstly, we propose a new lifelong hybrid approach, combining the advantages of Generative Adversarial Net (GAN) and Variational Autoencoder (VAE) for lifelong generative modelling. The proposed model can learn a robust generative replay network that provides high-quality generative replay samples to relieve forgetting, while we also train inference models to capture meaningful latent representations over time. Secondly, to learn a long sequence of tasks, we propose a novel dynamic expansion model that can reuse existing network parameters and knowledge to learn a related task while building a new component to deal with a novel task. Thirdly, we propose a novel lifelong teacher-student framework where a dynamic expansible GAN mixture model implements the teacher module. Then, we introduce a novel self-supervised learning approach for the Student that allows capturing cross-domain latent representations from the entire knowledge accumulated by the Teacher as well as from novel data. Finally, we extend the lifelong teacher-student framework to task-free continual learning, where the task information is unavailable. The proposed model can adaptively expand its network architecture when detecting the data distribution shift during the training, which can be applied to infinite data streams.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my gratitude to my primary supervisor, Dr. Adrian G. Bors, who guided me throughout this project. He has regular meetings with me to discuss the research, including helpful corrections on the writing and constructive suggestions. I have learned a lot of research and writing skills from those meetings. In addition, he supports me in publishing the research findings at several top-tier conferences, which provides opportunities for me to communicate with international scholars. Finally, I am pleased to work with him on the research project.

I would also like to thank my assessor, Prof. Richard Wilson, for the helpful discussions in the TAP meetings and his precise assessment of my work. In addition, he also provides valuable suggestions on my research and thesis writing, which is important for my thesis. Finally, I also wish to acknowledge the help provided by my colleagues in the Vision, Graphics and Learning Research group, who usually held several interesting research seminars that provided valuable research insights for me.

I would like to thank my friends, Guoxi Huang, ZeChao Hu, Jingbo Yang and Cameron Kyle-Davidson in the UK, for their support and help during my PhD study.

Finally, I would like to thank my family members for their support. My mother, YueZhen Hou, and my father, YuFu Ye encourage me to pursue a PhD degree at a top university while they also provide a kind of spiritual encouragement and support for me.

Declaration

I declare that this thesis represents my own original work, which I undertook at the University of York during 2018 - 2022, and I am the sole author. This work has not previously been presented included in a thesis submitted to this or any other institution for a degree, diploma or other qualifications.

Some parts of this thesis have been published or accepted in journals and conference proceedings; where items were published jointly with collaborators, the author of this thesis is responsible for the material presented here.

Conference Papers:

- Fei, Ye, and Adrian G. Bors. "Learning latent representations across multiple data domains using lifelong vaegan." In European Conference on Computer Vision (ECCV 2020), pp. 777-795. Springer, Cham, 2020.

- Fei, Ye, and Adrian G. Bors. Lifelong Generative Modelling Using Dynamic Expansion Graph Model. Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2022 **Oral**), pp. 8857-8865, 2022,

Journal Papers:

- F. Ye and A. G. Bors, "Dynamic Self-Supervised Teacher-Student Network Learning," in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022,

doi: 10.1109/TPAMI.2022.3220928.

Notations

Notations	Description
N	The total number of tasks in general continual learning.
N'	The total number of training times/steps for a data stream under task-free continual learning.
D_S^i	The i-th labelled training dataset.
D_D^i	The task label dataset, which represents the task information for $D^i{}_S$ from the i-th task.
D_{SU}^i	The i-th unlabelled training dataset.
D_T^i	The i-th labelled testing dataset.
$p(\mathbf{x}_{SU}^i)$	The data distribution formed by samples from the i-th training dataset D_{SU}^i.
$\{\mathbf{x}_i, \mathbf{y}_i\}$	An image with the associated with the class label.
$\mathbf{d}_i^\star$	The task label (one-hot vector) for $\mathbf{x}_i$.
$\mathbf{z}$	The continuous latent variable.
$\mathbf{u}$	The discrete latent variable, representing the class information.
$\mathbf{d}$	The discrete latent variable, representing the domain/task information.
$\mathbf{e}$	The discrete latent variable, representing expert identity information.
$\mathcal{X}$	The data space.
W	The dimension of a data sample.
L	The dimension of a latent variable $\mathbf{d}$.
Q	The dimension of a latent variable $\mathbf{u}$.
V	The dimension of a latent variable $\mathbf{e}$.
$\mathcal{Z}$	The space of the latent variable $\mathbf{z}$.
m	The size of a data batch.
K	The current number of components/experts in the model.
$\mathcal{G}_\theta$	The generator with the parameter θ.
$\mathcal{D}_\eta$	The discriminator with the parameter η.
$\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\}$	The labelled data batch.
$\mathbf{D}_{batch}$	The data batch of the task label.
D_G	The labelled dataset generated using the model.
D_{GU}	The unlabelled dataset generated using the model.
D_{GU}	The unlabelled dataset generated using the model.
$D_{GU}^{\widehat{s}}$	The unlabelled dataset produced by the $\widehat{s}$-th expert.
$D_G^{\widehat{s}}$	The labelled dataset produced by the $\widehat{s}$-th expert.
D_D^t	The task label set of the t-th task.
$\mathcal{A}_i$	The i-th component/expert in the mixture framework.
$\mathcal{A}$	A single model.
$\widehat{s}$	The index of the current selected component/expert in the mixture system.

Chapter 1

Introduction

Humans have an inherent ability to memorise, interpret and transfer knowledge across tasks, [4]. Lifelong/Continual learning represents the capability of people or animals of being able to continually acquire new skills or novel knowledge from a sequence of tasks while also maintaining their performance on previously learnt tasks [44]. When presented with a new task, humans would use their previously learnt experience in order to understand it. This ability is essential for adaptation and solving many real-world problems and would be very useful if it could be implemented in artificial systems in order to advance their capabilities. Artificial learning systems, able to learn new information from multiple sources while expanding their already assimilated cognitive abilities, would be able to solve multiple challenges [125]. However, lifelong learning remains a serious challenge for deep learning applications. While deep learning approaches perform well in many specific data classification applications [64], they suffer from the catastrophic forgetting problem [6, 136, 129, 163] when attempting to learn new tasks. This happens because a deep learning model, which had been trained initially on a specific database, loses that knowledge when it is trained for a new task on a novel data set. While deep learning was concerned mostly with language and image processing, the goal of lifelong learning is to process a wide variety of data. In this PhD thesis, we limit our study to learning databases of images.

Lifelong learning has attracted much attention in recent years because of its potential applications in many real-world systems such as autonomous learning agents and robots [125]. The natural property of lifelong learning is also relevant to many practical applications. For example, the bank fraud detection system should be able to continuously learn new cases to improve the system's robustness. Moreover, a robot should be able to acquire new knowledge in

a dynamically changing environment without forgetting it in order to make a reasonable decision when the environment changes. In addition, studying lifelong learning can bridge the gap between the computational model and the brain-like machine and promote the development of artificial intelligence research.

A lifelong learning problem can be defined by several individual tasks and a model, where each task is defined by sets of images from several incremental classes or from a single larger data set, which is maybe provided through a continuous data stream. During lifelong learning, only one set of training examples from the current task is available for training the model, while the model cannot access all previously seen tasks. Once training is complete, the model's effectiveness on all test examples is evaluated using a predefined performance metric such as Freschet Inception Distance (FID) [63]. This thesis also considers a more realistic setting for lifelong learning called Task-Free Continual Learning (TFCL) [7], which assumes there is no task information access during the training.

Prior research aiming to alleviate catastrophic forgetting was often focused on regularization [191], dynamic architectures [142] and memory-based methods [106]. For instance, regularisation-based approaches would generally impose a more significant penalty for changing the model parameters in order to relieve catastrophic forgetting [99]. However, these approaches do not work well when learning entirely different data sets. Dynamic architecture approaches would either freeze the weights for sections of the network or add new processing nodes when learning new tasks [129]. The drawback of these approaches is that they invariably require additional network structures, thus increasing the number of parameters requiring training for storing additional information. Other continual learning methods rely on a small memory buffer that stores some past samples [22] or uses a generator to reproduce previously learned samples [175]. The memory-based approaches have shown impressive results in continual learning but cannot handle an increasing number of tasks due to limited memory capacity.

On the other hand, most of the existing continual learning studies mainly focus on the classification task, where a classifier can continuously learn new classes [134]. Training deep generative models using lifelong learning has been recently investigated in several works [193, 2]. These studies have shown that generative models in lifelong learning can benefit many downstream tasks, including the image interpolation [2], disentangled representation learning [2] and image to image translation task [193]. These approaches typically use generative technologies such as the Variational Autoencoder (VAE) [85] and Generative Adversarial Net (GAN) [52]

to generate previously learned samples that are used to re-train the model to avoid forgetting. However, since the performance of these approaches depends heavily on the quality of the generative replay samples, they would suffer from catastrophic forgetting when learning a long series of tasks. In addition, when a single generative technology (GAN or VAE) is used for generative replay, it still suffers from the limitation of either lacking an inference mechanism (GAN) or producing blurred images (VAE). In this thesis, our research mainly focuses on training generative models under lifelong learning and aims to develop several methodologies to overcome the limitations of prior studies. Moreover, this research also investigates three different settings for lifelong learning : 1) learning a finite number of tasks; 2) learning an infinite number of tasks; 3) learning infinite data streams without knowing the task information; These settings represent the basis for developing practical applications in lifelong learning. Therefore, we implement these settings by developing several new models, briefly described in the following.

First, we address the first learning situation by proposing a new hybrid approach that combines the advantages of GANs and VAEs in a unified optimisation framework. Here, the GAN model is trained to produce high-quality generative replay patterns, while inference models are enforced to learn cross-domain latent representations for both past and current tasks using VAE losses. However, the proposed lifelong hybrid approach can only be applied to a few tasks and would suffer from serious forgetting problems when trying to learn an unlimited number of tasks. This inspires us to address the second learning setting by developing a lifelong learning framework, called the Dynamic Growing Mixture Model (GMM), which dynamically expands its network architecture to cope with an increasing number of tasks. To test whether the proposed GMM model is more advantageous than a single/static model in learning a long series of tasks, we construct a series of experiments on both supervised and unsupervised learning. The empirical results show that dynamically expanding the network architecture can significantly reduce forgetting compared to a single/static model.

Although GMMs achieve promising results in lifelong learning, they require a multi-head structure for performing the component selection during the testing phase. To address this issue, we develop a dynamic self-supervised Teacher-Student framework which can continually embed different data domains into a single latent space modelled by a lightweight student module. During the testing phase, the proposed model only requires the student module for the inference and discards the teacher module, accelerating the processing required for inference while also reducing the memory requirement.

The third learning context, called Task-Free Continual Learning (TFCL) [7] is a special case of continual learning in which the model can not access the task information and boundaries. To address this challenging learning paradigm, we develop a new teacher-student framework, where the teacher learns a dynamic expansion model as a parameterised memory to retain long-term information. We also introduce a short-term memory (STM) for temporarily storing the incoming information. To enable the model expansion, we develop a novel dynamic expansion criterion that evaluates the probabilistic distance between the STM and each previously learnt teacher component. Such a mechanism can avoid frequent expansion and ensure knowledge diversity among components during the training. Finally, to embed the stored knowledge into a single latent space, we propose to learn a VAE-based student module and a new knowledge distillation approach that transfers the teacher's and STM's knowledge into the student module.

We outline the remainder of this thesis as follows :

- **Chapter 2. Literature review :** We introduce the related works of general continuous learning and task-free continual learning. We also discuss the drawbacks of the existing lifelong learning approaches, which inspire us to propose several new lifelong learning models described in the following chapters.

- **Chapter 3. Lifelong learning with a hybrid approach :** We propose to learn a generative model under continual learning by taking the properties of GAN and VAE models that can learn cross-domain latent representations over time.

- **Chapter 4. Dynamic growing mixture model :** We develop the Dynamic Growing Mixture Model (GMM) model that can address the limitations of the static network architecture in learning an infinite number of tasks.

- **Chapter 5. Dynamic Self-Supervised Teacher-Student Network Learning :** Although GMMs can achieve promising results in lifelong generative modelling, they require a mechanism for performing the component selection at the testing phase, which leads to additional inference times. This chapter proposes a new lifelong learning framework based on the teacher-student structure, which aims to transfer the teacher's knowledge to a lightweight student module without forgetting. The proposed framework can significantly reduce inference times and maintain a lightweight student module during testing.

- **Chapter 6. Task-free continual learning using a teacher-student framework :** In this chapter, we extend the proposed teacher-student framework (chapter 5) for task-free continual learning and introduce a new dynamic expansion criterion that does not require accessing the task information during the training. The proposed framework can be applied to learning infinite data streams without forgetting.

- **Chapter 7. Conclusion and future works :** We provide a detailed discussion of the conclusion and hints about the future development of the research results from this thesis.

Chapter 2

Background and Literature Review

Although lifelong learning can be applied to a variety of applications such as natural language processing, text or speech, the research results presented in this thesis are focused on processing images. In this chapter, we first introduce several methods used in this thesis, including the Variational Autoencoder (VAE) and the Generative Adversarial Network (GAN). Then we provide an overview of related work on general continual learning, where the task label is always given during training. The lifelong approaches can be divided into four branches: regularisation-based methods, dynamic architectures, memory-based and knowledge distillation-based methods. Since task-free continual learning represents an important and realistic setting, we also provide an overview of related work on task-free continual learning in the remainder of this chapter.

2.1 Background

2.1.1 Generative Adversarial Nets (GANs)

GANs represent a promising approach for both unsupervised and semi-supervised learning [52]. The learning procedure of a GAN is presented in Figure 2.1. A GAN is made up of two different networks which are coupled: the Generator and the Discriminator. These two networks are separately trained while playing a Min-Max game where the Discriminator network is trained to receive both fake and real data samples while learning to distinguish them from each other. Meanwhile, the generator is trying to produce more realistic data, such as images, that would fool the Discriminator network. The loss for training GANs is defined as, [52]:

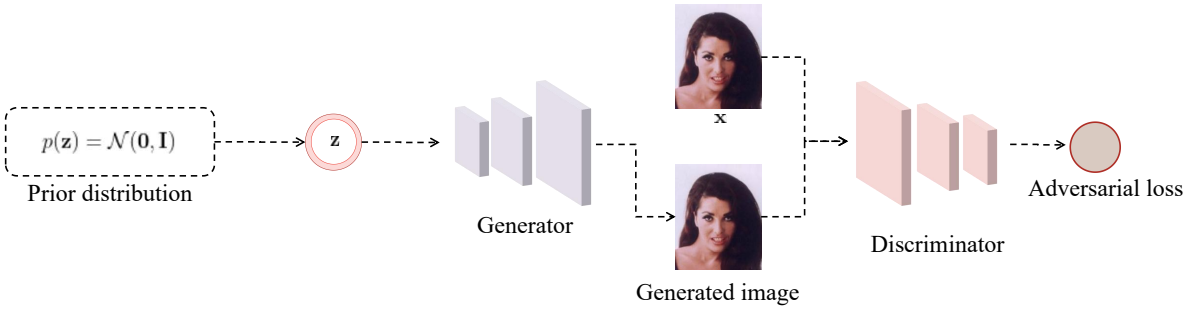


Figure 2.1: The learning procedure of GAN, which involves two components, the generator and the discriminator.

$$\mathcal{L}_{\mathcal{G}} = \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \left[\log \left(1 - \mathcal{D}_{\eta} \left(\mathcal{G}_{\theta} \left(\mathbf{z} \right) \right) \right) \right] , \tag{2.1}$$

$$\mathcal{L}_{\mathcal{D}} = -\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \left[\log \mathcal{D}_{\eta} \left(\mathbf{x} \right) \right] - \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \left[\log \left(1 - \mathcal{D}_{\eta} \left(\mathcal{G}_{\theta} \left(\mathbf{z} \right) \right) \right) \right] , \tag{2.2}$$

where $\mathbf{z}$ is a low-dimensional random noise vector sampled from a simple prior distribution $p\left(\mathbf{z}\right) = \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$ and $\mathbf{x}$ is a real training sample drawn from a data distribution [1] $p(\mathbf{x})$. $\mathcal{G}_{\theta}(\mathbf{z})$ is parameterized by θ and is implemented by a generator that receives a random vector $\mathbf{z}$ and returns an image $\mathbf{x}$. $\mathcal{D}_{\eta}(\mathbf{x})$ is parameterized by η and is implemented by a discriminator network which receives an image $\mathbf{x}$ and returns a scalar. In practice, the generator and discriminator can be implemented by using the deep CNN networks and the output of the discriminator in the original GAN [52] is confined to $[0, 1]$ by using the sigmoid function. Although the GAN model has shown impressive results in image generation, it still encounters two main drawbacks: (1) Mode collapse [156] happens when the generator tends to generate images of a limited number of categories; (2) The unstable learning [143] induces a strong generator and a weak discriminator, which can not find the optimal solution for the GAN and thus produce unrealistic image generation results.

2.1.2 Variational Autoencoder

The Variational Autoencoder (VAE) is a popular latent variable generative model, firstly proposed in [85]. The learning procedure of VAEs is shown in Figure 2.2 where a VAE is shown as having two module components, namely the encoder and decoder. Let $\mathbf{x}$ and $\mathbf{z}$ be the

[1] Following from the original GAN [52] works and other existing studies in generative modelling fields [11, 55], we adopt the data distribution to represent the probability distribution of data samples.

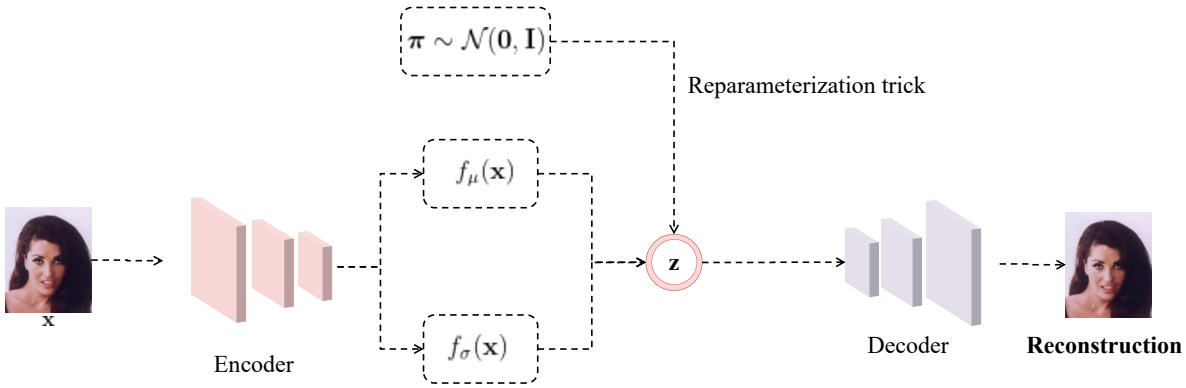


Figure 2.2: The learning procedure of VAE, where the encoder and decoder are used to model the encoding and decoding distributions, respectively.

observed and the latent variable, respectively. Let $p(\mathbf{x})$ represent a data distribution and $p_\theta(\mathbf{x} \,|\, \mathbf{z})$ be a decoding distribution which is usually implemented as the Gaussian distribution $p_\theta(\mathbf{x} \,|\, \mathbf{z}) = \mathcal{N}(f(\mathbf{z}; \theta), \sigma^2 \mathbf{I})$, where $f(\mathbf{z}; \theta)$ parameterized by θ is the decoder that receives a latent code $\mathbf{z}$ and its output is used as the mean vector of $\mathcal{N}(f(\mathbf{z}; \theta), \sigma^2 \mathbf{I})$. σ is the standard deviation which is fixed for the decoding process. The learning goal of a VAE aims to maximize the real sample log-likelihood, defined as:

$$\log \int p_\theta(\mathbf{x} \,|\, \mathbf{z}) p(\mathbf{z}) \, \mathrm{d}\mathbf{z} \,, \tag{2.3}$$

where $p(\mathbf{z})$ is the prior distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. To solve Eq. (2.3), the VAE framework derives a new objective function, described as follows.

In addition to the decoding distribution $p_\theta(\mathbf{x} \,|\, \mathbf{z})$, we can replace the true posterior distribution $p_\theta(\mathbf{z} \,|\, \mathbf{x})$ with an approximation distribution $q(\mathbf{z})$. We start by defining the Kullback-Leibler divergence between $p_\theta(\mathbf{z} \,|\, \mathbf{x})$ and $q(\mathbf{z})$, following from [34]:

$$D_{KL}[q(\mathbf{z}) \,||\, p_\theta(\mathbf{z} \,|\, \mathbf{x})] = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z})}[\log q(\mathbf{z}) - \log p_\theta(\mathbf{z} \,|\, \mathbf{x})] \,. \tag{2.4}$$

According to Bayes rule, we can rewrite Eq. (2.4) as:

$$D_{KL}[q(\mathbf{z}) \,||\, p_\theta(\mathbf{z} \,|\, \mathbf{x})] = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z})}[\log q(\mathbf{z}) - \log p_\theta(\mathbf{z} \,|\, \mathbf{x}) - \log p(\mathbf{z})] + \log p(\mathbf{x}) \,. \tag{2.5}$$

Then we rearrange Eq. (2.5), resulting in:

$$\log p(\mathbf{x}) - D_{KL}[q(\mathbf{z}) \,||\, p_\theta(\mathbf{z} \,|\, \mathbf{x})] = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}[\log p_\theta(\mathbf{x} \,|\, \mathbf{z})] - D_{KL}[q(\mathbf{z}) \,||\, p(\mathbf{z})]\,. \tag{2.6}$$

In practice, $q(\mathbf{z})$ is implemented by a parameterized distribution $q_\varsigma(\mathbf{z} \,|\, \mathbf{x}) = \mathcal{N}(f_\mu(\mathbf{x}), (f_\sigma(\mathbf{x}))^2\mathbf{I})$, where $f_\mu(\mathbf{x})$ and $f_\sigma(\mathbf{x})$ are the functions to return the mean and standard deviation vector, implemented by a neural network with the trainable parameters ς. Eq. (2.6) can be rewritten by replacing $q(\mathbf{z})$ by $q_\varsigma(\mathbf{z} \,|\, \mathbf{x})$:

$$\begin{aligned}\log p(\mathbf{x}) - D_{KL}[q_\varsigma(\mathbf{z} \,|\, \mathbf{x}) \,||\, p_\theta(\mathbf{z} \,|\, \mathbf{x})] = \mathbb{E}_{\mathbf{z}\sim q_\varsigma(\mathbf{z}\,|\,\mathbf{x})}[\log p_\theta(\mathbf{x} \,|\, \mathbf{z})] \\ - D_{KL}[q_\varsigma(\mathbf{z} \,|\, \mathbf{x}) \,||\, p(\mathbf{z})]\,.\end{aligned} \tag{2.7}$$

The Right-Hand-Side (RHS) of Eq. (2.7) is called the Evidence Lower Bound (ELBO) [85]. The VAE loss function is the minimization of the negative ELBO:

$$\mathcal{L}_{\text{VAE}} = -\mathbb{E}_{\mathbf{z}\sim q_\varsigma(\mathbf{z}\,|\,\mathbf{x})}[\log p_\theta(\mathbf{x} \,|\, \mathbf{z})] + D_{KL}[q_\varsigma(\mathbf{z} \,|\, \mathbf{x}) \,||\, p(\mathbf{z})]\,, \tag{2.8}$$

where the first term in RHS of Eq. (2.8) is the negative reconstruction error defined as:

$$\log p_\theta\left(\mathbf{x} \,|\, \mathbf{z}\right) = -\frac{1}{2\sigma^2}\|\mathbf{x} - f\left(\mathbf{z}; \theta\right)\|^2 - \frac{1}{2}W \log 2\pi\sigma^2\,, \tag{2.9}$$

where W is the dimension of $\mathbf{x}$. The second term in RHS of Eq. (2.8) is the KL divergence term. In the VAE optimization, one simple approach to enable the sampling procedure is by using the reparameterization trick [85] which firstly draws $\boldsymbol{\pi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then the latent variable $\mathbf{z}$ is calculated as $\mathbf{z} = f_\mu(\mathbf{x}) + f_\sigma(\mathbf{x}) \bigodot \boldsymbol{\pi}$, which allow the differentiable optimization process using Eq. (2.8), where $\bigodot$ is the element-wise product.

The VAE's performance can be improved by deriving a tight ELBO, using the Importance Weighted Autoencoder (IWELBO) [18] in which the tightness is controlled by the number of weighted samples considered. Other approaches focus on the choice of the approximate posterior distribution, including by using normalising flows [84, 137], implicit distributions [115] or using hierarchical variational inference [70]. The IWELBO bound can be used with any of these approaches to further improve their performance [151]. Additionally, online variational inference [119] has been used in VAEs, but requires storing the past samples for computing the

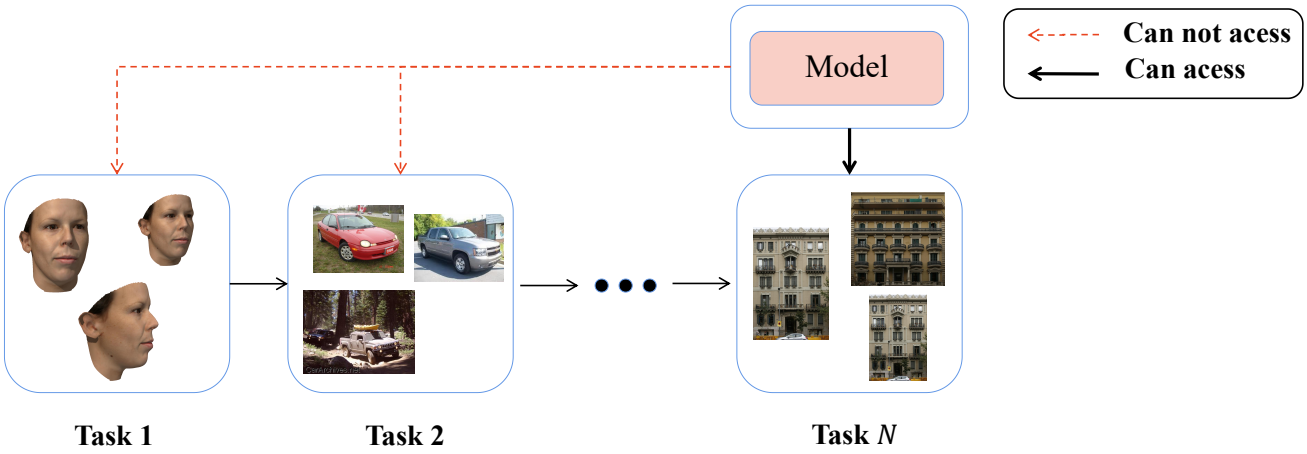


Figure 2.3: The procedure of general continual learning where each task is associated with a different data domain. During the N-th task learning, the model can access the training dataset of the N-th task while all previously learnt datasets are not available.

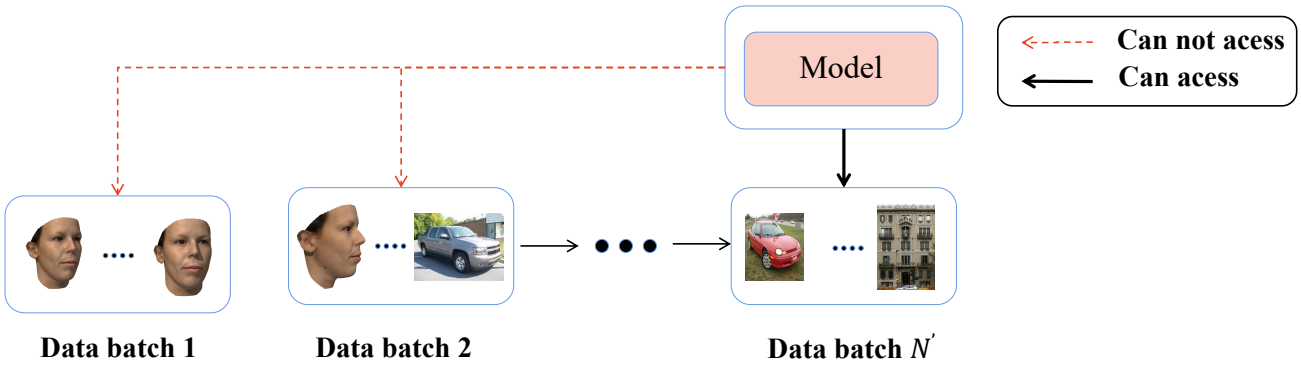


Figure 2.4: The procedure of task-free continual learning, where each data batch contains training samples from different data domains while the task information and boundaries are unavailable. During the N'-th training time/step, the model can only access the N'-th data batch while all previous data batches are not available.

approximate posterior, which is intractable when learning an infinite number of tasks.

2.1.3 Lifelong Learning and Its Settings

Continual/Lifelong learning represents a long-standing challenge for the machine learning approaches [125, 58]. Different from the traditional deep learning paradigm, which can access all training samples and aims to learn a single dataset, lifelong learning considers learning successively a sequence of tasks where each one is associated with a different dataset or with groups of samples from various data categories. During each task learning, the model can access the training samples from the current task while all previously learnt tasks are unavailable. Without accessing past samples, deep learning frameworks gradually lose their previously learnt knowledge, leading to catastrophic forgetting [125]. In this thesis, we study two main lifelong

learning scenarios, described as follows.

General continual learning is the most popular scenario in which the task information and label are given during the training [134]. For a given sequence of N tasks, each task (the t-th task) consists of a training set $D_S^t = \{\{\mathbf{x}_j, \mathbf{y}_j\}\}_{j=1}^{N_S^t}$ and testing set $D_T^t = \{\{\mathbf{x}_j, \mathbf{y}_j\}\}_{j=1}^{N_T^t}$, respectively, where $\mathbf{x}_j \in \mathcal{X}$ and $\mathbf{y}_j \in \mathcal{Y}$ are the data sample and its corresponding class label (one-hot vector), respectively. $\mathcal{X}$ and $\mathcal{Y}$ are the data and class label space, respectively. N_S^t and N_T^t represent the total number of samples for D_S^t and D_T^t, respectively. For unsupervised learning, each task (the t-th task) has an unlabelled training set[2] $D_{SU}^t = \{\mathbf{x}_j\}_{j=1}^{N_S^t}$ and testing set $D_{TU}^t = \{\mathbf{x}_j\}_{j=1}^{N_T^t}$, respectively. In the N-th task learning (Supervised learning), the model can only access the training samples from D_S^N, as shown in Fig. 2.3. Once the learning of all tasks is finished, the performance of the model is evaluated on all testing datasets $\{D_T^1, \cdots, D_T^N\}$.

Task-free continual learning is a more realistic scenario where the model sees only a small set of samples in a training period/step [7] without knowing task boundaries. Let $D_{TU}^i = \{\mathbf{x}_j\}_{j=1}^{N_i^T}$ and $D_{SU}^i = \{\mathbf{x}_j\}_{j=1}^{N_i^S}$ be the unlabelled test and training sets, for the i-th data domain/dataset, where N_i^T and N_i^S represent the number of samples for the test and training sets, respectively. In task-free continual learning, we create a joint dataset $\mathcal{S} = \{D_{SU}^1 \cup D_{SU}^2 \cup \cdots \cup D_{SU}^N\}$ as a data stream $\mathcal{S}$, where N is the total number of datasets. For a given data batch size m (We consider $m = 64$ in the experiments), the data stream can be divided into N' disjoint data batches $\{\mathbf{X}_{batch}^1, \mathbf{X}_{batch}^2, \cdots, \mathbf{X}_{batch}^{N'}\}$ in a batch-to-batch learning manner [7] and each data batch $\mathbf{X}_{batch}^t$ contains m samples. For a given set of N' training times/steps, each training step (the t-th step) is associated with a small batch of training samples $\mathbf{X}_{batch}^t := \{\mathbf{x}_j\}_{j=1}^m$. In the N'-th training period/step, the model can only access the data batch $\mathbf{X}_{batch}^{N'}$ while all previously learnt data batches $\{\mathbf{X}_{batch}^1, \cdots, \mathbf{X}_{batch}^{N'-1}\}$ are unavailable, as shown in Fig. 2.3. After all training steps are finished, the model's performance is evaluated on all testing samples. The primary difference between general continual learning and task-free continual learning is that TFCL does not access any task information and boundaries during training.

2.1.4 Disentangled Representation Learning

Learning a disentangled representation from data distributions is an important problem in computer vision. Generative models provide a general way to achieve this goal by learning a

[2]Since D_{SU}^t has the same dataset size with D_S^t, we employ N_S^t to denote the total number of samples for D_{SU}^t.

set of critical latent variables that can determine meaningful variations in visual spaces. The definition of disentangled representation is an open problem, but most studies consider it to be a data decomposition into sets of statistically and syntactically independent variables. In order to understand what is the disentangled representations, we show an example in Figure 2.5 (ref from [64]) where the face has meaningful variations as changing one single latent variable from -3 to 3 while fixing other variables. This shows that a single latent variable captures a specific meaningful variation in face images.

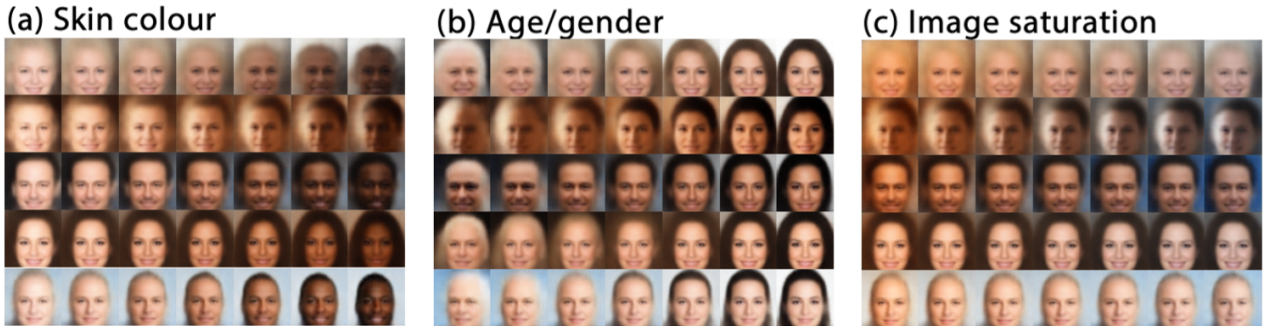


Figure 2.5: The disentangled representation results from [64].

In the following, we show how β-VAE [64] can learn disentangled representations by simply modifying the objective function of VAEs. The main change in the objective function is to set a large penalty factor $\beta > 1$ on the KL divergence term:

$$\mathcal{L}_{\text{BVAE}} = -\mathbb{E}_{q_\varsigma(\mathbf{z}\,|\,\mathbf{x})}\left[\log p_\theta\left(\mathbf{x}\,|\,\mathbf{z}\right)\right] + \beta D_{KL}\left[q_\varsigma\left(\mathbf{z}\,|\,\mathbf{x}\right)\,||\,p\left(\mathbf{z}\right)\right], \tag{2.10}$$

where β controls the balance between reconstruction fidelity and the degree of disengagements in latent variables [64]. To understand why increasing β can help disengagements, some works [112] [80] have decomposed the KL divergence term into two terms:

$$\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[D_{KL}\left(q_\varsigma\left(\mathbf{z}\,|\,\mathbf{x}\right)\,||\,p\left(\mathbf{z}\right)\right)\right] = I(\mathbf{x};\mathbf{z}) + D_{KL}\left[\tilde{q}\left(\mathbf{z}\right)||p\left(\mathbf{z}\right)\right], \tag{2.11}$$

where $\tilde{q}(\mathbf{x}) = \int_{\mathbf{x}} q_\varsigma(\mathbf{z}\,|\,\mathbf{x})p(\mathbf{x})\,\mathrm{d}\mathbf{x}$ is the aggregated posterior [112]. Minimising the last term can encourage disengagements in latent variables. However, minimising the above equation results in reducing the mutual information $I(\cdot;\cdot)$ between latent variables $\mathbf{z}$ and observed data $\mathbf{x}$. In practice, this can lead to an increasing reconstruction error during the training. Recently, many technologies have been developed to induce disengaged representation without sacrificing reconstruction quality. For instance, Burgess *et al.* [19] provided a deep analysis on the emergence of

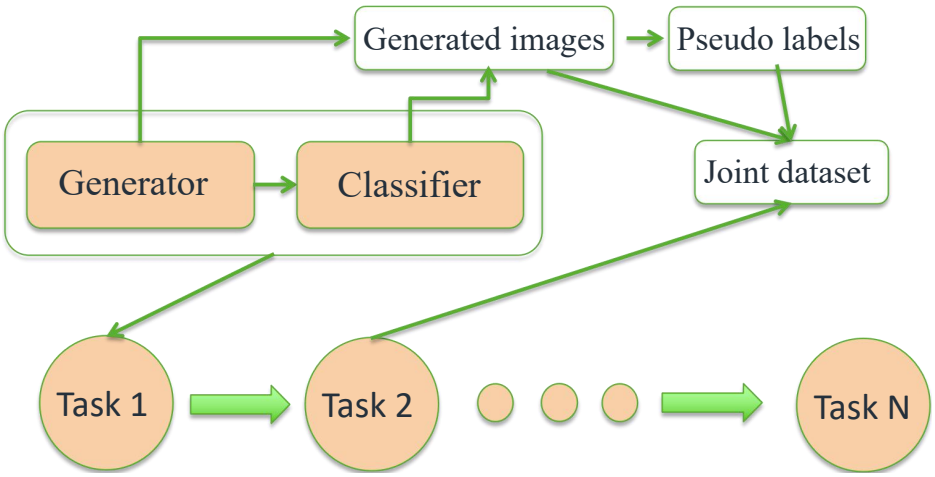


Figure 2.6: The learning procedure of GRM, which involves two components, the generator and the classifier.

disentangled representation in β-VAEs and proposed a new training procedure which progressively increases the penalty of the KL divergence term in order to balance disengagements and reconstruction accuracy. Gao *et al.* [48] proposed to use the multivariate mutual information, namely total correlation, to help find interpretable representations in VAEs. Kim *et al.* [80] proposed a VAE-based disentangled learning model, called FactorVAE, which encourages the distribution of latent representation to be independent and factorial over the dimensions by penalising the total correlation in the VAE objective function. A similar idea has been employed in [26] where the author demonstrated that the total correlation is the more important penalty in the VAE objective function for inducing disentangled representation.

Learning only continuous latent representations has limitations in the case of databases with complex images that contain discrete variations. Many research studies attempt to learn continuous and discrete representations to capture both the class and the continuously changing properties of data. Dupont *et al.* [41] proposed jointly learning continuous and discrete latent variables in VAEs to discover disentangled and interpretable representations. This joint VAE model introduces gradually increasing the penalty of two KL divergence terms in the VAE objective function to balance disengagements and reconstruction quality. The InfoGAN [27] is another kind of unsupervised disentangled learning method, which learns a subset of codes to capture meaningful representations. The InfoGAN derived a lower bound on the mutual information between latent codes and the generation process, treated as an additional term in the GAN objective function. In our research, we also explore the learning of the disentangled representations in the lifelong learning setting.

2.1.5 Generative Replay Mechanism for Lifelong Learning

Since this research focuses, among other research directions, on training generative models in lifelong learning, we give a brief overview of the generative replay mechanism (GRM). The idea behind the GRM is to train a deep generative model, such as a GAN or VAE, as a generative replay network that provides past examples to relieve forgetting. A GRM consists of two components: the generator and classifier, and its application to classification is shown in Fig. 2.6. When GRM sees a new task, it first generates the previously learned samples through the generation process, and then the classifier predicts the pseudo labels for each generated sample. Finally, these pseudo-samples are merged with new given samples to form a joint dataset which is used to train the model for the subsequent task learning. In this thesis, we show that GRM can be used for classification tasks and lifelong generative modelling.

2.2 General Continual Learning

Most of the existing works focus on general continual learning. They usually require task labels to define the task-specific component and loss function evaluation. In this section, we give an overview of related work on general continual learning.

2.2.1 Regularization Based Methods

Regularization approaches normally impose constraints on the objective function during training in order to alleviate catastrophic forgetting [191]. Changes in the neural network weights are penalized by considering a regularization term in the objective function, for instance. Kirkpatrick *et al.* [86] introduced the Elastic Weight Consolidation (EWC) algorithm, which encourages the weights of a neural network deemed significant to keep close their previous values when learning a new task. This approach proposes using a quadratic penalty on the difference between the parameters associated with the old and new tasks, which aims to minimize the change in the previously learnt parameters when learning a new task. Empirical results [77] have shown that EWC is good at the permutation tasks where each task is built using MNIST with a different random permutation of the image pixels. However, EWC performs worse when learning completely new categories incrementally. In addition, one disadvantage for EWC is that the Fisher matrix is kept for each task and is used in the loss function when learning a

new task. This can increase the computational complexity of learning a long sequence of tasks. This problem was addressed by [145], which only employs a single Fisher matrix to preserve the information of all previously learned tasks and therefore, the computational costs do not increase when the number of tasks grows.

More recently, regularization approaches have been developed based on the Bayesian Inference framework. For instance, Nguyen *et al.* [119] introduced a new continual learning framework, called Variational Continual Learning (VCL), which employs the Bayesian principle to overcome forgetting. Typically, VCL defines a projection operator at the t-th task learning [119]:

$$q_t(\epsilon) = \arg\min_{q \in \mathcal{Q}} D_{\mathrm{KL}}\big[q(\epsilon) \,||\, \frac{1}{Z_t} q_{t-1}(\epsilon) p(D_S^t \,|\, \epsilon)\big], t = 1, \cdots, N\,, \tag{2.12}$$

where ϵ is the model's parameter and D_{KL} is the Kullback-Leibler (KL) divergence. D_S^t is the dataset of the t-th task and $\frac{1}{Z_t}$ is the intractable normalizing constant. $q_t(\epsilon)$ is the tractable normalised approximation, and $\mathcal{Q}$ is a family of approximations. However, to solve Eq. (2.12) without accumulating errors, VCL requires storing a few samples for each task to calculate the task-specific approximate posterior, which is then used to regularise the updating of the model's parameters on a new task. VCL was shown to be successfully used in both the classifier and generative models, such as VAEs. Ahn *et al.* [3] analysed the drawbacks of VCL, including the computation time and space complexity and proposed a new solution, which introduces two additional regularisation terms that preserve old knowledge by freezing important parameters while allocating the remaining capacity to tackle a new task. Moreover, several works have proposed to regulate the learned representations that are robust to forgetting in continual learning. Ebrahimi *et al.* [43] introduced a new lifelong learning framework, which learns a disjoint representation for task-specific and task-invariant features. The main idea of this approach is to employ adversarial learning to induce task-invariant features that are combined with newly created task-specific features to learn new tasks. Khurram *et al.* [73] proposed a new approach that employs a deep Representation Learning Network (RLN) and a prediction learning network for continual learning. Specifically, the proposed approach employs the meta objective function [46] to update RLN that can be composed with the prediction learning network to make predictions for data samples. The empirical results show that the meta-learning can induce suitable representations that can improve the model's performance

in continual learning. However, these approaches require using significant computational costs, especially for learning a growing number of tasks [106], since they involve the inner iterative optimisation paradigm.

2.2.2 Dynamic Architectures

Dynamic architectures employ a flexible network architecture, which can be dynamically changed when learning new tasks. Resu *et al.* [142] proposed the Progressive Neural Network, which starts with a basic structure and increases its complexity when training with new information. In order to avoid catastrophic forgetting, this approach considers sub-networks for each newly learnt task, whose parameters are then frozen when learning new tasks. Aljundi *et al.* [6] proposed a mixture model, namely the Expert Gate for continual learning, where each expert is implemented as an autoencoder for learning a new task. Once all tasks have been learnt, the Expert Gate employs the reconstruction error as a component selection criterion at the testing phase. A component with the minimum reconstruction error is selected for the evaluation of the given task. The empirical results have shown that the Expert Gate can achieve impressive results when learning a long sequence of tasks. Wen *et al.* [174] introduced a new ensemble model for continual learning, called BatchEnsemble. This ensemble model consists of several components and each of these can have an independent weight matrix $\mathbf{W}_i$. However, the ensemble model would require many more parameters when learning a long sequence of tasks. To solve this issue, BatchEnsemble generates each weight matrix $\mathbf{W}_i$ by using a tuple of trainable vectors $\mathbf{r}_i$ and $\mathbf{s}_i$, expressed as:

$$\mathbf{W}_i = \mathbf{W} \circ \mathbf{F}_i, \mathbf{F}_i = \mathbf{r}_i \mathbf{s}_i^{\mathrm{T}}, \tag{2.13}$$

where $\mathbf{W}$ is the joint weight matrix which is frozen after the first task learning to avoid forgetting. $\circ$ is the element-wise multiplication operator. Eq. (2.13) is used to generate the specific network parameters for the i-th ensemble member. The previously learned information can be entirely preserved since each member has its own trainable parameters. However, the main disadvantage of using the BatchEnsemble for continual learning is that the number of tasks must be known before training, which is not applicable in a realistic scenario in which we usually do not know the number of tasks. In addition, each ensemble member in BatchEnsemble can only learn a unique task and can not learn several similar tasks, which leads to more parameters when

learning a long sequence of tasks. Yan *et al.* [179] proposed a new dynamic expansion model for class incremental learning. The main idea of this approach is to dynamically build a new feature extractor, which is combined with the previously learnt feature to form a super-feature extractor. Then a task-specific classifier is built on the top of this super-feature extractor. This approach can preserve the best performance on all previously learnt tasks since it only updates partial parameters when learning a new task. The recent study [38] has implemented the dynamic expansion model using an advanced neural network called Vision Transformer (ViT) [36] in which the task token θ_i is assigned to a certain task and a sub-classifier is built on the top the task-attention block for making predictions for a given sample. However, similar to BatchEnsemble, this approach does not reuse an existing component to learn several similar tasks. The total number of parameters will grow linearly as the number of tasks increases. In addition, most existing dynamic expansion models mentioned above [38, 179, 174, 142] can only be applied to the classification task, limiting their usability to a wide range of applications such as image reconstruction, image interpolation and disentangled representation learning.

2.2.3 Memory Based Approaches

Memory-based approaches can be generally divided into two categories: storing a few samples as a memory buffer and training a generator as a generative replay network. Firstly, we review the former.

Lopez-Paz *et al.* [106] introduced a general continual learning framework called the Gradient Episodic Memory (GEM) that employs a small memory buffer to store a subset of samples of each task. An important component in this framework is the task descriptors preserving the additional information for each task, such as the task label. During each task learning, the memory replays previously stored samples which are used to train the model to relieve forgetting. Chaudhry *et al.* [12] analysed GEM and found that GEM encounters a huge computational burden when the memory buffer size and the number of tasks are increased. Therefore, the author introduced an improved version of GEM, called Averaged GEM (A-GEM), which reduces the computational costs by introducing a new objective function that replaces $(t-1)$ constraints of GEM with a single constraint, where t is the number of tasks. A-GEM has been shown to reduce computational complexity and achieve better performance than GEM. More recently, Bang *et al.* [16] has shown that the diversity of samples in an episodic memory plays

an important role in the performance. The authors proposed a new diversity-aware sampling approach that promotes the diversity of samples by evaluating the classification uncertainty. Then a data augmentation approach is used to augment the memory to further encourage sample diversity and improve performance. Yoon *et al.* [188] introduced a simple yet effective sample selection approach that selectively stores the most representative and informative samples into the memory buffer, called the Online Coreset Selection (OCS). This approach utilises the gradient information to select samples that minimise the angle between the gradient vector of samples from the current task and the memory preserving all past samples. Sun *et al.* [157] investigated the memory selection from an information-theoretic perspective and proposed a stochastic information-theoretic reservoir sampler (InfoRS) that aims to store informative data points while filtering out uninformative ones. Specifically, InfoRS detects informative data points by calculating the negative log conditional probability $-\log p(\mathbf{y}_\star \,|\, \mathbf{Y}_\mathcal{M}; \mathbf{X}_\mathcal{M}, \mathbf{x}_\star)$ where $\{\mathbf{y}_\star, \mathbf{x}_\star\}$ is a pair of new training samples and $\{\mathbf{Y}_\mathcal{M}, \mathbf{X}_\mathcal{M}\}$ are memorized samples. Intuitively, if $\{\mathbf{y}_\star, \mathbf{x}_\star\}$ is obtained from a new task, $-\log p(\mathbf{y}_\star \,|\, \mathbf{y}_\mathcal{M}; \mathbf{X}_\mathcal{M}, \mathbf{x}_\star)$ will be large and $\{\mathbf{y}_\star, \mathbf{x}_\star\}$ will be added into the memory buffer to ensure sample diversity. Wang *et al.* [166] proposed employing a data compression method to store more memorised samples. The main idea of this approach is to use the JPEG algorithm [165] to compress the stored samples. In this way, the memory buffer can store a large number of compressed samples, which would preserve more information of previously learnt tasks. Empirical results indicate that this data compression technology can relieve forgetting of past tasks and improve performance. Madaan *et al.* [108] proposed a new approach for learning unsupervised representations under continual learning, which is based on the data augmentation approach called Mixup [195]. Typically, Mixup creates virtual training examples by performing the interpolations on a pair of training samples. In the case of continual learning, Mixup is used to create additional examples by interpolating between the samples uniformly obtained from the memory buffer and the current task, which is called Lifelong Unsupervised Mixup [108]. In addition, the memory-based approach can also be combined with the regularization methods to further enhance its performance, including the Bayesian inference [162], kernel function [32], gradient optimization [161, 102, 100, 167]. Although memory-based approaches provide promising results, using the additional storage space for learning is still necessary. This issue is addressed in [130], which introduced a memory recovery paradigm that synthesises samples from the learned classifier, called the transfer set, which is used for learning the next task to relieve forgetting. This approach can be used in a

CL scenario that does not require preserving past data samples.

In addition to using a memory buffer, recent works have explored the intrinsic properties of generative models for continual learning. Typical approaches for the generative replay are using Generative Adversarial Networks (GAN) [52] or Variational Autoencoders (VAE) [85] as generative models. Shin et al. [57], proposed a dual-model architecture consisting of a deep generative model and a classifier. This computational framework replays past knowledge by generating pseudo-data using generative models trained on previous tasks, while the learned classifier predicts the label for these generative replay samples. Then, this pseudo-dataset is used with the new samples to train the system on the next task learning. A similar idea is presented in [175], where the authors also use a GAN as a generative replay network that retains previously learned samples and reproduces them in the subsequent learning task to relive forgetting. This approach has been shown to improve both classification and generation tasks. Liu *et al.* [104] presented a new approach for incremental class learning. Unlike existing GRM methods that generate data samples in the image space, this approach introduces a feature replay mechanism that generates previously learned feature vectors to relieve forgetting. The advantage of such an approach is that it reduces memory requirements and speeds up the learning process. In addition to the generative replay mechanism, Cong *et al.* [29] extended the style transfer techniques [198, 127] to enable GAN to learn a sequence of tasks without forgetting. However, such an approach still requires adding new network parameters to adapt to a new task, which is not scalable for learning an infinite number of tasks. Moreover, GANs would suffer from mode collapse such that the model only captures a few modes of the true distribution [156]. Mode collapse in continual learning usually occurs when the model is trained on a long sequence of tasks where each task has a completely different data domain [181].

GAN-based GRM usually lacks inference mechanisms, and consequently, it cannot learn meaningful latent representations that support many downstream tasks, including image interpolation and reconstruction. By employing VAEs in lifelong learning, we can provide both generative replay and inference mechanisms, given that it consists of a decoder (generator) and an encoder. The first work using VAEs in lifelong learning was proposed in [2], which aims to learn disentangled representations under lifelong learning, called Variational Autoencoder with Shared Embeddings (VASE). To enable learning meaningful latent variables across multiple domains, VASE introduces a new loss function based on the Minimum Description Length (MDL) principle [64], which progressively increases the representational capacity to accommo-

date learning new data. The empirical results show that VASE can learn several disentangled representations across multiple data domains under lifelong learning. More recently. Ramapuram *et al.* [132] proposed the Lifelong Generative Modeling (LGM), which employs VAEs for two networks working in tandem: a teacher and a student module. The teacher network replays past knowledge through the generation process, as shown in Fig. 2.7, where the decoder of the teacher module receives a random vector sampled from a normal distribution and outputs the generation, while the student module continually learns knowledge from the data generated by the teacher module. At the subsequent task learning, the teacher and student exchange their roles in order to accumulate all previously learnt knowledge. The quality of the data generated from previously learnt knowledge in algorithms such as LGM [132] or VASE [2] depends on the generative abilities of VAEs, which are not that great, when compared to those of GANs, and usually result in blurred images. These models do not perform well in the case of complex data due to a rather poor replay of the knowledge from the previously learnt databases. Seff *et al.* [146] proposed the Augmented Generator objective function, based on a GAN, which is known as a better data generator than VAEs. Nevertheless, this model is applied on rather simple data. Kuzina *et al.* [89] introduced a lifelong learning approach for VAE, namely the Boosting Approach for continual learning of VAE (BooVAE), which learns the approximation of the aggregated posterior as a prior for each learned task. BooVAE employs the trainable pseudo-inputs as the parameters of the approximation of the posterior, and these pseudo-inputs preserve the past knowledge used to relieve forgetting. BooVAE has shown promising results in image generation tasks under lifelong learning. However, since BooVAE uses the static network architecture, it is suitable for learning several tasks for a single data domain and hardly dealing with a long sequence of different data domains.

2.2.4 Knowledge Distillation

Methods for transferring information from one model to another have been studied in recent years [62, 65]. This process is also called Knowledge Distillation (KD), where a classifier is trained on the predictions made by another classifier, [128]. A single classifier was trained using an ensemble of networks in order to achieve higher performance with fewer computations [9].

Knowledge distillation has recently been used to relieve forgetting in continual learning. Li

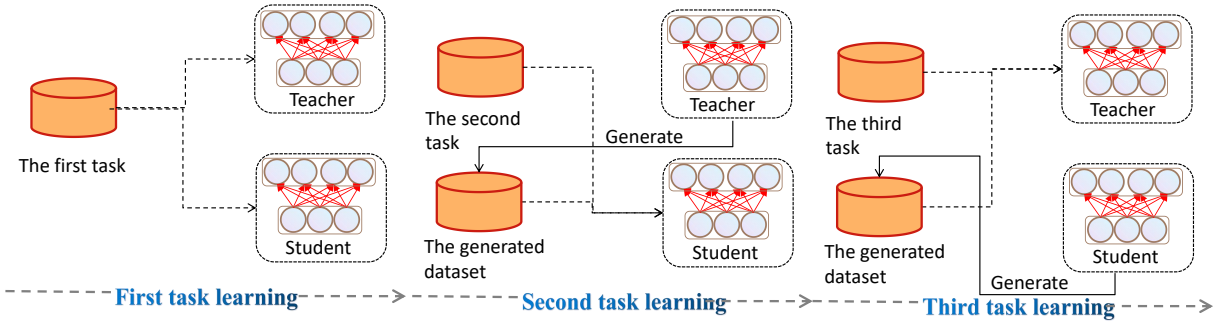


Figure 2.7: The learning process of LGM, which includes two components, the teacher and student module. Both teacher and student modules are implemented by VAEs. During the second task learning, the student module is fixed and provides the knowledge for the teacher's learning. In the next task learning, the teacher and student exchange their roles.

et al. [99] introduced a lifelong learning system, called Learning without Forgetting (LwF), which encourages the predictions for each data sample to be similar to the outputs from the original network by using Knowledge Distillation [65]. Zhai *et al.* [194] introduced a GAN-based learning framework for conditional image generation called Lifelong GAN. Unlike existing GAN-based GRM, which mainly focuses on the classification task, Lifelong GAN can be used for classification and conditional image generation tasks. The main idea of Lifelong GAN is to employ knowledge distillation that enforces the matching between the latent representations of the current task and the previously learnt task. However, this approach still requires to use the auxiliary data during the knowledge distillation process, which is intractable when learning an infinite number of tasks. Then Zhai *et al.* [192] proposed an efficient lifelong learning framework for conditional image generation called Piggyback GAN. Instead of using the generative replay or memory buffer for either generating or storing data, Piggyback GAN dynamically adds new network parameters for learning a new task while all previously learnt parameters are frozen to avoid forgetting. However, the total number of network parameters is also increased as learning more tasks. Buzzega *et al.* [20] introduced a simple and effective approach for continual learning using knowledge distillation, called the Dark Experience Replay (DER). This approach employs a memory buffer to store samples for past tasks and then minimize the distance on the network's output between the current task and memorized samples. DER was initially designed for general continual learning and can be extended for task-free continual learning by adapting reservoir sampling [164] that randomly selects samples from the data stream and then adds them into the memory buffer. Dhar *et al.* [33] introduced a new knowledge distillation approach for continual learning without memory buffers. This approach treats the previously

learnt model as the teacher and a newly initialized model as the student module. Then it generates attention maps for both the student and teacher module by using Grad-CAM [147] and introduces a loss term that minimizes the distance on the attention maps between the teacher and the student module. Fini *et al.* [45] introduced a two-stage processing algorithm to relieve catastrophic forgetting in continual learning. The first stage, called the warm-up stage, minimizes the classification loss over the new-task classifier using the incoming samples. The second stage, called the joint training, performs the knowledge distillation and the training with the new task, simultaneously. These approaches were also extended in [59] to enable incremental learning in the online scenario. Douillard *et al.* [37] formulated continual learning as representation learning and introduced a new knowledge distillation loss that minimizes the distance to the outputs of the final and intermediate layers between the trained teacher and a newly initialized student module. A similar idea was introduced in [149] to minimize the dissimilarity of the network's output between previous and current tasks from the low-dimensional subspace.

In addition to relieving forgetting, knowledge distillation-based methods have also been used to solve data imbalance problems in continual learning. Hou *et al.* [69] introduced to learn a unified classifier that treats both old and new tasks uniformly, while the knowledge distillation loss was used for relieving forgetting. Zhao *et al.* [196] found that optimising the knowledge distillation loss can not help the model treat old and new tasks fairly. To solve this problem, the author first trains a new model on both incoming and memorised samples using cross-entropy and knowledge distillation losses. Then a new approach, called Weight Aligning [196], is proposed to correct the model trained in the first stage.

Furthermore, knowledge distillation has also been used for transferring relevant knowledge when learning a new task. Ke *et al.* [76] employed a fully connected network or a CNN architecture as the knowledge base that preserves the knowledge from all learnt tasks. A knowledge distillation approach, called knowledge transfer attention [76], was proposed to selectively transfer useful information from the knowledge base when learning a new task, promoting the forward knowledge transfer. Such an idea was also used in [94] to transfer the knowledge to a subset of network layers selected by the Expectation-Maximization (EM) algorithm, which can be applied to other existing continual learning methods to further improve their performance.

2.3 Task-Free Continual Learning

Task-free continual learning represents a special form of lifelong learning in which access to task information is not required during the training. In this section, we summarize the current works of task-free continual learning into two branches: memory-based approaches and dynamic expansion models.

2.3.1 Memory Based Approaches

The Reservoir [164] was a simple but effective approach for continual learning, which shows impressive results compared to other continual learning methods [22]. The Reservoir can be seen as a popular baseline for task-free continual learning, which manages a memory buffer $\mathcal{M}$ with a fixed capacity. Although the Reservoir has several advantages, it still lacks an appropriate sample selection criterion that can selectively store informative data points into the memory buffer. Recently, several memory-based works have focused on designing an effective sample selection approach for the memory buffer. Aljundi *et al.* [7] proposed to employ a small memory buffer and a simple sample selection approach that maintains the samples with the highest loss among the new samples and the current memory buffer. The empirical results show that using a small memory buffer benefits task-free continual learning. The memory replay approach was then combined with the generative replay mechanism for training both the classifier and the Variational Autoencoders (VAEs) [85], resulting in a model called Maximal Interfered Retrieval (MIR), [5]. Specifically, MIR proposes a new retrieval mechanism which favours to store the samples with the highest loss value. Unlike existing work that focuses only on the classification task, MIR can be used for both the classification and image generation tasks. However, since the sample selection in MIR strongly depends on the loss value, MIR needs to use different selection criteria for the classification and generation tasks. The Gradient Sample Selection (GSS) [8] is another approach that treats sample selection as a constrained optimization reduction. More recently, a Learner-Evaluator framework was proposed for TFCL, called the Continual Prototype Evolution (CoPE) [30], which aims to store the same number of samples for each class in the memory in order to ensure the balance replay. CoPE consists of two components, the evaluator and the learner, enabling learning and evaluation at any time. The empirical results show that CoPE achieves impressive results, especially for learning highly imbalanced data streams. Jin *et al.* [75] proposed to modify the stored samples in

order to create more "challenging" examples for replay, called Gradient-based Memory EDiting (GMED). Unlike the other existing memory approaches that store real training samples, GMED directly edits the training samples via gradient updates and then stores them in a memory buffer. Moreover, GMED can be combined with any existing memory-based methods to further improve their performance.

2.3.2 Dynamic Expansion Model

Although memory-based approaches show promising results in task-free continual learning, learning an infinite number of data streams using those approaches is still challenging due to the fixed-length memory buffer. The other challenge for the memory-based approach is the negative backward transfer caused by some stored samples that interfere with updating the model using incoming samples [22]. A dynamic expansion model (DEM) can address these constraints from two aspects:1) DEM relieves the negative backward transfer by preserving previously learned knowledge into frozen components from a mixture system, in which the frozen components do not update their parameters when learning new samples and can thus maintain the optimal performance ; 2) DEM can achieve better generalisation performance under TFCL by allowing each component to model one or only a few similar underlying data distributions. The first work employing DEM for TFCL was proposed in [133] which introduces a new learning system for TFCL called the Continual Unsupervised Continual Learning (CURL). Unlike existing TFCL works that focus on the classification task, CURL studies a more sophisticated setting in continual learning where the task and class labels are unavailable. In order to adapt to changes in the data distribution during training, CURL dynamically builds new inference models to capture the information from the incoming samples. An expansion criterion evaluating the log-likelihood of the incoming samples is introduced in CURL to balance the complexity of the model and its generalisation performance. However, CURL has two drawbacks:1) CURL has only a single generator that continuously updates all its parameters, which leads to catastrophic forgetting when it learns a long sequence of data streams because the frequent updating of the generator leads to losing the information of the earlier learnt tasks; 2) CURL learns several inference models which benefit the clustering task and cannot handle many downstream tasks, including image interpolation across domains. Another DEM approach was proposed in [95] in which a new Neural Dirichlet process-based expansion mech-

anism is introduced for component expansion in a mixture system called the Continual Neural Dirichlet Process Mixture (CN-DPM). Unlike CURL, which expands its network architecture only for the inference models, CN-DPM expands the network architecture for both the generator and inference models. The advantage of CN-DPM over CURL is that CN-DPM does not use GRM to relieve forgetting and thus can preserve the complete information of the previously learnt samples. However, like CURL, CN-DPM does not embed the information of all tasks into a single latent space. Therefore, it is suitable for classification and image generation tasks and cannot perform image interpolation and disentangled representation tasks [181].

2.4 Conclusion

In this chapter, we discuss the research studies from the areas of general continual learning and task-free continual learning. The former assumes that the task label is given while there are no task labels for the second category of continual learning settings. To prevent forgetting, various methods are used, including memory-based, regularisation-based, dynamic architecture and knowledge distillation. Among these, the generative replay mechanism has the ability to generate data which is probabilistically consistent with the probabilistic representations of the tasks learnt in the past. By having a small buffer memory to store previous samples can also accomplish this task while requiring additional storage space. Moreover, the memory-based approaches are not scalable when learning a growing number of tasks. In this thesis, we explore the generative replay mechanism and memory buffer used for lifelong generative modelling.

Recently, significant attention has been paid to the dynamic extension model because of its scalability and generalisation performance. Thus, a static architecture model cannot handle an infinite number of tasks due to its limited model capacity. In addition, the static architecture model cannot guarantee the best performance for past tasks because it constantly updates the entire network parameters when it sees new samples. The Dynamic Extension Model (DEM) can overcome the limitations of a static architecture model through two mechanisms: 1) DEM freezes previously learned parameters to preserve the knowledge of past tasks; 2) DEM dynamically builds new components to adapt to new tasks;

Although existing approaches work well in general continual learning, applying these technologies in task-free continual learning is still challenging due to the absence of task information. Two approaches, the memory buffer and the dynamic expansion model have been shown to be

useful for task-free continual learning. However, the memory-based approach is suitable for processing a fixed-length data stream and would suffer from catastrophic forgetting when learning infinite data streams. This motivates us to combine the memory buffer and DEM to deal with real-world applications where we do not know precisely the task boundaries or the length of a data stream. Moreover, existing studies for DEM do not take into account the knowledge similarity when performing model expansion, which leads to non-optimal network architectures. In this thesis, we address this issue by developing a new dynamic expansion mechanism that ensures learning components charactering diverse knowledge in a mixture system. Furthermore, this thesis proposes a new expert pruning approach to further remove unimportant experts/components.

Chapter 3

Lifelong Learning Using VAEGAN

3.1 Introduction

In order to alleviate catastrophic forgotten, memory-based approaches introduce employing a memory buffer to store a small subset of previously seen data samples [5, 8, 22]. However, such an approach requires designing the criteria that would dynamically remove or add data samples in the buffer. Additionally, as the number of tasks increases, memory-based approaches would require large buffers, which is unsuitable in practical applications. Another solution, called the Generative Replay Mechanism (GRM) [57], consists of enabling a generator as a generative replay network for reproducing past samples when learning new tasks.

Many lifelong learning approaches consider using Generative Adversarial Networks (GANs) to implement the generative replay mechanism. Such an approach was first proposed in [57], where a classifier is considered in the GRM framework, learning from the samples associated with a new task while the generated samples are drawn from the outputs of the generator. However, such an approach requires generating a large number of samples after each task switch during lifelong learning, which would result in significant additional memory requirements. More recently, GRM-based methods have been combined with knowledge distillation for conditional image generation in a method called Lifelong GAN [193], which is built upon the BicycleGAN framework [199]. Lifelong GAN mainly focuses on the conditional image generation task while requiring to store past samples when learning new tasks, which is not applicable for learning an infinite number of tasks. Additionally, these approaches require designing a specific network architecture such as the classifier [57] and an encoding-decoding framework [193] to support learning many downstream tasks. Such methods are not able to learn meaningful

latent representations within a single latent space. This represents a challenge for many tasks, including image interpolation [123] or disentangled representation learning [80].

Learning meaningful and disentangled data representations has been shown to benefit many applications [182]. Recently, learning disentangled representations under lifelong learning was explored by introducing a framework based on the Variational Autoencoder (VAE), called VAE with Shared Embeddings (VASE) [2], which uses an environment-dependent mask to learn domain-specific latent representations. Additionally, VASE also uses the GRM to relieve forgetting. However, VAE-based GRM methods usually would yield blurred generative replay samples when compared with using GANs as GRM, leading to a degenerated performance on the past tasks. In this chapter, we develop the Lifelong Generative Adversarial Autoencoders (LGAA), representing a new approach to lifelong learning which combines the advantages of both GAN and VAEs models. We propose to train a robust generative replay network by using adversarial learning while also training the inference models on the joint data samples corresponding to the new task and the generated samples by the generator through a new objective function. The trained inference models can be used in a variety of applications, such as classification, image interpolation and learning disentangled representations. The advantage of the proposed LGAA over existing GRM-based methods is that LGAA can learn a robust generative replay network compared to VAE-based methods, while it can also capture meaningful latent representations across domains when compared to GAN-based approaches.

The rest of this chapter consists of Section 3.2, which introduces the proposed Lifelong Generative Adversarial Autoencoder and Section 3.3, where we provide the training algorithm of the proposed model. Finally, Section 3.4 contains the experimental results and their discussion, while the conclusions of this chapter are drawn in Section 3.5.

3.2 Lifelong Generative Adversarial Autoencoder

The quality of the generated samples is the key to the performance of the GRM-based models during lifelong learning [181]. Additionally, most GRM-based approaches do not have an inference model to use in training [57], which prevents them from extracting meaningful representations for downstream tasks. This inspires us to propose a novel lifelong learning approach, Lifelong Generative Adversarial Autoencoder (LGAA), which not only learns a robust generative replay network but also trains accurate inference models for representation learning.

The key idea of the proposed LGAA is to combine the advantages of GAN and VAE into a unified framework, in which GAN learning enables to produce high-quality generative replay samples while VAE learning encourages representation learning. To implement this goal, we propose to design GAN and VAE models, respectively, and then combine them into a unified framework. Firstly, we formulate a GAN generator as a generative replay network in supervised learning, which receives two additional hidden variables $\mathbf{d}$ and $\mathbf{u}$ to represent the unknown class and task information. Secondly, we introduce a VAE framework with the same hidden variables that are modelled by additional appropriate distributions approximated using the inference models. Using two models to implement lifelong generative modelling can have two main advantages : (1) It learns additional inference models when compared to the GAN-based methods, enabling to implement many downstream tasks, including image reconstruction, image interpolation and classification tasks; (2) It uses the GAN loss to improve the quality of generative replay samples, benefiting from relieving network forgetting when compared to the VAE-based methods that usually produce blurry generative replay samples; One simple approach for training such two models in a unified framework is to combine GAN and VAE losses into a simple objective function. However, such an approach has two main weaknesses : (1) It requires designing a hyper-parameter to balance the importance of GAN and VAE losses, making the model more complex; (2) It requires performing several runs in order to select an appropriate hyper-parameter, which is time-consuming. In this chapter, we introduce an alternative optimization approach that formulates GAN and VAE losses into a two-step updating procedure. Specifically, we update the generator using the GAN loss in the first step to provide good initial parameters for the generator. Then the second step updates the generator and inference models using the VAE loss, aiming to learn meaningful latent representations for the data. Such a design does not require the additional hyper-parameter selection process and is easy to implement.

3.2.1 Problem Formulation

Let $\mathcal{X}$ and $\mathcal{Z}$ represent the data and latent variable space, respectively. For a given sequence of N tasks, each task has a training set $D_S^t = \{\{\mathbf{x}_j, \mathbf{y}_j\}\}_{j=1}^{N_S^t}$ and a testing set $D_T^t = \{\{\mathbf{x}_j, \mathbf{y}_j\}\}_{j=1}^{N_T^t}$, respectively. $\mathbf{x}_j \in \mathcal{X}$ and $\mathbf{y}_j \in \mathcal{Y}$ (one-hot vector) are an observed variable, and the target variable (class label), where $\mathcal{Y}$ is the space of the class label. N_S^t and N_T^t are the total number

of samples for D_S^t and D_T^t, respectively. In unsupervised learning where we do not have class labels, each task (the t-th task) is assigned with an unlabelled training set[1] $D_{SU}^t = \{\mathbf{x}_j\}_{j=1}^{N_S^t}$ and testing set $D_{TU}^t = \{\mathbf{x}_j\}_{j=1}^{N_T^t}$, respectively. Since this chapter focuses on task-aware continual learning, we also define a task label dataset[2] $D_D^t = \{\mathbf{d}_j^\star\}_{j=1}^{N_S^t}$ to represent the task information for D_S^t and D_{SU}^t. Specifically, D_D^t has the same dataset size with D_S^t and each task label $\mathbf{d}_j^\star$ (one-hot vector) represents the task information for the j-th labelled sample $\{\mathbf{x}_j, \mathbf{y}_j\}$ and unlabelled sample $\mathbf{x}_j$ in D_S^t and D_{SU}^t, respectively. Our learning goal is to learn an inference model which can accumulate the representation information from novel concepts without forgetting previously learnt knowledge. This can allow us to perform many downstream tasks, such as classification, reconstruction and interpolation tasks, by using the inference model.

3.2.2 Training A Robust Generative Replay Network

We consider that the underlying information of observed data samples is defined by three latent variables $\{\mathbf{u}, \mathbf{d}, \mathbf{z}\}$ where $\mathbf{d} = \{d_i \,|\, i = 1, \ldots, L\}$ is the discrete variable (one-hot vector) representing the domain information, where L is the number of domains. Each dimension d_i in the vector $\mathbf{d}$ is either 0 or 1, and the summation of all dimensions of $\mathbf{d}$ is equal to 1. Meanwhile, similarly, $\mathbf{u} = \{u_i \,|\, i = 1, \ldots, Q\}$ is the Q-dimensional one-hot vector, which represents the class information. $\mathbf{z}$ is the continuous variable with 256 dimensions, which is drawn from a Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The generation process is defined by:

$$
\begin{aligned}
&\mathbf{d} = f_{\text{OneHot}}(d, L), d \sim \text{Cat}\left(L, p_1^{\mathbf{d}}, \cdots, p_L^{\mathbf{d}}\right) , \\
&\mathbf{u} = f_{\text{OneHot}}(u, Q), u \sim \text{Cat}\left(Q, p_1^{\mathbf{u}}, \cdots, p_Q^{\mathbf{u}}\right) , \\
&\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) , \\
&\mathbf{x}' = \mathcal{G}_\theta(\mathbf{z}, \mathbf{d}, \mathbf{u}) ,
\end{aligned}
\tag{3.1}
$$

where $\text{Cat}(\cdot)$ is the Categorical distribution and $p_j^{\mathbf{d}} = 1/L, j = 1, \cdots, L$ is the probability for each outcome. Each $p_j^{\mathbf{u}} = 1/Q, j = 1, \cdots, Q$ has the same probability. $f_{\text{OneHot}}(d, L)$ is a function that transfers the category variable d to the L-dimensional one-hot vector $\mathbf{d}$ where the d-th vector's entry is 1, and all others are 0. $\mathbf{x}'$ is the image drawn from the

[1] Since D_{SU}^t has the same dataset size with D_S^t, we employ N_S^t to denote the total number of samples for D_{SU}^t.

[2] We employ $\mathbf{d}_j^\star$ to denote a label (one-hot vector) that represents the task information for the j-th labelled $(\mathbf{x}_j, \mathbf{y}_j)$ and unlabelled training sample $\mathbf{x}_j$ in D_S^t and D_{SU}^t, respectively.

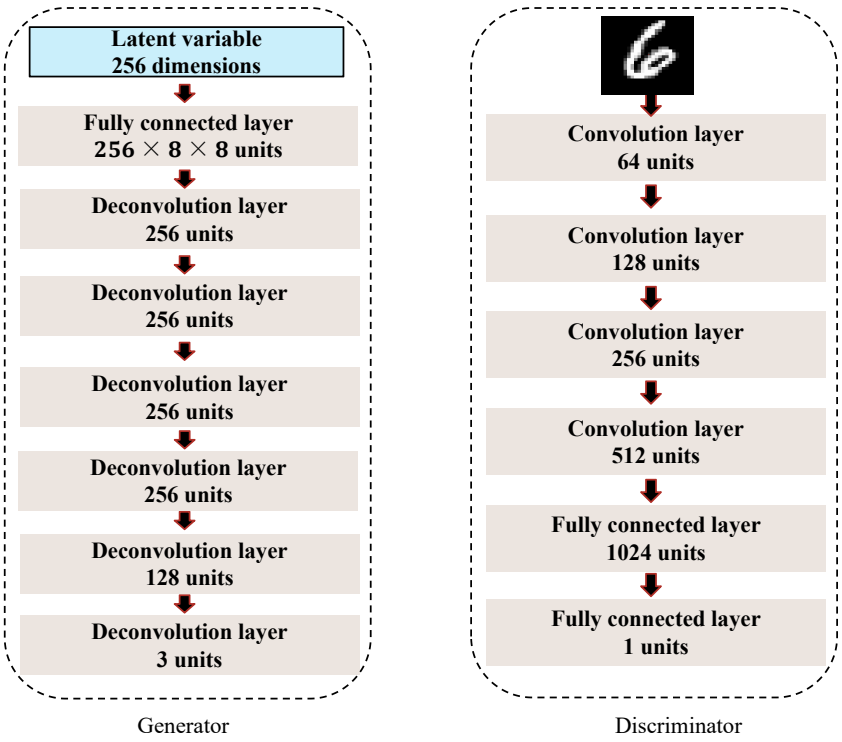


Figure 3.1: The detailed network architecture for the generator and discriminator.

generator distribution, modelled by the generator $\mathcal{G}_\theta(\mathbf{z}, \mathbf{d}, \mathbf{u})$ implemented by a neural network with trainable parameters, θ. In order to train a powerful generative replay network, we use the Wasserstein GAN (WGAN) [11] loss with the gradient penalty [55], which consists of a generator $\mathcal{G}_\theta$ and a discriminator network $\mathcal{D}_\eta$ with the parameter set η. The main goal of the discriminator network in the GAN training paradigm is to distinguish fake images (generated images) from real ones [52]. In the WGAN-GP framework, the final layer of the discriminator network outputs a single scalar and does not use any activation functions to restrict its outputs [11]. The loss functions used for WGAN-GP are defined by [55]:

$$\mathcal{L}_{\mathcal{G}^{\text{LGAA}}}(\theta) = \frac{1}{m}\sum_{i=1}^{m}\Big\{ -\mathcal{D}_\eta(\mathcal{G}_\theta(\mathbf{u}_i, \mathbf{z}_i, \mathbf{d}_i))\Big\}. \tag{3.2}$$

$$\mathcal{L}_{\mathcal{D}^{\text{LGAA}}}(\mathbf{X}_{batch}, \eta) = \frac{1}{m}\sum_{i=1}^{m}\Big\{\mathcal{D}_\eta(\mathcal{G}_\theta(\mathbf{u}_i, \mathbf{z}_i, \mathbf{d}_i)) - \mathcal{D}_\eta(\mathbf{x}_i) + \lambda(\|\nabla_{\tilde{\mathbf{x}}_i}\mathcal{D}_\eta(\tilde{\mathbf{x}}_i)\|_2 - 1)^2\Big\}, \tag{3.3}$$

where $\lambda = 10$ is considered in our all experiments according to [55]. $\mathcal{L}_{\mathcal{G}^{\text{LGAA}}}$ and -$\mathcal{L}_{\mathcal{D}^{\text{LGAA}}}$ are the loss functions for the generator and discriminator, respectively. In each training iteration of the mini-batch learning [92], we obtain a real data batch $\mathbf{X}_{batch} = \{\mathbf{x}_1, \mathbf{x}_1, \cdots, \mathbf{x}_m\}$ from the

training dataset and m is the batch size in the mini-batch learning [92] (during the experiments we considered the batch size as $m = 64$). In addition, the generator $\mathcal{G}_\theta(\mathbf{u}_i, \mathbf{z}_i, \mathbf{d}_i)$ in each training iteration produces a fake image data batch $\mathbf{X}'_{batch} = \{\mathbf{x}'_1, \mathbf{x}'_2, \cdots, \mathbf{x}'_m\}$ in the mini-batch learning where $\mathbf{x}_i$ and $\mathbf{x}'_i$ are the i-th sample from $\mathbf{X}_{batch}$ and $\mathbf{X}'_{batch}$, respectively. $\mathbf{u}_i$ and $\mathbf{d}_i$ in the generation process are given by the class and task labels associated with $\mathbf{x}_i$ and $\mathbf{z}_i$ is drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. $\tilde{\mathbf{x}}_i$ is the interpolated image produced by $\tilde{\mathbf{x}}_i = s\mathbf{x}_i + (1-s)\mathbf{x}'_i$ where s is drawn from a uniform distribution $U(0, 1)$. $\mathcal{D}_\eta$ represents the discriminator that receives an image $\mathbf{x}_i$ and returns a scalar, implemented by a neural network with trainable parameters η. We provide the detailed network architecture for the generator and discriminator in Fig. 3.1. The third term from the right-hand side of Eq. (3.3) is the gradient weighted by the penalty λ [55]. This penalty term aims to enforce a Lipschitz constraint on the discriminator. WGAN [11] implements the Lipschitz constraint by considering weight clipping, which can lead to undesired convergence behaviour during the training [55]. This issue is addressed by WGAN-Gradient Penalty (GP) [55], which introduces a penalty term that penalizes the norm of the gradient of the discriminator with respect to its input, leading to better performance while maintaining a stable learning procedure. The main idea of the WGAN-GP regularisation term is inspired by Corollary 1 from [55], which demonstrates that the optimal WGAN discriminator has unit gradient norm almost everywhere under the data distribution and the generator distribution.

If the class labels and task/domain labels are available, $\mathbf{u}_i$ and $\mathbf{d}_i$ in Eq. (3.2) and Eq. (3.3) can be obtained from the training dataset; otherwise, they are obtained from Eq. (3.1). The adversarial loss allows the generator and discriminator to be trained alternately such that the discriminator aims to distinguish real from generated data, while the generator tends to fool the discriminator by generating realistic data [52, 11].

3.2.3 The Inference Mechanism of LGAA

Most GAN-based lifelong methods [57, 176, 193] do not learn an accurate inference model and, therefore can not derive a meaningful data representation. For the model proposed in this chapter, we consider three differentiable non-linear functions $F_\varsigma(\cdot)$, $F_\varepsilon(\cdot)$, $F_\delta(\cdot)$, implemented by three encoders. These encoders aim to infer three different types of latent variables $\{\mathbf{z}, \mathbf{d}, \mathbf{u}\}$, as indicated in Section 3.2.2 and shown in Fig. 3.2. We implement $F_\varsigma(\cdot)$ considering the underlying Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$, where $\{\boldsymbol{\mu} = F^{\boldsymbol{\mu}}_\varsigma(\mathbf{x}), \boldsymbol{\sigma} = F^{\boldsymbol{\sigma}}_\varsigma(\mathbf{x})\}$ are the hyperparameters of

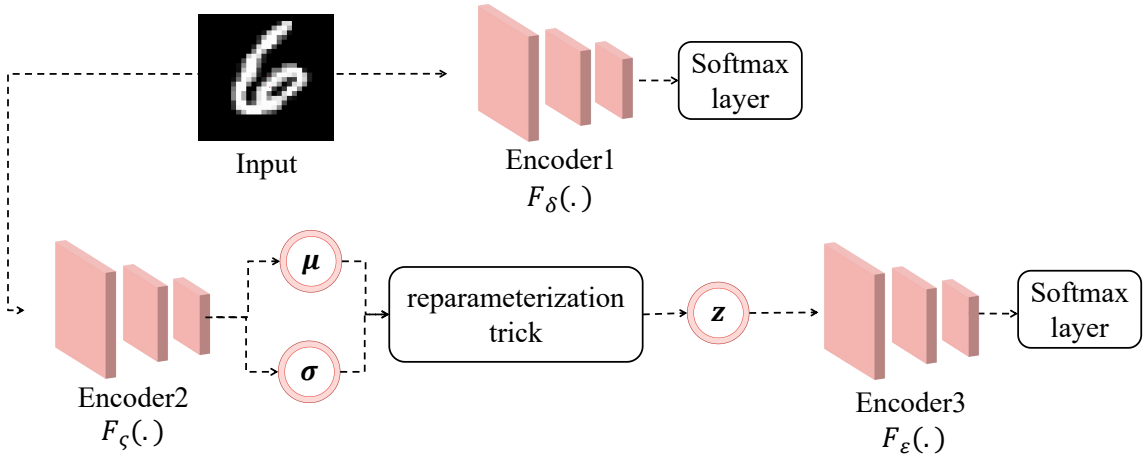


Figure 3.2: The structure of three differentiable non-linear functions $F_\varsigma(\cdot)$, $F_\varepsilon(\cdot)$, $F_\delta(\cdot)$ implemented by three encoders.

the Gaussian distribution, returned by $F_\varsigma(\mathbf{x})$, where $F_\varsigma^{\boldsymbol{\mu}}(.)$ and $F_\varsigma^{\boldsymbol{\sigma}}(.)$ denote the output for the single hyperparameter vector. We use the reparameterization trick [85, 138] for sampling $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\pi} \bigodot \boldsymbol{\sigma}$, where $\boldsymbol{\pi}$ is a random noise vector sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\bigodot$ is the element-wise product, in order to ensure end-to-end training. When designing $F_\varepsilon(\cdot)$ and $F_\delta(\cdot)$, we usually require implementing two deep convolutional neural networks with a considerable number of layers to process high-dimensional data, which leads to more parameters and computational costs. In this thesis, we desire to reduce the number of parameters by designing a lightweight neural network. One reasonable choice is to take a low-dimensional feature vector of the data as an input and process this representation using a simple neural network. However, since the encoder $F_\delta(\cdot)$ aims to make predictions on the class labels across different tasks, using the original image $\mathbf{x}$ as an input can improve the performance of the classification task because $\mathbf{x}$ contains additional category information than the low-dimensional variable $\mathbf{z}$. As a result, we only consider implementing $F_\varepsilon(\cdot)$ using a simple fully connected network that takes $\mathbf{z}$ as the input, where the softmax function implements the last layer's activation function that returns a set of probabilities $\{d'_1, \cdots, d'_L\}$. By considering the performance of the classification task, we implement $F_\delta(\cdot)$ using a deep neural network with a softmax layer that returns a set of probabilities $\{u'_1, \cdots, u'_Q\}$. The detailed network architecture for the inference models is shown in Fig. 3.3.

One can infer the discrete latent variables $\mathbf{d}$ and $\mathbf{u}$ is by transferring the probability outputs $\{d'_1, \cdots, d'_L\}$ and $\{u'_1, \cdots, u'_Q\}$ of $F_\varepsilon(\cdot)$ and $F_\delta(\cdot)$ to one-hot vectors. However, such one-hot vectors can not be used in the training process because we can not use backpropagation to learn $F_\varepsilon(\cdot)$ and $F_\delta(\cdot)$ [41]. In order to mitigate this, we use the Gumbel-Max trick [56] for

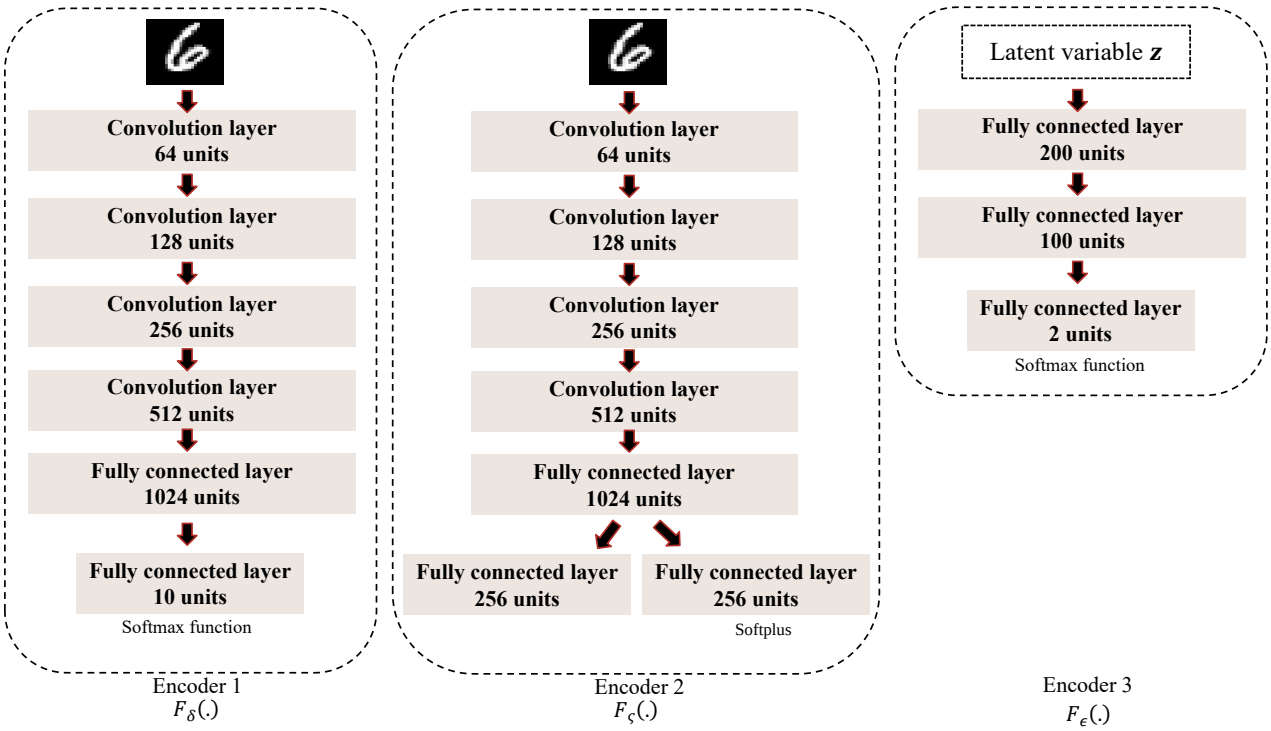


Figure 3.3: The detailed network architecture for the inference models. The "Encoder 2" has two output layers, and we employ the "Softplus" activation function to ensure the non-negativity of the hyperparameter $\boldsymbol{\sigma}$. The final layer in "Encoder 3" gives two probability outputs when the number of tasks is two. If we know the number of tasks, we need to redesign the final layer in "Encoder 3" such that the number of probability outputs matches the number of tasks. Such a limitation is discussed in Section 3.5 at the end of this chapter.

achieving the differentiable relaxation of discrete random variables. The Gumbel-Max trick was firstly used in [72] to define a continuous distribution called the Gumbel-Softmax distribution, which can sample continuous relaxations of one-hot vectors. Such a distribution was independently considered in [109], and is also called as the Concrete distribution. The sampling procedure of the discrete variable from a categorical distribution is not differentiable [72]. The Gumbel-Softmax distribution solves this drawback by using the softmax function as a differentiable approximation to the non-differentiable arg-max while employing samples drawn from the standard Gumbel distribution as independent noise in the sampling process. Thanks to the differentiable property, the Gumbel-Softmax distribution has been successfully applied to various deep generative models. Wang et al. [168] introduced using the Gumbel-Softmax distribution to draw the continuous relaxation of the model prediction, where each class probability is estimated using a classifier $q(\mathbf{y} \mid \mathbf{x})$. This approach was used in the generative model's training process and has been empirically found to reduce the variance of gradients [168]. Dupont et al. [41] proposed learning disentangled joint continuous and discrete representations for a VAE framework, in which the discrete variable is sampled through the Gumbel-Softmax distribution.

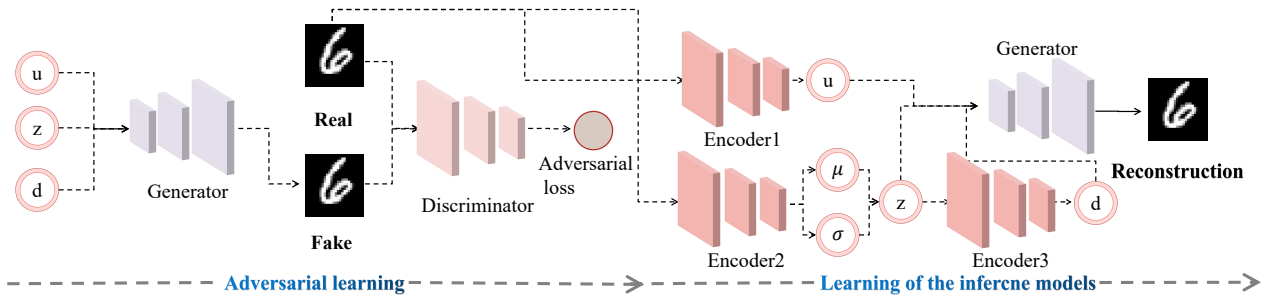


Figure 3.4: The network structure of the proposed LGAA model. The whole learning procedure is divided into two steps. In the first step, we draw random vectors $\{\mathbf{u}, \mathbf{z}, \mathbf{d}\}$ from the prior distributions and then consider them as input for the generator for producing the fake image. The adversarial loss, defined by Eq. (3.2), is used for both the generator and discriminator. In the second step, the objective function Eq. (3.14), is used to update the inference and generator models.

However, these approaches only consider learning a single data domain/set and can not learn a sequence of tasks. In this thesis, we explore the Gumbel-Softmax distribution for generating the continuous relaxation of the discrete variables in the context of lifelong learning.

The sampling process of discrete latent variables using the Gumbel-Max trick is defined as follows [72]:

$$\hat{d}_j = \frac{\exp((\log d'_j + g_j)/T)}{\sum_{i=1}^{L} \exp((\log d'_i + g_i)/T)}, \tag{3.4}$$

where d'_i is the i-th entry of the probability defined by the softmax layer characterizing $F_\varepsilon(\cdot)$ and $\hat{d}_j$ is the continuous relaxation of the j-th dimension of the variable $\mathbf{d}$, while g_i is sampled from the distribution Gumbel$(0, 1)$ and T is the temperature parameter that controls the degree of smoothness. A small T indicates that the variable $\mathbf{d}$ sampled using Eq. (3.4) is close to the one-hot vector. In contrast, a large T indicates that the process of sampling $\mathbf{d}$ is similar to that of sampling from a uniform distribution [72]. In our experiments, we set $T = 0.5$ to encourage the sampling as close as possible to that of the one-hot representation. We use the Gumbel softmax distribution for sampling both domain $\mathbf{d}$ and discrete $\mathbf{u}$ variables.

3.2.4 The Objective Function for the Inference Models

GANs lack an inference mechanism and use a random number generator for the generation process, preventing them from capturing data representations properly. In this chapter, we aim to derive an objective function to train the inference models (encoders). The VAE model [85] is one of the most popular representation learning frameworks, which consists of a decoder and an inference model. Despite the fact that VAE models have been successfully applied to image

generation, reconstruction and interpolation tasks [18, 84, 137, 70], most of them only learn a single latent representation $\mathbf{z}$ which can not capture the discrete information such as the class label. Recently, learning both continuous and discrete representations in a VAE framework was investigated in [41]. However, the model proposed in [41] can only learn a single and static data domain and would suffer from catastrophic forgetting when learning a sequence of tasks. Unlike the existing approaches discussed above, we aim to learn inference models that can infer latent representations across different data domains without forgetting.

Due to the robust inference mechanism, the VAE model is suitable for representation learning, and its objective function is defined as [85]:

$$\mathcal{L}_{\text{VAE}}^{\text{LGAA}}(\mathbf{x}, \theta, \varsigma) = -\mathbb{E}_{\mathbf{z}\sim q_\varsigma(\mathbf{z}\,|\,\mathbf{x})}[\log p_\theta(\mathbf{x}\,|\,\mathbf{z})] + D_{KL}[q_\varsigma(\mathbf{z}\,|\,\mathbf{x})\,||\,p(\mathbf{z})]\,, \tag{3.5}$$

where $q_\varsigma(\mathbf{z}\,|\,\mathbf{x})$ and $p_\theta(\mathbf{z})$ are the variational and prior distributions (Gaussian), respectively. $p_\theta(\mathbf{x}\,|\,\mathbf{z})$ is the decoder used to reconstruct images. However, Eq. (3.5) can only allow the VAE model to capture the continuous latent representations, which can not be used in our model since we have three different latent variables $\mathbf{z}, \mathbf{u}, \mathbf{d}$. Inspired by [41], which derives an objective function to regulate both continuous and discrete representations, we consider learning three latent variables within a VAE framework [85]. Before we derive the objective function, we first define the regularization term for latent variables, which are usually implemented by the KL divergence [85, 41, 64, 159]. Following from [41], we assume all latent variables $\{\mathbf{z}, \mathbf{d}, \mathbf{u}\}$ are conditionally independent and we have the variational distribution $q_{\varsigma,\varepsilon,\delta}(\mathbf{z}, \mathbf{d}, \mathbf{u}|\mathbf{x}) = q_\varsigma(\mathbf{z}\,|\,\mathbf{x})q_\varepsilon(\mathbf{d}\,|\,\mathbf{x})q_\delta(\mathbf{u}\,|\,\mathbf{x})$. We also assume all latent variables $\{\mathbf{z}, \mathbf{d}, \mathbf{u}\}$ are independent and we have the prior distributions $p(\mathbf{z}, \mathbf{u}, \mathbf{d}) = p(\mathbf{z})p(\mathbf{u})p(\mathbf{d})$. We can derive the

KL divergence term that regulates the learning of $\{\mathbf{z}, \mathbf{u}, \mathbf{d}\}$, expressed as:

$$
\begin{aligned}
D_{KL}[q_{\varsigma,\varepsilon,\delta}(\mathbf{z},\mathbf{d},\mathbf{u}\,|\,\mathbf{x})\,||\,p(\mathbf{z},\mathbf{d},\mathbf{u})] &= \mathbb{E}_{q_{\varsigma,\varepsilon,\delta}(\mathbf{z},\mathbf{d},\mathbf{u}\,|\,\mathbf{x})}\left[\log\frac{q_{\varsigma,\varepsilon,\delta}(\mathbf{z},\mathbf{d},\mathbf{u}\,|\,\mathbf{x})}{p(\mathbf{z},\mathbf{d},\mathbf{u})}\right] \\
&= \mathbb{E}_{q_\varsigma(\mathbf{z}\,|\,\mathbf{x})q_\varepsilon(\mathbf{d}\,|\,\mathbf{x})q_\delta(\mathbf{u}\,|\,\mathbf{x})}\left[\log\frac{q_\varsigma(\mathbf{z}\,|\,\mathbf{x})q_\varepsilon(\mathbf{d}\,|\,\mathbf{x})q_\delta(\mathbf{u}\,|\,\mathbf{x})}{p(\mathbf{z})p(\mathbf{d})p(\mathbf{u})}\right] \\
&= \mathbb{E}_{q_\varsigma(\mathbf{z}\,|\,\mathbf{x})q_\varepsilon(\mathbf{d}\,|\,\mathbf{x})q_\delta(\mathbf{u}\,|\,\mathbf{x})}\left[\log\frac{q_\varsigma(\mathbf{z}\,|\,\mathbf{x})}{p(\mathbf{z})}\right] \\
&+ \mathbb{E}_{q_\varsigma(\mathbf{z}\,|\,\mathbf{x})q_\varepsilon(\mathbf{d}\,|\,\mathbf{x})q_\delta(\mathbf{u}\,|\,\mathbf{x})}\left[\log\frac{q_\varepsilon(\mathbf{d}\,|\,\mathbf{x})}{p(\mathbf{d})}\right] \\
&+ \mathbb{E}_{q_\varsigma(\mathbf{z}\,|\,\mathbf{x})q_\varepsilon(\mathbf{d}\,|\,\mathbf{x})q_\delta(\mathbf{u}\,|\,\mathbf{x})}\left[\log\frac{q_\delta(\mathbf{u}\,|\,\mathbf{x})}{p(\mathbf{u})}\right] \\
&= \mathbb{E}_{q_\delta(\mathbf{u}\,|\,\mathbf{x})q_\varsigma(\mathbf{z}|\mathbf{x})}D_{KL}[q_\varepsilon(\mathbf{d}\,|\,\mathbf{x})\,||\,p(\mathbf{d})] \\
&+ \mathbb{E}_{q_\varepsilon(\mathbf{d}\,|\,\mathbf{x})q_\varsigma(\mathbf{z}\,|\,\mathbf{x})}D_{KL}[q_\delta(\mathbf{u}\,|\,\mathbf{x})\,||\,p(\mathbf{u})] \\
&+ \mathbb{E}_{q_\delta(\mathbf{u}\,|\,\mathbf{x})q_\varepsilon(\mathbf{d}\,|\,\mathbf{x})}D_{KL}[q_\varsigma(\mathbf{z}\,|\,\mathbf{x})\,|\,p(\mathbf{z})] \\
&= a_1 D_{KL}[q_\varsigma(\mathbf{z}\,|\,\mathbf{x})\,||\,p(\mathbf{z})] + a_2 D_{KL}[q_\varepsilon(\mathbf{d}\,|\,\mathbf{x})\,||\,p(\mathbf{d})] \\
&+ a_3 D_{KL}[q_\delta(\mathbf{u}\,|\,\mathbf{x})\,||\,p(\mathbf{u})]\,.
\end{aligned}
\tag{3.6}
$$

Using the KL divergence to regulate both continuous and discrete variables in a generative framework is used in [41, 28, 50], which only considers learning a static dataset and two latent variables. In order to allow the latent variables to encode the data information, we define the objective function that combines the reconstruction error and KL divergence terms (Eq. (3.6)), expressed as:

$$
\begin{aligned}
\mathcal{L}_{\text{VAE}}^{\text{LGAA}}(\mathbf{x},\theta,\varsigma,\varepsilon,\delta) = -\mathbb{E}_{q_{\varsigma,\varepsilon,\delta}(\mathbf{z},\mathbf{d},\mathbf{u}\,|\,\mathbf{x})}\log[p_\theta(\mathbf{x}\,|\,\mathbf{z},\mathbf{d},\mathbf{u})] + D_{KL}[q_\varsigma(\mathbf{z}\,|\,\mathbf{x})\,||\,p(\mathbf{z})] \\
+ D_{KL}[q_\varepsilon(\mathbf{d}\,|\,\mathbf{x})\,||\,p(\mathbf{d})] + D_{KL}[q_\delta(\mathbf{u}\,|\,\mathbf{x})\,||\,p(\mathbf{u})]\,.
\end{aligned}
\tag{3.7}
$$

Since $q_\varepsilon(\mathbf{d}\,|\,\mathbf{x})$ takes a high-dimensional image as the input, we usually process this input data using a deep CNN network with several convolutional layers, which leads to more computational complexity and parameters. To further reduce the model size, we then consider replacing the high-dimensional data $\mathbf{x}$ in $q_\varepsilon(\mathbf{d}\,|\,\mathbf{x})$ with a low-dimensional latent variable $\mathbf{z}$, resulting in a lightweight inference model $q_\varepsilon(\mathbf{d}\,|\,\mathbf{z})$ implemented by a simple fully connected network with fewer parameters that are enough to process $\mathbf{z}$. Such a replacement still ensures that the domain variable $\mathbf{d}$ can be drawn from $q_\varepsilon(\mathbf{d}\,|\,\mathbf{z})$ while $\mathbf{z}$ can also be seen as a compact representation of

$\mathbf{x}$. Then Eq. (3.7) is rewritten as:

$$\begin{aligned}\mathcal{L}_{\text{VAE}}^{\text{LGAA}}(\mathbf{x},\theta,\varsigma,\varepsilon,\delta) = -\mathbb{E}_{q_{\varsigma,\varepsilon,\delta}(\mathbf{z},\mathbf{d},\mathbf{u}\,|\,\mathbf{x})}\log[p_\theta(\mathbf{x}\,|\,\mathbf{z},\mathbf{d},\mathbf{u})] + D_{KL}[q_\varsigma(\mathbf{z}\,|\,\mathbf{x})\,||\,p(\mathbf{z})] \\ + D_{KL}[q_\varepsilon(\mathbf{d}\,|\,\mathbf{z})\,||\,p(\mathbf{d})] + D_{KL}[q_\delta(\mathbf{u}\,|\,\mathbf{x})\,||\,p(\mathbf{u})]\,,\end{aligned} \tag{3.8}$$

where the latent variable $\mathbf{z}$ in the second KL divergence term of Eq. (3.8) can be drawn from $q_\varsigma(\mathbf{z}\,|\,\mathbf{x})$. In Eq. (3.8), we have separated KL divergence components for the continuous $\mathbf{z}$ space, as well as for the discrete and domain spaces $\mathbf{u}$ and $\mathbf{d}$, respectively. Meanwhile, $\theta, \varsigma, \varepsilon, \delta$ represent the parameters of the corresponding networks. $q_\varsigma(\mathbf{z}\,|\,\mathbf{x})$ is a variational distribution (Gaussian) whose hyperparameters are estimated by the encoder $F_\varsigma(\cdot)$. $q_\varepsilon(\mathbf{d}\,|\,\mathbf{z})$ is considered as a categorical distribution whose parameters $\{d'_1, \cdots, d'_L\}$ are predicted by $F_\varepsilon(\cdot)$. Similarly, $q_\delta(\mathbf{u}\,|\,\mathbf{x})$ is considered as a categorical distribution whose parameters are predicted by $F_\delta(\cdot)$. We also employ the Gumbel-Max trick (Eq. (3.4)) to draw the continuous relaxation of the discrete variables from $q_\delta(\mathbf{u}\,|\,\mathbf{x})$ and $q_\varepsilon(\mathbf{d}\,|\,\mathbf{z})$ to ensure end-to-end training [109]. $p(\mathbf{u}) = \text{Cat}(Q, p_1^{\mathbf{u}}, \cdots, p_Q^{\mathbf{u}})$ is the prior implemented by the uniform categorical distribution where each parameter is set to $p_i^{\mathbf{u}} = 1/Q, i = 1, \cdots, Q$. Similarly, $p(\mathbf{d}) = \text{Cat}(L, p_1^{\mathbf{d}}, \cdots, p_L^{\mathbf{d}})$ is the prior implemented by the uniform categorical distribution where each parameter is set to $p_i^{\mathbf{d}} = 1/L, i = 1, \cdots, L$. Training a VAE framework with the discrete variable using the Gumbel Softmax distribution has been investigated in [109, 41]. However, those approaches can only learn a static and pre-defined data distribution, which can not be applied to lifelong learning.

For the supervised learning setting, auxiliary information such as class and task labels can be used to guide the inference model. For supervised learning we employ the cross-entropy loss $f_{\text{CE}}(\cdot,\cdot,\cdot)$ which is defined as:

$$f_{\text{CE}}(\mathbf{y}', \mathbf{y}, Q) = -\sum_{i=1}^{Q} y_i \log y'_i\,, \tag{3.9}$$

where y_i and y'_i are the i-th entry of the class label $\mathbf{y}$ (one-hot) and the prediction $\mathbf{y}'$, respectively. Q is the dimension of the variable $\mathbf{u}$ and the class label $\mathbf{y}$. By employing the cross-entropy loss $f_{\text{CE}}(\cdot,\cdot,\cdot)$, the inference models $q_\varepsilon(\mathbf{d}\,|\,\mathbf{z})$ and $q_\delta(\mathbf{u}\,|\,\mathbf{x})$ can be updated by minimizing:

$$\mathcal{L}_{\mathbf{d}}(\mathbf{x}, \mathbf{d}^\star, \varepsilon, L) = f_{\text{CE}}(F_\varepsilon(F_{\mathbf{z}}(\mathbf{x})), \mathbf{d}^\star, L), F_{\mathbf{z}}(\mathbf{x}) = F_\varsigma^{\boldsymbol{\mu}}(\mathbf{x}) + \boldsymbol{\pi} \bigodot F_\varsigma^{\boldsymbol{\sigma}}(\mathbf{x})\,, \tag{3.10}$$

$$\mathcal{L}_{\mathbf{u}}(\mathbf{x}, \mathbf{y}, \delta, Q) = f_{\mathrm{CE}}(F_{\delta}(\mathbf{x}), \mathbf{y}, Q)\,, \tag{3.11}$$

where $\boldsymbol{\pi}$ is a random noise vector sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\bigodot$ is the element-wise product. $F_{\varsigma}^{\boldsymbol{\mu}}(\mathbf{x})$ and $F_{\varsigma}^{\boldsymbol{\sigma}}(\mathbf{x})$ return the Gaussian hyperparameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, respectively. $\mathbf{y}$ and $\mathbf{d}^{\star}$ are the class and task labels, respectively, which can be obtained from the empirical training datasets. The network architecture of the generator and inference networks of the proposed LGAA is shown in Fig. 3.4. The proposed model is flexible to be extended for recognizing new tasks by automatically appending the domain variable $\mathbf{d}$ and optimizing the task-inference model $q_{\varepsilon}(\mathbf{d} \,|\, \mathbf{z})$ when faced with learning a new task.

The inference models in the proposed LGAA aim to map an input into three compact low-dimensional feature vectors $\{\mathbf{z}, \mathbf{d}, \mathbf{u}\}$, with each describing different characteristics of the input data. The decoder (generator) will recover an image from these three feature vectors. In the supervised learning framework, we optimize the inference models $q_{\varepsilon}(\mathbf{d} \,|\, \mathbf{z})$ and $q_{\delta}(\mathbf{u} \,|\, \mathbf{x})$ by introducing additional cross-entropy loss functions, such as Eq. (3.10) and Eq. (3.11), where each inference model is encouraged to capture different characteristics of an input. It allows for performing many downstream tasks, including image classification and interpolation, within a unified framework at the same time. Additionally, LGAA introduces a robust generative replay network for producing past samples, statistically consistent with those learnt previously from various databases. This allows the inference models to capture the context information and implicitly model the correlation between the new task and the previously learnt knowledge. The following section introduces the algorithm used for training LGAA.

3.3 Lifelong Training Algorithm for LGAA

In the following, we introduce a new training algorithm that enables LGAA to learn knowledge from a sequence of tasks without forgetting. The key idea of the proposed training algorithm consists of two distinct updating procedures where we firstly update the parameters of the generative replay network and afterwards those of the whole model. Our algorithm is different from existing hybrid models in three aspects: 1) Existing hybrid models are only trained for a single dataset [113]. However, the proposed LGAA is able to learn several data domains successively without forgetting; 2) Existing hybrid models usually train the generator and inference modules with a single optimization function [91], which requires balancing the GAN

and VAE losses. The other hybrid models learn an optimal coupling between the generator and inference modules using adversarial learning [91, 113, 25, 35, 40, 96, 115, 156]. However, those models can not achieve high-quality data reconstructions since using only adversarial learning can not learn an optimal inverse mapping of the generator. The proposed LGAA introduces a training algorithm consisting of two updating procedures for updating the inference model and generator separately. The advantage of the proposed two-step optimization is that adversarial learning can provide good initial parameters for the generator, and then the VAE inference-based learning for the whole model encourages learning meaningful latent representations that lead to good reconstruction performance; 3) Existing hybrid models usually learn a single latent variable during the training [115], which is not applicable for a wide range of applications. The proposed LGAA learns both discrete and continuous variables, which can be used in classification and disentangled representation learning. In the following, we introduce different loss functions for LGAA, which allow for the model to be used for supervised, semi-supervised, and unsupervised learning.

3.3.1 Supervised Learning

During the training, we first update the generator's parameters by using the WGAN-GP loss function at the t-th task learning. The adversarial loss function for the generator ($\mathcal{G}_{\theta^t}$) and discriminator ($\mathcal{D}_{\eta^t}$) at the t-th task learning is defined as:

$$\mathcal{L}_{\mathcal{G}}^{sup}(\mathbf{Y}_{batch}, \mathbf{D}_{batch}, \theta^t, \eta^t) = \frac{1}{m}\sum_{i=1}^{m}\Big\{ - \mathcal{D}_{\eta^t}(\mathcal{G}_{\theta^t}(\mathbf{y}_i, \mathbf{z}_i, \mathbf{d}_i^\star))\Big\}, \tag{3.12}$$

$$\begin{aligned}\mathcal{L}_{\mathcal{D}}^{sup}(\mathbf{X}_{batch}, \mathbf{Y}_{batch}, \mathbf{D}_{batch}, \theta^t, \eta^t) = \frac{1}{m}\sum_{i=1}^{m}\Big\{&\mathcal{D}_{\eta^t}(\mathcal{G}_{\theta^t}(\mathbf{y}_i, \mathbf{z}_i, \mathbf{d}_i^\star)) \\ &- \mathcal{D}_{\eta^t}(\mathbf{x}_i) + \lambda(\|\nabla_{\tilde{\mathbf{x}}_i}\mathcal{D}_{\eta^t}(\tilde{\mathbf{x}}_i)\|_2 - 1)^2\Big\},\end{aligned} \tag{3.13}$$

where $\mathcal{G}_{\theta^t}$ denotes that the generator's parameters θ^t are updated at the t-th task learning. $\{\mathbf{x}_i, \mathbf{y}_i\}$ is the i-th labelled sample of the data batch $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^m$ obtained from a joint dataset $D_S^t \bigcup D_G$ where D_S^t is the training dataset of the t-th task, where m represents the number of data samples in the batch. D_G is a generated dataset[3], where each image $\mathbf{x}$ is generated using the generator $\mathcal{G}_{\theta^{t-1}}$ and the associated class label $\mathbf{y}$ is predicted

[3]We do not use the superscript in D_G for denoting task information since the generation process of the dataset D_G at each task learning is clearly described in **Algorithm 1**.

Algorithm 1: The supervised learning for LGAA

Input: All training databases
Output: The model's parameters

for $t < N$ **do**
  if $t == 1$ **then**
    Obtain the training dataset D_S^t ;
    Obtain the task label dataset D_D^t ;
  end
  else
    Generative replay process ;
    for $c < datasetSize$ **do**
      Generate $\mathbf{x}_c$ using the generator ;
      Predict the class label $\mathbf{y}_c$ using the inference model $q_{\delta^{t-1}}(\mathbf{u} \mid \mathbf{x})$;
      Obtain $\mathbf{z}_c$ using $q_{\varsigma^{t-1}}(\mathbf{z} \mid \mathbf{x})$ and predict the task label $\mathbf{d}_c^\star$ using $q_{\varepsilon^{t-1}}(\mathbf{d} \mid \mathbf{z})$;
      Form a labelled sample $\{\mathbf{x}_c, \mathbf{y}_c\}$;
      $D_G = D_G \bigcup \{\mathbf{x}_c, \mathbf{y}_c\}$;
      $D_{GD} = D_{GD} \bigcup \mathbf{d}_c^\star$;
    end
    Form a joint dataset $D_S^t = D_S^t \bigcup D_G$;
    Form a joint dataset $D_D^t = D_D^t \bigcup D_{GD}$;
  end
  for $epoch < 20$ **do**
    for $j < batchCount$ **do**
      Obtain the data batches $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\}$ from D_S^t ;
      Obtain the data batch $\mathbf{D}_{batch}$ from D_D^t ;
      Adversarial learning ;
      Update the generator and discriminator only once on $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}, \mathbf{D}_{batch}\}$ using Eq. (3.12) and Eq. (3.13) ;
      Learning by the VAE loss ;
      Update all components only once on $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}, \mathbf{D}_{batch}\}$ using Eq. (3.14) ;
      Updating the inference models ;
      Update $q_{\varepsilon^t}(\mathbf{d} \mid \mathbf{z})$ only once on $\{\mathbf{X}_{batch}, \mathbf{D}_{batch}\}$ using Eq. (3.10) ;
      Update $q_{\delta^t}(\mathbf{u} \mid \mathbf{x})$ only once on $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\}$ using Eq. (3.11) ;
    end
  end
end

using the inference model $q_{\delta^{t-1}}(\mathbf{u} \mid \mathbf{x})$. Since the class labels are available in the joint dataset $D_S^t \bigcup D_G$, $\mathbf{u}_i$ in Eq. (3.12) and Eq. (3.13) can be assigned by $\mathbf{y}_i$ from the data batch $\mathbf{Y}_{batch}$. $\mathbf{D}_{batch} = \{\mathbf{d}_i^\star\}_{i=1}^m$ in Eq. (3.13) is the data batch and obtained from a joint dataset $D_D^t \bigcup D_{GD}$ that represents the task information for $D_S^t \bigcup D_G$, where $D_D^t = \{\mathbf{d}_j^\star\}_{j=1}^{N_S^t}$ is the task label dataset of the t-th task and has the same dataset size with D_S^t. D_{GD} is a generated dataset and has the same dataset size with D_G, where each task label in D_{GD} is produced as follows : (1) Obtain a sample $\mathbf{x}$ from D_G; (2) Obtain the latent representation $\mathbf{z}$ for $\mathbf{x}$ using the inference model $q_{\varsigma^{t-1}}(\mathbf{z} \mid \mathbf{x})$; (3) Predict the task label for the latent code $\mathbf{z}$ using the inference model $q_{\varepsilon^{t-1}}(\mathbf{d} \mid \mathbf{z})$;

In the second updating step, we update the parameters of the whole model by using the

objective function, defined as:

$$\begin{aligned}\mathcal{L}_{\text{VAE}}^{Sup}(\mathbf{X}_{batch}, \theta^t, \varsigma^t, \varepsilon^t, \delta^t) = \frac{1}{m}\sum_{i=1}^{m}\Big\{ &- \mathbb{E}_{q_{\varsigma^t,\varepsilon^t,\delta^t}(\mathbf{z},\mathbf{d},\mathbf{u}\,|\,\mathbf{x}_i)} \log[p_{\theta^t}(\mathbf{x}\,|\,\mathbf{z},\mathbf{d},\mathbf{u})] \\ &+ D_{KL}[q_{\varsigma^t}(\mathbf{z}\,|\,\mathbf{x}_i)\,||\,p(\mathbf{z})] + D_{KL}[q_{\varepsilon^t}(\mathbf{d}\,|\,\mathbf{z}_i)\,||\,p(\mathbf{d})] \\ &+ D_{KL}[q_{\delta^t}(\mathbf{u}\,|\,\mathbf{x}_i)\,||\,p(\mathbf{u})]\Big\},\end{aligned} \tag{3.14}$$

where $\mathbf{z}_i$ in the third term is inferred by $q_{\varsigma^t}(\mathbf{z}\,|\,\mathbf{x}_i)$ with the sample $\mathbf{x}_i$ and $\mathbf{X}_{batch}$ is the data batch. Note that Eq. (3.14) does not require the class label data batch $\mathbf{Y}_{batch}$. We also update the inference models $q_{\delta^t}(\mathbf{u}\,|\,\mathbf{x}_i)$ and $q_{\varepsilon^t}(\mathbf{d}\,|\,\mathbf{z}_i)$ using the cross-entropy losses (Eq. (3.10) and Eq. (3.11)) on the data batches $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}, \mathbf{D}_{batch}\}$. During the testing phase, the inference model $q_\delta(\mathbf{u}\,|\,\mathbf{x})$ can be used for classification.

We provide the pseudocode for the supervised learning of the LGAA in Algorithm 1, which can be summarized into three steps:

Step 1. Generative replay process: At the first task learning, the proposed LGAA does not require the generative replay process to relieve forgetting and therefore is directly trained on the first task. The training and task label datasets in the following task are created by the generative replay process, described as follows. We assume that the model was trained on $t-1$ tasks. In a new task learning (t-th task), we perform the generative replay process to create a joint dataset $D_S^t \bigcup D_G$ consisting of the real training dataset D_S^t from the t-th task and a dataset D_G generated using the model trained on $t-1$ tasks. We also form a joint dataset $D_D^t \bigcup D_{GD}$, which represents the task information for all samples.

Step 2. Adversarial learning: We update the discriminator and generator in the mini-batch learning manner, in which the data batches $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}, \mathbf{D}_{batch}\}$ are obtained from the datasets $D_S^t \bigcup D_G$ and $D_D^t \bigcup D_{GD}$, while the parameters of the discriminator and generator are only updated once using Eq. (3.12) and Eq. (3.13).

Step 3. The whole model updating: In the mini-batch learning, we update all components only once on a data batch $\mathbf{X}_{batch}$ obtained from the joint dataset $D_S^t \bigcup D_G$ using Eq. (3.8). We then update $q_{\varepsilon^t}(\mathbf{d}\mid\mathbf{z})$ and $q_{\delta^t}(\mathbf{u}\mid\mathbf{x})$ only once on the data batches $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}, \mathbf{D}_{batch}\}$ obtained from the datasets $D_S^t \bigcup D_G$ and $D_D^t \bigcup D_{GD}$ using Eq. (3.10) and Eq. (3.11), respectively. As shown in Algorithm 1, the number of training epochs for each task is 20 and the number of iterations ($batchCount$) in each training epoch is determined by $batchCount = datasetSize/m$,

where $m = 64$ is the batch size and $datasetSize$ is training dataset size. If the total number of training iterations for the current task learning is not finished, we continually perform **Step 2** and **Step 3**, otherwise, we perform **Step 1** for the next task learning.

3.3.2 Semi-Supervised Learning

We apply our model to the lifelong semi-supervised learning setting where only a small subset of samples from each task has known labels, while the rest of the data is unlabelled. We design different objective functions for the labelled and unlabelled samples. The generator and discriminator training are the same as in Eq. (3.12) and Eq. (3.13). The whole model is optimized by the objective function for the labelled data without the inference model $q_\delta(\mathbf{u}\,|\,\mathbf{x})$ at the t-th task learning:

$$\begin{aligned}\mathcal{L}^{S}_{\text{VAE}}(\mathbf{X}_{batch}, \mathbf{Y}_{batch}, \theta^t, \varsigma^t, \varepsilon^t) = \frac{1}{m}\sum_{i=1}^{m}\Big\{ &- \mathbb{E}_{q_{\varsigma^t}(\mathbf{z}\,|\,\mathbf{x}_i), q_{\varepsilon^t}(\mathbf{d}\,|\,\mathbf{x}_i)} \log[p_{\theta^t}(\mathbf{x}\,|\,\mathbf{z}, \mathbf{d}, \mathbf{y}_i)] \\ &+ D_{KL}[q_{\varsigma^t}(\mathbf{z}\,|\,\mathbf{x}_i)\,||\,p(\mathbf{z})] + D_{KL}[q_{\varepsilon^t}(\mathbf{d}\,|\,\mathbf{z}_i)\,||\,p(\mathbf{d})]\Big\},\end{aligned} \tag{3.15}$$

where $\mathbf{y}_i$ is the i-th label from the data batch $\mathbf{Y}_{batch}$. In addition, we model the unlabeled data samples by using $\mathcal{L}^{\text{LGAA}}_{\text{VAE}}(\mathbf{x}, \theta^t, \varsigma^t, \varepsilon^t, \delta^t)$, defined in Eq. (3.8), where the discrete variable $\mathbf{u}$ is sampled from the Gumbel-softmax distribution whose probability vector is obtained by the encoder $q_{\delta^t}(\mathbf{u}\,|\,\mathbf{x})$. The semi-supervised loss used to train the proposed model is defined as:

$$\begin{aligned}\mathcal{L}^{Semi}_{\text{VAE}}(\mathbf{X}_{batch}, \mathbf{Y}_{batch}, \mathbf{X}^{unsupervised}_{batch}, \theta^t, \varsigma^t, \varepsilon^t, \delta^t) = &\mathcal{L}^{S}_{\text{VAE}}(\mathbf{X}_{batch}, \mathbf{Y}_{batch}, \theta^t, \varsigma^t, \varepsilon^t, \delta^t) \\ &+ \beta^{Semi}\mathcal{L}^{Sup}_{\text{VAE}}(\mathbf{X}^{unsupervised}_{batch}, \theta^t, \varsigma^t, \varepsilon^t),\end{aligned} \tag{3.16}$$

where $\beta^{Semi} = 0.5$ is used to control the importance of unsupervised learning when compared with the component associated with supervised learning [185]. $\mathcal{L}^{Sup}_{\text{VAE}}(\mathbf{X}^{unsupervised}_{batch}, \theta^t, \varsigma^t, \varepsilon^t)$ is the mini-batch learning form of $\mathcal{L}^{\text{LGAA}}_{\text{VAE}}(\mathbf{x}, \theta^t, \varsigma^t, \varepsilon^t, \delta^t)$ and can be used for unlabelled data samples $\mathbf{X}^{unsupervised}_{batch}$ without requiring class labels. $\mathcal{L}^{S}_{\text{VAE}}(\mathbf{X}_{batch}, \mathbf{Y}_{batch}, \theta^t, \varsigma^t, \varepsilon^t, \delta^t)$ is used to calculate the loss for a pair of labelled data batches $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\}$. In addition, the entropy loss $\mathcal{L}_{\mathbf{u}}(\mathbf{x}, \mathbf{y}, \delta, Q)$ is also performed with the labelled data samples in order to enhance the prediction ability of $q_\delta(\mathbf{u}\,|\,\mathbf{x})$, as in Eq. (3.11).

3.3.3 Unsupervised Learning

In this section, we apply the proposed LGAA for the lifelong unsupervised learning setting, where the class labels for samples are unavailable. Similarly to the supervised learning framework, the first updating step of the training at the t-th task learning employs adversarial loss:

$$\mathcal{L}_{\mathcal{G}}^{U}(\mathbf{D}_{batch}, \theta^t, \eta^t) = \frac{1}{m}\sum_{i=1}^{m}\Big\{ -\mathcal{D}_{\eta^t}(\mathcal{G}_{\theta^t}(\mathbf{z}_i, \mathbf{d}_i^{\star}))\Big\}, \tag{3.17}$$

$$\begin{aligned}\mathcal{L}_{\mathcal{D}}^{U}(\mathbf{X}_{batch}, \mathbf{D}_{batch}, \theta^t, \eta^t) = \frac{1}{m}\sum_{i=1}^{m}\Big\{&\mathcal{D}_{\eta^t}(\mathcal{G}_{\theta^t}(\mathbf{z}_i, \mathbf{d}_i^{\star})) - \mathcal{D}_{\eta^t}(\mathbf{x}_i) \\ &+ \lambda(\|\nabla_{\tilde{\mathbf{x}}_i}\mathcal{D}_{\eta^t}(\tilde{\mathbf{x}}_i)\|_2 - 1)^2\Big\},\end{aligned} \tag{3.18}$$

where $\mathbf{x}_i$ is the i-th sample from the data batch $\mathbf{X}_{batch}$. $\mathbf{d}_i^{\star}$, obtained from the data batch $\mathbf{D}_{batch}$, is the task label for $\mathbf{x}_i$. In the second updating step, the whole model is updated by:

$$\begin{aligned}\mathcal{L}_{\text{VAE}}^{U}(\mathbf{X}_{batch}, \theta^t, \varsigma^t, \varepsilon^t) = \frac{1}{m}\sum_{i=1}^{m}\Big\{&-\mathbb{E}_{q_{\varsigma^t}(\mathbf{z}\,|\,\mathbf{x}_i), q_{\varepsilon^t}(\mathbf{d}\,|\,\mathbf{x}_i)}\log[p_{\theta^t}(\mathbf{x}\,|\,\mathbf{z}, \mathbf{d})] \\ &+ D_{KL}[q_{\varsigma^t}(\mathbf{z}\,|\,\mathbf{x}_i)\,||\,p(\mathbf{z})] \\ &+ D_{KL}[q_{\varepsilon^t}(\mathbf{d}\,|\,\mathbf{z}_i)\,||\,p(\mathbf{d})]\Big\}.\end{aligned} \tag{3.19}$$

Learning meaningful and disentangled representations has become a new and important topic in computer vision [64, 171]. The disentangled representations aim to learn a set of variables, where each one (scalar) captures the variation of a specific characteristic of the image and is independent of the other variables. Learning disentangled representations can be implemented by using the VAE framework, which was firstly proposed in [64], called β-VAE where a hyperparameter $\beta > 1$ is used to regularize the KL divergence term in the VAE's loss function to encourage learning disentangled representations. One downside for β-VAE is that the model usually produces blurred reconstruction results when using a large hyperparameter β. This is because penalizing the KL divergence term leads to the reduction of the mutual information between the observed variable $\mathbf{x}$ and the latent variable $\mathbf{z}$ (See details in Section 2.1.4). This negative effect can be relieved by an improved β-VAE objective function [19], which progressively increases the information capacity of the latent representation during training. However, most existing disentangled representation methods only consider learning a static and pre-defined data distribution, while the lifelong learning of disentangled representations is

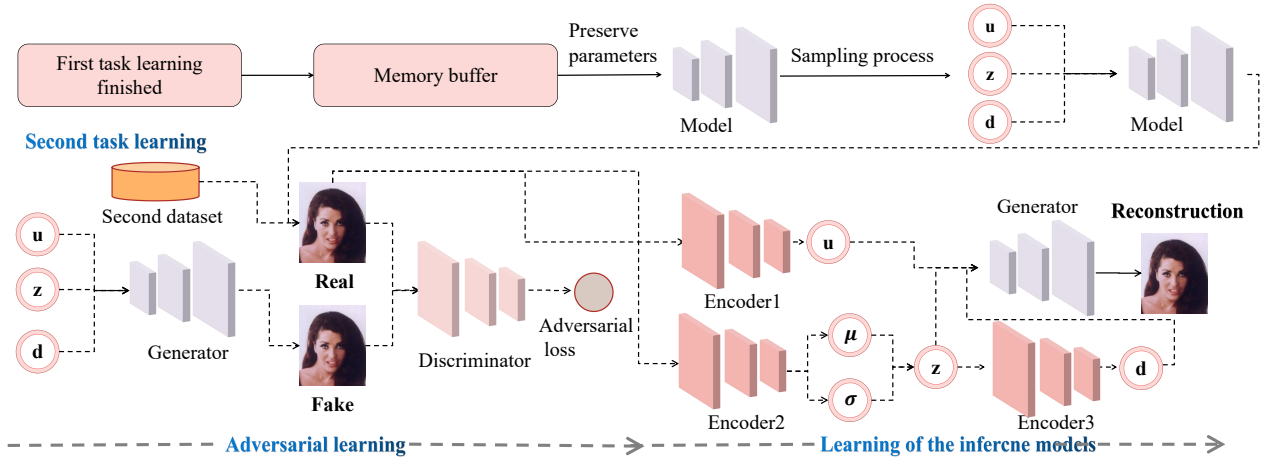


Figure 3.5: Using the memory buffer in the LGAA framework. Once the first task is learnt, we use a buffer to preserve the generator's parameters. Then, during the second task learning, the preserved generator is used as a generative replay mechanism, producing a batch of samples. The generated data samples are incorporated together with new samples drawn from the second task for training LGAA. Then, the process of creating buffers for temporary storing generator parameters is repeated each time when learning a new task.

a new research topic [139]. In this chapter, we employ the improved β-VAE method [19] to replace the second term from Eq. (3.19), aiming to learn disentangled representations in the context of lifelong learning:

$$\begin{aligned}\mathcal{L}_{\text{VAE}}^{U_{dis}}(\mathbf{X}_{batch}, \theta^t, \varsigma^t, \varepsilon^t) = \frac{1}{m}\sum_{i=1}^{m}\Big\{ &- \mathbb{E}_{q_{\varsigma^t}(\mathbf{z}\,|\,\mathbf{x}_i), q_{\varepsilon^t}(\mathbf{d}\,|\,\mathbf{x}_i)} \log[p_{\theta^t}(\mathbf{x}\,|\,\mathbf{z}, \mathbf{d})] \\ &+ \gamma |D_{KL}[q_{\varsigma^t}(\mathbf{z}\,|\,\mathbf{x}_i)\,||\,p(\mathbf{z})] - C| \\ &+ D_{KL}[q_{\varepsilon^t}(\mathbf{d}\,|\,\mathbf{z}_i)\,||\,p(\mathbf{d})]\Big\},\end{aligned} \tag{3.20}$$

where γ and C are a multiplicative and a linear constant used for controlling the degree of disentanglement. For the experiments, we consider the recommended hyperparameters from [19], where the multiplicative parameter is $\gamma = 4$, while we increase the hyperparameter C from 0.5 to 25.0 during the training.

3.3.4 Using A Memory Buffer for Storing Model Parameters, in Lifelong Learning

Instead of generating a collection of data samples by the generator, we can employ a small memory buffer to preserve the current model's parameters before learning the next task. Then, the preserved model is used to generate a batch of data to be used for training together with

data sampled from the database corresponding to the subsequent task learning. The buffer is always fixed in size while increasing the number of tasks to be learnt during lifelong learning. After learning the current task, the old model parameters stored in the buffer will be replaced by the current model parameters. Then, during the new task learning, the parameters from this buffer are used by the model for generating a batch of data corresponding to the stored model. The buffer used in our model can achieve a similar performance without the need to increase the required memory when adding new tasks to be learnt. This mechanism provides a reduced memory requirement in the proposed model, and its learning structure is displayed in Fig. 3.5.

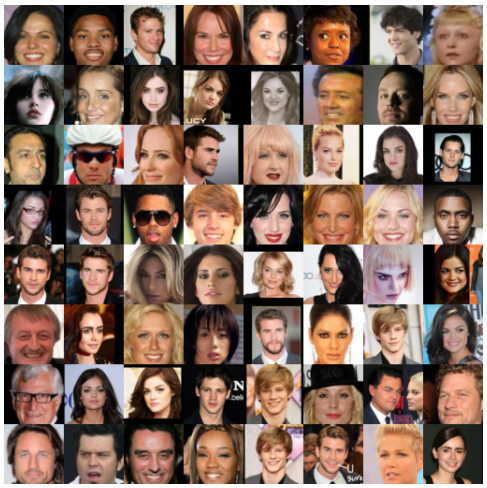
3.4 Experiments

In this section, we investigate how the proposed Lifelong Generative Adversarial Autoencoder (LGAA) model learns meaningful and interpretable image representations under the lifelong learning of several tasks.

3.4.1 Reconstruction and Interpolation Results Following Unsupervised Lifelong Learning

We train the LGAA model using the loss functions $\mathcal{L}_{\mathcal{D}}^{U}$, $\mathcal{L}_{\mathcal{G}}^{U}$ and $\mathcal{L}_{\text{VAE}}^{U}$ from equations (3.17), (3.18) and (3.19), which contain adversarial and VAE objective functions, respectively, and we consider to use Adam algorithm [82] with the learning rate of 0.001 for training. We resize all images of CelebA and CACD to the resolution $64 \times 64 \times 3$. The results for the unsupervised lifelong learning of CelebA [105] to CACD [24] are provided in Figures 3.6a-c where we show the real images, generated images, and the real image reconstructions, respectively. Meanwhile, in Figures 3.7a-c, we provide real images, generated images and the reconstructions of the real images from Fig. 3.7a, after the lifelong learning of CelebA to 3DChair [13]. From these results, it can be observed that the proposed approach can learn different data domains sequentially and provide good reconstruction results.

In the following, we perform data interpolation experiments under the lifelong learning setting in order to evaluate the manifold continuity in the latent space. We call lifelong interpolation when the interpolation is performed between multiple data domains by considering



(a) Real samples.

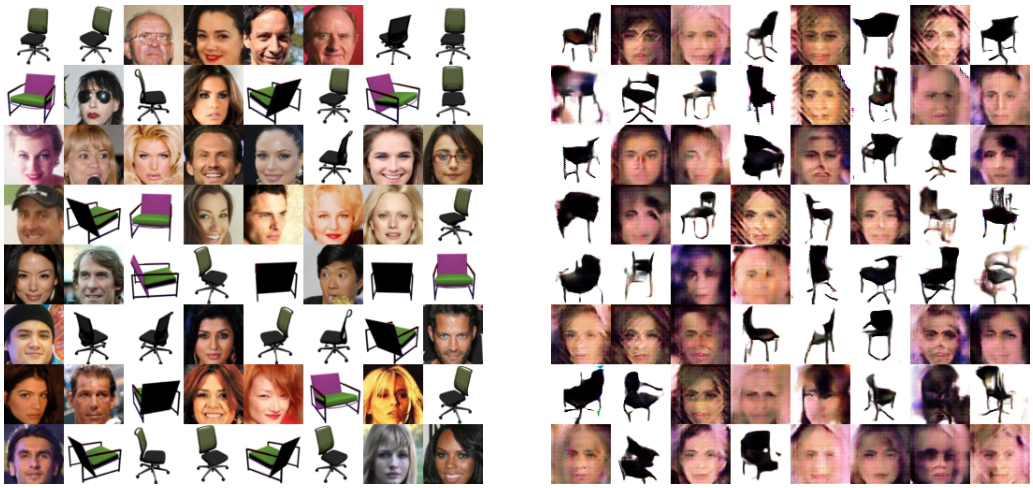
(b) Generations.

(c) Reconstructions.

Figure 3.6: The reconstruction and generation results under the CelebA to CACD lifelong learning.

data from different databases, under the lifelong learning setting. We randomly select two images and then infer their discrete $\mathbf{u}$, and continuous $\mathbf{z}$ latent variables by using the inference model. Then, we perform the interpolation on these latent variables and the resulting interpolated variables are used as inputs to the generator for the image reconstruction. The interpolation results are shown in Fig. 3.8-a for CelebA to CACD, and in Fig. 3.8-b for CelebA to 3D-chair lifelong learning. We can observe from the images from the last two rows of Fig. 3.8-b that a chair is transformed into a human face, where the chair's seat and backside are smoothly changed into the eyes and hair of a person. This shows that the LGAA model can learn the joint latent space of two completely different data configurations.

In the following, we evaluate the reconstruction quality of the images generated by the proposed LGAA and other VAE-based methods under CelebA to 3D-Chair lifelong learning.



(a) Real samples.

(b) Generations.

(c) Reconstructions.

Figure 3.7: The reconstruction and generation results under the CelebA to 3D-Chair lifelong learning.

To achieve this, we calculate the reconstruction error using the Mean Square Error, summing up the square error between all real testing samples and their reconstructions and then computing the average. The results are provided in Table 3.1, which show that the proposed LGAA outperforms other VAE-based methods on the image reconstruction performance. This result also indicates that the proposed LGAA can learn meaningful latent representations across different data domains over time, which can be potentially used for data compression in the context of continual learning.

3.4.2 Lifelong Disentangled Representations

Within the unsupervised lifelong learning framework, we train the LGAA model under the CelebA to 3D-Chairs lifelong learning by using the loss function from Eq. (3.20) in order to

(a) CelebA-CACD (b) CelebA-3DChair

Figure 3.8: Interpolation results after lifelong learning.

Table 3.1: Reconstruction error under CelebA to 3D-Chair lifelong learning.

Dataset	LGAA	LGM [132]	VAEGAN [115]
CelebA	313.20	351.23	357.25
3D-Chair	328.05	330.12	341.23
Average	**320.62**	340.67	349.24

achieve unsupervised disentangled representations, as described in Section 3.3.3. We consider the multiplicative parameter $\gamma = 4$, while increasing the linear one C from 0.5 to 25.0 in Eq. (3.20), during the training. After the training, we change one of the dimensions of a continuous latent representation $\mathbf{z}$, inferred by using the inference model, for a given input, and then map it back to the visual data space by using the generator. The disentangled results are presented in Figures 3.9a-f, indicating changes in the appearance of gender, facial narrowing, skin tone variation, skin face pose, chairs' size, and chairs' style. These results show that the LGAA model can discover various disentangled representations in CelebA and 3D-Chairs databases following lifelong learning.

3.4.3 Quality Assessment of the Generated Images

We use Inception score (IS) [144] and Fréchet Inception Distance (FID) [63] in order to evaluate the quality of generated image results after lifelong learning. We train various methods under the CelebA to CACD lifelong learning setting. The FID scores, calculated between 5,000 target images and 5,000 generated images, where target images include samples from both CelebA and CACD databases, are provided in Fig. 3.10-a. The results of the proposed LGAA are

(a) Gender change

(b) Face width manipulation

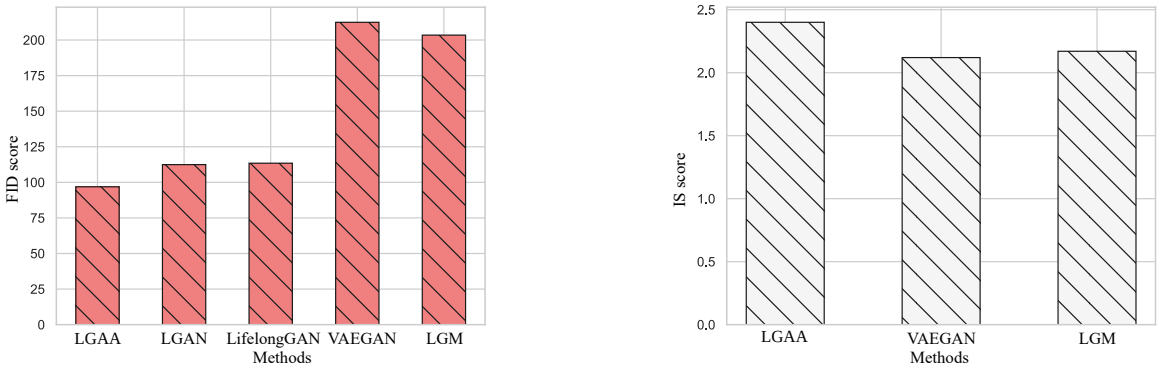
(c) Skin variation

(d) Pose change

(e) Chair style in 3D chairs

(f) Chair size variation in 3D chairs

Figure 3.9: Results when manipulating the latent variables under the CelebA to 3D-Chair lifelong learning when considering the loss function from Eq. (3.20). We change a single latent variable in the latent space from -3.0 to 3.0 while fixing all others.



(a) Fréchet Inception Distance (FID) for generated images after CelebA and CACD lifelong learning.

(b) Inception Score (IS) for image reconstructions after Cifar10 to MNIST lifelong learning.

Figure 3.10: Evaluation of the image reconstruction quality for various lifelong learning methods.

Table 3.2: Classification results following the lifelong learning of MNIST (M) and Fashion (F)

Dataset	Lifelong	LGAA	LGAN [57]	LGM [132]	EWC [86]	Transfer	MeRGANs [176]
MNIST	M-F	**98.76**	98.41	97.29	37.7	40.63	98.34
MNIST	F-M	98.77	98.32	98.85	**99.12**	98.25	98.27
Fashion	M-F	**92.01**	91.42	91.71	91.38	91.01	91.12
Fashion	F-M	**89.24**	89.15	86.05	54.53	37.92	88.86

compared with three other lifelong learning approaches: LGAN [57], LifelongGAN [194] and LGM [132]. A low FID score indicates that the model is able to generate more realistic images with respect to the real data distribution. We also consider Cifar10 [87] to MNIST database lifelong learning. The IS score on the reconstructions of 5,000 CIFAR10 testing samples, is provided in Fig. 3.10-b, where we compare with VAEGAN [115] and LGM [132]. A higher IS score indicates a better diversity of reconstructed images. The results from Fig. 3.10-b show that the proposed LGAA achieves a higher IS score than VAE-based methods, where VAE-based methods usually generate blurred images. The approach proposed in this chapter produces higher-quality generative replay images and can learn better data representations than other GAN-based lifelong learning approaches.

3.4.4 Lifelong Supervised Learning

We compare LGAA with various methods under the lifelong supervised learning setting as described in Section 3.3.1. LGAN [57] typically trains a classifier (called Solver) on both the images generated by the GAN and the training samples from the current task. We also consider an auxiliary classifier for LGM [132] by training it on the mixed data consisting of images generated by LGM and the training samples of the current task.

We train the LGAA model under the MNIST to Fashion [177] (M-F) lifelong learning and also when considering the reversed order of database learning, as F-M. The classification results, after the lifelong training, are reported in Table 3.2, where 'Transfer' is a baseline that is directly trained on a new task after each task switch and we compare the proposed LGAA with several other models from the literature. We observe that GRM-based methods can prevent forgetting, and their performance relies on the quality of the generative replay samples. GAN-based methods provide slightly better results than the VAE-based method since the generative replay network using a GAN can produce higher-quality data samples compared with models using VAEs. From Table 3.2, we can also find that only the learning of the first

task would suffer from the degenerated performance caused by forgetting. When comparing with the results provided by the baselines, the proposed LGAA model achieves the best results on the first task, demonstrating that LGAA can generate high-quality samples which allow the inference model to predict the accurate class label for each sample, resulting in a small target risk on the first task. Additionally, LGAA also outperforms other baselines on the average classification accuracy.

In addition, we find that LGAN achieves similar performance compared with the proposed LGAA. The reason is that LGAN also employs the GAN as a generative replay mechanism, which can provide high-quality generative replay samples for lifelong learning. Therefore, LGAN can achieve competitive performance on the classification task compared with the proposed LGAA. However, LGAN lacks an inference mechanism and, therefore, can only perform the classification and image generation tasks. In contrast, one of the main advantages of the proposed LGAA over LGAN is the inference mechanism that enables LGAA to implement many downstream tasks, including image interpolation and disentangled representation learning.

Furthermore, the results from Table. 3.2 show that all models suffer a significant drop in performance when changing the lifelong learning setting from M-F to F-M. The main reason for this phenomenon is that the Fashion database is more complex than MNIST because it contains images of various clothing items, which are more complex than those of hand-written digits which make up MNIST. When the Fashion dataset is used as the first task to be learnt, we replay samples corresponding to Fashion when learning the second task in which the performance loss on Fashion is mainly caused by the generative replay mechanism. In contrast, when MNIST is used as the first task, we do not see a significant drop in performance after lifelong learning. This is because the MNIST is a simple dataset and the generative replay mechanism can produce more realistic images corresponding to MNIST after learning the database as the second task learning. These results also indicate that changing the order of tasks has an influence on the performance of LGAA, which is not addressed by the proposed model.

We also investigate the performance of the proposed approach when learning a long sequence of tasks, which involves several similar and dissimilar datasets. We train various models under lifelong learning of MNIST, SVHN, Fashion, InverseFashion, InverseMNIST and CIFAR10, namely MSFIIC. For InverseFashion and InverseMNIST, we inverse the pixels $\mathbf{x}$ from every image from the database as 255-$\mathbf{x}$. We report the classification results in Table 3.3. We can observe that the proposed LGAA achieves the best results for almost every task when compared

Table 3.3: Classification results under MSFIIC lifelong learning

Dataset	LGAA	LGM [132]	MeRGANs [176]
MNIST	86.79	86.14	82.08
SVHN	52.18	23.87	34.20
Fashion	64.37	55.00	61.20
InverseFashion	78.60	49.83	74.17
InverseMNIST	97.33	86.49	93.44
CIFAR10	52.36	57.09	58.49
Average	**71.94**	59.74	67.27

to the other methods. We can also observe that GAN-based models can relieve forgetting better than the VAE-based models outperforming the latter in terms of average classification accuracy.

3.4.5 Semi-Supervised Learning

For the semi-supervised training of LGAA, described in Section 3.3.2, we consider only a small number of labelled images from each database (1,000 for MNIST and 10,000 for Fashion) while the other images are not labelled (59,000 for MNIST and 50,000 for Fashion). The classification results following lifelong learning when using LGAA compared to other semi-supervised learning methods are provided in Table 3.5. These results show that the proposed approach LGAA outperforms LGAN [57], under the semi-supervised learning setting,

In the following, we train LGAN and LGAA under the lifelong learning of SVHN and Fashion, where we only access 1,000 and 10,000 labelled training samples for SVHN and Fashion, while the remaining 72,257 and 50,000, for SVHN and Fashion, respectively, are unlabelled. We report the results in Table 3.4, which shows that the proposed LGAA still outperforms LGAN for lifelong semi-supervised learning. In Tables 3.5 and 3.4, we include the results for methods which are only trained on one database, and we can observe that LGAA provides similar results to these, despite being at a significant disadvantage when learning successively two databases.

Since we consider the lifelong semi-supervised learning setting, the performance of LGAA and LGAN degenerates when learning a new task, due to the forgetting process. However, other baselines are trained for learning a single task where the model can access a set of labelled samples and all unlabelled samples during the training. There are no forgetting issues for such baselines. These results show that the proposed LGAA still achieves competitive results when compared to the state-of-the-art models that are not trained using lifelong learning, successively

Table 3.4: Semi-supervised classification error results on SVHN database, under the SVHN to Fashion lifelong learning.

Methods	Lifelong	Error
LGAA	Yes	60.36
LGAN [57]	Yes	63.25
kNN [83]	No	77.93
TSVN [83]	No	66.55
M1+KNN [83]	No	65.63
M1+TSVM [83]	No	54.33
M1+M2 [83]	No	36.02

Table 3.5: Semi-supervised classification error results on MNIST database, under the MNIST to Fashion lifelong learning.

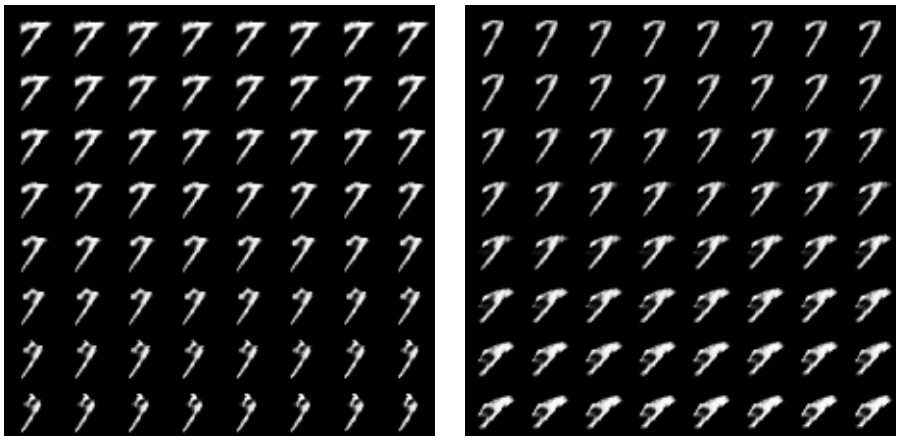
Methods	Lifelong	Error
LGAA	Yes	4.34
LGAN [57]	Yes	5.46
Neural networks (NN) [83]	No	10.7
Deep networks (CNN) [83]	No	6.45
TSVM [83]	No	5.38
CAE [83]	No	4.77
M1+TSVM [83]	No	4.24
M2 [83]	No	3.60
M1+M2 [83]	No	2.40
Semi-VAE [117]	No	2.88

on multiple databases but using conventional learning on a single database, which entails to a significant disadvantage for LGAA. These results also demonstrate that the proposed approach can be potentially used in other semi-supervised learning tasks.

3.4.6 Ablation Study

In this section, we investigate the importance of different latent variables used in the proposed LGAA model.

Choosing the categories of latent variables. First, we consider the proposed framework with only the continuous latent variable $\mathbf{z}$ as a baseline for comparison. Afterwards, we train the proposed framework with two inference models $\{\mathbf{z}, \mathbf{d}\}$ by considering the discrete variable $\mathbf{d}$, representing the domain, as well. We investigate whether using the task inference model can degenerate the performance of the proposed approach. The average reconstruction error (MSE) across all testing data is reported in Table 3.6. We also train a simple classifier on the reconstructions produced by the model on all training samples. Then we evaluate the



(a) Changing the first dimension of **z** from -2 to 2 .

(b) Changing the second dimension of **z** from -1 to 1.

Figure 3.11: Reconstruction results on MNIST when changing a single continuous latent variable while fixing all others.

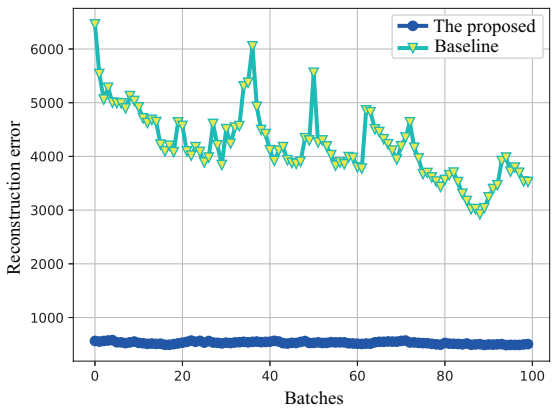
Table 3.6: Quantitative evaluation on the representation learning ability

MNIST and Fashion				
Methods	Lifelong	Dataset	Reconstruction error (MSE)	Accuracy (%)
LGAA	M-F	MNIST	4.75	92.53
Baseline	M-F	MNIST	4.71	91.29
LGAA	M-F	Fashion	17.44	67.66
Baseline	M-F	Fashion	16.54	67.97
LGAA	F-M	MNIST	4.92	93.29
Baseline	F-M	MNIST	5.14	92.34
LGAA	F-M	Fashion	13.16	66.97
Baseline	F-M	Fashion	14.78	66.45

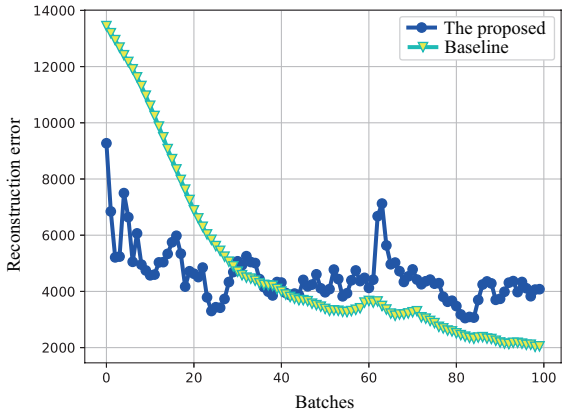
classification accuracy on all testing samples using the classifier. This test can reflect the quality of the reconstruction results and is reported in Table 3.6. We observe that the proposed LGAA model's performance, when considering task inference, does not deteriorate while the model learns the information from several databases.

Then we perform the task inference experiments under the lifelong learning of MNIST followed by the Fashion database, as well as the other way around under the F-M sequence. The results, when estimating the domain **d**, are reported in Table 3.7. We find that the task-inference model can infer, in most cases, the task ID for the given data. This result also demonstrates that the latent variable **z** captures the task and implicitly domain information, which enables the task-inference model $q_\varepsilon(\mathbf{d} \,|\, \mathbf{z})$ to make accurate predictions.

Enforcing the disentanglement between the latent variables z and u. We train the



(a) Average reconstruction error on CACD during the lifelong CelebA to CACD



(b) Average reconstruction error on 3D-Chair during the lifelong CelebA to 3D-Chair

Figure 3.12: Assessing the knowledge transferability, calculated using Eq. (3.21), for the LGAA model under the CelebA to CACD, and for CelebA to 3D-Chair lifelong learning. The average reconstruction errors are calculated based on samples from CACD and 3D-Chair datasets during the second task learning.

Table 3.7: Task inference accuracy after the lifelong learning of MNIST and Fashion.

MNIST and Fashion			
Methods	Lifelong	Dataset	Accuracy (%)
LGAA	M-F	MNIST	91.26
LGAA	M-F	Fashion	91.12
LGAA	F-M	MNIST	94.25
LGAA	F-M	Fashion	97.48

proposed model considering three latent vectors $\{\mathbf{u}, \mathbf{z}, \mathbf{d}\}$ under the lifelong supervised learning setting. After training, the inference model $q_\delta(\mathbf{u}\,|\,\mathbf{x})$ is used to make predictions. Then we change one dimension of the latent vector $\mathbf{z}$ inferred by $q_\varsigma(\mathbf{z}\,|\,\mathbf{x})$ while fixing the others. In Figures 3.11-a and 3.11-b, we show the results for the images showing the digit '7' from MNIST when changing the first and the second dimension of $\mathbf{z}$ within the ranges $[-2, 2]$ and $[-1, 1]$, respectively. From the results in Fig. 3.11 we observe that the latent variable $\mathbf{z}$ only represents the handwriting styles instead of the digit types in the images, which is modelled by the variable $\mathbf{u}$.

3.4.7 Transfer Metric and Transfer Learning

By using the generative replay mechanism, the proposed approach can accelerate the training speed for learning future tasks by transferring the previously learned knowledge. The transfer of knowledge between a past task and the currently given task is stronger when the new task

is related to the previously learned statistical representation. In this case, the model should be able to adapt quickly when learning the new task. In order to measure the task knowledge transferability in the network, we define the performance criterion for a model trained in the past when shown a newly given batch sample while learning a newly given i-th task:

$$F_\alpha(\mathbf{X}_{i,j}, f_{\theta_{i,j-1}}(\mathbf{X}_{i,j})), \quad j > 0\,. \tag{3.21}$$

Eq. (3.21) evaluates the performance for the j-th batch of training samples $\mathbf{X}_{i,j}$ on the i-th task (batch size is 64), achieved by the model which was trained with the $j-1$-th data batch from the i-th task. $F_\alpha(\cdot,\cdot)$ is a performance metric which can be implemented as the Mean Square Error (MSE) or by evaluating the classification accuracy, depending on the given task. $f_{\theta_{i,j-1}}(\cdot)$ is the model which has been updated using the $(j-1)$-th batch during the learning of the i-th task. Once a model f_θ has been trained on the $i-1$ task, we can use Eq. (3.21) to evaluate the capacity of transfer learning from one task (the $i-1$-th task) to another (the i-th task).

In the following, we train the proposed model under the CelebA to CACD and from the CelebA to 3D-Chair lifelong learning, respectively. Since we investigate task knowledge transferability, we consider a baseline that is only trained on a single dataset with randomly initialized parameters. Therefore, we consider training this baseline on a single dataset (CACD or CelebA) only and then evaluate the performance in each training step using Eq. (3.21). The baseline employs the same network architecture from LGAA. In contrast, we train the proposed LGAA in lifelong learning by evaluating its performance in each training step at the second task learning using Eq. (3.21) to investigate whether the previously learnt parameters are benefiting the learning of a new dataset. We use the MSE as the performance metric $F_\alpha(\cdot,\cdot)$ in Eq. (3.21) and the results are shown in Figures 3.12-a and 3.12-b for the CelebA to CACD database, and CelebA to 3D-Chair, respectively. From Fig. 3.12-a we observe that the model provides reasonable reconstruction errors in the initial training phase of the second task while the learning for the baseline proceeds rather slowly. This is due to the fact both CACD and CelebA are human face datasets, which means that they share similar facial feature information with each other. So the model can quickly adapt to a new task, which is similar to the previously learnt task, as we can observe from the decreasing average reconstruction errors during the learning steps. From Fig. 3.12-b, we observe that the proposed LGAA approach achieves lower reconstruction

errors than the baseline in the beginning stage of the training procedure. Then the baseline learns better than the proposed approach, while the proposed LGAA approach is actually a lifelong learning approach which also knows the information from the past learnt datasets, while aiming to learn the information from the current dataset as well. The reason behind this is that the human face image dataset shares few features with the 3D-chair images, which have a completely different appearance. The knowledge learned from CelebA is less likely to have a positive transferable effect when learning an entirely different dataset.

3.5 Conclusion and Limitations

In this chapter, we propose a new approach for lifelong learning, called the Lifelong Generative Adversarial Autoencoder (LGAA), which relies upon using the advantages of jointly using GAN and VAE generative deep learning methods into a unified lifelong learning framework. The proposed LGAA can learn meaningful representations across domains without forgetting them under lifelong learning. This approach can be used in a wide range of applications, including data classification, semi-supervised learning, reconstruction, generation and interpolation. In addition, we investigate the transfer learning ability of the proposed LGAA and find that the proposed LGAA can learn fast for a new task when the previously learnt tasks share similar features and information with respect to new data samples.

In the following, we summarise the limitations of the proposed LGAA into two aspects:

- The proposed LGAA model is still limited when learning an infinite number of tasks since the GAN would suffer from the mode collapse [156] when learning several entirely different datasets. This causes catastrophic forgetting for the proposed LGAA given that the model can not obtain generative replay samples of reasonably good quality.
- The proposed LGAA still requires knowing the maximum number of tasks before the training since $q_{\varepsilon}(\mathbf{d} \,|\, \mathbf{z})$ is a static inference model and can not dynamically increase the dimension of the latent variable $\mathbf{d}$.

Those limitations inspire us to develop a new lifelong learning approach enabled by the dynamic expansion mechanism for learning a growing number of tasks, described in the next chapter.

Chapter 4

Dynamic Growing Mixture Model

4.1 Introduction

Lifelong learning represents a learning paradigm in which a model is trained to successively learn a series of tasks without forgetting the information associated with any of these tasks. Modern deep learning models have been successfully used in a wide range of applications, including image translation [103], image processing [183], object recognition [135] and image synthesis [122]. Existing deep models tend to perform well only when accessing all training samples during a single learning stage. When attempting to learn multiple tasks, such models would only know the last task and would fail on those learnt in the past. The collapse in performance is caused by the fact that the parameters of the model are changed when training on a new task, and this phenomenon is called catastrophic forgetting [125].

One popular approach to relieve forgetting the Generative Replay Mechanism (GRM) [57], which can be implemented by either the Variational Autoencoder (VAE) [85], or the Generative Adversarial Net (GAN) [52]. A generative model aims to provide pseudo data when learning a new task in order to avoid forgetting by using GRM [2, 132, 141, 175, 180]. GRM-based models have shown good performance in the continual classification task but would gradually lose their performance when learning an increasing number of tasks. Additionally, the other drawback of GRMs is that of suffering from the mode collapse [156], especially when learning several entirely different data domains. To address these two problems, it was proposed to combine the dynamic expansion and GRM into a unified optimization framework [133] in which the model's capacity is increased to adapt to the data distribution shift. However, this approach still suffers from forgetting due to the repeated recursive use of many GRM processes. Other

attempts are focused on the dynamic expansion mechanism [95] and on using ensemble structures [43, 74, 174], in which each component is built on the top of a joint network backbone. These approaches can avoid forgetting since they do not update the previously learnt network parameters when learning new data [43]. However, most of the dynamic expansion models only focus on supervised learning and are not suitable for unsupervised image generation tasks.

After considering the limitations of previous research studies and the method described in Chapter 3, this chapter introduces a new approach that combines the generative replay and dynamic expansion mechanisms into a unified framework, aiming to address the challenges of lifelong learning from two aspects: network forgetting and complexity. First, the GRM-based methods usually see a reduction in the performance of all previous tasks when learning a new task. This result is caused by the generative replay process that repeatedly generates previous data during the new task learning while the quality of generative replay samples is gradually decreased as the number of tasks increases. To reduce the performance degeneration caused by GRM, we propose a dynamic expansion method that builds a new component while freezing the others to preserve the previously learnt information. Second, to reduce the model's complexity while maintaining its performance, we introduce a new knowledge-measure approach for evaluating the knowledge similarity between each previously learnt component and the data distribution of the dataset corresponding to the incoming task. The proposed knowledge measure approach chooses an appropriate component for learning a related task that shares similar visual concepts with the knowledge of the selected component. This approach reduces the accumulated errors caused by the GRM processes. Furthermore, accumulating knowledge into a single latent space using a student module, which can infer the information across domains, is an attractive feature in lifelong learning. Teacher-student architectures have not been explored so far in other lifelong mixtures or ensemble models [95, 182]. We adopt a knowledge distillation (KD) method for transferring the knowledge learnt by each component of the mixture learning system into a lightweight student module.

In this chapter, we summarise our contributions as follows:

- We propose a new lifelong learning model, which can dynamically expand its network architectures according to the novelty of the incoming task while reusing an existing component to learn a related task.
- We propose a new knowledge distillation approach that enables a lightweight student

module to learn the knowledge from the Generative Mixture Model (GMM) as well as from the new tasks. Furthermore, the student module can capture the cross-domain latent representations over time while enjoying fast inference during the testing phase.

- We perform several experiments, and the empirical results demonstrate that the proposed approach achieves the state of the art performance in both the generative modelling and classification tasks.

The rest of this chapter, discusses the methodology, including the architecture, learning algorithm and its implementation, in Section 4.2. The experiments are detailed in Section 4.3 and the conclusions from the research presented in this chapter are drawn in Section 4.4.

4.2 Methodology

The VAE framework has shown promising performance in lifelong generative modelling [2, 132] due to its powerful inference mechanism that supports many downstream tasks such as image reconstructions and interpolations. However, these works [2, 132] employ a single VAE framework, which fails to learn a growing number of tasks due to the fixed model capacity. In this section, we introduce a new VAE-based approach to address a long sequence of tasks. The key idea of the proposed approach is to learn a dynamic expansion model (mixture system) in which each component is implemented using a single VAE model which can be dynamically built when learning a new task. In order to reduce the model size, we propose to share many parameters among components. Specifically, these shared parameters are formulated as a shared encoder and decoder, respectively, which are only built once at the initial training phase and updated only at the first task learning to avoid forgetting, as described in Section 4.2.1. In addition, dynamically building and learning a new component for each task still leads to a considerable number of parameters, especially when learning a growing number of tasks. We address this issue by proposing a component expansion and selection mechanism, which evaluates the novelty of a new task as the signal for the model expansion, as described in Section 5.4.6. The key idea of the proposed mechanism is to calculate the difference in the loss value between the generated images from each component and the training samples from the new task. A high measure indicates that the new task contains different information from the already learnt knowledge and we can build a new component to learn this new task. In contrast, a low measure evaluated by

the expansion criterion will choose an appropriate component to learn this new task through the component selection process, which can reduce the model size. In addition, based on the mixture of the VAE framework, we extend the proposed GMM to implement two applications: classification and image-to-image translation, as described in Section 4.2.4. Specifically, we employ the concept of Conditional VAE (CVAE) [153] to implement each component of GMM since CVAE can enable prediction tasks within a VAE framework.

However, one weakness of the proposed GMM framework is the model size, which will be increased over time when learning a growing number of tasks. Such a design can not deploy the proposed GMM into a resource-constrained machine. In addition, since each VAE-based component in the proposed GMM only embeds information from the associated tasks into its own latent space, we can not perform the image interpolation across multiple data domains due to lacking of a shared latent space. To solve these issues, one efficient approach is to formulate the proposed GMM as the teacher module and transfer its learned knowledge to a fixed-size student module through knowledge distillation, described in Section 4.2.5. Specifically, we implement the student module using a single VAE model, aiming to embed the information from all previously learnt tasks into a shared latent space, which supports image interpolations across different data domains over time. The other advantage of introducing the student module is to enjoy a fast inference process during the testing phase.

Learning setting. In this chapter, we focus on a general setting in lifelong learning where the task label is provided during the training [125]. Let $\mathcal{X}$ and $\mathcal{Z}$ represent the data and latent variable space, respectively. Let us consider the learning of a sequence of N tasks, where each task (the t-th task) is assigned with a training set $D_S^t = \{\{\mathbf{x}_j, \mathbf{y}_j\}\}_{j=1}^{N_S^t}$ and a testing set $D_T^t = \{\{\mathbf{x}_j, \mathbf{y}_j\}\}_{j=1}^{N_T^t}$. $\mathbf{x}_j \in \mathcal{X}$ and $\mathbf{y}_j \in \mathcal{Y}$ (one-hot vector) are an observed variable, and the target variable (class label), where $\mathcal{Y}$ is the space of the class label. N_S^t and N_T^t are the total number of samples for D_S^t and D_T^t, respectively. In addition, let $D_{SU}^t = \{\mathbf{x}_j\}_{j=1}^{N_S^t}$ and $D_{TU}^t = \{\mathbf{x}_j\}_{j=1}^{N_T^t}$ be the unlabelled training and testing dataset from the t-th task. Let $p(\mathbf{x}_{TU}^t)$ and $p(\mathbf{x}_{SU}^t)$ denote the data distribution[1] for D_{TU}^t and D_{SU}^t, respectively[2]. Our goal in lifelong learning is to learn a model that only accesses samples from the associated training set (D_S^t or D_{SU}^t) in the t-th task learning while it cannot access data corresponding to the previously

[1]Following from the original GAN paper [52] and other extensive works [55, 11], we employ the concept of the data distribution.

[2]Since the dataset size of D_{TU}^t and D_{SU}^t is equal to that of D_T^t and D_S^t, we employ N_T^t and N_S^t to denote the total number of samples for D_{TU}^t and D_{SU}^t, respectively.

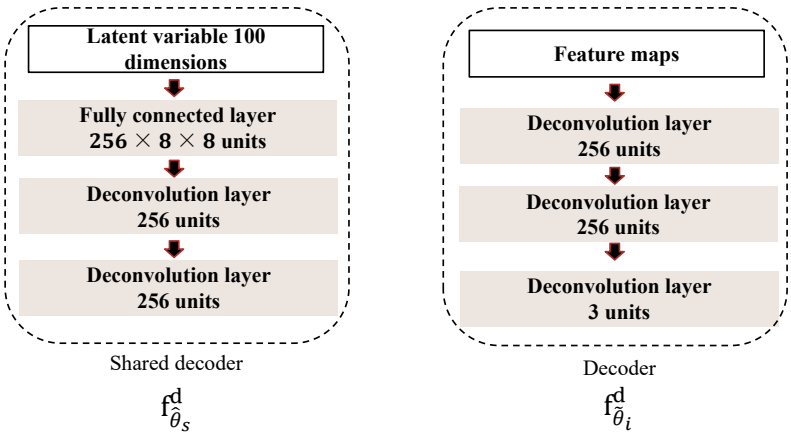


Figure 4.1: The network architecture of the generator under unsupervised learning.

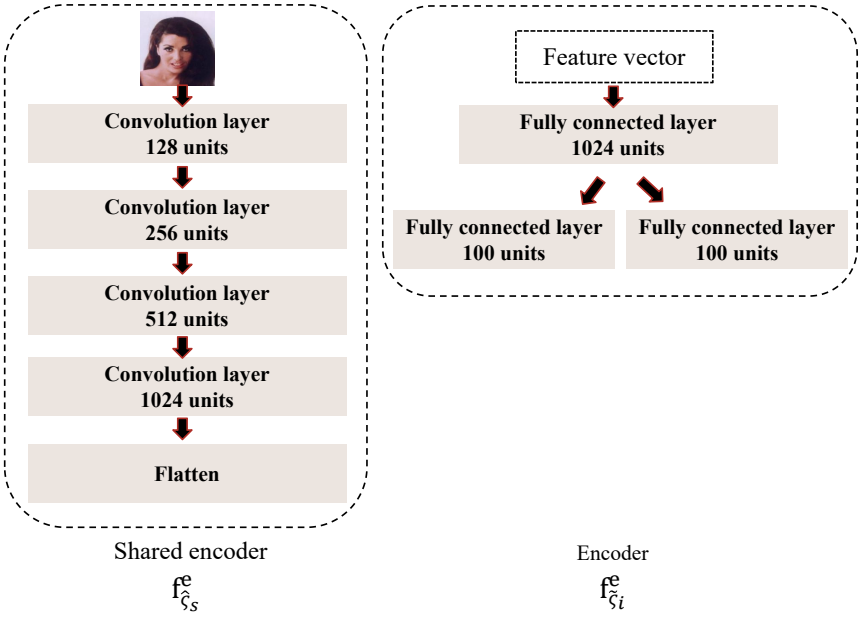


Figure 4.2: The network architecture of the encoder under unsupervised learning.

learnt $(t-1)$ tasks. Once all tasks are completed, we evaluate the performance of the trained model on all testing sets ($\{D_T^1, \cdots, D_T^N\}$ or $\{D_{TU}^1, \cdots, D_{TU}^N\}$), where N represents the number of tasks.

To relieve network forgetting, a mixture system can be adopted. The optimal performance on the target datasets can be achieved when the number of components in the mixture model matches the number of tasks because the model does not suffer from the forgetting caused by the generative replay mechanism. However, such a mixture system would require using many additional network parameters and is not scalable for learning a growing number of tasks. Inspired by this conclusion, we develop a new Growing Mixture Model (GMM) that satisfies two aspects. First, we address forgetting in lifelong learning by preserving the learned knowledge in the frozen network structure while dynamically building and adding and training new compo-

nents to the growing mixture, for learning new tasks. Second, to reduce the complexity of the model while maintaining performance, we share some of the parameters between these components and introduce a new knowledge-measure approach that guides the GMM to reuse some of the network parameters for learning a related task through a component selection process.

4.2.1 Network Architecture

In this section, we detail the proposed GMM's network architecture. In order to reduce the whole model's parameters, GMM introduces a shared module that consists of two neural networks $\mathrm{f}^{\mathrm{e}}_{\hat{\varsigma}_s} : \mathcal{X} \to \mathcal{Z}'$ and $\mathrm{f}^{\mathrm{d}}_{\hat{\theta}_s} : \mathcal{Z} \to \mathcal{X}'$, corresponding to the shared VAE encoder and decoder[3], respectively, where $\mathcal{X}$ and $\mathcal{Z}'$ are the space of the image $\mathbf{x}$ and the output (feature vector) of $\mathrm{f}^{\mathrm{e}}_{\hat{\varsigma}_s}$. The dimension of $\mathcal{Z}'$ is larger than $\mathcal{Z}$ that is the space of the latent variable $\mathbf{z}$. $\mathcal{X}'$ is the output space of $\mathrm{f}^{\mathrm{d}}_{\hat{\theta}_s}$, which has smaller dimensions than the data space $\mathcal{X}$. $\{\hat{\varsigma}_s, \hat{\theta}_s\}$ are the parameters of the shared module. When GMM expands, we dynamically build a component-specific module that consists of two neural networks $\mathrm{f}^{\mathrm{e}}_{\tilde{\varsigma}_i}$ and $\mathrm{f}^{\mathrm{d}}_{\tilde{\theta}_i} : \mathcal{X}' \to \mathcal{X}$, corresponding to the specific VAE encoder and decoder, where i represents the component index and $\{\tilde{\varsigma}_i, \tilde{\theta}_i\}$ is the parameter set. $\mathrm{f}^{\mathrm{e}}_{\tilde{\varsigma}_i}$ receives the output from the shared encoder $\mathrm{f}^{\mathrm{e}}_{\hat{\varsigma}_s}(\mathbf{x})$ and returns Gaussian hyperparameters. Therefore, the encoder for the i-th component is implemented by $\mathrm{f}^{\mathrm{e}}_{\tilde{\varsigma}_i}(\mathrm{f}^{\mathrm{e}}_{\hat{\varsigma}_s}((\mathbf{x})))$ which outputs the Gaussian hyperparameters $\{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$ used to form an encoding distribution $q_{\{\hat{\varsigma}_s, \tilde{\varsigma}_i\}}(\mathbf{z} \,|\, \mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$ (Gaussian), where $\{\hat{\varsigma}_s, \tilde{\varsigma}_i\}$ denotes the specific and shared parameters. Similarly to the encoder, we implement the decoder $p_{\{\hat{\theta}_s, \tilde{\theta}_i\}}(\mathbf{x} \,|\, \mathbf{z})$ for the i-th component by $\mathrm{f}^{\mathrm{d}}_{\tilde{\theta}_i}(\mathrm{f}^{\mathrm{d}}_{\hat{\theta}_s}(\mathbf{z}))$. The network architecture of the decoder and encoder for one VAE component is shown in Fig. 4.1 and Fig. 4.2. We also provide the overall structure of the proposed GMM in Fig. 4.3, where each component can be seen as a single VAE model. Let $\mathcal{A}_i$ to denote the i-th component consisting of $q_{\{\hat{\varsigma}_s, \tilde{\varsigma}_i\}}(\mathbf{z} \,|\, \mathbf{x})$ and $p_{\{\hat{\theta}_s, \tilde{\theta}_i\}}(\mathbf{x} \,|\, \mathbf{z})$, and the optimization corresponds to maximize the Evidence Lower Bound (ELBO) [85]:

$$\begin{aligned} \mathcal{L}_{ELBO}(\mathbf{x}, \hat{\theta}_s, \tilde{\theta}_i, \hat{\varsigma}_s, \tilde{\varsigma}_i) = \mathbb{E}_{\mathbf{z} \sim q_{\{\hat{\varsigma}_s, \tilde{\varsigma}_i\}}(\mathbf{z} \,|\, \mathbf{x})} \left[\log p_{\{\hat{\theta}_s, \tilde{\theta}_i\}}(\mathbf{x} \,|\, \mathbf{z}) \right] \\ - D_{KL} \left[q_{\{\hat{\varsigma}_s, \tilde{\varsigma}_i\}}(\mathbf{z} \,|\, \mathbf{x}) \,\|\, p(\mathbf{z}) \right] , \end{aligned} \tag{4.1}$$

[3]We employ $\hat{\theta}_s$ and $\hat{\varsigma}_s$ to denote the parameters of the shared module. The goal of the shared VAE encoder and decoder is to output the feature vectors, which are used as input for the specific encoder and decoder.

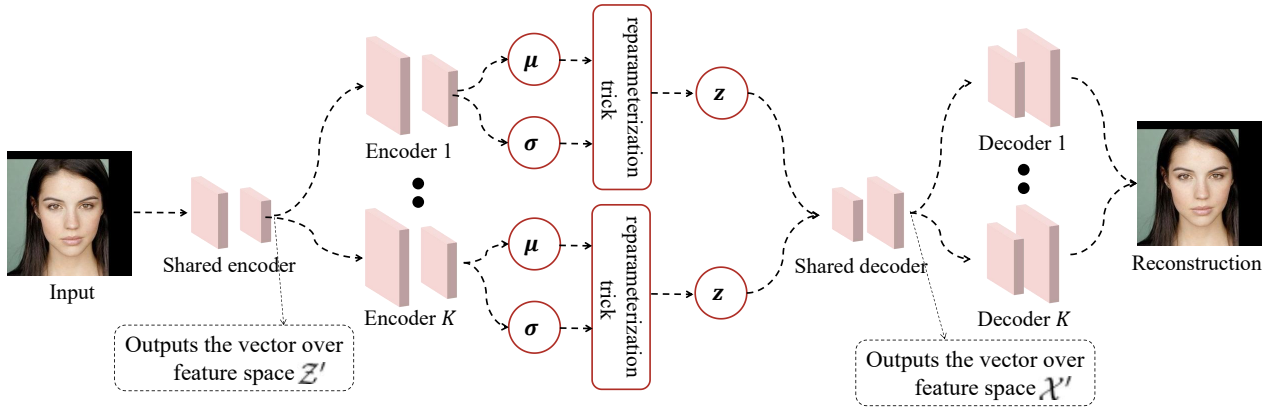


Figure 4.3: The network architecture of the proposed GMM consisting of K components. Each component can be seen as a single VAE model.

where the shared modules are only updated by using Eq. (4.1) for the first task learning in order to avoid forgetting the prior knowledge in the subsequent task learning. In practice, the first term in Eq. (4.1) is implemented as the negative reconstruction error [148].

4.2.2 Component Selection and Mixture Expansion Mechanism

When the number of components is equal to that of tasks, the learning errors are not cumulating successively after learning several tasks. However, such a learning paradigm requires to train a task-specific component of a mixture system, which would lead to substantial memory budgets as the number of tasks grows. To address this problem, we introduce a knowledge-evaluation approach that guides component selection and mixture expansion. The primary motivation is that a certain component can easily learn several similar tasks and benefit from positive knowledge transfer when learning a new task. To achieve this goal, a reasonable solution is for a given component to learn tasks of a similar nature, by evaluating a discrepancy distance on the data corresponding to different tasks.

First we can consider estimating the discrepancy distance [114] between the new task and each of the previously trained components. A high discrepancy distance indicates that the new task is novel enough, and that we should build a new component for learning this task. The discrepancy distance between two distributions $p(\mathbf{x}^i_{SU})$ and $p(\mathbf{x}^j_{SU})$ is defined as, [114]:

$$\begin{aligned}\mathcal{L}_{\text{discrepancy}}\left(p(\mathbf{x}^i_{SU}), p(\mathbf{x}^j_{SU})\right) = \sup_{(h,h')\in\mathcal{H}^2} \Big| & \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}^i_{SU})}\left[\mathcal{L}\left(h'\left(\mathbf{x}\right), h\left(\mathbf{x}\right)\right)\right] \\ & - \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}^j_{SU})}\left[\mathcal{L}\left(h'\left(\mathbf{x}\right), h(\mathbf{x})\right)\right]\Big|,\end{aligned} \tag{4.2}$$

where $\mathcal{H}$ is the space of the classifiers. In practice, the discrepancy value in Eq. (4.2) can be estimated by training two classifiers h and h' on two labelled datasets D_S^i and D_S^j, respectively. However, such an approach requires classifiers that are assumed to be trained on the labelled dataset, which is not available in unsupervised learning. In this study, we introduce an alternative approach to estimate task similarity, which does not require extra neural networks and computational costs. We assume that at the t-th task learning, we have already learned K components. When we see a new task (the $(t+1)$-th task), we propose to compare the loss value between each learned component and the new task (the $(t+1)$-th task):

$$F_S(\mathcal{A}_i, t+1, D_{SU}^{t+1}) = \frac{1}{m'} \sum_{k=1}^{m'} \left| \mathcal{L}_{ELBO}(\mathbf{x}'_{(i,k)}, \hat{\theta}_s, \tilde{\theta}_i, \hat{\varsigma}_s, \tilde{\varsigma}_i) - \mathcal{L}_{ELBO}(\mathbf{x}_{(t+1,k)}, \hat{\theta}_s, \tilde{\theta}_i, \hat{\varsigma}_s, \tilde{\varsigma}_i) \right|, \quad (4.3)$$

where m' is the number of samples that are used for the evaluation and $\mathcal{A}_i$ is the i-th component. $|\cdot|$ denotes the absolute value. D_{SU}^{t+1} is the unlabelled training dataset from the $(t+1)$-th task. In practice, we set $m' = 5000$ for all experiments. Since we can not access all past samples, we use each component to generate pseudo sample $\mathbf{x}'_{(i,k)}$, where i represents the component index. $\mathbf{x}_{(t+i,k)}$ represents the k-th real training sample obtained from the unlabelled training set of the $(t+1)$-th task. Eq (4.3) compares the given data with the new task (the $t+1$-th task) with the information already known by the model. We test the loss value provided by the mode with the newly given samples $X_{(t+1,k)}$ with the that obtained with the samples generated by the model. This measures the novelty of the new data for the model.

The main reason for comparing the losses can be summarized into two aspects. Firstly, since each component in the GMM is a VAE model, the first term in the loss function is implemented using the reconstruction error. For a given image that was not seen before and was entirely different from the previously learnt images, the VAE model would not produce an accurate image reconstruction, resulting in a large reconstruction error. As a result, comparing the loss value between the generated images and new training samples can indicate the novelty of the data corresponding to a new task. Secondly, comparing the loss value is an efficient approach as it does not require significant computational costs compared to other distance measures, while is also easy to implement. Eq. (4.3) can measure the knowledge similarity between each component and the new task, which can be used for selecting a component or for deciding when to expand the architecture with a new component:

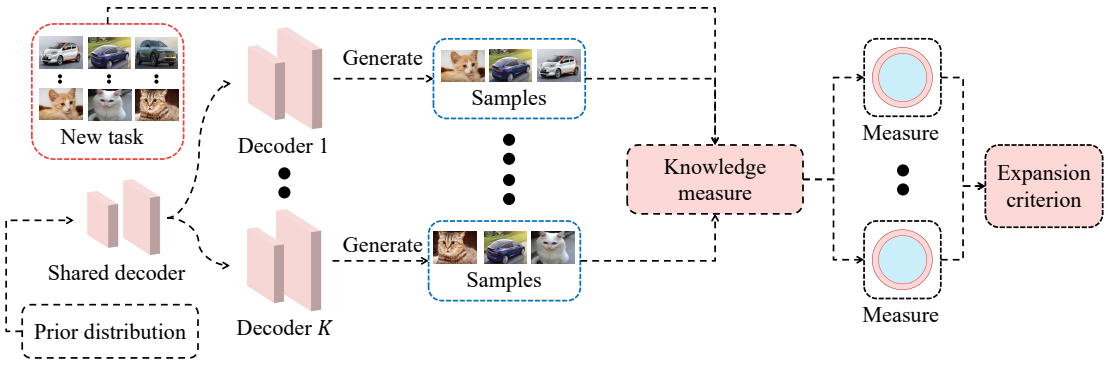


Figure 4.4: The procedure of the proposed knowledge measure approach. When seeing a new task, we generate a set of samples using each component. Then these generated samples and real samples from the new task are used for checking the model expansion (Eq. (4.4)).

$$\min_{i=1,\cdots,K} \left\{ F_S(\mathcal{A}_i, t+1, D_{SU}^{t+1}) \right\} \geq \lambda^{\text{GMM}}, \tag{4.4}$$

where λ^{GMM} is a threshold which controls the model size of the GMM and K is the current number of components. The procedure of the proposed knowledge measure is illustrated in Fig. 4.4. If Eq. (4.4) is satisfied, the GMM will add a new component for learning the $(t+1)$-th task, otherwise, it will select a component according to the knowledge measure:

$$F_{\text{find}}^{\text{GMM}}(\mathcal{A}_1, \cdots, \mathcal{A}_K, t+1, D_{SU}^{t+1}) = \arg\min_{i=1,\cdots,K} \left\{ F_S(\mathcal{A}_i, t+1, D_{SU}^{t+1}) \right\}, \tag{4.5}$$

where $\widehat{s} = F_{\text{find}}^{\text{GMM}}(\mathcal{A}_1, \cdots, \mathcal{A}_K, t+1, D_{SU}^{t+1})$ is the selected component index. Then we choose the $\widehat{s}$-th component for learning the $(t+1)$-th task with the generative replay process.

The choice of λ^{GMM} in the expansion criterion, defined by Eq. (4.4), represents a trade-off between model complexity (size) and performance. For example, if λ^{GMM} is very large, the GMM tends to reuse a component for learning a new task more frequently, while the performance of previous tasks would be reduced due to the repetitive usage of the generative reply mechanism (GRM). On the other hand, when λ^{GMM} is very small, GMM tends to create more components, increasing the complexity of the model while reducing the accumulated errors caused by the GRM processes.

4.2.3 Algorithm Implementation

We present the pseudocode in **Algorithm 2**, for the unsupervised learning algorithm for GMM, which is summarized in three steps in the following:

- **Step 1. Training phase:** If GMM has no components, we build a new component at the beginning of the first task learning, otherwise, we check the flag 'isAdd' in **Algorithm 2** to decide either the mixture expansion or the component selection process. If the flag "isAdd == True" at the t-th task learning, we dynamically build a new component and train it on D_{SU}^{t}, otherwise, we train the selected component ($\widehat{s}$-th component) on the joint dataset $D_{SU}^{t} \bigcup D_{GU}^{\widehat{s}}$ consisting of the training and generated dataset. Note that we use D_{SU}^{t} to denote the unlabelled dataset, which is different from the labelled dataset (D_{S}^{t}). $D_{GU}^{\widehat{s}}$ is the unlabelled dataset generated by using the selected component ($\mathcal{A}_{\widehat{s}}$). The total number of training iterations ($iterationNumber$ in **Algorithm 2**) is determined by $iterationNumber = (dataSize/batchSize) \times epoch$, where $batchSize = 64$ and $epoch = 20$ are the batch size and training epochs, respectively. $dataSize$ is the total number of samples for the current training dataset.

- **Step 2. The evaluation of knowledge measure:** After the current task learning (the t-th task) is finished, we evaluate the knowledge comparison measure between each learnt component and the dataset of the next task (the $(t+1)$-th task) by using Eq. (4.3).

- **Step 3. The expansion and selection process:** We employ the knowledge measure to decide either the expansion or the selection process, controlled by an expansion threshold λ^{GMM}. If Eq. (4.4) is satisfied, then we set the flag 'isAdd = True', otherwise, we set 'isAdd = False' in **Algorithm 2**. We then return back to **Step 1** to perform the next task learning.

4.2.4 Supervised Learning Task

Although the proposed GMM mainly focuses on the unsupervised generative modelling task, it can be extended to two applications: image classification and image-to-image translation (ITIT). Therefore, we only modify the network architecture of individual components to implement these applications, and the training process is similar to that required in unsupervised learning (**Algorithm 2**). In the following, we introduce the detailed implementation for two applications. Note that we usually use θ and ς to represent the parameters of the generator (decoder) and the encoder for both applications for the sake of reducing the number of notations.

Algorithm 2: Unsupervised learning of GMM

(Input:All training databases) ;
for $t < N$ **do**
  if $t == 1$ **then**
    isAdd = True ;
  end
  Get the unlabelled training set D_{SU}^{t} ;
  if isAdd == *False* **then**
    Generate the dataset $D_{GU}^{\widehat{s}}$ using the $\widehat{s}$-th component ;
    From a joint dataset $D_{SU}^{t} = D_{SU}^{t} \bigcup D_{GU}^{\widehat{s}}$;
  end
  else
    Build a new component for GMM ;
  end
  for *index* $<$ *iterationNumber (Only update each model once in each iteration)* **do**
    Get the data batch $\mathbf{X}_{batch}$ from the training set D_{SU}^{t} ;
    if isAdd == *False* **then**
      Update the selected component with $\mathbf{X}_{batch}$ using $\mathcal{L}_{ELBO}(\mathbf{x}, \hat{\theta}_s, \tilde{\theta}_{\widehat{s}}, \hat{\varsigma}_s, \tilde{\varsigma}_{\widehat{s}})$
      $\widehat{s}$ is the selected component index ;
    end
    else
      Update the new component with $\mathbf{X}_{batch}$ using $\mathcal{L}_{ELBO}(\mathbf{x}, \hat{\theta}_s, \tilde{\theta}_{K+1}, \hat{\varsigma}_s, \tilde{\varsigma}_{K+1})$;
    end
  end
  The evaluation of knowledge measure ;
  Calculate the knowledge measure using Eq. (4.3) ;
  if $\min_{i=1,\cdots,K} \left\{F_S(\mathcal{A}_i, t+1, D_{SU}^{t+1})\right\} \geq \lambda^{\text{GMM}}$, **then**
    isAdd = True ;
  end
  else
    Select a component ;
    $\widehat{s} = \arg\min_{i=1,\cdots,K} \left\{F_S(\mathcal{A}_i, t+1, D_{SU}^{t+1})\right\}$;
    isAdd = False ;
  end
end

Image to Image Translation (ITIT) task. Unlike the image reconstruction task [85], which receives an original image and aims to give an accurate reconstruction, the ITIT aims to predict and generate a target image for a given source image [71]. To implement the image-to-image translation task, we need to redesign the network architecture of each component in the GMM. Conditional VAE (CVAE) [153] is a well-known variant of VAEs, which enables to make prediction tasks. We employ the concept of the CVAE for implementing ITIT for two reasons: (1) The CVAE is a VAE-based model, which can be used as the component/expert in the GMM; (2) The CVAE is mainly used for the prediction task, which is suitable for ITIT. The loss

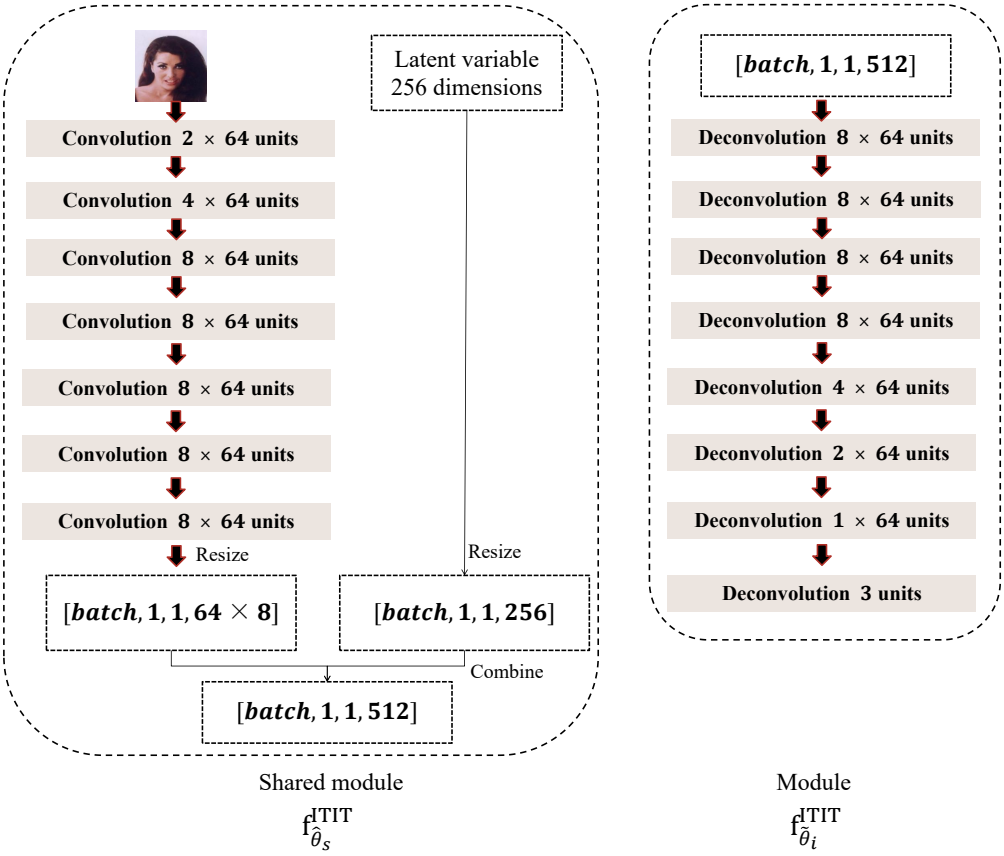


Figure 4.5: The network architecture of the encoding-decoding network under the image-to-image translation task, where input size is $256 \times 256 \times 3$. "batch" is the batch size, which is set to 8 in our experiment.

function of the CVAE is defined by:

$$\begin{aligned}\mathcal{L}_{\text{CVAE}}(\mathbf{g}, \mathbf{x}, \hat{\varsigma}_s, \tilde{\varsigma}_i, \hat{\theta}_s, \tilde{\theta}_i) = &-\mathbb{E}_{q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}(\mathbf{z}\,|\,\mathbf{x},\mathbf{g})}[\log p_{\{\hat{\theta}_s,\tilde{\theta}_i\}}(\mathbf{g}\mid\mathbf{x},\mathbf{z})] \\ &+ D_{KL}\left[q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}(\mathbf{z}\mid\mathbf{x},\mathbf{g}) \,||\, p_{\{\hat{\upsilon}_s,\tilde{\upsilon}_i\}}(\mathbf{z}\mid\mathbf{x})\right],\end{aligned} \tag{4.6}$$

where $p_{\{\hat{\upsilon}_s,\tilde{\upsilon}_i\}}(\mathbf{z}\mid\mathbf{x})$ represents the prior network that receives $\mathbf{x}$ and returns $\mathbf{z}$ [153]. $p_{\{\hat{\theta}_s,\tilde{\theta}_i\}}(\mathbf{g}\mid\mathbf{x},\mathbf{z})$ and $q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}(\mathbf{z}\mid\mathbf{x},\mathbf{g})$ are the generation and recognition network [153], respectively, where $\mathbf{g}$ belongs to the image domain and has the same size with $\mathbf{x}$ in the image-to-image translation task. Although the original objective function of CVAE, defined by Eq. (4.6), can ensure a lower bound of the conditional log-likelihood [153], implementing each component of the GMM by using the CVAE would lead to more parameters and computational complexity, especially when the GMM continually builds several components during lifelong learning. In this chapter, we aim to design a lightweight network architecture and modified objective function for each component to minimize the overall computational complexity. We find that the prior network $p_{\{\hat{\upsilon}_s,\tilde{\upsilon}_i\}}(\mathbf{z}\mid\mathbf{x})$ requires processing the high-dimensional data $\mathbf{x}$ using a deep neural network

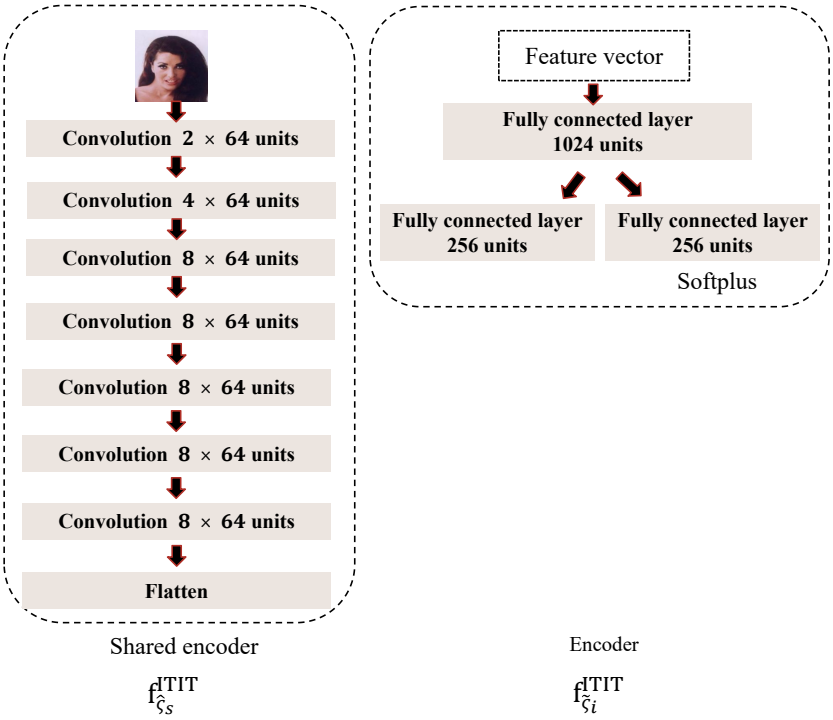


Figure 4.6: The network architecture of the encoder under the image-to-image translation task.

with numerous parameters. Additionally, when the variable $\mathbf{g}$ is a high-dimensional data in the image-to-image translation task, $q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}(\mathbf{z} \mid \mathbf{x}, \mathbf{g})$ would lead to more computational complexity as its input involves two high-dimensional data $\mathbf{x}$ and $\mathbf{g}$.

To reduce the total number of parameters for each component in the GMM, we firstly consider replacing $q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}(\mathbf{z} \mid \mathbf{x}, \mathbf{g})$ by employing an encoder $q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}(\mathbf{z} \mid \mathbf{g})$, which only processes a certain high-dimensional data and forms a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}, \boldsymbol{\sigma}^2_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}\mathbf{I})$ where $\{\boldsymbol{\mu}_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}, \boldsymbol{\sigma}_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}\}$ are hyperparameters. Then, we consider replacing the prior network $p_{\{\hat{\theta}_s,\tilde{\theta}_i\}}(\mathbf{z} \mid \mathbf{x})$ [153] by adopting a normal distribution $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ which does not involve any trainable parameters, further reducing the model size. As image-to-image translation is challenging in lifelong learning, we implement $p_{\{\hat{\theta}_s,\tilde{\theta}_i\}}(\mathbf{g} \mid \mathbf{x}, \mathbf{z})$ by employing an encoding-decoding network $f^{ITIT}_{\hat{\theta}_s}(f^{ITIT}_{\tilde{\theta}_i}(\mathbf{x}, \mathbf{z}))$ whose network architectures are shown in Fig. 4.5, where $f^{ITIT}_{\hat{\theta}_s}$ and $f^{ITIT}_{\tilde{\theta}_i}$ are the shared and individual module, respectively. $q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}(\mathbf{z} \mid \mathbf{g}) = \mathcal{N}(\boldsymbol{\mu}_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}, \boldsymbol{\sigma}^2_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}\mathbf{I})$ is a Gaussian distribution implemented by an encoding network $f^{ITIT}_{\tilde{\varsigma}_i}(f^{ITIT}_{\hat{\varsigma}_s}((\mathbf{g})))$ where $f^{ITIT}_{\hat{\varsigma}_s}(.)$ is the shared encoder and $f^{ITIT}_{\tilde{\varsigma}_i}(\cdot)$ receives the output from $f^{ITIT}_{\hat{\varsigma}_s}(\mathbf{g})$ and returns the Gaussian hyperparameters $\{\boldsymbol{\mu}_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}, \boldsymbol{\sigma}_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}\}$. We provide the detailed network architectures of $f^{ITIT}_{\tilde{\varsigma}_i}(\cdot)$ and $f^{ITIT}_{\hat{\varsigma}_s}(\mathbf{g})$ in Fig. 4.6.

Each component $\mathcal{A}_i$ in the image to image translation task consists of $p_{\{\hat{\theta}_s,\tilde{\theta}_i\}}(\mathbf{g} \mid \mathbf{x}, \mathbf{z})$ and $q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}(\mathbf{z} \mid \mathbf{g})$, where i represents the component index. Then, Eq. (4.6) is reformulated to be

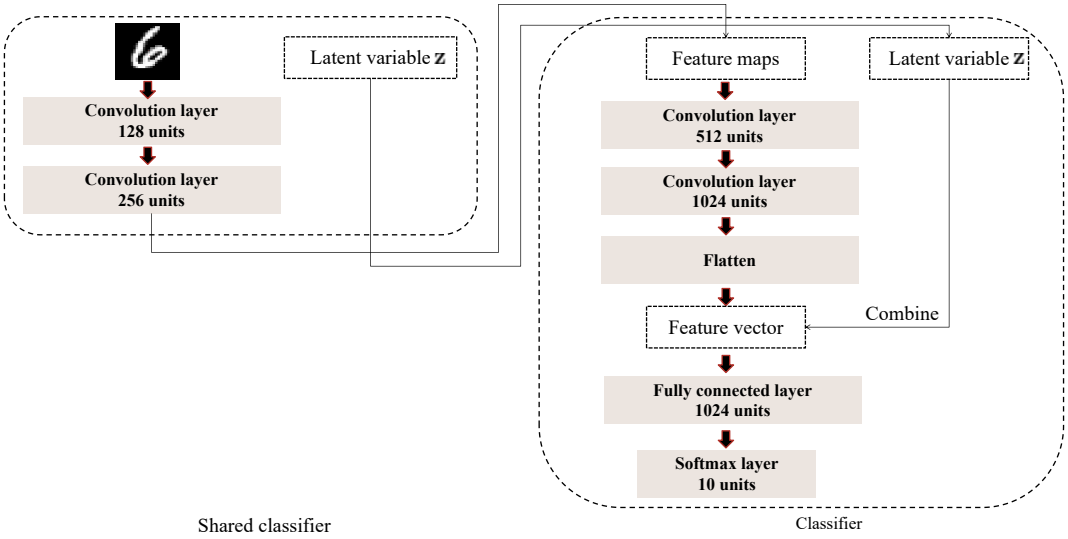


Figure 4.7: The network architecture of the classifier under the classification task.

used as the objective function for the i-th component in the Image-to-Image translation task, $\mathcal{L}_{\text{ITIT}}(\mathbf{x}, \mathbf{g}, \hat{\varsigma}_s, \tilde{\varsigma}_i, \hat{\theta}_s, \tilde{\theta}_i)$:

$$\begin{aligned}\mathcal{L}_{\text{ITIT}}(\mathbf{x}, \mathbf{g}, \hat{\varsigma}_s, \tilde{\varsigma}_i, \hat{\theta}_s, \tilde{\theta}_i) = &-\mathbb{E}_{q_{\{\hat{\varsigma}_s, \tilde{\varsigma}_i\}}(\mathbf{z} \mid \mathbf{g})}[\log p_{\{\hat{\theta}_s, \tilde{\theta}_i\}}(\mathbf{g} \mid \mathbf{x}, \mathbf{z})] \\ &+ D_{KL}[q_{\{\hat{\varsigma}_s, \tilde{\varsigma}_i\}}(\mathbf{z} \mid \mathbf{g}) \mid\mid p(\mathbf{z})]\,.\end{aligned} \tag{4.7}$$

The first term in Eq. (4.7) is implemented by the negative reconstruction error, which has been widely used in other works [148]. Compared to the original objective function, defined by Eq. (4.6), our objective function, defined by Eq. (4.7) can not ensure a lower bound of the conditional log-likelihood. We can observe that $\mathbf{g}$ in Eq. (4.7) belongs to the image domain, and i is the component index. Compared to the unsupervised image reconstruction task, the generator $p_{\{\hat{\theta}_s, \tilde{\theta}_i\}}(\mathbf{g} \mid \mathbf{x}, \mathbf{z})$ in the image-to-image translation task can only map the source image $\mathbf{x}$ to the target image $\mathbf{g}$ and is unable to generate paired images $\{\mathbf{x}, \mathbf{g}\}$. As a result, each component in the image-to-image generation task fails to implement the GRM procedure. Reusing the same component to continually learn several tasks would suffer from catastrophic forgetting due to the absence of generative replay samples. To avoid this issue, the GMM in the image-to-image translation task dynamically adds a new component when seeing a new task (Each component learns a unique task.).

Classification task. For the classification task, we consider $\mathbf{u}$ to be the discrete variable (one-hot vector) representing the class label. Therefore, we implement $p_{\{\hat{\delta}_s, \tilde{\delta}_i\}}(\mathbf{u} \mid \mathbf{x}, \mathbf{z})$ as a classifier which also has the shared parameters $\hat{\delta}_s$ and the component-specific parameters $\tilde{\delta}_i$. We provide

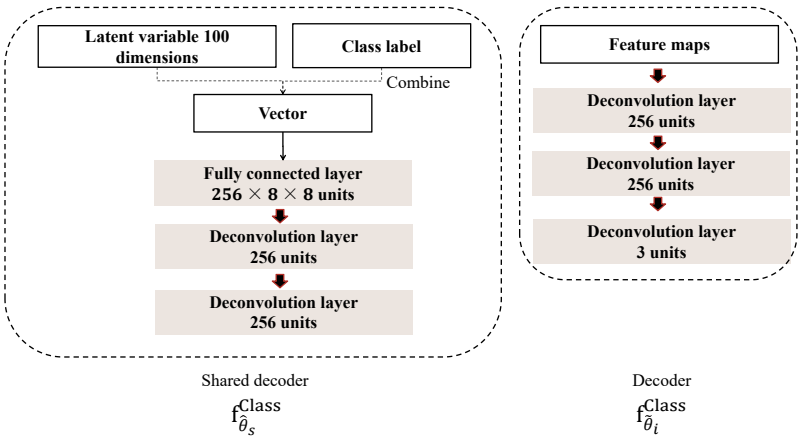


Figure 4.8: The network architecture of the generator under the classification task.

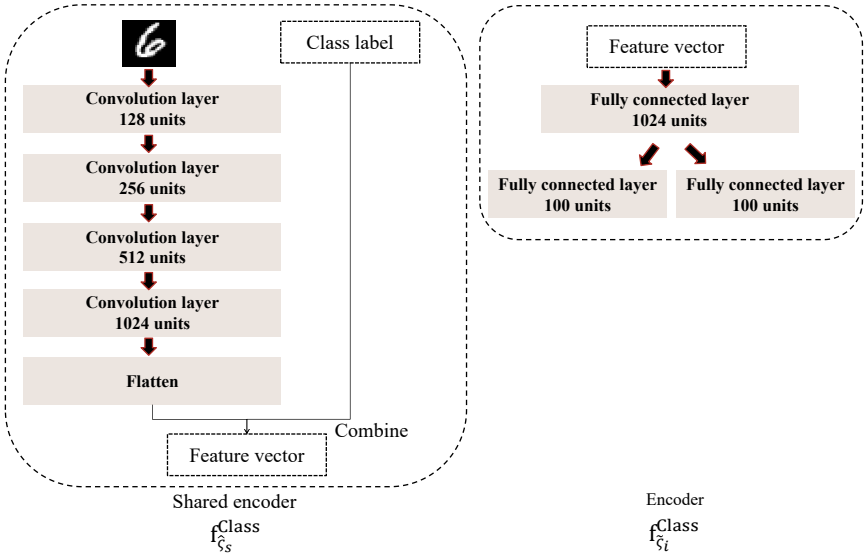


Figure 4.9: The network architecture of the encoder under the classification task.

the network architecture of the classifier in Fig. 4.7. In order to reduce the whole model size, we also replace $p_{\{\hat{\upsilon}_s,\tilde{\upsilon}_i\}}(\mathbf{z} \mid \mathbf{x})$ of Eq. (4.6) by a simple normal distribution $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, and then the objective function used for training the classifier of the i-th component is defined as:

$$\begin{aligned}\mathcal{L}_{\mathrm{C}}(\mathbf{x}, \mathbf{y}, \hat{\varsigma}_s, \tilde{\varsigma}_i, \hat{\delta}_s, \tilde{\delta}_i) = f_{\mathrm{CE}}(p_{\{\hat{\delta}_s,\tilde{\delta}_i\}}(\mathbf{u} \,|\, \mathbf{x}, \mathbf{z}), \mathbf{y}, Q) \\ + D_{KL}[q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}(\mathbf{z} \mid \mathbf{x}, \mathbf{y}) \,||\, p(\mathbf{z})]\,,\end{aligned} \tag{4.8}$$

where the first term is the cross-entropy loss and $\mathbf{z}$ in the first term is given by the inference model $q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}(\mathbf{z} \mid \mathbf{x}, \mathbf{y})$. In the testing phase, the classifier $p_{\{\hat{\delta}_s,\tilde{\delta}_i\}}(\mathbf{u} \,|\, \mathbf{x}, \mathbf{z})$ can be used to implement the classification task, where $\mathbf{z}$ is drawn from the prior $p(\mathbf{z})$. Since Eq. (4.8) is used to train the classifier, it still requires learning the generator in order to implement the GRM

procedure. We propose the objective function for the generator of the i-th component as:

$$\begin{aligned}\mathcal{L}_{\text{Gen}}(\mathbf{x},\mathbf{y},\hat{\varsigma}_s,\tilde{\varsigma}_i,\hat{\theta}_s,\tilde{\theta}_i) = &-\mathbb{E}_{q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}(\mathbf{z}\,|\,\mathbf{x},\mathbf{y})}[\log p_{\{\hat{\theta}_s,\tilde{\theta}_i\}}(\mathbf{x}\,|\,\mathbf{y},\mathbf{z})] \\ &+ D_{KL}[q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}(\mathbf{z}\mid\mathbf{x},\mathbf{y})\;||\;p(\mathbf{z})]\,,\end{aligned} \tag{4.9}$$

where $p_{\{\hat{\theta}_s,\tilde{\theta}_i\}}(\mathbf{x}\,|\,\mathbf{y},\mathbf{z})$ is implemented by $\mathrm{f}^{\text{Class}}_{\tilde{\theta}_i}(\mathrm{f}^{\text{Class}}_{\hat{\theta}_s}(\mathbf{y},\mathbf{z}))$ where $\mathrm{f}^{\text{Class}}_{\hat{\theta}_s}(\mathbf{y},\mathbf{z})$ is the shared generator (decoder) that receives $\mathbf{y}$ and $\mathbf{z}$ and returns a feature/tensor, which is fed into $\mathrm{f}^{\text{Class}}_{\tilde{\theta}_i}(\cdot)$ to produce $\mathbf{x}$. Note that $\mathbf{y}$ can be obtained from the training dataset when the class label is available. We provide the network architecture of the generator in Fig. 4.8. $q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}(\mathbf{z}\mid\mathbf{x},\mathbf{y})$ is implemented by a Gaussian distribution $\mathcal{N}(\tilde{\boldsymbol{\mu}}_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}},\tilde{\boldsymbol{\sigma}}^2_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}\mathbf{I})$ whose hyperparameters $\{\tilde{\boldsymbol{\mu}}_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}},\tilde{\boldsymbol{\sigma}}_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}\}$ are provided by a neural network $\mathrm{f}^{\text{Class}}_{\tilde{\varsigma}_i}(\mathrm{f}^{\text{Class}}_{\hat{\varsigma}_s}((\mathbf{x},\mathbf{y})))$ where $\mathrm{f}^{\text{Class}}_{\hat{\varsigma}_s}(\mathbf{x},\mathbf{y})$ is the shared encoder that receives $\mathbf{x}$ and $\mathbf{y}$ and returns a feature, which is fed into $\mathrm{f}^{\text{Class}}_{\tilde{\varsigma}_i}(\cdot)$ to produce the Gaussian hyperparameters $\{\tilde{\boldsymbol{\mu}}_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}},\tilde{\boldsymbol{\sigma}}^2_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}\}$ used to form the Gaussian distribution $q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}(\mathbf{z}\mid\mathbf{x},\mathbf{u}) = \mathcal{N}(\tilde{\boldsymbol{\mu}}_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}},\tilde{\boldsymbol{\sigma}}^2_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}\mathbf{I})$. We provide the network architecture of the encoder in Fig. 4.9. Each component $\mathcal{A}_i$ in the classification task consists of $p_{\{\hat{\theta}_s,\tilde{\theta}_i\}}(\mathbf{x}\,|\,\mathbf{y},\mathbf{z})$, $q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_i\}}(\mathbf{z}\mid\mathbf{x},\mathbf{y})$ and $p_{\{\hat{\delta}_s,\tilde{\delta}_i\}}(\mathbf{u}\mid\mathbf{x},\mathbf{z})$.

In practice, we update the model's parameters using the objective functions $\mathcal{L}_{\text{Gen}}$ and $\mathcal{L}_{\text{C}}$ from equations Eq. (4.9) and Eq. (4.8) with the mini-batch of data samples. In addition, the objective function ($\mathcal{L}_{\text{Gen}}$) is also used for the model expansion in the classification task after each task finishes:

$$\min_{i=1,\cdots,K}\left\{\mathrm{F}^c_S(\mathcal{A}_i,t+1,D_S^{t+1})\right\} \geq \lambda^{\text{GMM}}\,, \tag{4.10}$$

where D_S^{t+1} is the labelled training dataset from the $(t+1)$-th task and $\mathrm{F}^c_S(\cdot,\cdot)$ is defined as:

$$\mathrm{F}^c_S(\mathcal{A}_i,t+1,D_S^{t+1}) = \frac{1}{m'}\sum_{k=1}^{m'}\left|\mathcal{L}_{\text{Gen}}(\mathbf{x}'_{(i,k)},\mathbf{y}'_{(i,k)},\hat{\varsigma}_s,\tilde{\varsigma}_i,\hat{\theta}_s,\tilde{\theta}_i) - \mathcal{L}_{\text{Gen}}(\mathbf{x}_{(t+1,k)},\mathbf{y}_{(t+1,k)},\hat{\varsigma}_s,\tilde{\varsigma}_i,\hat{\theta}_s,\tilde{\theta}_i)\right|, \tag{4.11}$$

where $\mathcal{A}_i$ has the parameter sets $\{\hat{\varsigma}_s,\tilde{\varsigma}_i,\hat{\theta}_s,\tilde{\theta}_i\}$ and we use the decoder $p_{\{\hat{\theta}_s,\tilde{\theta}_i\}}(\mathbf{x}\,|\,\mathbf{y},\mathbf{z})$ of $\mathcal{A}_i$ to generate the pseudo sample $\mathbf{x}'_{(i,k)}$, where $\mathbf{z}\sim p(\mathbf{z})$ and $\mathbf{y}$ is a randomly generated class label (one-hot vector). i represents the component index. We also use the classifier (inference model) $p_{\{\hat{\delta}_s,\tilde{\delta}_i\}}(\mathbf{u}\mid\mathbf{x},\mathbf{z})$ of $\mathcal{A}_i$ to predict the class label $\mathbf{y}'_{(i,k)}$ for $\mathbf{x}'_{(i,k)}$ and therefore $\mathbf{y}$ in $\mathcal{L}_{\text{Gen}}$ is given by the class label $\mathbf{y}'_{(i,k)}$. $\mathbf{x}_{(t+i,k)}$ and $\mathbf{y}_{(t+i,k)}$ represent the k-th real training sample and label obtained from the training set of the $(t+1)$-th task. If the expansion criterion (Eq. (4.10)) is

Algorithm 3: GMM for the classification task

(Input:All training databases) ;
for $t < N$ **do**
  if $t == 1$ **then**
    isAdd = True ;
  end
  Get the labelled training set D_S^t from the t-th task ;
  if isAdd == *False* **then**
    Generate the labelled dataset $D_G^{\widehat{s}}$ from the $\widehat{s}$-th component ;
    Combine two datasets $D_S^t = D_S^t \bigcup D_G^{\widehat{s}}$;
  end
  else
    Build a new component ;
  end
  for *index < iterationNumber (Only update each model once in each iteration)* **do**
    Get the data batches $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\}$ from the set D_S^t ;
    if isAdd == *False* **then**
      Update the parameters $\{\tilde{\varsigma}_{\widehat{s}}, \tilde{\delta}_{\widehat{s}}\}$ with $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\}$ using $\mathcal{L}_{\mathrm{C}}(\mathbf{x}, \mathbf{y}, \hat{\varsigma}_s, \tilde{\varsigma}_{\widehat{s}}, \hat{\delta}_s, \tilde{\delta}_{\widehat{s}})$;
      Update the parameters $\{\tilde{\varsigma}_{\widehat{s}}, \tilde{\theta}_{\widehat{s}}\}$ with $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\}$ using $\mathcal{L}_{\mathrm{Gen}}(\mathbf{x}, \mathbf{y}, \hat{\varsigma}_s, \tilde{\varsigma}_{\widehat{s}}, \hat{\theta}_s, \tilde{\theta}_{\widehat{s}})$;
    end
    else
      Update the parameters of the new component ;
      Update the parameters $\{\tilde{\varsigma}_{K+1}, \tilde{\delta}_{K+1}\}$ with $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\}$ using $\mathcal{L}_{\mathrm{C}}(\mathbf{x}, \mathbf{y}, \hat{\varsigma}_s, \tilde{\varsigma}_{K+1}, \hat{\delta}_s, \tilde{\delta}_{K+1})$;
      Update the parameters $\{\tilde{\varsigma}_{K+1}, \tilde{\theta}_{K+1}\}$ with $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\}$ using $\mathcal{L}_{\mathrm{Gen}}(\mathbf{x}, \mathbf{y}, \hat{\varsigma}_s, \tilde{\varsigma}_{K+1}, \hat{\theta}_s, \tilde{\theta}_{K+1})$;
    end
  end
  The evaluation of knowledge measure ;
  Calculate the knowledge measure using Eq. (4.11) ;
  if $\min_{i=1,\cdots,K} \left\{\mathrm{F}_S^c(\mathcal{A}_i, t+1, D_S^{t+1})\right\} \geq \lambda^{\mathrm{GMM}}$, **then**
    isAdd = True ;
  end
  else
    Select a component ;
    $\widehat{s} = \arg\min_{i=1,\cdots,K} \left\{\mathrm{F}_S^c(\mathcal{A}_i, t+1, D_S^{t+1})\right\}$;
    isAdd = False ;
  end
end

Algorithm 4: Algorithm for the teacher-student

Input:All training databases) ;
for $t < N$ **do**
  if *t == 1* **then**
    isAdd = True ;
  end
  Get the unlabelled training set D_{SU}^{t} ;
  if *isAdd == False* **then**
    Generate dataset $D_{GU}^{\widehat{s}}$ from the $\widehat{s}$-th component ;
    $D_{SU}^{t} = D_{SU}^{t} \bigcup D_{GU}^{\widehat{s}}$ Form a joint dataset ;
  end
  else
    Build a new component ;
  end
  for *index < iterationNumber (Each module in each iteration is updated only once)* **do**
    Get the data batch $\mathbf{X}_{batch}$ from D_{SU}^{t} ;
    if *isAdd == False* **then**
      Update the selected component using $\mathcal{L}_{ELBO}(\mathbf{x}, \hat{\theta}_s, \tilde{\theta}_{\widehat{s}}, \hat{\varsigma}_s, \tilde{\varsigma}_{\widehat{s}})$ and $\mathbf{X}_{batch}$;
    end
    else
      Update the new component using $\mathcal{L}_{ELBO}(\mathbf{x}, \hat{\theta}_s, \tilde{\theta}_{K+1}, \hat{\varsigma}_s, \tilde{\varsigma}_{K+1})$ and $\mathbf{X}_{batch}$;
    end
    The learning of the student module ;
    Update the student module by using Eq. (4.13) with $\mathbf{X}_{batch}$ and generative replay samples ;
  end
  The evaluation of knowledge measure: ;
  Calculate the knowledge measure using Eq. (4.4) ;
  if $\min_{i=1,\cdots,K} \left\{ \mathrm{F}_S(\mathcal{A}_i, t+1, D_{SU}^{t+1}) \right\} \geq \lambda^{\mathrm{GMM}}$, **then**
    isAdd = True ;
  end
  else
    Select a component ;
    $\widehat{s} = \underset{i=1,\cdots,K}{\arg\min} \left\{ \mathrm{F}_S(\mathcal{A}_i, t+1, D_{SU}^{t+1}) \right\}$;
    isAdd = False ;
  end
end

not satisfied, we perform the component selection in the classification task by:

$$F_{\text{findSup}}^{\text{GMM}}(\mathcal{A}_1, \cdots, \mathcal{A}_K, t+1, D_S^{t+1}) = \underset{i=1,\cdots,K}{\arg\min} \left\{ \mathrm{F}_S^c(\mathcal{A}_i, t+1, D_S^{t+1}) \right\} , \tag{4.12}$$

where $\widehat{s} = F_{\text{findSup}}^{\text{GMM}}(\mathcal{A}_1, \cdots, \mathcal{A}_K, t+1, D_S^{t+1})$ is the index of the selected component in supervised learning. We provide the pseudocode for the classification task in **Algorithm 3**.

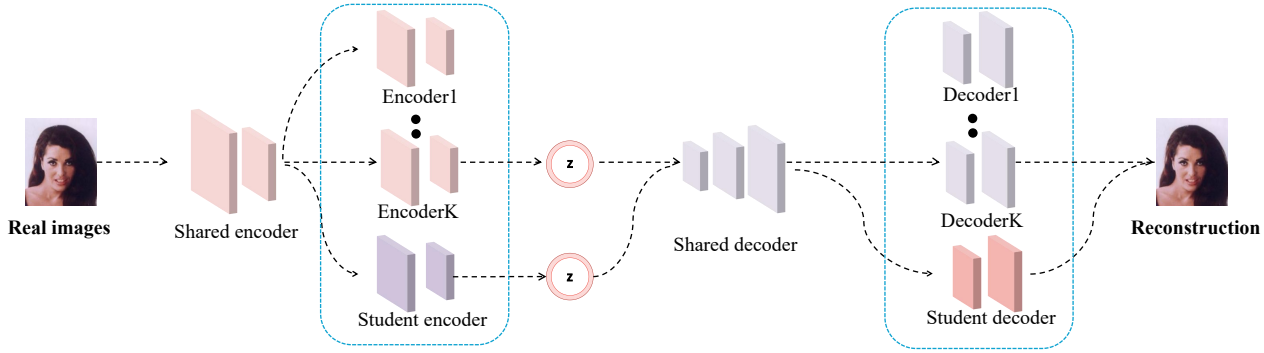


Figure 4.10: Diagram showing the learning structure for the proposed GMM mixture model. Only a few components ('Encoder K', 'Student Encoder', 'Decoder K', and 'Student Decoder') update parameters during each stage of lifelong learning. Meanwhile, we always train the student module in each task learning by using the objective function from Eq. (4.13).

4.2.5 Learning A Compact Student Module and the Training Algorithm

In order to map multiple tasks into a single latent space, we propose training a compact student (VAE) model under unsupervised learning. The advantage of training a student is that of being able to compress the information learnt by a complex Teacher module, such as the GMM, described in Section 4.2.3, into a lightweight model which would not have to perform a component selection process and thus enjoys fast inference during the testing phase. Furthermore, the student module enables many applications, including image reconstruction and interpolation across multiple domains. To implement these goals, we propose a Teacher-Student framework, whose structure is shown in Fig. 4.10, where the GMM is used as the teacher module, while a student module is built based on the top of a shared encoder and a shared decoder. We then propose a new objective function for the student's learning, which consists of the optimization of the current task (the t-th task) and the knowledge distillation, expressed as:

$$\mathcal{L}_{\text{Stu}}^{\text{GMM}}(\mathbf{x}^t, \hat{\varsigma}_s, \tilde{\varsigma}_{\text{stu}}, \hat{\theta}_s, \tilde{\theta}_{\text{stu}}) = \underbrace{\mathbb{E}_{q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_{\text{stu}}\}}(\mathbf{z}\,|\,\mathbf{x}^t)}\left[\log p_{\{\hat{\theta}_s,\tilde{\theta}_{\text{stu}}\}}(\mathbf{x}\mid\mathbf{z})\right] - D_{KL}\left[q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_{\text{stu}}\}}(\mathbf{z}\mid\mathbf{x}^t)\,||\,p(\mathbf{z})\right]}_{\text{ELBO on the t-th task}} +$$

$$\underbrace{\sum_{i=1}^{K}\left(\mathbb{E}_{q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_{\text{stu}}\}}(\mathbf{z}\,|\,\mathbf{x}'_i)}\left[\log p_{\{\hat{\theta}_s,\tilde{\theta}_{\text{stu}}\}}(\mathbf{x}\mid\mathbf{z})\right] - D_{KL}\left[q_{\{\hat{\varsigma}_s,\tilde{\varsigma}_{\text{stu}}\}}(\mathbf{z}\mid\mathbf{x}'_i)\,||\,p(\mathbf{z})\right]\right)}_{\text{Knowledge distillation optimization}}, \tag{4.13}$$

where $\mathbf{x}'_i$ is a sample generated from the i-th VAE component of the GMM and $\mathbf{x}^t$ is a real training sample from the t-th task. K is the total number of components. We provide the

network architecture of a GMM as a teacher with a student module in Fig. 4.10, where the student module has subsets of parameters $\{\tilde{\theta}_{stu}, \tilde{\varsigma}_{stu}\}$ and also shares parameters with the joint network that consists of shared VAE encoder and decoder. This can further reduce the complexity of the whole system while ensuring the usage of a lightweight student module in the testing phase. Furthermore, during training, the student module is always activated to compress the knowledge from the new task together with the knowledge of the trained K components of the GMM in each task learning. The detailed training algorithm for the teacher-student system is shown in the pseudocode from **Algorithm 4**.

4.3 Experiments

4.3.1 Hyperparameter Setting and Network Architecture

We use TensorFlow [1] for the implementation of all models. We adopt the Adam optimization algorithm [82] where we consider a learning rate of 0.0002 and $\beta = 0.5$. We consider the input image size as $32 \times 32 \times 3$ pixels while the kernel size is 3×3. We implement the shared encoder $\mathrm{f}^{\mathrm{e}}_{\hat{\varsigma}_s}$ by using a CNN with four layers with $\{128, 256, 512, 1024\}$ processing units. We implement each sub-encoder $\mathrm{f}^{\mathrm{e}}_{\tilde{\varsigma}_i}$ by using a fully connected network with two layers of $\{1024, 100\}$ processing units. For the decoding process, we implement the shared part $\mathrm{f}^{\mathrm{d}}_{\hat{\theta}_s}$ by using a CNN that has three layers consisting of a fully connected layer with 256 * 8 * 8 processing units and other two convolution layers with $\{256, 256\}$ units. Then we implement the sub-encoder $\mathrm{f}^{\mathrm{d}}_{\tilde{\theta}_i}$ by using a CNN that consists of three layers with $\{256, 256, 3\}$ units. For the large input size of $(64 \times 64 \times 3)$ pixels, the shared encoder $f^{e}_{\hat{\varsigma}_s}$ is implemented by a CNN with $\{64, 128, 256\}$ units. We also implement the sub-encoder $\mathrm{f}^{e}_{\tilde{\varsigma}_i}$ by using a network that consists of a convolution layer with 256 units and two fully connected layers with $\{1024, 256\}$ units. For the decoding process, we implement the shared decoder $\mathrm{f}^{d}_{\hat{\theta}_s}$ by using a network that consists of three convolution layers with $\{256, 256, 256\}$ units and a fully connected layer with 256 * 8 * 8 units. We then implement the sub-decoder $\mathrm{f}^{d}_{\tilde{\theta}_i}$ by using a CNN that consists of $\{256, 128, 3\}$ processing units.

4.3.2 Datasets and Evaluation Criteria

This section introduces detailed information about the dataset and evaluation criteria.

- In unsupervised learning, we consider learning a sequence of tasks where each task is

associated with a dataset. The sequence of databases used for training includes MNIST [93], SVHN [118], Fashion [177], InverseFashion (IFashion) and Rated MNIST (RMNIST). We call this learning setting as MSFIR.

- If we incorporate CIFAR10 [87] after MSFIR, as the last learning task, resulting in MSFIRC sequence for supervised classification. We resize all images to the resolution $32 \times 32 \times 3$ pixels.

Evaluation criteria: In supervised learning, we calculate the average classification accuracy on all tasks as the performance criterion. In unsupervised learning, we adapt the structural similarity index measure (SSIM) [68], the Mean Squared Error (MSE) and the Peak-Signal-to-Noise Ratio (PSNR) [68] in order to evaluate the image reconstruction quality. The calculation form of the SSIM and PSNR criteria are provided as follows:

$$\text{SSIM}(\mathbf{x}, \mathbf{x}') = \frac{(2\mu_{\mathbf{x}}\mu_{\mathbf{x}'} + c_1)(2\sigma_{\mathbf{x}\mathbf{x}'} + c_2)}{(\mu_{\mathbf{x}}^2 + \mu_{\mathbf{x}'}^2 + c_1)(\sigma_{\mathbf{x}}^2 + \sigma_{\mathbf{x}'}^2 + c_2)}\,, \tag{4.14}$$

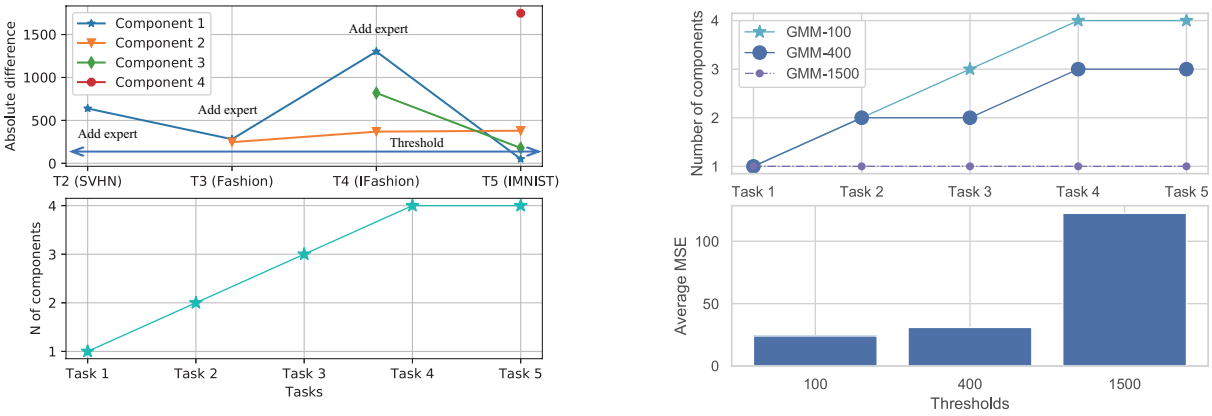
$$\text{PSNR}(\mathbf{x}, \mathbf{x}') = 10 \log_{10} \frac{\max(\mathbf{x})^2}{\text{MSE}(\mathbf{x}, \mathbf{x}')}\,, \tag{4.15}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are the real testing and reconstructed image, respectively. $\mu_{\mathbf{x}}$ and $\mu_{\mathbf{x}'}$ are the pixel sample mean of $\mathbf{x}$ and $\mathbf{x}'$, respectively. $\sigma_{\mathbf{x}}^2$ and $\sigma_{\mathbf{x}'}^2$ are the variance of $\mathbf{x}$ and $\mathbf{x}'$, respectively. $\text{MSE}(\cdot, \cdot)$ is the mean square error. $c_1 = (k_1 L_{\text{image}})^2$ and $c_1 = (k_2 L_{\text{image}})^2$ are two variables where L_{image} is the dynamic range of the pixel-values and $k_1 = 0.01$ and $k_2 = 0.03$. We employ the skimage library to implement the PSNR and SSIM criteria.

Baselines: In this chapter, we compare with three popular continual learning models: CURL [133], LGM [132] and BatchEnsemble (BE) [174]. BE was used in supervised learning. In order to compare the proposed model with BE for unsupervised learning, we implement BE as a mixture model where each component is a VAE model that has a tuple of trainable vectors built on the top layer of a joint neural network. We only update this joint neural network at the first task learning afterwards freezing it to avoid forgetting.

4.3.3 Generative Modelling Tasks

We train various models on MSFIR lifelong learning tasks, and the number of training epochs for each task learning is 20. We present the results obtained after the lifelong learning of



(a) (b)

Figure 4.11: (a) The evaluation of the knowledge similarity between the given new task and the information already known by the GMM under the MSFIR lifelong learning. We also plot the number of components in each task learning. (b) The model's performance and complexity change when using different thresholds λ^{GMM} in Eq. (4.4) during MSFIR lifelong learning. 'GMM-100' denotes that the GMM model is trained using the threshold $\lambda^{\text{GMM}} = 100$.

Table 4.1: The performance of various models after the MSFIR lifelong learning.

	MSE					**SSMI**					**PSNR**				
Datasets	LGM	CURL	BE	GMM	Stud	LGM	CURL	BE	GMM	Stud	LGM	CURL	BE	GMM	Stud
MNIST	129.93	211.21	19.24	26.64	176.82	0.45	0.46	0.92	0.88	0.42	14.52	13.27	22.57	21.02	13.72
Fashion	89.28	110.60	38.81	33.67	178.04	0.51	0.44	0.61	0.75	0.37	15.82	14.89	14.46	19.68	8.81
SVHN	169.55	102.06	39.57	30.27	146.70	0.24	0.26	0.66	0.64	0.47	8.11	10.86	18.90	15.55	13.58
IFashion	432.90	115.29	36.52	35.03	158.18	0.26	0.54	0.75	0.77	0.43	9.04	15.51	19.32	19.47	14.17
RMNIST	130.28	279.47	25.41	22.97	157.55	0.45	0.29	0.88	0.90	0.43	14.51	10.84	21.31	21.71	14.18
Average	190.38	163.72	31.91	**29.71**	163.45	0.38	0.39	0.76	**0.78**	0.42	12.40	13.07	19.31	**19.48**	12.89

MSFIR in Table 4.1 where "Stu" represents the performance of the student model which is worse than that of GMM because the student module is trained on the generated knowledge from GMM. From Table 4.1, we observe that the proposed GMM outperforms the baseline in each task, demonstrating the advantage of the proposed approach. We also evaluate the novelty of the information contained in the new data compared with the already learnt information by the model, calculated using the left-hand side of Eq. (4.10) and the number of components in Fig. 4.11a, where GMM had trained four components after lifelong learning. This result shows that the first component that has already learnt MNIST, is used to learn a similar dataset, RMNIST.

(a) Testing samples. (b) Reconstructions.

Figure 4.12: Reconstructed image results achieved by the student module of the GMM after CCCOS lifelong learning.

4.3.4 The Lifelong Learning of Complex Datasets

In this section, we evaluate the proposed GMM model when considering more complicated tasks. Firstly, we introduce several datasets.

Sub-ImageNet (Sub-IM). To balance the number of samples in each task, we build a subset of ImageNet, namely Sub-ImageNet, by randomly collecting 60,000 samples from the ImageNet [88], which contains a variety of images of higher complexity. We then select 50,000 samples as the training set while all remaining samples are used as the testing set.

CelebA. It is a large-scale face attributes dataset which has more than 200K celebrity images [105].

CACD. This dataset contains 163,446 images from 2,000 celebrities collected from the internet [24].

3D-Chair. This dataset contains rendered images of around 1000 different three-dimensional chair models [14].

For CelebA, CACD and 3D-Chair, we randomly select ninety per cent of all samples from each dataset as the training dataset while all remaining samples are employed as the testing dataset. We build a sequence of complex tasks, including CelebA, CADS, 3D-Chair, Omniglot and Sub-ImageNet databases, namely CCCOS, where we resize all images as $64 \times 64 \times 3$ pixels. Various models are trained under CCCOS, and we report the results in Table 4.2. We can observe that the proposed GMM outperforms other baselines on both CCCOS and MSFIR

Figure 4.13: Interpolation results obtained by the student module of GMM, after CCCOS lifelong learning.

settings, according to the results from Table. 4.2, respectively. We present the reconstruction results achieved by the GMM in Fig. 4.14, where the GMM model is shown that it can provide accurate image reconstructions for each task. We also show the image reconstruction results achieved by the student module of the GMM in Fig. 4.12, which show that the proposed approach produces high-quality image reconstructions.

In the following, we also consider exploring joint latent spaces through the image interpolation experiment. We employ the inference model of the GMM's student module to infer the latent codes for two different images. Then, we generate the intermediary images by smoothly changing the latent code of one image to that of another image, and the results are shown in Fig. 4.13. From the results from rows 3-4 and 5-6 from Fig. 4.13 we can observe that GMM's student module can generate smooth interpolations even when the original images are from completely different domains, such as face images and 3D chairs, or faces and specific signs, respectively. These results demonstrate that the proposed knowledge distillation loss can encourage the Student module to learn cross-domain representations within a single latent space.

Table 4.2: The performance of various models under the CCCOS learning setting.

	MSE					SSMI					PSNR				
Datasets	LGM	CURL	BE	GMM	Stud	LGM	CURL	BE	GMM	Stud	LGM	CURL	BE	GMM	Stud
CelebA	1536.06	1446.86	209.93	214.55	646.95	0.33	0.34	0.69	0.69	0.49	15.14	15.42	23.61	23.52	18.71
CACD	2348.35	2202.88	459.93	363.17	1394.11	0.26	0.27	0.55	0.62	0.38	13.16	13.40	20.21	21.28	15.38
3D-Chair	1430.87	1258.02	629.55	483.29	1527.70	0.43	0.47	0.73	0.80	0.47	15.68	16.18	19.26	20.72	15.60
Omniglot	3356.40	2464.04	753.30	361.33	4258.15	0.20	0.26	0.78	0.89	0.28	11.76	13.13	18.55	21.99	10.75
Sub-IM	1147.64	1336.58	773.89	783.21	1064.51	0.30	0.32	0.37	0.37	0.32	15.80	16.07	18.47	18.44	17.06
Average	1963.86	1741.68	565.30	**441.11**	1778.29	0.30	0.33	0.62	**0.67**	0.39	14.31	14.84	20.02	**21.19**	15.50

(a) CelebA. (b) CACD. (c) 3D-chair. (d) Omniglot. (e) Sub-ImageNet.

(f) Task 1. (g) Task 2. (h) Task 3. (i) Task 4. (j) Task 5.

Figure 4.14: Reconstructed image results achieved by the GMM after CCCOS lifelong learning. The first row represents testing images and the second row are their reconstructions using GMM.

4.3.5 Classification Tasks

In this section, we compare GMM with the state-of-the-art approaches for the classification task. LGM [132] is designed for the unsupervised learning task, and we can use LGM for the classification task by training a classifier on the joint dataset consisting of real samples and the generated samples from LGM. We report the classification accuracy in Table 4.3, which shows that the proposed GMM outperforms other baselines. Additionally, BE also achieves good performance since its number of components is equal to the number of tasks, and consequently suffers less from forgetting.

Additionally, we investigate the performance of various models when learning a long sequence of tasks. We consider training various models considering seven tasks, including MNIST, SVHN, Fashion, IFashion, RMNIST, rotated Fashion (RFashion which contains the same images as Fashion database, but which are rotated with 180 angles) and CIFAR10 (MSFIRRC).

Table 4.3: Classification accuracy of various models after the MSFIRC lifelong learning.

Dataset	LGM [132]	CURL [133]	BE[174]	GMM	MRGANs [175]
MNIST	90.54	91.30	99.40	99.44	91.24
SVHN	22.56	62.05	74.46	85.13	64.12
Fashion	68.29	79.18	88.95	91.49	80.10
IFashion	73.70	82.51	86.45	68.75	82.19
RMNIST	90.52	98.56	99.10	98.50	98.30
CIFAR10	57.43	67.34	52.48	65.27	67.19
Average	67.17	80.16	83.47	**84.76**	80.52

Table 4.4: Classification accuracy of various models after the MSFIRRC's lifelong learning.

Dataset	GMM	BE [174]
MNIST	86.01	99.28
SVHN	86.91	74.84
Fashion	90.68	87.60
IFashion	91.02	86.03
RMNIST	99.01	98.77
RFashion	91.43	86.60
CIFAR10	64.61	54.79
Average	**87.10**	83.99

The threshold λ^{GMM} from Eq. (4.4), which defines the selection of the components from GMM is set to 180 when training on MSFIRRC. After the learning of all given tasks is finished, the GMM eventually has learnt five components. The GMM reuses the first component that has learnt MNIST, in order to learn a similar task (RMNIST), and reuses the third component to firstly learn Fashion and then RFashion. The classification results are reported in Table 4.4, where the proposed GMM outperforms BE in terms of the average classification accuracy. Moreover, the proposed GMM achieves the best performance on almost every dataset when compared to BE, except for the MNIST database, where BE has better results.

Although the proposed GMM is mainly used in task-incremental learning, we also investigate the performance of GMM in the class-incremental setting. Following the setting from [124], we create a new dataset, namely Permuted MNIST, where we have ten tasks and each task is defined by the MNIST database with a fixed permutation of pixels [124]. We also consider the Split MNIST [191], which splits MNIST into five tasks where each task contains samples belonging to two successive classes. For Permuted MNIST, the classifier in each expert is implemented by using a Multilayer Perceptron (MLP) consisting of two hidden layers, where each layer has 100 hidden units, according to [124]. For Split MNIST, we implement the classifier of each expert by a neural network comprised of two layers where each layer has 256

Table 4.5: Results of continuous learning benchmark.

Methods	Permuted MNIST	Split MNIST
Improved VCL* [158]	93.1 ± 1	98.4 ± 0.4
EWC* [86]	84	63.1
DLP* [150]	82	61.2
SI* [191]	86	98.9
FROMP* [124]	94.9 ± 0.1	99.0 ± 0.1
FRCL-TR* [162]	94.3 ± 0.2	97.8 ± 0.7
FRCL-RND* [162]	94.2 ± 0.1	97.1 ± 0.7
GMM	**96.46** $\pm$ 0.03 (10 C)	**99.21** $\pm$ 0.04 (5 C)
GMM	88.78 (7 C)	96.77 (4 C)
GMM	95.25 (8 C)	91.37 (3 C)

Table 4.6: Results of Split CIFAR.

Methods	Split CIFAR
FROMP-L_2	75.6 ± 0.4
FROMP	76.2 ± 0.4
SI	73.5 ± 0.5
VCL + random coreset	67.4 ± 1.4
EWC	71.6 ± 0.9
GMM	**76.40** $\pm$ 0.3 (6 C)
GMM	65.70 (5 C)

hidden units, according to [124]. We search the threshold λ^{GMM} of Eq. (4.10) from 80 to 120 for Permuted MNIST and from 30 and 60 for Split MNIST, resulting in employing different numbers of components.

In addition, we also consider a challenging dataset, namely Split CIFAR, because it contains more complex images, Split CIFAR [191], in continual learning. Following the same procedure from [124], Split CIFAR considers the CIFAR10 that is employed as the first task, and then we consider the following five tasks where we select training samples from 10 consecutive categories from CIFAR100 as a task and so we have six tasks for Split CIFAR. Following from [124], we adopt a network architecture that consists of four convolutional layers and two fully connected layers. It notes that the shared classifier is implemented by means of four convolutional layers. During the GMM expansion process, we build a sub-classifier by using two fully connected layers, based on the top layer of the shared classifier. Therefore, each sub-classifier reuses parameters from this shared classifier, which can reduce the whole model size. We only update the parameters of the shared classifier when learning the first task in order to relieve forgetting. Compared to FROMP [124], the one major weakness of the proposed GMM is the training process of the shared modules whose parameters are only updated at the first task learning

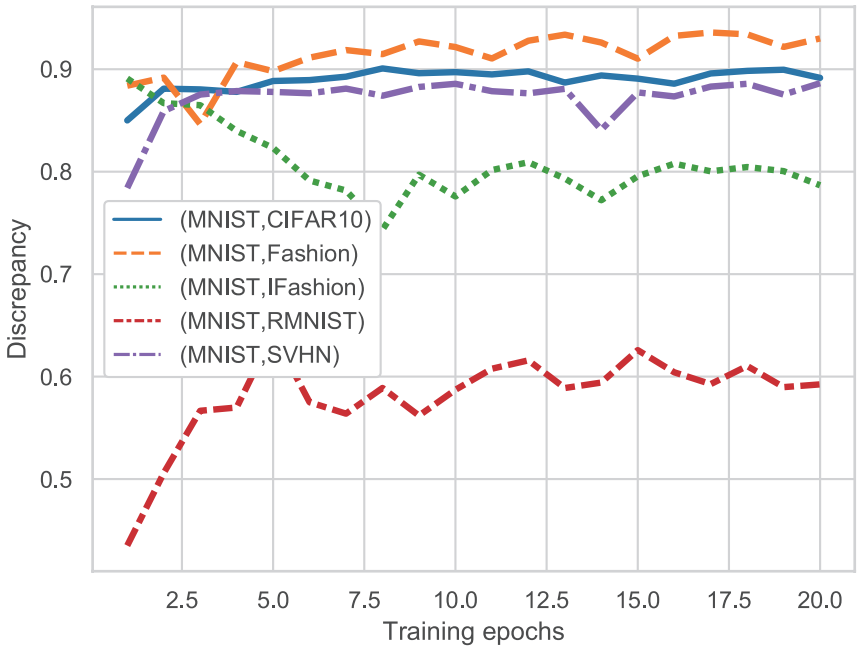
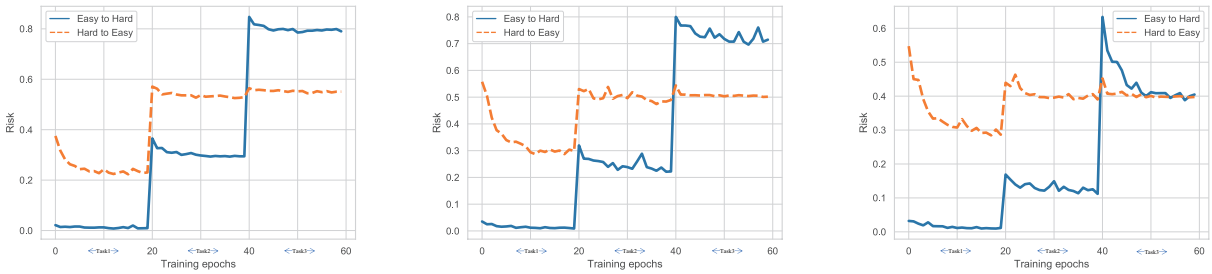


Figure 4.15: The estimation of the discrepancy distance between different domains, where the first one was assumed to have already been learnt by the model.

and are frozen in subsequent task learning. As a result, the proposed GMM does not explore the full model capacity for learning new tasks. In addition, FROMP can keep a fixed model structure during the training but requires a memory buffer to store many data samples of all prior tasks. In contrast, the proposed GMM does not need any memorized examples but continually adds and freezes parameters during the training, aiming to preserve all previously learned information. Furthermore, the main idea and technology from FROMP can be used in the training process of the shared module of the proposed GMM, which enables updating shared parameters across multiple tasks and thus would improve model performance.

During this experiment, we search for the best threshold $\lambda^{\text{GMM}} \in [80, 100]$ from Eq. (4.10) for Permuted MNIST and $\lambda^{\text{GMM}} \in [30, 60]$ for Split MNIST and Split CIFAR. The best threshold λ^{GMM} for Permuted MNIST, Split MNIST and Split CIFAR is 98, 56 and 55, respectively. We perform five independent runs and calculate the average classification accuracy for Permuted MNIST, Split MNIST and Split CIFAR, which are shown in Table 4.5 and Table 4.6 where we cite all results[4] from [124] except for the proposed GMM which are provided by this experiment. We denote by "N C" the fact that GMM has learnt N components. These results demonstrate that we can achieve optimal performance by ensuring that the number of components in GMM is equal to the number of learnt tasks where the model does not suffer from forgetting.

[4]Some results were reported in supplementary material from [124].



(a) (b) (c)

Figure 4.16: The target risk (classification error) on all datasets, achieved by the proposed GMM when learning a sequence of the MNIST, SVHN and CIFAR10, namely MSC. We employ 'Easy-to-Hard' and 'Hard-to-Easy' to denote that the model is trained under MSC and CMS lifelong learning, respectively. (a) The results from BatchEnsemble. (b) The results of the GMM that shares part of the parameters between components. (c) The results of the GMM when do not share parameters among components.

4.3.6 Ablation Study

This section investigates the importance of each module in the proposed GMM. Firstly, we evaluate the performance of the proposed GMM by varying the threshold λ^{GMM} from Eq. (4.10), for deciding when to add new components to the mixture model, and the results are shown in Fig. 4.11b. A large threshold λ^{GMM} allows GMM to reuse components frequently, resulting in worse performance but a more compact architecture for GMM. In contrast, a small threshold allows GMM to expand more often during the training, increasing the model's complexity while improving performance.

Additionally, we investigate whether the proposed component selection process can find a component that does not suffer from much degeneration during learning.. We train two classifiers h, h' on MNIST and a joint dataset (MNIST and A) where A is another dataset, respectively. Then, we estimate the discrepancy distance between MNIST and A by evaluating the outputs of these two classifiers h and h' using Eq. (4.2). We present the results in Fig. 4.15 where "$(MNIST, Fashion)$" represents the discrepancy distance between MNIST and Fashion databases, estimated using the two classifiers h and h'. From Fig. 4.15, we observe that the discrepancy distance is small when two tasks are related (for example, MNIST and RMNIST). The proposed component selection procedure reuses the component trained on MNIST to learn a similar dataset, such as RMNIST, demonstrating that the proposed selection procedure can help GMM choose an appropriate component when just a small discrepancy between the learnt knowledge and the information representation of the new task is detected.

Table 4.7: The number of parameters of various models under MSFIR unsupervised learning.

Model	LGM [132]	CURL [133]	BE [174]	GMM	Stu
Parameters	3.3×10^8	2.3×10^8	3.6×10^8	2.1×10^8	1.4×10^8

Table 4.8: The number of parameters of various models under the CelebA, CACD, 3D-Chair, Omniglot and Sub-ImageNet (CCCOS) lifelong learning setting.

Model	LGM [132]	CURL [133]	BE [174]	GMM	Stu
Parameters	1.9×10^9	2.0×10^9	2.0×10^9	7.2×10^8	1.7×10^8

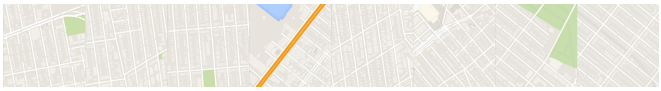
In the following, we investigate the performance of the proposed GMM when changing the order of tasks during lifelong learning. We train the proposed GMM under MNIST, SVHN and CIFAR10 (MSC) lifelong learning, and compare with training with the same databases, but in a different order, CMS. We show the empirical results in Figures 4.16-a and b, which indicate that GMM and BE have a significant difference in the results when changing the order of tasks during lifelong learning. This is because GMM and BE share most of the parameters between the components, while the shared module is only updated at the first task learning. As a result, the shared module can provide low-level feature information extracted from the first task for the subsequent task learning. However, such feature information can be changed and thus affect the performance of the subsequent tasks when changing the first task. These results show that the proposed GMM is sensitive to changes in the order of task learning. In order to address this issue, we also investigate the performance of the GMM that does not share parameters among components. In this case, each newly created component in the GMM has independent parameters and does not share its parameters with other components. We plot the target risks of this model in Fig. 4.16c. From this plot, we can observe that by employing independent parameters for each newly created component, such components are less sensitive to changes in the learning tasks order, while achieving better performance compared with the models that share parameters among components.

4.3.7 Image to Image Translation Task

In this section, we apply our model GMM for image-to-image translation tasks. We build a sequence of Map [71], CMP [131], and Shoe [190] datasets. We train GMM on this sequence by using the objective function from Eq. (4.7), and the visual results are shown in Fig. 4.17. We observe that the proposed GMM achieves high-quality image-to-image translation results without forgetting. These results show that the proposed GMM can be potentially applied to

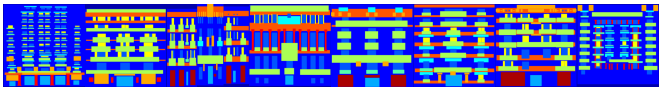
Table 4.9: The number of parameters of various models under the lifelong supervised learning (MSFIRC).

Model	LGM	CURL	BE [174]	GMM	MRGANs [175]
Parameters	5.9×10^8	3.3×10^8	3.9×10^8	3.4×10^8	3.3×10^8



(a) Maps.

(b) Reconstructions.



(c) testing samples from CMP [131]

(d) Reconstructions.

(e) Testing samples from [190].

(f) Reconstructions.

Figure 4.17: Image to Image translation results when learning three different tasks under the lifelong learning.

the image-to-image translation task under lifelong learning.

4.3.8 Model's Complexity

In this section, we evaluate the model size when experimenting with various methods. The number of parameters required by various methods are reported in the Tables 4.7, 4.8, 4.9 where 'Stu' denotes the number of parameters for the Student module of GMM. These results show that the proposed GMM requires fewer parameters for training than other models.

4.4 Conclusion and Limitations

In this chapter, we propose a new dynamic expansion model (GMM) and introduce a new knowledge measure that evaluates the novelty of the incoming task before expanding the network architecture. This knowledge measure provides guidelines for the expansion and selection mechanisms while ensuring that a certain component is reused for learning a related task, thus ensuring that the number of parameters for the model is appropriate for the given tasks. The proposed knowledge measure leads the proposed GMM to learn a reasonable network architecture while providing good performance in all test sets.

We conduct a series of experiments for lifelong supervised and unsupervised tasks. We also evaluate the effectiveness of the student module of the proposed GMM under the cross-domain image reconstruction and interpolation tasks. The empirical results show that we can transfer the knowledge from the GMM into a lightweight student model that captures different underlying data distributions into a single latent space and implicitly models the connections between different data domains. Compared to the LGAA described in Chapter 3, the proposed GMM can learn an unlimited number of tasks without forgetting, depending only on the computational resources available.

In the following, we summarise some limitations of the proposed GMM:

- Sharing parameters among components in the GMM can reduce the model size. However, such an approach can lead to unstable performances when changing the order of tasks, discussed in Section 4.3.6.
- The proposed knowledge measure approach relies on the inference mechanism of each component. As a result, powerful implicit generative models [116] such as GANs can not be efficiently used as components in the GMM.
- The GMM trains a rather weak student module, producing blurred image reconstructions and interpolations. This inspires us to develop a new lifelong teacher-student framework to learn a robust student module, as described in the next chapter.

Chapter 5

Dynamic Self-Supervised Teacher-Student Network

5.1 Introduction

In Chapter 4, we have shown how to train a dynamic expansion model for continual learning. However, due to the poor generation capacity of VAEs, the proposed GMM learns a weak student module which is not able to produce satisfactory results on the cross-domain image reconstruction and interpolation tasks. In addition, the expansion processes in the proposed GMM and existing models [95, 133, 184] rely on modelling the sample log-likelihood/density estimation and the inference mechanism of each component, which is limited when considering a more powerful implicit generative model [116].

To address the drawbacks of prior works, a new lifelong learning framework is proposed in this chapter, called the Dynamic Teacher-Student (D-TS), where the Teacher module is implemented by a dynamically expandable GAN mixture model which expands its network architecture according to the novelty of given tasks. The idea of expanding neural networks has been used in GMM, described in Section 4.2.2 of Chapter 4, which compares the difference between the already-learnt knowledge and a new task. Specifically, GMM evaluates the distance on the sample log-likelihood between the generated samples yielded by each component and compares it to the sample data corresponding to a new task. However, such an expansion criterion can not be directly used in D-TS because each teacher expert in D-TS is implemented using an implicit deep generative model (GAN) that can not estimate the sample log-likelihood. In order to solve this issue, this chapter introduces a new criterion, called the Knowledge

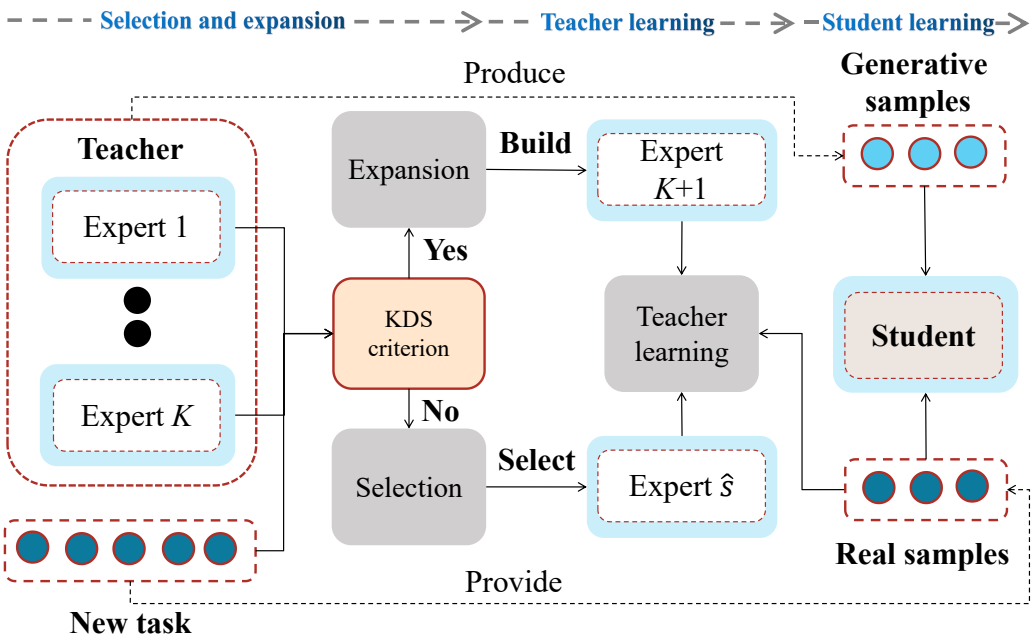


Figure 5.1: The diagram illustrating the learning procedure for the proposed lifelong framework, which consists of three steps (See details in Section 5.2.7). First, when seeing a new task (the t-th task), we perform the Knowledge Discrepancy Score evaluation, by employing SE or KFD criterion, which guides us to perform either the selection or expansion process (Eq. (5.6)). Second, we update the teacher module by using Eq. (5.8) and Eq. (5.9) where we omit the GRM process when a selected expert is reused for learning a new task. Third, we update the student module on real samples from the current task combined with generative replay samples drawn by the teacher module. The more detailed pseudo code is provided in Section 5.2.7.

Discrepancy Score (KDS) that evaluates the relevance between each learnt teacher expert and the incoming task. Specifically, KDS is employed to determine the novelty of an incoming task after each task switch, guiding us to either reuse an existing expert for learning a related task or build a new expert for learning an entirely different task. To model the correlations on the underlying factors between multiple domains, a lightweight latent variable generative model is developed as the student module and a self-supervised learning approach is proposed that trains the student module on mixing real training samples with generative replay samples drawn from the teacher module, as shown in Fig. 5.1. In addition, a new regularized term is introduced in the student's objective function, which minimizes the distance between the posterior distribution learnt by the student and the conditional distribution parameterized by the identity information of each expert. This regularized term encourages embedding multiple knowledge sources from the teacher module into several clusters in the latent space of the student module, which further improves the cross-domain reconstruction and interpolation performance. This chapter summarizes the difference between D-TS and GMM (described in Chapter 4) in three aspects: (1) The GMM employs the VAE as the component of the teacher

module, which results in rather poor generation results and can not provide a high-quality knowledge transfer for the student's learning. In contrast, the D-TS can dynamically build GAN-based components/experts, providing better generation results. (2) The GMM employs the sample log-likelihood evaluation as the dynamic expansion criterion. In contrast, D-TS provides a more flexible dynamic expansion criterion, which can enable the implicit generative models, such as GANs, to be used as the components of the teacher module. (3) The student module in GMM is designed differently from that in D-TS. In addition, the D-TS introduces a new conditional prior that regularizes the student's learning, resulting in better reconstruction performance.

The main contributions of this chapter are:

- This chapter studies a more challenging lifelong learning setting in which we desire to learn domain-specific representations while also inferring the characteristics of these representations into a single latent space.

- This chapter proposes a new lifelong learning framework, namely the Dynamic Teacher-Student Network (D-TS), which enables the teacher module to expand its network architecture in order to learn a growing number of tasks. Meanwhile, the student module in D-TS is self-supervised trained to learn both predictive as well as generative representations across domains.

- This chapter proposes a new criterion, called the Knowledge Discrepancy Score (KDS) which controls the selection and expansion of the teacher module. The proposed KDS enables each expert/component to be implemented by a GAN model.

- This chapter introduces a new conditional prior that employs the identity information of the teacher expert to embed different knowledge sources (modelled by different teacher experts) into different clusters in the latent space of the student module during the training, leading to better image reconstruction performance.

The rest of the chapter is organized as in the following. The Dynamic Teacher-Student (D-TS) framework is described in Section 5.2 and some of its applications are in Section 5.3. The experimental results are provided in Section 5.4 and the conclusions of this chapter are drawn in Section 5.5.

5.2 Dynamic Self-Supervised Teacher-Student Network (D-TS) Framework

The dynamic expansion models have shown promising performance in lifelong learning [95, 133, 184] due to the network expansion mechanisms that can dynamically increase the model's capacity to deal with new tasks. These methods are based on the VAE-based framework [133, 184], which can learn meaningful latent representations that support image reconstructions and interpolations. However, one major weakness of these methods is the inability to perform image interpolations across different data domains due to the lack of a shared latent space. This issue is addressed by the teacher-student framework (GMM), proposed in Chapter 4, which learns a VAE-based student module to embed information from all prior tasks into a single latent space. However, GMM employs a VAE-based teacher framework, which can not provide high-quality knowledge transfer for the student module learning, resulting in poor performance. In this chapter, we solve this issue by developing a dynamically expandable GAN teacher module, which implements each teacher expert using the GAN model [52] that can produce high-quality generative replay samples.

One major challenge for the GAN-based teacher module is the expansion criterion. Comparing the sample log-likelihood as the expansion signal has shown promising results for the VAE-based teacher framework, described in Chapter 4, which, can not be applied to the GAN-based teacher framework since the GAN model does not have inference mechanisms. We solve this issue by introducing a new expansion criterion, namely the Knowledge Discrepancy Score (KDS), described in Section 5.2.3. The key idea of KDS is to evaluate the distance between the generated images from each teacher expert and the training samples from a new task using various measures. Such a design does not require the inference mechanisms of each teacher expert and thus can implement the teacher module using any deep generative technologies. The proposed KDS can guide to dynamically build a new expert for the teacher module when the new task contains novel enough information while selecting an appropriate to learn several similar tasks, aiming to maintain a compact network architecture for the teacher module. The detailed teacher module updating process is presented in Section 5.2.4.

However, the GAN-based teacher framework can not learn meaningful latent representations and thus fails to implement image reconstructions and interpolations across multiple data

domains over time. We address this issue by introducing a VAE-based student module and a new loss function that transfers the knowledge from the teacher module and the new task to the student module. In addition, mapping the information from multiple data domains into a single cluster would lead to performance loss in the image interpolation task because the different domains share the same region of the latent space. We solve this issue by employing a conditional prior distribution to regulate the latent representation optimization of the student module, which can encourage embedding the information from different data domains into different regions of the latent space. We provide the detailed student model design and its updating process in Section 5.2.5 and Section 5.2.6, respectively.

The main contributions of this section are:

- We propose to employ the WGAN-GP technology to implement a dynamic expansion framework as the teacher module, which provides high-quality knowledge transfer for the student module learning.
- We propose a new expansion criterion (KDS) to control the network expansion process of the teacher module. The proposed KDS can also help the teacher module reuse an appropriate teacher expert to learn several similar tasks.
- We propose to use a conditional prior distribution to regulate the latent variable in the student module learning, which can improve the image interpolation tasks.

5.2.1 Problem Definition

Following the notations from Chapter 4, let $p(\mathbf{x}^i_{SU})$ and $p(\mathbf{x}^i_{TU})$ be the data distribution[1] for the unlabelled training set D^i_{SU}, and testing set D^i_{TU} of the i-th task, respectively. For a sequence of N domains (tasks), we assume that each data distribution $p(\mathbf{x}^i_{SU})$, defined on the data space $\mathcal{X} \in \mathbf{R}^W$, is obtained from the i-th task, where W is the dimension[2] of the data sample. In the context of lifelong learning, a model $\mathcal{A}$ (generator or classifier) only accesses samples drawn from $p(\mathbf{x}^i_{SU})$ at the i-th task learning. Our learning goal is to train $\mathcal{A}$ to capture the generative factors from a sequence of N tasks without forgetting previously learnt latent representations, where N is the total number of tasks. Our model consists of a teacher module made up of a

[1]Following from the original GAN paper [52] and other extensive works [55, 11], we employ the concept of the data distribution.

[2]Since we mainly focus on the image dataset, we have $W = Wight \times Hight \times Channel$, where $Weight$, $Hight$ and $Channel$ are the image weight, hight and channel, respectively.

mixture of Generative Adversarial Networks (GAN) and a student module, implemented by a generative latent variable model.

5.2.2 Preliminaries

In the following, we describe the GAN model, which is used as an expert in a mixture system in the teacher module [180]. A GAN model [52] consists of two components: a generator network $\mathcal{G}_\theta \colon \mathcal{Z} \to \mathcal{X}$ and a discriminator network $\mathcal{D}_\eta \colon \mathcal{X} \to \mathrm{R}$, of parameters θ and η, respectively. The generation process is started by drawing a random noise vector $\mathbf{z} \in \mathcal{Z}$ from a fixed multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ as the input of the generator $\mathcal{G}_\theta(\mathbf{z})$ which outputs a fake image $\mathbf{x}'$. The discriminator network $\mathcal{D}_\eta(\mathbf{x})$, of parameters η, is trained to distinguish $\mathbf{x}'$ from a real image $\mathbf{x}$, while the generator $\mathcal{G}_\theta(\mathbf{z})$, of parameters θ, is trained to generate fake images $\mathbf{x}'$ that can fool the discriminator. In this chapter, we employ the Wasserstein GAN-Gradient Penalty (WGAN-GP) [11, 55] in the teacher module whose training is defined by the loss function: [55]:

$$\mathcal{L}_{\mathcal{G}}^{\mathrm{DTS}}(\theta) = \frac{1}{m} \sum_{i=1}^{m} \Big\{ - \mathcal{D}_\eta(\mathcal{G}_\theta(\mathbf{z}_i)) \Big\} . \tag{5.1}$$

$$\mathcal{L}_{\mathcal{D}}^{\mathrm{DTS}}(\mathbf{X}_{batch}, \eta) = \frac{1}{m} \sum_{i=1}^{m} \Big\{ \mathcal{D}_\eta(\mathcal{G}_\theta(\mathbf{z}_i)) - \mathcal{D}_\eta(\mathbf{x}_i) + \lambda(\|\nabla_{\tilde{\mathbf{x}}_i} \mathcal{D}_\eta(\tilde{\mathbf{x}}_i)\|_2 - 1)^2 \Big\} , \tag{5.2}$$

where $\lambda = 10$ is the penalty term [55], which is considered in our all experiments according to [55]. $\mathcal{L}_{\mathcal{G}}^{\mathrm{DTS}}$ and $\mathcal{L}_{\mathcal{D}}^{\mathrm{DTS}}$ are the loss functions for the generator and discriminator, respectively. $\mathbf{X}_{batch} = \{\mathbf{x}_1, \mathbf{x}_1, \cdots, \mathbf{x}_m\}$ is a real data batch obtained from the training dataset and m is the batch size in the mini-batch learning [92] (during the experiments we considered the batch size as $m = 64$). $\tilde{\mathbf{x}}_i$ is the interpolated image produced by $\tilde{\mathbf{x}}_i = s\mathbf{x}_i + (1 - s)\mathbf{x}'_i$ where s is drawn from a uniform distribution $U(0, 1)$. $\mathcal{D}_\eta$ represents the discriminator that receives an image $\mathbf{x}_i$ and returns a scalar, implemented by a neural network with trainable parameters η.

5.2.3 The Knowledge Discrepancy Score (KDS)

Before we describe the D-TS framework, we first introduce the Knowledge Discrepancy Score, which represents the criterion used to control the expansion of the teacher module. Existing mixture models use the log-likelihood/density estimation [95, 133], evaluated by each component with a new training set, for the expansion or selection process. However, these mixture

models require each component to have an inference mechanism (sample log-likelihood/loss estimation), which does not allow for the use of explicit generative models as experts. The sample log-likelihood estimation is also used in the GMM, described in Eq. (4.3) of Chapter 4, for the selection and expansion process. However, the GMM has two main weaknesses. Firstly, the sample log-likelihood evaluation limits the GMM use in a more powerful generative model such as GANs. Secondly, the teacher module in GMM can not provide high-quality knowledge transfer for the student's learning due to the poor generation results yielded by VAEs. In this section, we introduce an alternative approach to evaluate the expansion and selection of components for the mixture model using the Knowledge Discrepancy Score, which addresses these two drawbacks. Firstly, we define the KDS, which aims to evaluate the knowledge similarities between the two datasets, described in the following.

Definition 1 *Knowledge discrepancy score (KDS). Given two unlabelled datasets D^i_{SU} and D^j_{SU} over the image space $\mathcal{X}$, let us define a distance measure function f_s. The KDS between two sets is defined as:*

$$\mathrm{KDS}_{f_s}\left(D^i_{SU}, D^j_{SU}\right) = f_s\left(\Phi\left(\mathbf{X}_{D^i_{SU}}\right), \Phi\left(\mathbf{X}_{D^j_{SU}}\right)\right), \tag{5.3}$$

where $\mathbf{X}_{D^i_{SU}} \in \mathbf{R}^{m' \times W}$ and $\mathbf{X}_{D^j_{SU}} \in \mathbf{R}^{m' \times W}$ are two data subsets formed by m' samples randomly obtained from D^i_{SU} and D^j_{SU}, respectively, where W is the dimension of each sample. $\Phi(\cdot)$ is a mapping function[3] which can be of arbitrary complexity. m' is considered as 5,000 in our experiments. In the following, we introduce two measures for implementing KDS.

Knowledge Fréchet Distance (KFD). A direct approach would evaluate the distance between two empirical data distributions in the high-dimensional data space by using a probabilistic measure. However, such an approach requires additional computations [51], or auxiliary training [17, 101]. Recently, perceptual features extracted from deep Convolutional Neural Networks (CNN), pre-trained on ImageNet [88], have been shown to be suitable for style matching [49] and transfer learning [189]. One of the advantages of using perceptual features is to capture a compact representation of the data, which can be used in many downstream tasks with reasonable computational overheads. This motivates us to consider evaluating the KDS in the feature space to reduce the required computation complexity. The Fréchet Inception Distance

[3]The function $\Phi(\cdot)$ in the definition of KDS is arbitrary complexity, which can be implemented by a feature extractor from a pre-trained model or a loss estimator. Such designs can allow us to explore more KDS measures by only modifying f_s and Φ, which will be investigated in future work.

(FID) was a popular approach first proposed in [63], representing a metric used to evaluate the distance between two empirical data distributions. Specifically, this approach first extracts a feature vector from each image using the last pooling layer of an Inception v3 model pre-trained on ImageNet. Then the FID score between the real image and the generated dataset is calculated based on these feature vectors using the Fréchet distance. The FID score has been widely used to evaluate the performance of GANs in image generation tasks [143, 121, 197]. A high FID score indicates that the GAN model can not generate more realistic images with respect to the samples from the real training dataset. In this section, we generalize FID for the selection and expansion criterion in the teacher module, which then can be used to evaluate the task similarity. Specifically, we consider using the Fréchet distance [39] for implementing $f_s(\cdot)$, evaluated on the low-dimensional feature space, as:

$$\begin{aligned}\mathrm{KDS}_{f_s}^{KFD}(D_{SU}^i, D_{SU}^j) = &\left\| f_e\left(\Phi\left(\mathbf{X}_{D_{SU}^i}\right)\right) - f_e\left(\Phi\left(\mathbf{X}_{D_{SU}^j}\right)\right)\right\|^2 \\ &+ \mathrm{Tr}\Big[\kappa\left(\Phi\left(\mathbf{X}_{D_{SU}^i}\right)\right) + \kappa\left(\Phi\left(\mathbf{X}_{D_{SU}^j}\right)\right) - \\ &2\left(\kappa\left(\Phi\left(\mathbf{X}_{D_{SU}^i}\right)\right)\kappa\left(\Phi\left(\mathbf{X}_{D_{SU}^j}\right)\right)\right)^{1/2}\Big],\end{aligned} \tag{5.4}$$

where $\mathrm{Tr}(\cdot)$ is the trace and $\Phi(\cdot)$ is implemented as a mapping function that transforms an image subset[4] $\mathbf{X}_{D_{SU}^i}$ from D_{SU}^i into the feature matrix $\mathbf{X}_{D_{SU}^i}^f \in \mathbf{R}^{\mathrm{m}'\times\mathrm{W}'}$ by using the feature extractor, which in our experiments is implemented by the last pooling layer of an Inception v3 model [160][5], trained on ImageNet, where $W' < W$ is the dimension of the feature space. $f_e(\cdot)$ and $\kappa(\cdot)$ are used to calculate the mean vector and covariance matrix for $\mathbf{X}_{D_{SU}^j}^f \in \mathbf{R}^{\mathrm{m}'\times\mathrm{W}'}$. We call Eq. (5.4) as the Knowledge Fréchet Distance (KFD), which represents the generalization of the Fréchet Distance Score [63].

Student's Evaluation (SE). For evaluating the novelty and similarity between the knowledge associated with each expert and that of the incoming task, we can use a measure of the knowledge learned by a student module, which is implemented as a VAE-based framework, and represents a depository of the knowledge from all previous tasks. The student module is able to estimate the loss value across domains. A similar loss value between the past and new data would indicate that the given task is known to the student module. In this case, the KDS implements a

[4]We employ $\mathbf{X}_{D_{SU}^i}$ to denote a subset formed by samples randomly selected from D_{SU}^i, which reduces the computational costs for the evaluation of the knowledge discrepancy score.

[5]The Inception v3 model was a pre-trained model [63], which has been used in the FID evaluation. As the same as FID, Eq. (5.4) can estimate the FID score based on two datasets.

distance measure f_s, evaluating the absolute differences in the loss function results between the data samples associated with the new tasks and those generated by each teacher expert from the mixture. We define the KDS using the student's evaluation as:

$$\mathrm{KDS}_{f_s}^{SE}(D_{SU}^i, D_{SU}^j) = \left| \Phi\left(\mathbf{X}_{D_{SU}^i}\right) - \Phi\left(\mathbf{X}_{D_{SU}^j}\right) \right|, \tag{5.5}$$

where we implement f_s as a function that calculates the absolute value $f_s = |\cdot|$ and $\Phi(\cdot)$ is implemented as the function that returns the loss value $\Phi(\mathbf{X}_{D_{SU}^i}) = (1/m') \sum_{i=1}^{m'} SE(\mathbf{X}_{D_{SU}^i}[i])$ where $\mathbf{X}_{D_{SU}^i}[i]$ is the i-th data sample of $\mathbf{X}_{D_{SU}^i}$ and $SE(\cdot)$ is the estimator for the loss value, implemented by the student's objective function, defined in Eq. (5.12). SE can be computed more efficiently than KFD because it is directly estimated by the student module and does not require an externally pre-trained network. Compared to the dynamic expansion mechanism of the GMM described in Chapter 4, which relies on the loss evaluation for each mixture component, both KFD and SE only access generated images from each component and can thus implement by a more powerful generative model (GAN) for each component, improving the model's performance. Furthermore, the other KDS criteria can be implemented by only modifying f_s and Φ, which will be investigated in our future study.

5.2.4 The Teacher Module

Existing teacher-student frameworks [180] use a single GAN as the teacher module, but such an approach has limitations when learning several different datasets due to the mode collapse problem [156] (see also the empirical results in Section 5.4.6). In this chapter, we develop a novel dynamically expandable experts-based memory system for the teacher module in order to learn a growing number of different tasks. To ensure a compact network architecture, we would require that a certain expert learns several data domains from tasks that have similarities with each other. We assume that after the $(t-1)$-th task learning is finished, we have trained K experts $\{\mathcal{G}_{\theta_1}, \ldots, \mathcal{G}_{\theta_K}\}$, where each expert is a GAN and the proposed model's architecture is shown in Fig. 5.2. Let P_{θ_i} represent the probability distribution of the data produced by the generator $\mathcal{G}_{\theta_i}$ of parameters θ_i. By employing the KDS, we can define the dynamic expansion and selection mechanism used to control the teacher expansion process, shown in Fig. 5.3, where $\mathrm{KDS}_{f_s}(\cdot,\cdot)$ is evaluated between the knowledge accumulated by each expert and that corresponding to the incoming task.

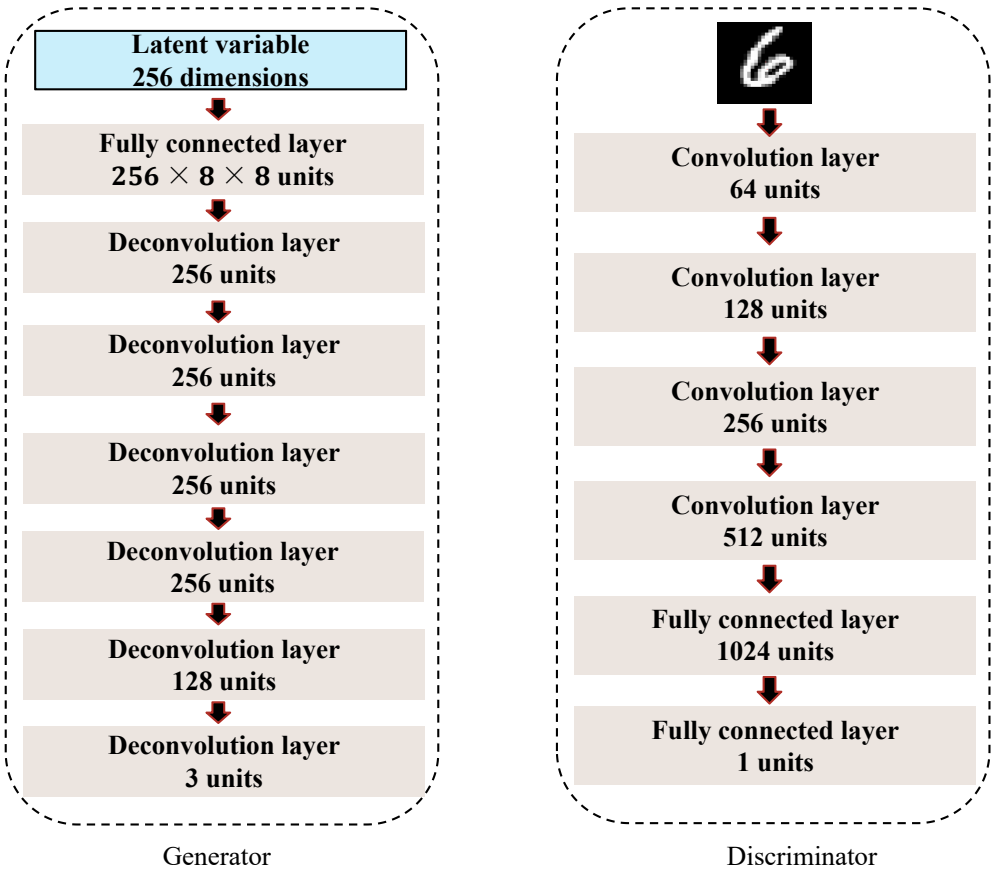


Figure 5.2: The network architecture of each teacher expert, involving a generator and a discriminator.

Selection and expansion using KDS. From Fig. 5.3, after learning the $(t-1)$-th task, the component selection and network expansion procedure are performed by a non-parametric inference process in which we first evaluate the Knowledge Discrepancy Score between the new dataset D^t_{SU}, and the set D^i_{GU} generated by each of the teacher's experts $\mathcal{G}_{\theta_i}$, $i = 1, \ldots, K$ and then we check the model expansion by:

$$\min_{i=1,\ldots,K} \left\{ \mathrm{KDS}_{f_s} \left(D^i_{GU}, D^t_{SU} \right) \right\} > hold \,, \tag{5.6}$$

where D^t_{SU} denotes the unlabelled training dataset from the t-th task, and K is the current number of experts in the teacher module. *hold* is the expansion threshold used to control the model expansion. If the expansion criterion, defined in Eq. (5.6), is satisfied, then the teacher module expands its capacity by building a new expert, otherwise the teacher module selects the most appropriate expert for learning the new task (the t-th task), according to:

$$F^{\mathrm{DTS}}_{\mathrm{selection}}(D^t_{SU}, \theta_1, \cdots, \theta_K) = \arg\min_{i=1,\cdots,K} \left\{ \mathrm{KDS}_{f_s}(D^i_{GU}, D^t_{SU}) \right\} \,, \tag{5.7}$$

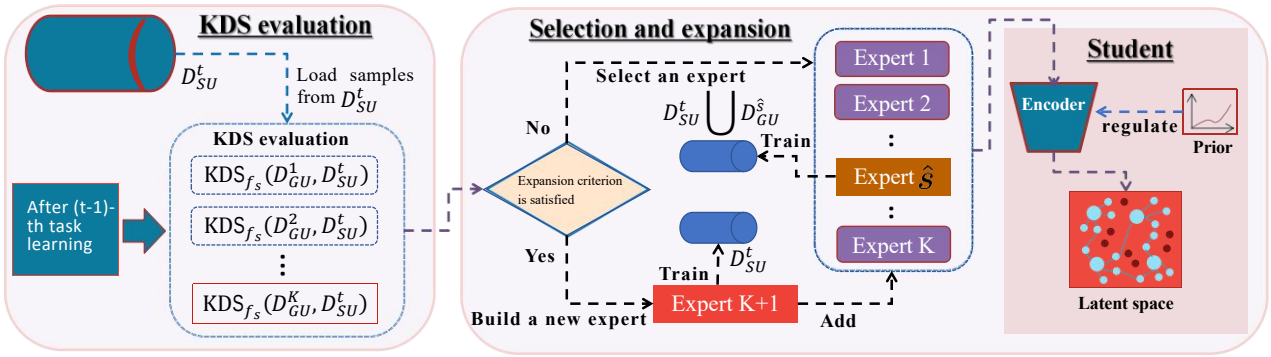


Figure 5.3: The unsupervised learning procedure for D-TS. When learning the t-th task learning, we perform the KDS evaluation between the new dataset D^t_{SU}, and the data sets generated by the teacher's experts, D^i_{GU}, $i = 1, \ldots, K$. If the minimum KDS is larger than a threshold *hold*, then we add a new expert to the mixture system (teacher module), otherwise, we select the expert with the minimum KDS for learning the t-th task. The activated experts are shown in red. The student is trained along with the teacher module, aiming to compress the knowledge from different sources (experts) into a compact latent space.

where $\widehat{s} = F^{\text{DTS}}_{\text{selection}}(D^t_{SU}, \theta_1, \cdots, \theta_K)$ is the index of the selected teacher expert. Given that the new task (the t-th task) shares similar knowledge with respect to the selected expert, we can use fewer training epochs for updating the $\widehat{s}$-th expert when compared to training a new component added to the teacher module. Compared to the expansion criterion in the GMM, both SE and KDS evaluations do not need the inference mechanism of experts. As a result, we can implement each expert in D-TS using arbitrary generative models.

Training the dynamic expansible mixture model. After selection or expansion, we define the teacher's loss function for the following t-th task in a mini-batch learning manner as:

$$\mathcal{L}^{\text{DTS}}_{\mathcal{G}}(\theta_{\widehat{s}}) = \frac{1}{m}\sum_{i=1}^{m}\Big\{ - \mathcal{D}_{\eta_{\widehat{s}}}(\mathcal{G}_{\theta_{\widehat{s}}}(\mathbf{z}_i))\Big\}. \tag{5.8}$$

$$\mathcal{L}^{\text{DTS}}_{\mathcal{D}}(\mathbf{X}_{batch}, \eta_{\widehat{s}}) = \frac{1}{m}\sum_{i=1}^{m}\Big\{\mathcal{D}_{\eta_{\widehat{s}}}(\mathcal{G}_{\theta_{\widehat{s}}}(\mathbf{z}_i)) - \mathcal{D}_{\eta_{\widehat{s}}}(\mathbf{x}_i) + \lambda(\|\nabla_{\tilde{\mathbf{x}}_i}\mathcal{D}_{\eta_{\widehat{s}}}(\tilde{\mathbf{x}}_i)\|_2 - 1)^2\Big\}, \tag{5.9}$$

where $\mathcal{D}_{\eta_{\widehat{s}}}(\cdot)$ and $\mathcal{G}_{\theta_{\widehat{s}}}$ are the discriminator and the generator of the selected teacher expert (the $\widehat{s}$-th teacher expert), parameterized by $\eta_{\widehat{s}}$ and $\theta_{\widehat{s}}$, respectively. If the expansion criterion, defined in Eq. (5.6), is satisfied at the t-th task learning, we dynamically build a new teacher expert $\{\mathcal{D}_{\eta_{K+1}}, \mathcal{G}_{\theta_{K+1}}\}$ and set the index $\widehat{s}$ of the selected teacher expert as $\widehat{s} = K + 1$ in Eq. (5.8) and Eq. (5.9), respectively. The newly created teacher expert will be trained on the training dataset D^t_{SU} from the t-th task learning and $\mathbf{X}_{batch}$ is a data batch from D^t_{SU}. If the expansion criterion, defined in Eq. (5.6), is not satisfied at the t-th task learning, we select

an appropriate teacher expert using Eq. (5.7) and $\widehat{s} = F^{\text{DTS}}_{\text{selection}}(D^t_{SU}, \theta_1, \cdots, \theta_K)$. Then we form a joint dataset $D^t_{SU} \bigcup D^{\widehat{s}}_{GU}$, including samples from the new training set D^t_{SU} from the t-th task and the dataset $D^{\widehat{s}}_{GU}$ generated by $G_{\theta_{\widehat{s}}}$, where $\widehat{s}$ is the index of the selected expert. We train the selected teacher expert $\{\mathcal{D}_{\eta_{\widehat{s}}}, \mathcal{G}_{\theta_{\widehat{s}}}\}$ in the mini-batch learning manner on the joint dataset $D^t_{SU} \bigcup D^{\widehat{s}}_{GU}$ using Eq. (5.8) and Eq. (5.9), respectively, in which $\mathbf{X}_{batch}$ is a data batch from $D^t_{SU} \bigcup D^{\widehat{s}}_{GU}$. We name D-TS-KFD the model when considering the Knowledge Fréchet Distance (KFD), and D-TS-SE when using the student's evaluation, for KDS in order to decide whether to select a new expert for the teacher module.

5.2.5 The Student Module

For the design of the student module, we consider two crucial requirements: (1) A light architecture with fewer parameters than the teacher module; (2) A powerful inference mechanism for representation learning. Following from [41], we define a latent variable generative model $p_{\theta_{stu}}(\mathbf{x}, \mathbf{z}, \mathbf{e}) = p_{\theta_{stu}}(\mathbf{x} \,|\, \mathbf{z}, \mathbf{e}) p(\mathbf{z}, \mathbf{e})$ as the student module, where the discrete variable $\mathbf{e}$ that is assumed to be the V-dimensional one-hot vector, also called expert-variable, represents the identity information associated with each expert, defined in the teacher module while the continuous variable $\mathbf{z}$ represents the fundamental generative factors. For instance, we can employ a V-th dimensional one-hot vector $\mathbf{e} = [1, 0, 0, \cdots, 0]^{\mathrm{T}}$ to represent the identity information of the first teacher expert, where $V = 10$ is the dimension in our experiments. The loss function of the student module for a single task is defined as [41]:

$$\begin{aligned} \mathcal{L}^{\text{DTS}}_{\text{single}}(\mathbf{x}, \theta_{stu}, \varsigma_{stu}, \psi_{stu}) = & -\mathbb{E}_{q_{\varsigma_{stu}, \psi_{stu}}(\mathbf{z}, \mathbf{e} \,|\, \mathbf{x})} \left[\log p_{\theta_{stu}}(\mathbf{x} \mid \mathbf{z}, \mathbf{e}) \right] \\ & + D_{KL}[q_{\varsigma_{stu}}(\mathbf{z} \,|\, \mathbf{x}) \,||\, p(\mathbf{z})] \\ & + D_{KL}[q_{\psi_{stu}}(\mathbf{e} \,|\, \mathbf{x}) \,||\, p(\mathbf{e})] \,, \end{aligned} \tag{5.10}$$

where $q_{\varsigma_{stu}}(\mathbf{z} \,|\, \mathbf{x})$ and $q_{\psi_{stu}}(\mathbf{e} \,|\, \mathbf{x})$ are two variational distributions. $q_{\varsigma_{stu}}(\mathbf{z} \,|\, \mathbf{x})$ is implemented as a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2 \mathbf{I})$, where $\{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$ are the hyperparameters of the Gaussian distribution, returned by a neural network of input $\mathbf{x}$. A latent variable $\mathbf{z}$ can be drawn using the reparameterization trick [85, 138] $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\pi} \bigodot \boldsymbol{\sigma}$, where $\boldsymbol{\pi}$ is a random noise vector sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\bigodot$ is the element-wise product. $q_{\psi_{stu}}(\mathbf{e} \,|\, \mathbf{x})$ is implemented as a Categorical distribution $\text{Cat}(V, p^{\mathbf{e}}_1, \cdots, p^{\mathbf{e}}_V)$ whose parameters $\{p^{\mathbf{e}}_1, \cdots, p^{\mathbf{e}}_V\}$ are given by the probability outputs $\{e'_1, \cdots, e'_V\}$ of an expert-inference network, where V is the total number

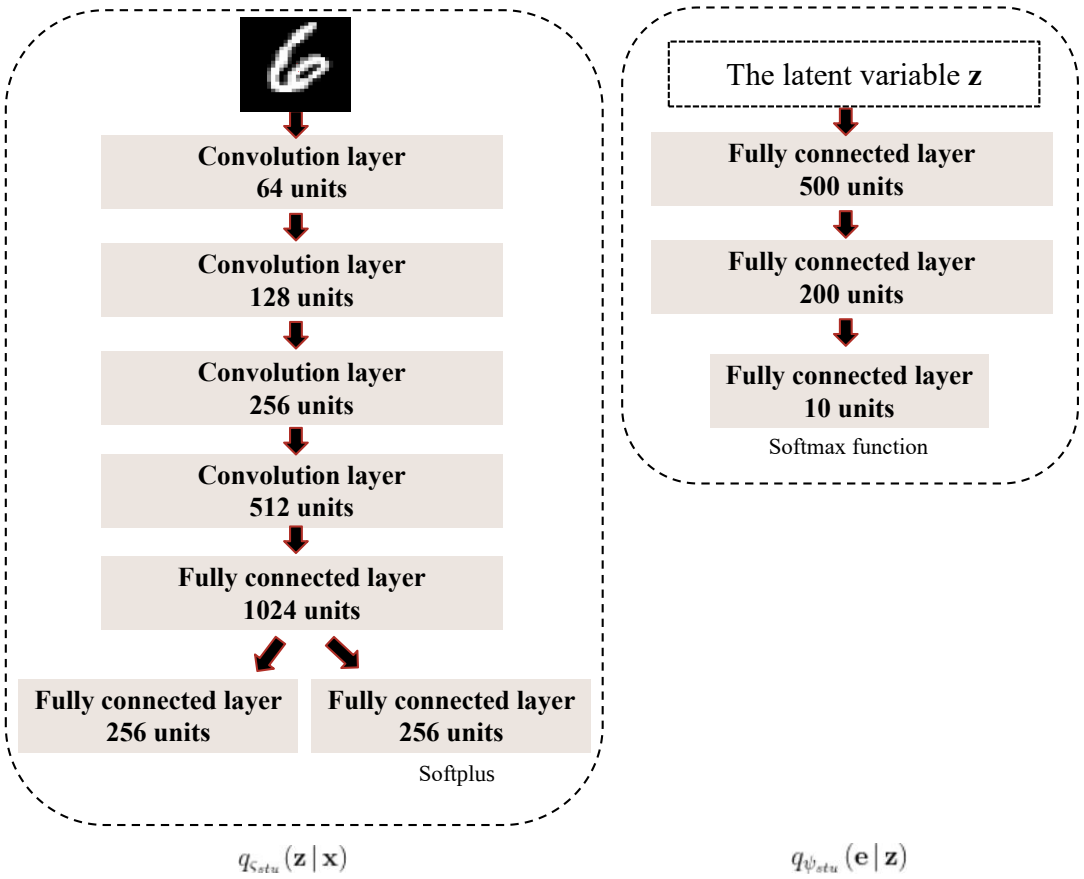


Figure 5.4: The network architecture of the inference models used for modelling $q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x})$ and $q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{z})$.

of the probability outputs. In order to reduce the variation of gradients [168] and use the one-hot form of $\mathbf{e}$ in the end-to-end training, we adopt the Gumbel-Max trick [56, 110], which was also used in [41, 72, 109, 168], to calculate a differentiable relaxation for the discrete variables:

$$\hat{e}_j = \frac{\exp\left((\log e'_j + g_j)/T\right)}{\sum_{i=1}^{V} \exp\left((\log e'_i + g_i)/T\right)}, \tag{5.11}$$

where e'_j is the j-th probability output obtained by the softmax layer of a neural network that models $q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{x})$ and $\hat{\mathbf{e}} = \{\hat{e}_1, \cdots, \hat{e}_V\}$ is the continuous relaxation of $\mathbf{e}$. g_j is sampled from Gumbel$(0, 1)$, while T is the temperature parameter controlling the smoothness. This sampling process is implemented during both inference and generation.

However, by employing a simple fixed prior $p(\mathbf{z})$ in Eq. (5.10) would lead to posterior collapse, [60], especially when learning several different data domains. In addition, employing a simple fixed prior $p(\mathbf{z})$ in Eq. (5.10) can not be helpful for embedding multiple knowledge sources from the teacher module into several clusters in the latent space of the student module. Furthermore, learning $q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{x})$, where ψ_{stu} represents the corresponding network parameters,

requires designing a large-scale neural network in order to process the high-dimensional variable $\mathbf{x}$. To address these issues, we propose to modify the two Kullback-Leibler (KL) terms in Eq. (5.10). Firstly, we replace $q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{x})$ with $q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{z})$ which can be implemented by using a simple neural network to process the low-dimensional latent variable $\mathbf{z}$. Therefore, the second KL term in Eq. (5.10) is replaced by $D_{KL}[q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{z})\,||\,p(\mathbf{e})]$. Secondly, we replace the prior $p(\mathbf{z})$ with a new distribution $p(\mathbf{z}\,|\,\mathbf{e}^{\star})$ by incorporating the identity information of the teacher experts, which uses the expert label[6] $\mathbf{e}^{\star}$ to regulate the information embedding process for the student module. Specifically, we can assign the expert label $\mathbf{e}^{\star}$ to each data sample $\mathbf{x}$ (generated or real) because we know that each data sample is learnt or generated by a specific teacher expert during the training phase. Then we implement $p(\mathbf{z}\,|\,\mathbf{e}^{\star})$ considering the Gaussian distribution $\mathcal{N}(B(\mathbf{e}), \mathbf{I})$, where $B(\cdot)$ is a function that firstly maps $\mathbf{e}^{\star}$ to a scalar $e^{\star}$ and then transforms $e^{\star}$ into a mean vector for the Gaussian distribution where each entry is $e^{\star}$, and $\mathbf{I}$ is the identity matrix. As a result, the first KL divergence term in the right-hand side of Eq. (5.10) is expressed by $D_{KL}[q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x})\,||\,p(\mathbf{z}\,|\,\mathbf{e}^{\star})]$, where ς_{stu} represents the corresponding network parameters. Therefore, $B(\mathbf{e}^{\star})$ can transfer the expert label $\mathbf{e}^{\star}$ to a mean vector that is used to regulate the updating of $q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x})$. When the proposed framework learns the first task, the objective function for the student module from Eq. (5.10) is defined as:

$$\begin{aligned}\mathcal{L}^{\text{DTS}}_{\text{single}}(\mathbf{x}, \theta_{stu}, \varsigma_{stu}, \psi_{stu}) = &- \mathbb{E}_{q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x}) q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{z})}\left[\log p_{\theta_{stu}}(\mathbf{x}\,|\,\mathbf{z}, \mathbf{e})\right] \\ &+ D_{KL}[q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x})\,||\,p(\mathbf{z}\,|\,\mathbf{e}^{\star})] \\ &+ D_{KL}[q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{z})\,||\,p(\mathbf{e})]\,,\end{aligned} \tag{5.12}$$

where $\mathbf{z}$ in the third KL divergence term is obtained from $q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x})$. $\mathbf{e}^{\star}$ in the prior distribution $p(\mathbf{z}\,|\,\mathbf{e}^{\star})$ is the expert label[7], which was associated with the data $\mathbf{x}$. We desire to minimize the KL divergence between $q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x})$ and the prior $p(\mathbf{z}\,|\,\mathbf{e}^{\star})$, assumed to be Gaussian distributions, in order to allow the student module to embed knowledge inferred from different sources (experts that are assigned by the unique expert-variable from the teacher module) into different regions of its latent space. We provide the detailed network design used for modelling $q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x})$ and $q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{z})$ in Fig. 5.4.

[6]Since the latent variable $\mathbf{e}$ can not be obtained from the dataset, we assign the expert label $\mathbf{e}^{\star}$ for each sample $\mathbf{x}$ to denote the identity information of the expert. $\mathbf{e}^{\star}$ (one-hot vector) has the same dimension as $\mathbf{e}$.

[7]We assign the expert label $\mathbf{e}^{\star}$ (one-hot vector) to each data sample $\mathbf{x}$, which is an observed variable and is different from the latent variable $\mathbf{e}$.

5.2.6 Student Learning

Existing KD approaches assume that data samples are provided by the user during the training [168, 53]. However, in the context of the lifelong learning setting, we do not have access to past samples, and these KD approaches can not be applied in our framework. In this chapter, we introduce a new training approach in which past data samples are generated by the teacher module. Then, these pseudo samples can be used for training the student module. Additionally, unlike existing KD approaches that transfer knowledge only at the logit-level [81, 126, 128], the proposed approach can transfer statistic data representations through sampling without accessing any real samples and labels of past tasks. Moreover, the proposed training approach transfers the knowledge from the teacher module represented by multiple source distributions, implemented by mixtures of expert GAN models, as described in Section 5.2.4, to a compact student latent space. By considering the teacher's knowledge and a new task, we design a loss function which is used to update the student module in the mini-batch learning manner:

$$\begin{aligned}\mathcal{L}_{\text{Stu}}^{\text{DTS}}(\mathbf{X}_{batch}, \theta_{stu}, \varsigma_{stu}, \psi_{stu}) = &\frac{1}{m}\sum_{j=1}^{m}\{-\mathbb{E}_{q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x}_j)q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{z})}\left[\log p_{\theta_{stu}}(\mathbf{x}\,|\,\mathbf{z},\mathbf{e})\right] \\ &+ D_{KL}[q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x}_j)\,||\,p(\mathbf{z}\,|\,\mathbf{e}_j^{\star})] \\ &+ D_{KL}[q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{z}_j)\,||\,p(\mathbf{e})]\}\,,\end{aligned} \tag{5.13}$$

where $\mathbf{z}_j$ is drawn from $q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x}_j)$ and $\mathbf{x}_j$ is the j-th sample of the data batch $\mathbf{X}_{batch}$. $\mathbf{e}_j^{\star}$ is the expert label, which was associated with the data $\mathbf{x}_j$. To form $\mathbf{X}_{batch}$ in each training step, we draw the same number of generative samples from each teacher expert. These generated samples are incorporated together with the real training samples of the current task. The number of generated samples by each mixture expert component of the teacher module is considered in the same proportion as those sampled from the database corresponding to the newly given task. Compared to the GMM described in Chapter 4, the student's learning in D-TS has several differences: (1) The D-TS employs a conditional prior distribution to regularize the student's learning, leading to better reconstruction performance; (2) The teacher module in D-TS can provide high-quality knowledge transfer for the student's learning compared with the GMM; (3) The D-TS extends the student module to the supervised learning task as described in Section 5.3.1.

5.2.7 The Training Algorithm

In the following, we provide the pseudocode in **Algorithm 5** describing the pipeline of the proposed lifelong unsupervised learning strategy, which is summarized into three steps:

- **Step 1. Selection and expansion mechanism:** When starting learning the first task, the teacher module has no experts. In this case, we build a new expert and learn it according to the training procedure from **Step 2**, otherwise, we verify the teacher's expansion and selection as follows: we evaluate the KDS between each teacher's expert and the training set of the current task (the t-th task) by using either KFD or SE as a criterion, as described in Section 5.2.4. Then we consider the threshold *hold* from Eq. (5.6), to decide either the selection of an expert to be updated, or initiating the expansion process for the teacher module. We employ *expansion* in **Algorithm 5** to denote the expansion state.

- **Step 2. Updating an expert for the teacher module:** If the teacher module performs the expansion at the current task learning (the t-th task), then we directly update the newly added expert on samples from the t-th task, otherwise, we form a joint dataset $D_{SU}^{t} \bigcup D_{GU}^{\widehat{s}}$, including samples from the new training set D_{SU}^{t} from the t-th task and the dataset $D_{GU}^{\widehat{s}}$ generated by $G_{\theta_{\widehat{s}}}$, where $\widehat{s}$ is the index of the selected expert. We employ D_{SU}^{t} to denote the unlabelled dataset, which differs from the labelled dataset D_{S}^{t}. We then update the selected expert $G_{\theta_{\widehat{s}}}$ on the data batch $\mathbf{X}_{batch}$ from the joint dataset $D_{SU}^{t} \bigcup D_{GU}^{\widehat{s}}$ at the t-th task by using Eq. (5.8) and Eq. (5.9) in the mini-batch learning manner [92]. Since we know the training dataset D_{SU}^{t} will be learnt by the selected teacher expert and the expert identity of each generated sample, we can assign the expert variable $\mathbf{e}$ for each sample from the joint dataset $D_{SU}^{t} \bigcup D_{GU}^{\widehat{s}}$. The total number of training steps for the teacher and student modules is determined by $iterations = epoch \times (dataSize/m)$ where $epcoh = 20$ and $dataSize$ are the number of training epochs and the dataset size. $m = 64$ is the batch size.

- **Step 3. Student updating:** During each batch learning, we draw the same number of generative samples from each teacher expert. These generative samples are incorporated together with the real training samples of the current task, considered in an equal probability with those generated by each of the experts. These will form a batch of samples $\mathbf{X}_{batch}$ for updating the student module using Eq. (5.13). We also update the $q_{\psi_{stu}}(\mathbf{e} \,|\, \mathbf{z})$

Algorithm 5: D-TS-KFD unsupervised training algorithm

Input: All training databases
Output: The model's parameters
for $t < N$ **do**
  Step 1: Selection and expansion mechanisms ;
  if $t == 1$ **then**
    Build a new expert $\{G_{\theta_1}, \mathcal{D}_{\eta_1}\}$ for the teacher module ;
  end
  else
    Calculate KDS between each teacher expert and D_{SU}^t ;
    Check the selection and expansion using Eq. (5.6) ;
    if *expansion = True* **then**
      Build a new expert for the teacher module ;
    end
    else
      Select an expert for the current task learning according to Eq. (5.7) ;
    end
  end
  Get the unlabelled training set D_{SU}^t ;
  if *expansion = False* **then**
    Form a joint dataset $D_{SU}^t = D_{SU}^t \bigcup D_{GU}^{\widehat{s}}$, $D_{GU}^{\widehat{s}}$ is generated by $G_{\theta_{\widehat{s}}}$;
  end
  Step 2 and 3: Teacher and student learning ;
  for *index < iterations (All models are only updated once in each iteration)* **do**
    Get the training data batch $\mathbf{X}_{batch}$ from D_{SU}^t ;
    Update the teacher on $\mathbf{X}_{batch}$ using Eq. (5.8) and Eq. (5.9) ;
    Generate the data batch $\mathbf{X}'_{batch}$ using the teacher module ;
    Combine $\mathbf{X}'_{batch}$ and $\mathbf{X}_{batch}$ to form a new data batch $\mathbf{X}_{batch} = \mathbf{X}_{batch} \bigcup \mathbf{X}'_{batch}$;
    Update networks of the student on $\mathbf{X}_{batch}$ using Eq. (5.13) ;
    Update $q_{\psi_{stu}}(\mathbf{e} \mid \mathbf{z})$ on $\mathbf{X}_{batch}$ with the associated expert labels using cross-entropy loss ;
  end
end

on the data batches $\mathbf{X}_{batch}$ with the associated expert labels using the cross-entropy loss, where $\mathbf{z}$ is obtained from $q_{\varsigma_{stu}}(\mathbf{z} \mid \mathbf{x})$ with the sample $\mathbf{x}$. The student model is each time initialized with the parameters learnt previously, while only when trained for the first time its parameters would be randomly generated.

5.3 Applications

In the following we provide some applications of the proposed lifelong learning framework.

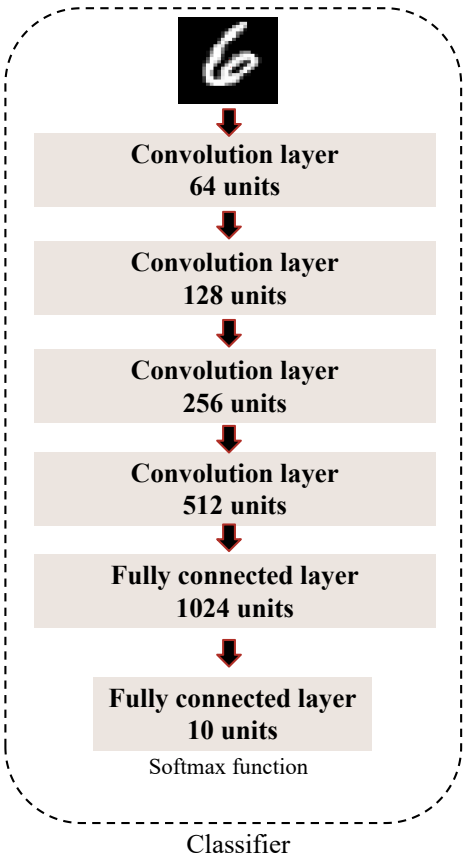


Figure 5.5: The network architecture of the classifiers $F_{\delta_{\widehat{s}}^{\text{Teah}}}(\mathbf{x})$ and $F_{\delta}(\mathbf{x})$. The final layer in the classifier outputs a Q-dimensional probability vector using the softmax activation function.

5.3.1 Prediction Tasks

In this section, we extend the D-TS framework for classification tasks. We implement each expert from the teacher module by using a combination of a generator and a solver. The solver is a neural network, which outputs the class probability, trained by minimizing the cross-entropy loss, defined as:

$$\mathcal{L}^{c}_{\text{Teach}}(\mathbf{x}, \mathbf{y}, \delta_{\widehat{s}}^{\text{Teah}}, Q) = f_{\text{CE}}(F_{\delta_{\widehat{s}}^{\text{Teah}}}(\mathbf{x}), \mathbf{y}, Q)\,, \tag{5.14}$$

where $f_{\text{CE}}(\cdot,\cdot,\cdot)$ is the cross-entropy loss function, which is defined as:

$$f_{\text{CE}}(\mathbf{y}', \mathbf{y}, Q) = -\sum_{i=1}^{Q} y_i \log y'_i\,, \tag{5.15}$$

where y_i and y'_i are the i-th entry of the class label $\mathbf{y}$ (one-hot vector) and the prediction $\mathbf{y}'$, respectively. Q is the dimension of the class label $\mathbf{y}$. $\{\mathbf{x}, \mathbf{y}\}$ is a paired sample from the labelled dataset. $F_{\delta_{\widehat{s}}^{\text{Teah}}}(\mathbf{x})$ is the solver (classifier), defined by parameters $\delta_{\widehat{s}}^{\text{Teah}}$ in the selected teacher

Algorithm 6: D-TS-KFD supervised training algorithm

Input: All training databases
Output: The model's parameters

for $t < N$ **do**
  Step 1: Selection and expansion mechanisms ;
  if $t == 1$ **then**
    Build a new expert $\{F_{\theta_1}, \mathcal{D}_{\eta_1}, F_{\delta_1^{\text{Teah}}}\}$ for the teacher module ;
  end
  else
    Calculate KDS between each teacher expert and D_S^t ;
    Check the selection and expansion using Eq. (5.6) ;
    if *expansion = True* **then**
      Build a new expert for the teacher module ;
    end
    else
      Select an expert for the current task learning according to Eq. (5.7) ;
    end
  end
  Get the training set D_S^t ;
  if *expansion = False* **then**
    $D_S^t = D_S^t \bigcup D_G^{\widehat{s}}$, $D_G^{\widehat{s}}$ generated by the selected teacher expert $G_{\theta_{\widehat{s}}}$;
  end
  Step 2 and 3: Teacher and student learning ;
  for *index < iterations (All models are only updated once in each iteration)* **do**
    Get the data batch $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\}$ from D_S^t and $\{\mathbf{X}'_{batch}, \mathbf{Y}'_{batch}\}$ generated by all teacher experts ;
    Update the teacher on $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\}$ using Eq. (5.8), Eq. (5.9) and Eq. (5.16) ;
    $\mathbf{X}_{batch} = \mathbf{X}_{batch} \bigcup \mathbf{X}'_{batch}$, $\mathbf{Y}_{batch} = \mathbf{Y}_{batch} \bigcup \mathbf{Y}'_{batch}$;
    Update the student on $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\}$ using Eq. (5.17) ;
    Update the classifier of the student module on $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\}$ using Eq. (5.16) ;
    Update $q_{\psi_{stu}}(\mathbf{e} \mid \mathbf{z})$ on $\mathbf{X}_{batch}$ with the associated expert variables using cross-entropy loss ;
  end
end

expert[8] (the $\widehat{s}$-th expert). In order to allow the student module to perform data classification tasks, we introduce a classifier model $F_{\delta_{stu}}(\mathbf{x})$ for the student module, trained on images and labels obtained from the data generated by the teacher module and by using image samples from the current database:

$$\mathcal{L}_{\text{Stu}}^{c}(\mathbf{X}_{batch}, \mathbf{Y}_{batch}, \delta_{stu}, Q) = \frac{1}{m} \sum_{j=1}^{m} \left\{ f_{\text{CE}} \left(F_{\delta_{stu}} \left(\mathbf{x}_j \right), \mathbf{y}_j, Q \right) \right\}, \tag{5.16}$$

where $\{\mathbf{x}_j, \mathbf{y}_j\}$ is the j-th paired sample from the labelled data batch $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\}$ which also contains the generated samples and the associated class labels obtained by each teacher

[8] We employ $\delta_{\widehat{s}}^{\text{Teah}}$ to denote the classifier's parameter set of the selected expert, which is different from δ_{stu} that denotes the classifier's parameter set of the student module.

expert. $m = 64$ is the data batch size. We provide the detailed network architecture of the classifiers $F_{\delta_{\widehat{s}}^{\text{Teah}}}(\mathbf{x})$ and $F_{\delta_{stu}}(\mathbf{x})$ in Fig. 5.5. Eq. (5.16) compares the outputs predicted by the student module against the ground-truth labels from the labelled data batch $\{\mathbf{X}_{batch}, \mathbf{Y}_{batch}\}$. Eq. (5.16) is only used to optimize $F_{\delta_{stu}}(\mathbf{x})$ and we also introduce an objective function to optimize both $F_{\delta_{stu}}(\mathbf{x})$ and other components of the student module by incorporating the variable $\mathbf{u}$ accounting for the class information, resulting in:

$$\begin{aligned}\mathcal{L}_{\text{Stu}}^{Sup}(\mathbf{X}_{batch}, \theta_{stu}, \varsigma_{stu}, \delta_{stu}) = &\frac{1}{m}\sum_{j=1}^{m}\Big\{ - \mathbb{E}_{q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x}_j)q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{z}_j)q_{\delta_{stu}}(\mathbf{u}\,|\,\mathbf{x}_j)}\left[\log p_{\theta_{stu}}(\mathbf{x}\,|\,\mathbf{z},\mathbf{u},\mathbf{e})\right] \\ &+ D_{KL}[q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x}_j)\,||\,p(\mathbf{z}\,|\,\mathbf{e}_j^{\star})] \\ &+ D_{KL}[q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{z})\,||\,p(\mathbf{e})] \\ &+ D_{KL}[q_{\delta_{stu}}(\mathbf{u}\,|\,\mathbf{x}_j)\,||\,p(\mathbf{u})]\Big\},\end{aligned} \tag{5.17}$$

where $\mathbf{e}_j^{\star}$ is the expert label, which was associated with the data $\mathbf{x}_j$. The variational distribution $q_{\delta_{stu}}(\mathbf{u}\,|\,\mathbf{x})$ is modelled by a categorical distribution $\text{Cat}(Q, u'_1, \cdots, u'_Q)$ whose parameters $\{u'_1, \cdots, u'_Q\}$ are given by the classifier $F_{\delta_{stu}}(\mathbf{x})$, where $Q = 10$ is the total number of classes. We also employ the Gumble-max distribution [56] to draw the continuous relaxation of the variable $\mathbf{u}$ in order to enable end-to-end optimization. $p(\mathbf{u})$ is a uniform categorical distribution $\text{Cat}(Q, p_1^u = 1/Q, \cdots, p_Q^u = 1/Q)$.

5.3.2 Learning Disentangled Representations

Most artificial intelligence learning approaches aiming to describe meaningful feature variations through generative learning are based on inference models such as VAEs. Certain constraints can be imposed on the VAE loss function in order to induce disentangled representations [19, 26, 41, 48, 64, 80]. For enticing the learning of disentangled representations under the lifelong learning framework, we consider the following loss function for the student module:

$$\begin{aligned}\mathcal{L}_{\text{Stu}}^{Dis}(\mathbf{X}_{batch}, \theta_{stu}, \varsigma_{stu}) = &\frac{1}{m}\sum_{j=1}^{m}\Big\{ - \mathbb{E}_{q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x}_j)q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{z})}\left[\log p_{\theta_{stu}}(\mathbf{x}\,|\,\mathbf{z},\mathbf{e})\right] \\ &+ \gamma\Big\|D_{KL}[q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x}_j)\,||\,p(\mathbf{z}\,|\,\mathbf{e}_j^{\star})] - C\Big\|\Big\},\end{aligned} \tag{5.18}$$

where γ and C control the degree of disentanglement. We set $\gamma = 4$ in our experiments to avoid sacrificing much reconstruction ability while C is linearly increased from a very small value of

0.5 to 25.0 during the training, [64]. We omit the KL term on $q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{z})$ since this term would not benefit from disentangled representation learning.

5.3.3 Inter-Domain Interpolation

After lifelong learning, the inference model of the student module can provide several latent representations $\{\mathbf{z}^1, \mathbf{z}^2 \ldots, \mathbf{z}^N\}$ where each $\mathbf{z}^t$ represents the generative factors for images obtained from the dataset D^t_{SU}. The lifelong learning model, besides representing the generative factors for each domain/task (the t-th task), it also captures the shared generative factors across domains. To evaluate this property, we manipulate the latent variable space by interpolating the images drawn from two different domains. Let $\mathbf{x}^i$ and $\mathbf{x}^j$ be two images obtained from the datasets of the i-th task and j-th task, respectively. Let we define a function that receives a pair of images and returns the interpolated image:

$$f_{\text{interpolation}}(i, j, c, \mathbf{x}^i, \mathbf{x}^j) = \text{Dec}(\text{Enc}(\mathbf{x}^j)c + \text{Enc}(\mathbf{x}^i)(1-c))\,, \tag{5.19}$$

where $\text{Enc}(\cdot)$ is the encoder that receives the data samples $\mathbf{x}^j$ and $\mathbf{x}^i$ and returns the latent variables $\mathbf{z}^j$ and $\mathbf{z}^i$, respectively. $\text{Dec}(\cdot)$ is the decoding function implemented by a decoder (We only consider the continuous variable $\mathbf{z}$ in the VAE framework), and $c \in [0, 1]$ is the interpolation parameter in Eq. (5.19). In order to simplify the notation, we employ $\tilde{\mathbf{x}}_{i\to j}(c)$ to denote the result from the function $f_{\text{interpolation}}(i, j, c, \mathbf{x}^i, \mathbf{x}^j)$. We extend the image interpolation from [123] into exploring the joint latent space of all previously learnt N domains/tasks, under the lifelong learning setting:

- **Boundary conditions** [123]**.** $\tilde{\mathbf{x}}_{i\to j}(0) = \mathbf{x}^i$ and $\tilde{\mathbf{x}}_{i\to j}(1) = \mathbf{x}^j$ when $\text{Dec}(\cdot)$ is the optimal decoding function.

- **Monotonicity** [123]**.** We assume that Dec(.) is the decoding function. For a given distance measure $f_{\text{similarity}}(\cdot, \cdot)$ evaluating the similarity between two images, we can define the distance from the interpolated image to the original input.

$$f_{\text{similarity}}(\tilde{\mathbf{x}}_{i\to j}(c), \mathbf{x}^i) \leq f_{\text{similarity}}(\tilde{\mathbf{x}}_{i\to j}(c'), \mathbf{x}^i)\,, \tag{5.20}$$

Table 5.1: The performance of various models under the MSFIR lifelong learning setting.

	MSE					SSMI					PSNR				
Datasets	LGM	D-TS-KFD	D-TS-SE	BE-Stu	LTS	LGM	D-TS-KFD	D-TS-SE	BE-Stu	LTS	LGM	D-TS-KFD	D-TS-SE	BE-Stu	LTS
MNIST	19.60	26.84	28.61	33.66	73.97	0.90	0.88	0.87	0.86	0.73	22.51	21.14	20.64	20.13	17.10
SVHN	292.15	29.67	31.04	71.58	42.98	0.36	0.65	0.64	0.47	0.54	11.33	12.65	12.58	12.13	11.91
Fashion	80.95	39.35	48.96	149.26	45.64	0.17	0.72	0.66	0.47	0.72	12.65	18.38	17.01	13.57	17.77
IFashion	94.32	35.92	37.94	83.44	37.60	0.58	0.74	0.75	0.61	0.76	16.35	18.70	17.86	15.89	18.34
RMNIST	19.58	24.09	23.81	24.29	21.97	0.90	0.89	0.89	0.90	0.90	22.51	21.46	21.53	21.57	21.64
Average	101.32	**31.17**	34.07	72.45	44.83	0.58	**0.78**	0.76	0.66	0.73	17.07	**18.47**	17.92	16.66	17.35

where $c' > c$ and

$$f_{\text{similarity}}(\tilde{\mathbf{x}}_{i\to j}(c'), \mathbf{x}^j) \leq f_{\text{similarity}}(\tilde{\mathbf{x}}_{i\to j}(c), \mathbf{x}^j)\,, \tag{5.21}$$

- **Smoothness** [123]**.** The interpolation function $f_{\text{interpolation}}(i, j, c, \mathbf{x}^i, \mathbf{x}^j)$ is Lipschitz continuous with a constant M.

$$\left\| f_{\text{interpolation}}(i, j, c, \mathbf{x}^i, \mathbf{x}^j) - f_{\text{interpolation}}(i, j, c + a, \mathbf{x}^i, \mathbf{x}^j) \right\| \leq M|a|\,. \tag{5.22}$$

Unlike in [123] which only considers a static dataset, we aim to learn latent representations under the lifelong learning setting, which is more challenging because neural network models would forget previously learnt latent representations when trained on a new task. Given the image interpolation properties from above, we define a new criterion evaluating the effectiveness of the model when performing image interpolation:

$$f_{\text{similarity2}}(\tilde{\mathbf{x}}_{i\to j}(c), \mathbf{x}^j), \quad c > 0.5\,, \tag{5.23}$$

where $f_{\text{similarity2}}(\cdot)$ is a pre-defined criterion, which can be implemented as the image reconstruction error. If $f_{\text{similarity2}}(\tilde{\mathbf{x}}_{i\to j}(c), \mathbf{x}^j)$ is small, this means that the interpolated result $\tilde{\mathbf{x}}_{i\to j}(c)$ is very similar to $\mathbf{x}^j$, as c increases.

5.4 Experiments

In the following, we evaluate the application of the Dynamic Teacher-Student Network (D-TS) in lifelong learning tasks. We use the Adam optimization algorithm [82], with a learning rate of 0.0002 and the hyperparameter $\beta = 0.5$. The number of training epochs for each task is set to

Table 5.2: The performance when learning a sequence of six tasks.

	MSE					SSMI				
Datasets	BE-Stu	D-TS-KFD	D-TS-SE	LGM	LTS	BE-Stu	D-TS-KFD	D-TS-SE	LGM	LTS
MNIST	24.32	23.71	23.21	18.97	26.96	0.89	0.90	0.90	0.90	0.89
SVHN	85.21	31.93	30.07	229.13	61.45	0.49	0.64	0.65	0.35	0.45
Fashion	167.38	43.72	42.14	90.62	81.56	0.47	0.71	0.71	0.15	0.56
IFashion	113.90	41.62	41.18	173.60	60.84	0.62	0.74	0.74	0.38	0.66
CIFAR10	359.09	203.70	208.72	676.13	220.72	0.21	0.35	0.33	0.04	0.33
Ommiglot	275.66	179.83	182.27	273.54	147.43	0.65	0.82	0.80	0.68	0.84
Average	170.93	**87.42**	87.93	243.66	99.83	0.56	0.69	0.69	0.42	0.62

Table 5.3: The lifelong learning of a sequence of six tasks.

	PSNR				
Datasets	BE-Stu	D-TS-KFD	D-TS-SE	LGM	LTS
MNIST	21.52	21.55	21.66	22.62	21.03
SVHN	11.80	12.77	13.20	11.45	13.42
Fashion	13.36	17.75	17.95	11.67	16.52
IFashion	15.10	17.57	17.19	12.58	16.62
CIFAR10	14.91	15.08	15.19	12.22	15.33
Ommiglot	16.91	18.80	18.36	17.46	19.26
Average	15.60	17.25	**17.26**	14.67	17.03

20. In all experiments, we consider 60,000 randomly selected images from each database that does not split the training and testing sets for training and 10,000 for testing, unless specified otherwise.

5.4.1 The Evaluation of Representation Learning During Unsupervised Lifelong Learning

We evaluate the performance of various methods for unsupervised lifelong learning. We consider five tasks, in a sequence called MFSIR, defined by the databases: MNIST [93], Fashion [177], SVHN [118], InverseFashion (IFashion) and Rotated MNIST (RMNIST). The results are reported in Table 5.1, where we use the threshold $hold = 150$ for D-TS-KFD and $hold = 50$ for D-TS-SE. We consider the Mean Square Error (MSE), the Structural Similarity Index Measure (SSIM) [68] and the Peak-Signal-to-Noise Ratio (PSNR) [68] for evaluating the image reconstruction quality.

For comparison, we consider LTS [180], which uses a large network architecture, defined as a single processing module, for the teacher module, and we consider the Lifelong Generative Modeling (LGM) [132]; we also adapt the BatchEnsemble [174] in order to train a student

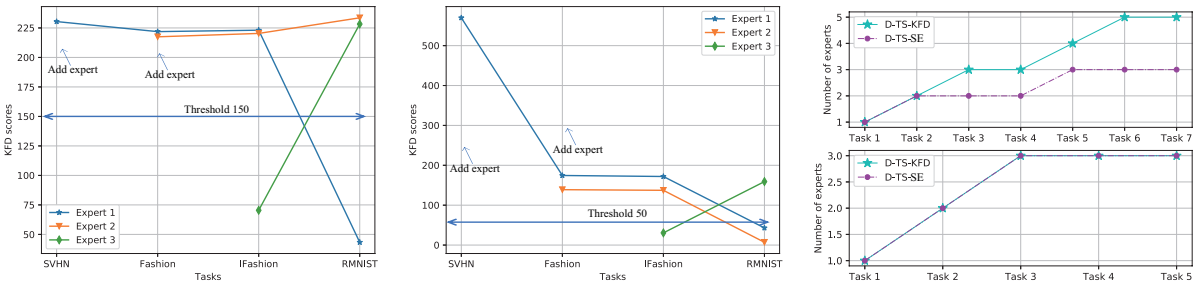
model, under the unsupervised lifelong learning setting. We consider building an ensemble of VAEs as the teacher module, where the number of components is equal to the number of tasks. Then we train a VAE model as the student module, which accumulates knowledge from both the teacher module and the tasks learnt during the lifelong learning. The models D-TS-SE or D-TS-KFD, which employ the knowledge distillation for D-TS, as explained in Section 5.2.3, by using either SE or KFD as the expert selection criterion, respectively, and achieve the best result for every task.

We evaluate the performance when learning a sequence of seven challenging tasks, defined by databases which contain complex and diverse images: MNIST, SVHN, Fashion, IFashion, CIFAR10 [87], Omniglot [90] and MNIST. We consider a threshold $hold = 150$ for both D-TS-KFD and D-TS-SE. The results are provided in Table 5.2 and Table 5.3. The proposed method outperforms other models in this challenging learning setting which, includes several databases, some with strong content similarities.

We also investigate how the proposed framework adds a new expert during lifelong learning by evaluating either KFD or SE after each task switch, and the results are shown in Figures 5.6-a and 5.6-b. After learning the first task, KFD between the first expert and the next task (SVHN database), is 230 and therefore the teacher module adds a new expert to learn SVHN. Then, after learning the third task, KFD between each expert and the next task (IFashion database) is smaller than 150, and therefore the teacher module reuses the third expert in order to learn IFashion. KFD and SE measures exhibit different characteristics. For instance, KFD is small when two tasks share similar visual concepts, while for example the SE score is small when two databases share similar global structures and colour palettes. The architecture expansion of the teacher module is shown in Fig. 5.6-c, where D-TS-KFD and D-TS-SE lead to a reasonable number of experts, each capturing specific knowledge from the databases.

5.4.2 Study of The Latent Space of The Student Module

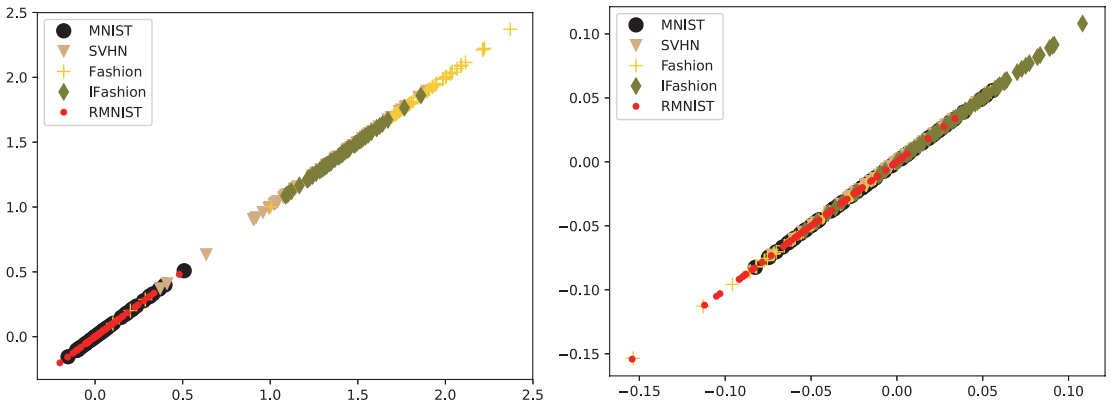
Projection of the latent variables. In the following, we project the latent variables extracted by the student module to show how similar knowledge sources are embedded into the same cluster in the latent space, considering images drawn from different domains: MNIST, SVHN, Fashion, IFashion and RMNIST (MSFIR sequence). For this analysis, we train D-TS-KFD under MSFIR lifelong learning with the threshold for adding a new component set to $hold = 220$



(a) KFD when learning 5 tasks. (b) SE when learning 5 tasks. (c) Network expansion in D-TS.

Figure 5.6: Knowledge discrepancy evaluation and the expansion of the network during the training.

in Eq. (5.6). After the training, we select a batch of 64 images for each domain. Then we use the inference model $q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x})$ to produce the mean vector (hyperparameter of the Gaussian distribution) for each image and we average the results as z^* which is used as coordinates (z^*, z^*) for each image in Fig. 5.7. We can observe that the student module embeds similar domains, as they are modelled by a certain expert from the teacher module, into the same cluster in the latent space. See the overlap between the coordinates of MNIST and RMNIST, as well as between Fashion and IFashion, where the latent spaces are better separated when using the conditional prior, according to the results from Fig. 5.7-a. When the proposed D-TS-KFD does not use the conditional prior, the knowledge learnt by each teacher expert is embedded into the same region of the latent space, as shown in Fig. 5.7-b.



(a) With conditional prior. (b) Without conditional prior.

Figure 5.7: Latent space projections for D-TS model.

Lifelong learnt disentangled representations. In the following experiments, we evaluate the ability of the D-TS model to create disentangled representations under lifelong learning, as discussed in Section 5.3.2. We train D-TS-KFD under CelebA to 3D-Chair lifelong learning using the loss function from Eq. (5.18), where $\gamma = 4$ and C is linearly increased from 0.5 to 25.0

during the training, [64]. We can observe from the disentangled results, shown in Fig. 5.8a-f, that the student module can capture meaningful generative factors of images, such as changing the gender of a person, face size, face makeup, hairstyle, chair size or by rotating the object (chair) shown in the image.

Table 5.4: MSE of the reconstructed interpolated images using Eq. (5.24).

Interpolation	D-TS-KFD	D-TS-KFD-Without	LTS
CelebA → CACD	208.53	249.51	440.11
CACD → CelebA	179.14	198.05	409.43
CIFAR10 → Sub-ImageNet	234.32	230.74	346.77
Sub-ImageNet →CIFAR10	221.35	218.82	316.61
Average	**210.84**	224.28	378.23

Inter-domain interpolations enabled by lifelong learning. Interpolations in the latent spaces were previously used for exploring model representations [183]. A good interpolation result indicates that the model can learn a meaningful latent representation of data by exploring the latent space between two locations from this space. Following the description from Section 5.3.3 we show that the proposed model not only that it can learn meaningful representations across domains over time, but it can also be used to explore the inter-domain latent spaces. We train the proposed D-TS-KFD model under the CelebA [105], CACD [24], 3D-Chair [14] and Omniglot (CCCO) lifelong learning for exploring their joint latent spaces. We show the interpolation results obtained when using the interpolation equation (Eq. (5.19)) from Section 5.3.3, when varying $c \in [0, 1]$, in Fig. 5.9, where we can observe how a human face can be smoothly transformed into images of multiple domains, while a chair is transformed into a human face while its frame gradually becomes the eyes and mouth in the face. These results indicate that the student module has additional modelling abilities and can capture surprising relationships between different latent space regions from multiple domains.

In the following, we evaluate the performance of the proposed D-TS-KFD in the image interpolation task. We train the proposed D-TS-KFD model considering CelebA, CACD, CIFAR10, Sub-ImageNet, SVHN and MNIST (CCCSSM) database lifelong learning. After the training, we extract the latent variables $\mathbf{z}^i$ and $\mathbf{z}^j$ from two images $\mathbf{x}^i$ and $\mathbf{x}^j$ belonging to the i-th and j-th domain/task, respectively, using the student module. We consider 1000 such image pairs from different domains. Then we generate the interpolated reconstructions by using the decoder:

$$\hat{\mathbf{x}} = p_{\theta_{stu}}(\mathbf{x} \,|\, 0.2\,\mathbf{z}^i + 0.8\,\mathbf{z}^j, \mathbf{e}^j)\,, \tag{5.24}$$

(a) Gender change

(b) Face size

(c) Makeup

(d) Hair style

(e) Chair size

(f) Chair turning

Figure 5.8: Results when varying the latent variables under the CelebA to 3D-Chair lifelong learning. We change a single latent variables in the latent space from -3.0 to 3.0 while fixing the others.

where $\mathbf{z}^i$ and $\mathbf{z}^j$ are the latent variable for $\mathbf{x}^i$ and $\mathbf{x}^j$, respectively, obtained using the inference model $q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x})$. $\mathbf{e}^j$ is the expert variable for $\mathbf{x}^j$, obtained using the inference model $q_{\psi_{stu}}(\mathbf{e}\,|\,\mathbf{x})$. Then we evaluate the interpolation performance of the D-TS-KFD by employing Eq. (5.23) in which we calculate MSE between the real image $\mathbf{x}^j$ and the interpolated reconstruction $\hat{\mathbf{x}}$. $p_{\theta_{stu}}(\mathbf{x}\,|\,0.2\,\mathbf{z}^i + 0.8\,\mathbf{z}^j, \mathbf{e}^j)$ is the decoder of the student module. We report the MSE results in Table 5.4, where 'D-TS-KFD-Without' represents D-TS-KFD without using the regularized variable $\mathbf{e}^\star$ in the KL divergence term $D_{KL}[q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x})\,||\,p(\mathbf{z}\,|\,\mathbf{e}^\star)]$ from Eq. (5.12) ($p(\mathbf{z}\,|\,\mathbf{e}^\star) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ is fixed). These results indicate that D-TS-KFD can provide a smaller reconstruction error than other baselines such as LTS, which demonstrates that D-TS-KFD can learn a smooth latent space for multiple domains under lifelong learning.

5.4.3 Lifelong Learning of Databases With Complex Images

For this experiment, we train various models under the CCCSSM database sequence lifelong learning. These databases contain a variety of rather complex images showing human faces as well as natural images among others. We evaluate the Mean Square Error (MSE), Structural Similarity Index Measure (SSMI) and PSNR results for the reconstructed images from the six

Table 5.5: Image reconstruction errors when learning datasets containing complex images, such as the CCCSSM sequence.

	MSE					**SSMI**				
Datasets	LGM	D-TS-KFD	D-TS-SE	BE-Stu	LTS	LGM	D-TS-KFD	D-TS-SE	BE-Stu	LTS
CelebA	703.62	137.67	141.47	153.25	215.43	0.05	0.55	0.56	0.54	0.40
CACD	979.18	160.66	123.49	265.80	246.99	0.03	0.58	0.65	0.45	0.44
CIFAR10	515.66	161.05	150.78	306.72	215.42	0.08	0.42	0.44	0.23	0.33
Sub-ImageNet	551.39	172.56	154.41	303.50	230.55	0.08	0.41	0.45	0.24	0.33
SVHN	62.15	28.76	34.08	52.71	34.90	0.20	0.65	0.62	0.50	0.60
MNIST	22.44	31.41	28.34	25.17	25.66	0.88	0.86	0.88	0.89	0.89
Average	472.51	115.35	**105.43**	184.53	161.49	0.22	0.58	**0.60**	0.48	0.50

Table 5.6: Image reconstruction errors when learning datasets containing complex images, such as the CCCSSM sequence.

	PSNR				
Datasets	LGM	D-TS-KFD	D-TS-SE	BE-Stu	LTS
CelebA	12.18	18.42	18.81	19.00	16.37
CACD	10.86	18.15	19.39	16.80	16.12
CIFAR10	13.35	16.23	16.82	15.81	15.32
Sub-ImageNet	13.14	16.00	16.82	15.88	15.08
SVHN	13.50	12.95	13.70	13.43	13.94
MNIST	21.74	20.18	20.69	21.27	21.16
Average	14.13	16.99	**17.71**	17.03	16.33

datasets, and the results are provided in Table 5.5 and Table 5.6. From these tables, we can observe that the proposed framework performs better on these complex image datasets, when compared to other methods by a large margin. The proposed D-TS-SE, employing the SE criterion for deciding whether to add or not a new expert, performs better than D-TS-KFD, which uses the KFD criterion, for the lifelong learning of sequences of both complex and simple image databases. Visual results for the lifelong learning of the CCCSSM sequence of databases, produced by D-TS-KFD and by D-TS-SE, are provided in Fig. 5.10 and Fig. 5.11, respectively, where each expert is able to capture information which is different from that associated with all other experts. Furthermore, the teacher module is able to embed, when appropriate, the information associated with two similar databases into a single expert, which accelerates the training speed and reduces the required memory. The teacher module expansion, when trained with the CCCSSM sequence, is analysed in the plots from Figures 5.12a-b, when considering the threshold $hold = 150$ for D-TS-KFD and $hold = 50$ for D-TS-SE, in Eq. (5.6). From Figures 5.12a, we can observe that D-TS-KFD requires four experts for the lifelong learning of the CCCSSM sequence and is able to distinguish between different visual concepts from

several tasks while assigning the relevant experts to the incoming tasks. For instance, the first expert only learns the data distribution of CelebA and CACD, while the second expert learns those of CIFAR10 and Sub-ImageNet databases. In contrast, D-TS-SE, as it can be seen from Fig. 5.12-b, requires only three experts after lifelong learning.

We can also observe that D-TS-SE performs better than D-TS-KFD in the MSFIR setting. However, D-TS-KFD is better than D-TS-SE in the CCCSSM setting involving complex datasets. As shown in Tab. 5.1, the performance gap between D-TS-SE and D-TS-KFD is not that large. Such a gap would be caused by the dynamic learning process of the GAN-based expert. However, as we know, GAN is in general a model which is not always stable [23], and can not ensure the same quality for all generated images during each run. Therefore, the variation in the quality of the knowledge generated by the teacher module can affect the performance of the student module which is trained with such generated data.

For the lifelong learning of the sequence of datasets CCCSSM, containing complex images, whose results are shown in Tab. 5.5 and Tab. 5.6. The number of experts for both D-TS-KFD and D-TS-SE is 4 and 3, and examples of their generated images are shown in Fig. 5.10 and Fig. 5.11, respectively. Learning the database MNIST is the last task in the lifelong learning of the CCCSSM setting, with the results reported in Tab. 5.5 and Tab. 5.6. We have several observations that explain the differences in performance between D-TS-SE and D-TS-KFD, shown as follows:

- From the generated images from Fig. 5.10 and Fig. 5.11, "Expert 3" of D-TS-KFD learns a rather simple dataset such as SVHN while all experts of D-TS-SE capture the information of datasets containing complex images, such as those from CACD, CelebA, CIFAR10 and Sub-ImageNet databases. This means that D-TS-SE can provide more information after learning datasets with complex images when training the student module. What was learnt by each teacher expert can lead to an imbalanced data issue in the batch learning of the student module, where the number of generated samples corresponding to each previously learnt dataset is different because each expert can only learn one task or several different tasks. Therefore, the imbalanced data issue can influence the performance of the student model on each dataset.

- In addition, D-TS-KFD provides clearer images for a dataset containing rather simple images, such as SVHN, compared with D-TS-SE, as shown in the generated image results

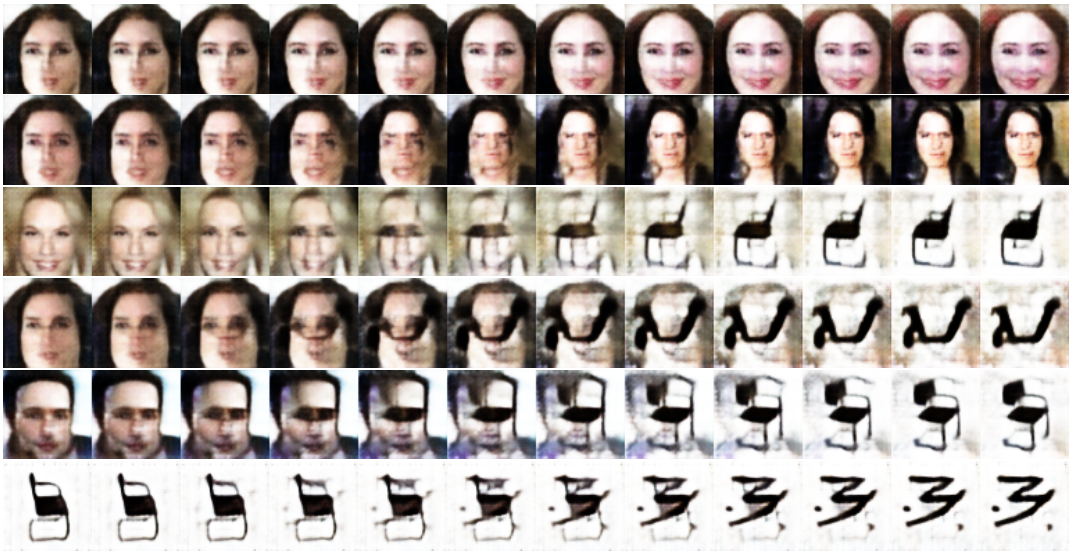


Figure 5.9: Interpolation results under the CelebA, CACD, 3D-Chair and Omniglot lifelong learning.

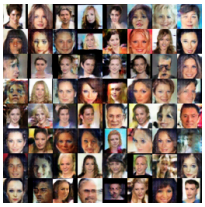
from Fig. 5.10 and Fig. 5.11. Since the simple dataset (SVHN) is sufficiently different from the datasets containing more complex images, learning clearer images of the simpler dataset would also affect the learning of the dataset with complex images because the student module has a fixed model capacity. This analysis can be empirically found in Tab. 5.5 and Tab. 5.6 where D-TS-SE mostly outperforms D-TS-KFD on the datasets with more complex images such as CACD, CIFAR10 and Sub-ImageNet, while D-TS-KFD outperforms D-TS-SE on SVHN.

- Moreover, it can be found from Fig. 5.11 where "Expert 1" of D-TS-SE tends to generate some fuzzed images for CelebA, which would lead to the degenerated performance for the student module of the D-TS-SE model trained on CelebA. This result can be found in the empirical results from Tab. 5.5 and Tab. 5.6, where D-TS-KFD outperforms D-TS-SE on CelebA.

- Eventually, the performance gap between D-TS-KFD and D-TS-SE in Tab. 5.5 and Tab. 5.6 is not that large. Such a gap is mainly caused by two factors: the dynamic learning process of the GAN-based experts and the imbalanced data sampling process for each dataset.

Table 5.7: Classification accuracy under the supervised lifelong learning of MNIST, Fashion, SVHN and InverseFashion (IFashion) databases.

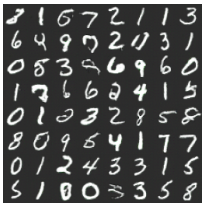
Dataset	D-TS-KFD	LGM [132]	LGAN [57]	TS-EWC [86]	EWC [86]	D-TS-SE	MeRGANs [176]	CURL [133]	BE-Stu [174]
MNIST	96.40	94.05	51.34	66.67	64.87	96.81	59.30	80.74	84.46
SVHN	65.21	47.24	48.16	55.63	54.12	68.68	55.31	68.46	62.78
Fashion	80.09	85.86	89.04	90.49	89.68	65.55	89.49	86.28	78.26
IFashion	86.68	89.08	92.15	92.30	92.76	88.48	92.17	91.48	81.94
Average	**82.09**	79.06	70.17	76.27	75.35	79.88	74.06	81.74	76.86



(a) Expert 1.

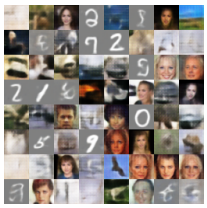
(b) Expert 2.

(c) Expert 3.



(d) Expert 4.

(e) Real testing.



(f) Reconstructions.

Figure 5.10: Generation and reconstruction of images when considering D-TS-KFD under CelebA, CACD, CIFAR10, Sub-ImageNet, SVHN and MNIST (CCCSSM) lifelong learning.

5.4.4 Supervised Learning

We evaluate the performance of the proposed approach in supervised classification tasks. The results when training for the lifelong learning of MNIST, SVHN, Fashion and IFashion, are provided in Table 5.7, where the number of training epochs for each task is 20. We observe that GRMs-based methods used for comparison provide good results on the most recently learned tasks and tend to achieve a lower performance on the earlier tasks. In contrast, the proposed approach is able to balance its performance across all learned tasks during supervised lifelong learning. CURL [133] uses a mixture model and is better in three tasks than D-TS-KFD. Similar to unsupervised learning, we also consider the BatchEnsemble (BE) [174] as a Teacher-Student model in the supervised lifelong learning setting. We first build an ensemble model as the teacher module, based on BE, where each expert contains a VAE and a classifier. We then train a classifier as the student module which accumulates the predictive knowledge from both the teacher module and the tasks learned during the lifelong learning. During

(a) Expert 1. (b) Expert 2. (c) Expert 3. (d) Real testing. (e) Reconstructions.

Figure 5.11: Image generation and reconstruction from D-TS-SE after the CCCSSM database sequence lifelong learning.

the lifelong supervised learning, each BE expert generates data samples and the associated classifier infers the class labels for the generated images. Then, the paired images and their corresponding classes are used to train the student module in order to overcome catastrophic forgetting. We name this supervised model as BE-Stu. However, from the results in Table 5.7, BE-Stu performs worse than D-TS in every task.

5.4.5 Model Complexity

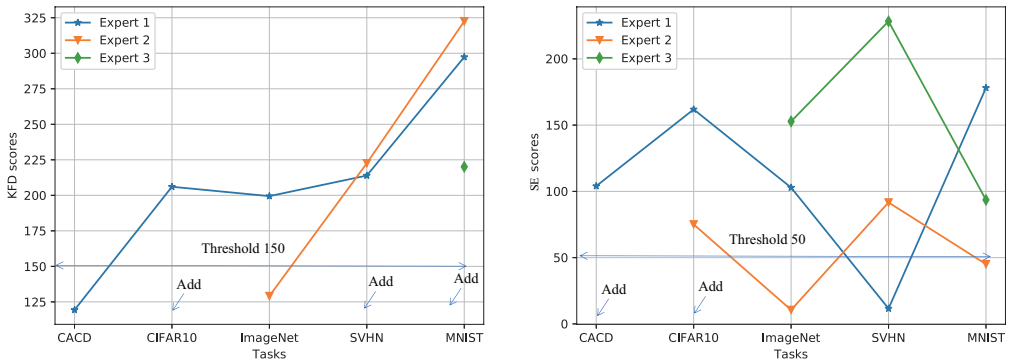
In the following, we evaluate the number of parameters used for various unsupervised lifelong learning methods when considering three sets of databases: MFSIR - which includes MNIST, Fashion, SVHN, IFashion and RMNIST; MSFICOM - including MNIST, SVHN, Fashion, IFashion, CIFAR10, Omniglot and MNIST; CCCSSM - including CelebA, CACD, CIFAR10, SubImageNet, SVHN and MNIST. The results for the number of parameters required are provided in Table 5.8. D-TS-Stu represents the number of parameters for the student module. From Table 5.8, it can be observed that the student module has significantly fewer parameters while achieving the state of the art results compared to other lifelong learning methods.

Table 5.8: The number of parameters required by various models for the unsupervised lifelong learning of MFSIR, MSFICOM and CCCSSM database sequences.

LLL sequence	LGM [132]	D-TS-KFD	D-TS-SE	D-TS-Stu	BE-Stu	LTS [180]
MFSIR	3.3×10^8	2.3×10^8	2.3×10^8	8.0×10^7	4.7×10^8	3.3×10^8
MSTICOM	3.3×10^8	3.1×10^8	2.3×10^8	8.0×10^7	5.2×10^8	3.3×10^8
CCCSSM	3.3×10^8	3.1×10^8	2.3×10^8	8.0×10^7	5.2×10^8	3.3×10^8

5.4.6 Ablation Study

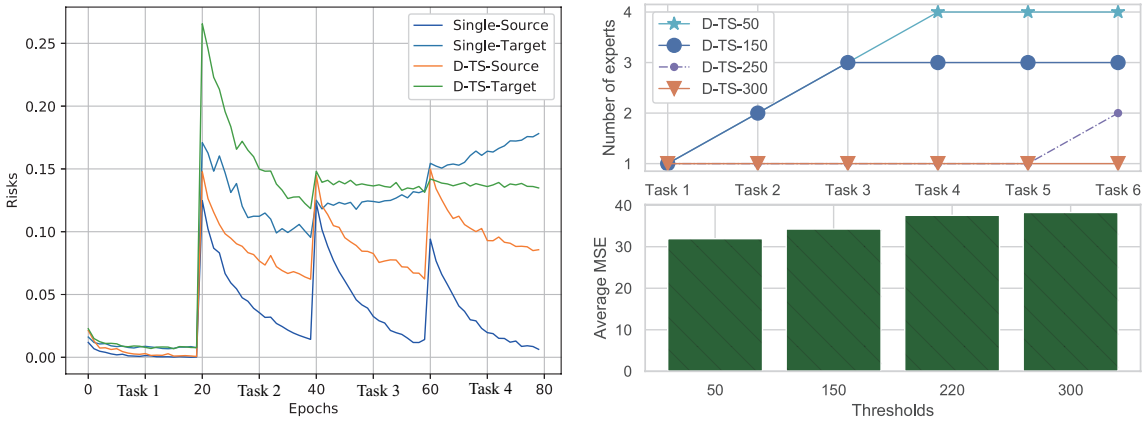
Firstly, we consider a baseline model which uses a single GAN for the teacher module, as in LTS [180], and does not use the selection and dynamical expansion mechanism as proposed



(a) KFD measure. (b) SE measure.

Figure 5.12: Results for the measures used for the Knowledge Discrepancy Score for the expansion of the teacher module under CelebA, CACD, CIFAR10, Sub-ImageNet, SVHN and MNIST lifelong learning.

for the D-TS model in this chapter. We evaluate the source and target risks (measured as the classification error), where the former is evaluated for the training data and the latter for the testing data. All risks are calculated as the average classification errors by using the student module, across the lifelong learning of MNIST, SVHN, Fashion, IFashion (MSFI). The results are provided in Fig. 5.13a where 'Single-Source' represents the source risk evaluated by the baseline and 'D-TS-Target' represents the target risk calculated on all testing samples by using D-TS-KFD. We can observe that both D-TS-KFD and the baseline achieve low source risks but the baseline has a higher target risk. These results show that multiple teacher experts can help relieve forgetting compared to using a single teacher expert.



(a) Risks for D-TS-KFD and baseline. (b) Varying the KDS threshold 'hold'.

Figure 5.13: The risk evaluation (classification error) and the performance of the student module when changing the KDS threshold in K-DS-hold where 'hold' is the threshold in Eq. (5.6).

We also consider the results when varying the threshold *hold* from Eq. (5.6), as described in Section 5.2.4, and the results for MSFI sequence are provided in Fig. 5.13b, where the plots

from the upper graph indicate D-TS-KFD (*hold*), with $hold \in \{50, 150, 250, 300\}$. We can observe that a lower threshold *hold* leads to more components, while for $hold = 300$ the system would use a single component. This result is because a large value for *hold* can prevent the expansion of the D-TS. From the bar-plot at the bottom of Fig. 5.13b we can observe that the reconstruction MSE error would decrease for $hold = 50$ resulting in more components than when using other KDS thresholds. These results show that changing the value of the KDS threshold (*hold*) can trade-off between the model size and performance.

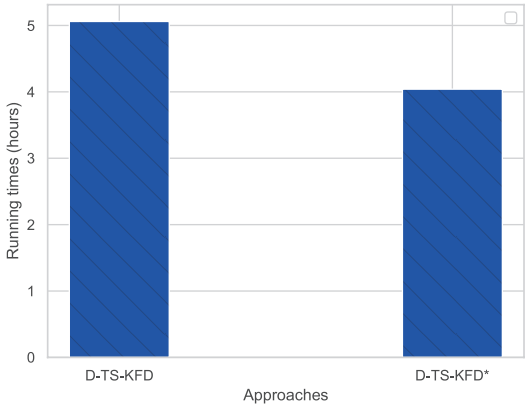
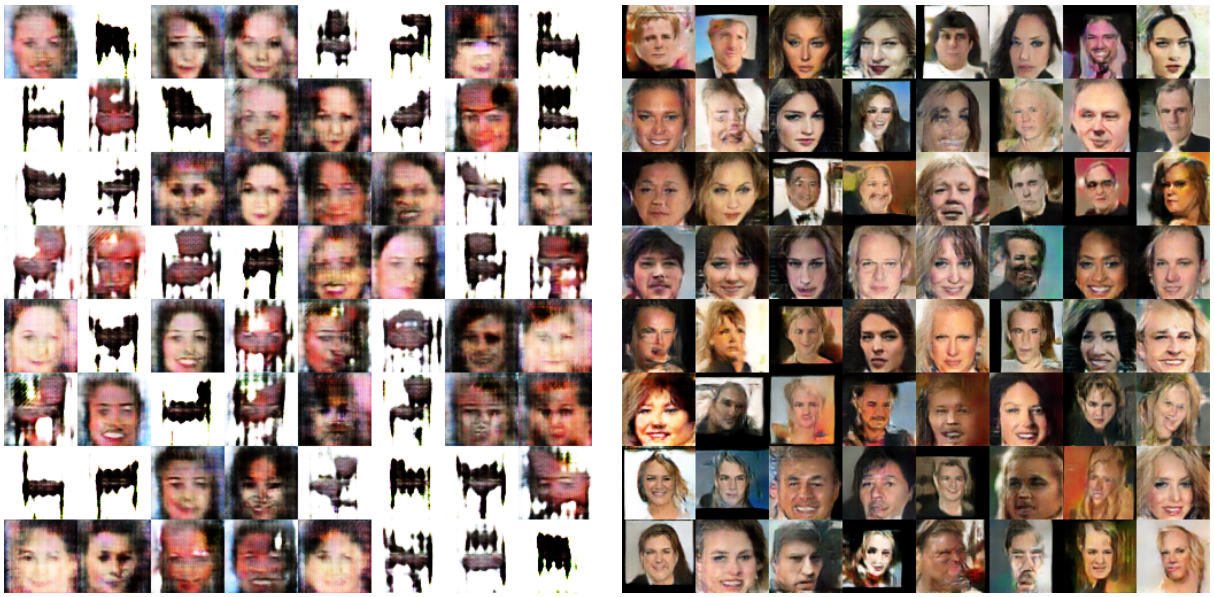


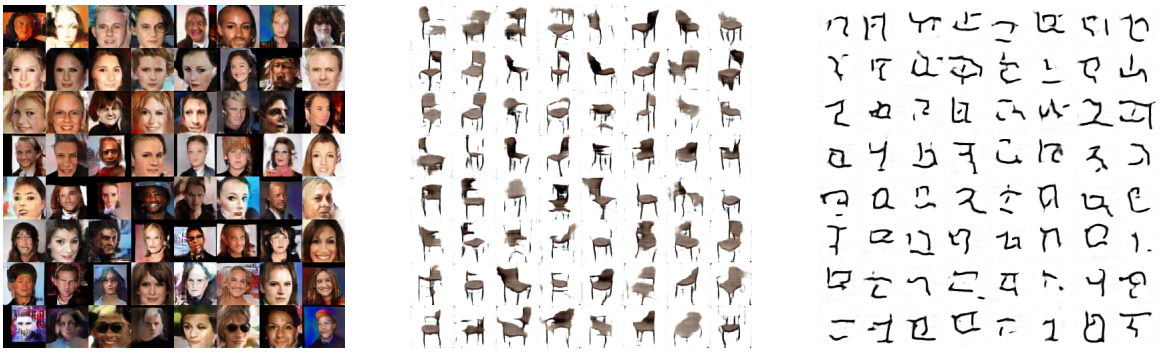
Figure 5.14: The running time for D-TS-KFD when considering 20 epochs for both training and updating a component, while for D-TS-KFD* we consider only 5 epochs when updating a component, according to Eq. (5.6), under CelebA, CACD, CIFAR10, Sub-ImageNet, SVHN and MNIST lifelong learning.



(a) CelebA to 3D-Chair. (b) CelebA to CACD.

Figure 5.15: Images generated by WGAN when considering GRMs.

Robustness to missing data during the training: We consider the following learning setting,



(a) Expert 1. (b) Expert 2. (c) Expert 3.

Figure 5.16: Images generated by the teacher module from the proposed lifelong teacher-student framework.

Table 5.9: The performance for all testing data samples when considering that training data are missing for certain databases (marked with '*'). The total number of training samples for CACD* and Sub-ImageNet* is 10,000, respectively.

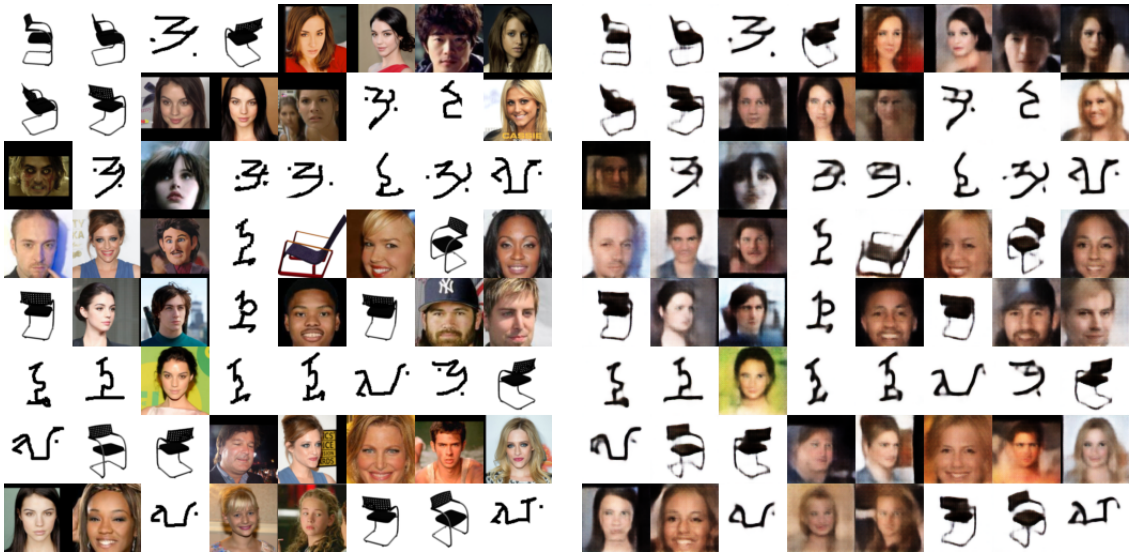
	MSE		**SSMI**	
Datasets	D-TS-KFD	LTS	D-TS-KFD	LTS
CelebA	185.54	312.24	0.43	0.26
CACD*	124.92	400.32	0.63	0.28
CIFAR10	164.68	330.85	0.41	0.21
Sub-ImageNet*	176.34	337.68	0.41	0.22
SVHN	31.37	40.47	0.63	0.54
MNIST	30.11	25.82	0.87	0.89
Average	118.83	241.23	0.56	0.40

where the model is trained under the CelebA, CACD*, CIFAR10, Sub-ImageNet*, SVHN and MNIST lifelong learning. We create CACD* and Sub-ImageNet* by considering only 10,000 samples, representing a small subset training set for CACD and Sub-ImageNet (the total number of samples for CACD and Sub-ImageNet is 60,000), respectively. We would like to evaluate whether the proposed framework can handle missing data well during lifelong learning. The average result is provided in Table 5.9. From these results we can observe that although we have much fewer training data from CACD* and Sub-ImageNet*, the proposed D-TS-KFD framework still achieves very good results. LTS [180], used for comparison, tends to forget more information from the tasks learned earlier on during lifelong learning.

Accelerating the future task learning: The proposed methodology is efficient in reusing the learned knowledge when updating an existing expert, based on the similarity of its accumulated knowledge compared to with the information from a new database. This results in the accelerated learning of those tasks which contain similar information to what was already learned by the D-TS model. In the following, we consider fewer training epochs when the model reuses a

Table 5.10: The results when considering just five training epochs for updating an existing component, when the condition to expand the model is not fulfilled in Eq. (5.6).

	MSE		SSMI	
Datasets	D-TS-KFD*	LTS	D-TS-KFD*	LTS
CelebA	117.61	215.43	0.60	0.40
CACD	148.95	246.43	0.59	0.44
CIFAR10	177.95	215.42	0.40	0.33
Sub-ImageNet	190.47	230.55	0.39	0.33
SVHN	33.03	34.90	0.63	0.60
MNIST	32.00	25.66	0.86	0.89
Average	116.6	161.49	0.58	0.50



(a) Real images. (b) Images reconstructed by the student module.

Figure 5.17: Image reconstructions by the student module after CelebA, CACD, 3D-Chair and Omniglot lifelong learning.

selected expert of the teacher module for learning the next task, according to Eq. (5.6). The results are provided in Table 5.10, where D-TS-KFD* denotes using only 5 training epochs for updating an existing component, while 20 epochs are used for training a new component. We observe that D-TS-KFD* still achieves good results while it also accelerates the training, as shown in the bar-plots from Figure 5.14, where D-TS-KFD* would reduce the time required for the full lifelong training by D-TS-KFD, where the latter uses 20 training epochs for both training a new component as well as when updating an existing one. Both D-TS-KFD* and D-TS-KFD use four experts for their teacher modules.

The mode collapse in GRM. We investigate how the mode collapse occurs during lifelong learning. We consider training a Wasserstein GAN (WGAN) [11] on two databases, CelebA and

3D-Chair, which do not have any common characteristics, with the first representing human faces while the other contains images of chairs. In order to overcome the forgetfulness of WGAN, we use the GRM [176] during the training. After the lifelong learning, we generate images using the WGAN, which are shown in Fig. 5.15a. From these images we can observe that WGAN with the GRM cannot generate clear images for the two given domains, CelebA and 3D-Chair. The reason for this is that CelebA contains images which have completely different characteristics from those of 3D-Chair database. In the following, we train a single WGAN with GRM under the CelebA and CACD lifelong learning and the images generated by the WGAN are shown in Fig. 5.15b. These generated images are of rather good quality. This shows that WGAN is able to learn multiple similar databases. However, existing GRMs-based methods cannot be applied to long sequences of tasks, where the datasets are entirely different from each other. The drawback outlined by this example motivates us to develop a novel dynamic memory system for the teacher module. The proposed Knowledge Discrepancy Score (KDS) can detect and identify the novelty of the incoming tasks and guides the teacher module to expand its capacity in order to learn databases containing images with entirely different characteristics. In the following, we train the proposed Dynamic Teacher-Student Learning model D-TS-KFD under the CelebA, CACD, 3D-chair and Omniglot lifelong learning. After the lifelong learning process is finished, our teacher module adds two new experts and the images generated by the teacher module are shown in Fig. 5.16, where each expert models well images from similar databases such as CelebA and CACD, with both containing face images. We also show the reconstructions made by the student module in Fig. 5.17. We can observe that the student module is also able to provide high-quality reconstructions across domains. These results indicate that the proposed Lifelong D-TS provides better results than Generative Reply Mechanisms (GRM) methods.

5.5 Conclusion and Limitations

A novel Dynamic Teacher-Student Network (D-TS) learning framework, capable of continually learning data representations without forgetting, is proposed in this chapter. The model is made up of a teacher module, which is allowed to expand its architecture with new components, and a student module. The Knowledge Discrepancy Score criterion is used for comparing the distance between the incoming data and the information already acquired by the teacher module. For implementing KDS we consider two measures: the Knowledge Frécher Distance (KFD) and the

Student's Evaluation (SE). A new component (expert) is added to the teacher module when KDS is above a certain threshold, when learning a new database. Otherwise, the most efficient and flexible component is selected by the KDS from the mixture in order to be updated with the information from the database. The selection mechanism contributes to reusing the learned knowledge for accelerating the future task learning. In the experimental results we show that D-TS can train a compressed student module which outperforms other methods in various multi-task applications while also requiring fewer parameters to train.

The one limitation of the proposed D-TS is that it still requires accessing the task information during the training, which can not be applied to a more realistic continual learning setting (TFCL). In a real-time application, the system should learn and acquire information from the streaming data without forgetting. The other limitation is that the teacher model size will grow forever when learning a growing number of tasks, which is not suitable for resource-constraint devices. In the next chapter, we introduce a new lifelong teacher-student framework to address these limitations discussed above.

Chapter 6

Teacher-Student Framework for TFCL

6.1 Introduction

In chapters 3, 4 and 5, we mainly focus on a general continual learning setting in which the task information and boundary are provided during the training. In this chapter, we study the lifelong generative modelling [132] into a more sophisticated learning scenario, called Task-Free Continual Learning (TFCL) [7], which does not require the task identity during the training and testing. Training a model in TFCL is more challenging than general continual learning because the data distribution can change at anytime during lifelong learning. One popular approach to reduce forgetting in TFCL is the memory-based approach, which manages a fixed-capacity memory buffer [30, 75] to store necessary data samples to be used for future training. During the subsequent learning, the memory buffer can replay past data samples for updating the model to relieve forgetting. The sample selection plays an important role in the performance of the memory-based approaches [7]. However, the memory-based approach could potentially lead to data privacy and security concerns [111]. Another approach to relieve forgetting in TFCL is to train a powerful generator that produces high-quality generative replay samples, which are statistically consistent with the training sets [181]. Compared to the memory-based approaches, which usually manage a capacity-fixed memory buffer, the generative replay network can produce an unlimited number of samples, which can be used for training. However, these approaches rely on a single memory and generation system, which is not scalable for learning infinite data streams due to their limited memory capacity while also requiring frequent retraining [181].

Recently, the Dynamic Expansion Model (DEM) [95, 133] has shown promising results for

TFCL and can potentially be applied to infinite data streams. The key idea of the DEM is to dynamically build a new expert/component when detecting data distribution changes while freezing all previously trained components to preserve past knowledge. Compared to the static model, the DEM is scalable to a dynamically changing learning environment and can achieve better generalization performance. The expansion criterion in the DEM plays a vital role in balancing the complexity of the model and the generalization performance. Existing DEM-based methods implement the expansion criterion by considering the sample log-likelihood evaluation [133] and Neural Dirichlet processes [95], but they cannot guarantee an optimal trade-off between the network architecture and performance. They also have a multi-head structure that requires performing the component selection process at the testing phase and cannot model correlations between different data domains in a single latent space leading to requiring additional computational resources. Meanwhile, its application is rather limited to certain tasks such as cross-domain image interpolation [123, 181].

In this chapter, we address the limitations of existing DEM approaches by proposing a scalable knowledge distillation framework to learn a data stream with a growing number of statistical representations in a TFCL scenario while ensuring a lightweight network architecture at the testing phase. To achieve this goal, the teacher-student framework [180, 184, 186] can be adopted since it can accumulate knowledge over time and compress the learnt information into a compact student module. The other advantage of the teacher-student framework over the DEM method is that we can flexibly design the student architecture aiming to support many applications, including unsupervised, supervised learning and disentangled representation learning, which has been investigated in Chapter 5. However, the existing lifelong teacher-student frameworks [180, 184, 186] require accessing the task information, which can not be used in a more realistic learning scenario, such as TFCL. To address this issue, this study proposes to train a dynamic expansion mixture model as the teacher module and manage a short-term memory buffer to store more recent samples from a data stream. The introduction of the short-term memory buffer has two main goals: (1) It can provide enough samples for training one of the teacher experts, enabling it to learn knowledge from a data stream without accessing task information; (2) It aims to store more recently seen data samples, representing short-term information, which can be combined with the knowledge preserved by the teacher module, representing the long-term information, to provide a better knowledge distillation process for student learning.

The other obstacle to applying the teacher-student framework for TFCL is the dynamic expansion criterion. The D-TS, described in Chapter 5, employs the Knowledge Discrepancy Score (KDS) in Eq. (5.3) to control the teacher's expansion process, which relies on task identity and boundaries. To tackle this issue, we propose the Knowledge Incremental Assimilation Mechanism (KIAM) that enables the teacher module to expand its network architecture in the absence of task information. The primary idea of the KIAM is to evaluate the distance between the current memory buffer (STM) and the already accumulated knowledge in the teacher module as a signal to increase the teacher's memorization capacity. A high measure in KIAM indicates that the current memory buffer stores novel enough information with respect to the already-learnt knowledge. As a result, we can freeze the current teacher expert to preserve the information from the current memory buffer while building a new expert for subsequent learning. Such a mechanism enables the teacher module to gradually accumulate new knowledge without forgetting while ensuring knowledge diversity among all frozen components in the teacher module. In addition, we further reduce the teacher model size by proposing a new expert pruning approach that selectively removes redundant experts from the teacher module. This pruning approach can induce a compact network architecture for the teacher module and reduce the computational costs for knowledge distillation. Furthermore, in order to embed multiple data domains into a single latent space, we propose to employ a data-free knowledge distillation approach, namely the Continual Generative Knowledge Distillation (CGKD), to transfer generative knowledge from the teacher to the student module without accessing any task information in an online learning manner.

In summary, our contributions in this chapter are as follows:

- We extend the teacher-student framework for lifelong generative modelling in a challenging learning setting such as TFCL;
- We propose the Knowledge Incremental Assimilation Mechanism (KIAM) for the teacher module, which enriches the teacher's knowledge incrementally while ensuring a minimal architecture;
- A new knowledge distillation learning approach is introduced to transfer generative knowledge from a teacher to a student module in an online manner;
- We propose a new expert pruning approach for the teacher-student framework, which

reduces the size of the teacher module while enriching its knowledge diversity.

The rest of the chapter provides the setting for the proposed methodology in Section 6.2. Afterwards, as part of the proposed methodology, in Section 6.2.2 is presented the knowledge incremental assimilation mechanism, in Section 6.2.3, the continual generative knowledge distillation mechanism, and the expert pruning approach in Section 6.2.4. The implementation is explained in Section 6.2.5, and the experimental results in Section 6.3. Eventually, Section 6.4 draws the conclusions and discusses the limitations of this chapter.

6.2 Method

The lifelong teacher-student frameworks, introduced in Chapters 4 and 5, have shown promising performance in lifelong generative modelling. However, these methods can not be used in the more challenging Task Free Continual Learning (TFCL) [7] since they require to access the task information to implement the network expansion mechanisms. In this chapter, we address this issue by developing a new dynamic expansion mechanism for the teacher module, which does not rely on the task information. The key idea of the proposed approach is inspired by the biological research [47] showing that short- and long-term information can be learned using distinct brain regions at different stages. To implement this goal, we formulate a dynamic expansion model (Teacher module) as a long-term memory system and introduce a short-term memory buffer to preserve more recently seen data samples. Specifically, the short-term memory buffer has a fixed size and we would like to increase the long-term memory's capacity in order to accumulate knowledge over time. This memory expansion process is implemented by the proposed FID-based expansion mechanism that evaluates the distance between the knowledge learnt by the teacher module and the samples from the short-term memory buffer and uses this measure which is then used to expand the network architecture of the teacher module. Such an approach can encourage the teacher module to gradually increase the model's capacity together with its knowledge without accessing task information. We provide the detailed dynamic expansion mechanism in Section 6.2.2.

In addition, when considering to deploy the learning model to a resource-constrained device, it is necessary to keep a fixed-size teacher module during the training. We implement this goal by proposing an expert pruning approach, described in Section 6.2.4, which automatically removes redundant teacher experts during the training. The key idea of the proposed expert

pruning approach is to evaluate the relationships among teacher experts and then remove the experts that share similar knowledge.

6.2.1 Problem Definition

Let $D^i_{TU} = \{\mathbf{x}_j\}_{j=1}^{N_i^T}$ and $D^i_{SU} = \{\mathbf{x}_j\}_{j=1}^{N_i^S}$ be the unlabelled test and training sets, for the i-th data domain/dataset, where N_i^T and N_i^S represent the number of samples for the test and training sets, respectively. In this work, we focus on sequential learning of different data domains without accessing the task/domain information. Therefore, we create a joint dataset as a data stream $\mathcal{S}$ in task-free continual learning (TFCL [7]) by including all incoming training sets in a sequence manner, expressed as:

$$\mathcal{S} = \{D^1_{SU} \cup D^2_{SU} \cup \cdots \cup D^N_{SU}\}, \tag{6.1}$$

where N is the total number of datasets considered in the continual learning sequence. In addition to the domain-incremental setting (Eq. (6.1)), we also consider a more challenging setting that involves the class and domain shifts, expressed as:

$$\mathcal{S} = \{D^{1,1}_{SU} \cup D^{1,2}_{SU} \cup \cdots \cup D^{1,5}_{SU} \cup \cdots \cup D^{N,1}_{SU} \cup D^{N,2}_{SU} \cup \cdots \cup D^{N,5}_{SU}\}. \tag{6.2}$$

Following from the class-incremental learning setting [30], we divide each single dataset D^i_{SU} into five parts $\{D^{i,1}_{SU}, \cdots, D^{i,5}_{SU}\}$ according to category information [7] and each data set $D^{i,1}_{SU}$ contains samples from two classes [7], where $i = 1, \cdots, N$. This setting is more challenging than Eq. (6.1) since the model requires dealing with the change in the class and domain over time. In task-free continual learning, the model is trained in a batch-to-batch learning manner [7]. To clearly describe this learning paradigm, the data stream $\mathcal{S}$ can be divided in N' disjoint data batches $\{\mathbf{X}^1_{batch}, \cdots, \mathbf{X}^{N'}_{batch}\}$ and each data batch $\mathbf{X}^i_{batch}$ contains m samples (In the experiment, we set $m = 64$ as the data batch size). Learning the data stream $\mathcal{S}$ requires a total of N' training steps/times. At a specific training step/time (the t-th training time), we only access a small data batch $\mathbf{X}^t_{batch}$, obtained from $\mathcal{S}$ while all previous data batches $\{\mathbf{X}^1_{batch}, \cdots, \mathbf{X}^{t-1}_{batch}\}$ are unavailable. After a training model is done with the N'-th training step/time, we evaluate the average performance of the model on all test sets $\{D^1_{TU}, \cdots, D^N_{TU}\}$. Compared with the problem definition described in Chapters 3, 4 and 5, which can access the task information during the

training, TFCL represents a more challenging learning scenario where task identities are not available while the model can only access a small batch of samples at a given time.

6.2.2 Knowledge Incremental Assimilation Mechanism (KIAM)

Humans can incrementally learn and memorize novel concepts throughout their entire lifespan [15]. Specifically, biological research [47] shows that short- and long-term information/skills can be learned using distinct brain regions at different stages. Inspired by this, we propose to manage two memory systems storing both short- and long-term information to address catastrophic forgetting in TFCL. Firstly, we introduce to employ a short-term memory buffer [95] to store more recently seen data samples from the data stream $\mathcal{S}$ during the training, aiming to preserve the up-to-date information. Then we introduce a long-term memory system to preserve permanent information during training. One reasonable approach for implementing a long-term memory system is employing a memory buffer with a fixed capacity to store many crucial real training samples during the training [7]. However, due to the memory capacity limitation, such an approach can not store enough information for the model training when the data stream $\mathcal{S}$ involves a large number of underlying data distributions. Inspired by the dynamic teacher-student framework described in Chapter 5, which enables the teacher module to capture multiple data distributions without forgetting, we propose to implement a long-term memory system using the dynamic expansion model (Teacher) that has several advantages compared to the use of a memory buffer: (1) The dynamic expansion model can generate an unlimited number of samples while the memory buffer can only store and replay a limited number of samples; (2) The dynamic expansion model is scalable to the data stream $\mathcal{S}$ that consists of multiple data domains; (3) Storing long-term information using a memory buffer requires carefully designing an appropriate sample selection approach to selectively filter out data samples deemed not important. However, such an approach would lose some previous important data samples, resulting in performance degeneration. In contrast, the dynamic expansion model can permanently preserve the past information by freezing all previously trained components and learn novel knowledge through an appropriate dynamic expansion mechanism.

The other challenge for the teacher module is its dynamic expansion process, which is hard to control in the TFCL since we can not access the task information during training.

In this chapter, we address this issue by introducing the Knowledge Incremental Assimilation Mechanism (KIAM) for the teacher module, which gradually increases the knowledge capacity of the teacher module without accessing the task information. Let $\mathcal{M}_t$ represent a short-term memory (STM) that has a fixed capacity, updated at the t-th training step and $\mathbf{A} = \{\mathcal{A}_1, \cdots, \mathcal{A}_K\}$ be a teacher module assumed to have already trained K experts up to the training step (the t-th training step), where each $\mathcal{A}_j$ is implemented by either a GAN [52] or a VAE to learn a probability distribution P_{θ_j} of data samples produced by the generator module with trainable parameters θ_j. Detecting when and what new concepts are provided is a real challenge under the TFCL framework, where we do not know the task labels. To address this problem, KIAM evaluates distances between the current memory (STM) and the already learnt information, aiming to detect when the data distribution shift occurs:

$$\min_{j=1,\cdots,K-1} \left\{ f_p^{\mathrm{CGKD}}(\tilde{D}'_j, \mathcal{M}_t) \right\} \geq \lambda^{\mathrm{CGKD}} , \tag{6.3}$$

where $\tilde{D}'_j$ is the dataset consisting of 200 samples drawn from P_{θ_j}, which represents a statistical data representation characterized by parameters θ_j. In practice, we calculate $f_p^{\mathrm{CGKD}}(\tilde{D}'_j, \mathcal{M}_t)$ by employing 200 samples randomly obtained from $\mathcal{M}_t$ in Eq. (6.3) in order to reduce the computational costs. $\lambda^{\mathrm{CGKD}} \in [0, 200]$ is a threshold for controlling the number of experts for the teacher module. $f_p^{\mathrm{CGKD}}(\cdot, \cdot)$ is a distance measure function that evaluates the similarity between two datasets. We omit the current expert $\mathcal{A}_K$ in Eq. (6.3) since $\mathcal{A}_K$ is knowledgeable about $\mathcal{M}_t$. One potential approach for implementing $f_p^{\mathrm{CGKD}}(\cdot, \cdot)$ is training a discriminator using adversarial learning [52] to estimate the discrepancy between the current memory buffer and the already-learnt knowledge. However, such an approach can lead to considerable computational costs, especially when the total number of training steps/times is large. Moreover, adversarial learning is prone to unstable training behaviour and mode collapse problems [156], leading to unstable signals for the model expansion using Eq. (6.3). In this chapter, we desire to implement the distance measure $f_p^{\mathrm{CGKD}}(\cdot, \cdot)$ using a simple computational form without extra training costs. The Fréchet Inception Distance (FID) score is a popular approach to evaluate the performance of GAN models [63] and has been used as one of the KDS criteria in Chapter 5. We consider the FID [63] as the $f_p^{\mathrm{CGKD}}(\cdot, \cdot)$ measure for evaluating the knowledge similarity between two datasets, because FID can be effectively evaluated on data samples without the need to know the density form for each dataset. A small FID score indicates that a GAN model

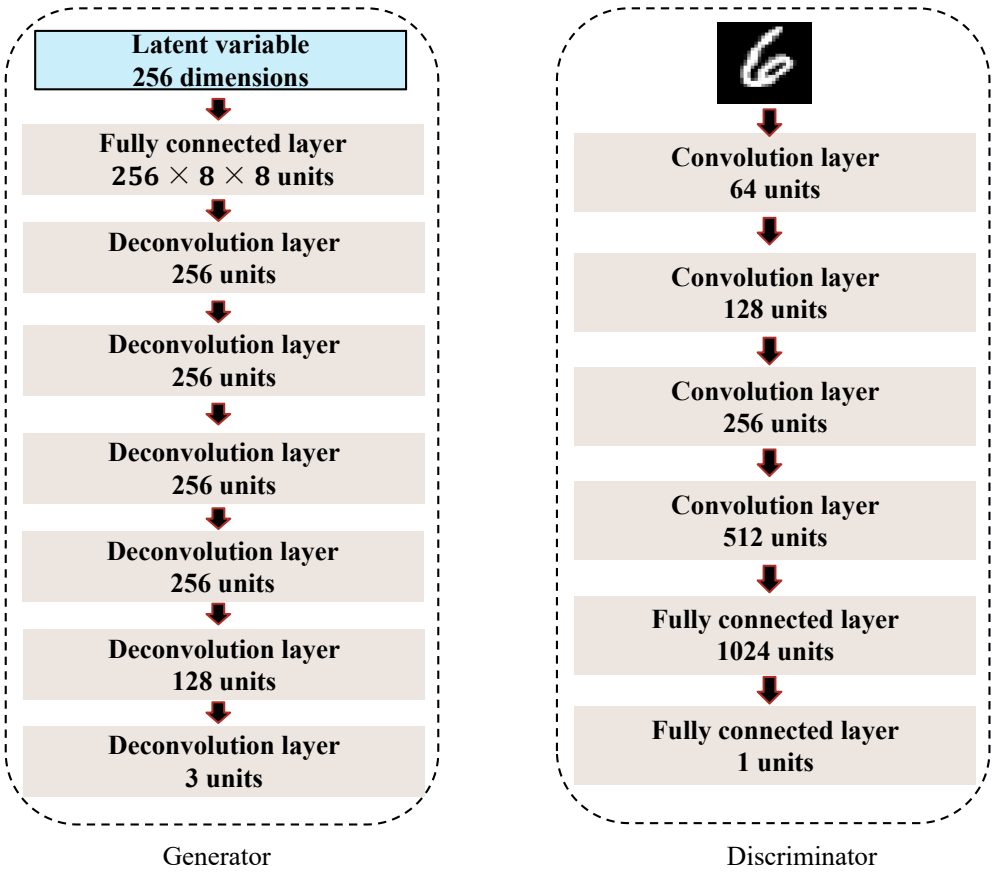


Figure 6.1: The network architecture of the generator and a discriminator.

can generate more realistic images with respect to the samples from the real training dataset [63]. As a consequence, we can employ the FID to evaluate the knowledge similarity between each previously trained expert and the current memory buffer (STM). Such measures can be used as signals for the model expansion. Compared to the existing dynamic expansion models, which employ the sample log-likelihood evaluation [133] and Neural Dirichlet processes [95] to implement the expansion criterion, Eq. (6.3) relies on a distance measure between two datasets which can better distinguish the new data samples from the already-learnt knowledge and thus can provide good expansion signals for the model.

If the dynamic expansion criterion (Eq. (6.3)) is satisfied at the t-th training step, we freeze $\mathcal{A}_K$ (the current expert) that has already preserved the knowledge of the current memory $\mathcal{M}_t$, while adding a new expert $\mathcal{A}_{K+1}$ for the next training step (the $(t+1)$-th training step). We also empty the current memory $\mathcal{M}_t$ so that the newly created expert $\mathcal{A}_{K+1}$ is able to learn statistically different data in the following training steps. Compared to the expansion process described in Chapter 5, Eq. (6.3) does not need accessing task information and thus is suitable for task-free continual learning. In addition, the existing dynamic expansion method (Continual Neural Dirichlet Process Mixture (CN-DPM)) [95] also employs the short-term

memory buffer to store recently received data samples for training, but the proposed KIAM has several differences from CN-DPM: (1) The CN-DPM does not train a student module, while the proposed KIAM implements the dynamic expansion model as a teacher module in a unified framework to support student learning; (2) The CN-DPM employs the Neural Dirichlet process for the model expansion, while the proposed KIAM uses the FID-based criterion; (3) The CN-DPM implements each expert using a VAE model, resulting in blurred image generation results. In contrast, the proposed KIAM can employ a more powerful generative model (GAN) to implement each expert, providing better generative knowledge for training the student module.

6.2.3 Continual Generative Knowledge Distillation

Most existing approaches to knowledge distillation usually transfer the category from a complex teacher module to a lightweight student module for the classification task [126]. These methods are not able to distil the knowledge for generative modelling because previously learnt samples are not available during the continual learning process. In this chapter, we introduce a data-free KD approach for lifelong generative modelling in TFCL. Let us consider a latent variable-based generative model $p_{\theta_{stu}}(\mathbf{x}, \mathbf{z}) = p_{\theta_{stu}}(\mathbf{x} \,|\, \mathbf{z})p(\mathbf{z})$ which is represented by the student module in our teacher-student framework, where $\mathbf{x}$ and $\mathbf{z}$ represent the observed and latent variables, respectively. $p_{\theta_{stu}}(\mathbf{x} \,|\, \mathbf{z})$ is the decoder and $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the prior distributions for the latent space. An approach for distilling knowledge when having a teacher module characterised by a probability distribution P_{θ_j} of data samples produced by the generator, would require to minimise the KL divergence between P_{θ_j} and $\mathrm{P}_{\theta_{stu}}$, where $\mathrm{P}_{\theta_{stu}}$ is the probability distribution formed by samples generated using the student model. However, this optimization is computationally infeasible due to the lack of an explicit density function for P_{θ_j}. Instead, we propose to implement KD by minimizing the negative sample log-likelihood with respect to the samples drawn from each P_{θ_j}:

$$\mathcal{L}_{\mathrm{KD}}^{\mathrm{CGKD}}(\theta_1, \theta_2, \cdots, \theta_K, \theta_{stu}, \varsigma_{stu}) = \sum\nolimits_{j=1}^{K-1} \Big\{ -\mathbb{E}_{\mathbf{x}\sim \mathrm{P}_{\theta_j}}[\log p_{\theta_{stu}}(\mathbf{x})] \Big\}, \tag{6.4}$$

where P_{θ_j} represents the probability distribution of the data generated by the j-th teacher expert. $\mathbf{A} = \{\mathcal{A}_\mathbf{1}, \cdots, \mathcal{A}_\mathbf{K}\}$ is the teacher module and $\{\theta_{stu}, \varsigma_{stu}\}$ is the parameter set of the student module. The direct calculation of Eq. (6.4) is intractable [85] and we introduce the use of a variational distribution $q_{\varsigma_{stu}}(\mathbf{z} \,|\, \mathbf{x})$, parameterised by ς_{stu}, to approximate the true posterior

$p_{\theta_{stu}}(\mathbf{z}\,|\,\mathbf{x})$, and therefore the marginal log-likelihood $\log p_{\theta_{stu}}(\mathbf{x})$ can be approximated by a lower bound [85]. Then, Eq (6.4) can be calculated as:

$$\begin{aligned}\mathcal{L}_{\mathrm{KD}}^{\mathrm{CGKD}}(\theta_1,\theta_2,\cdots,\theta_K,\theta_{stu},\varsigma_{stu}) = \sum\nolimits_{j=1}^{K-1}\big\{\mathbb{E}_{\mathbf{x}\sim \mathrm{P}_{\theta_j}}[-\mathbb{E}_{q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x})}\left[\log p_{\theta_{stu}}\left(\mathbf{x}\,|\,\mathbf{z}\right)\right] \\ + D_{KL}\left[q_{\varsigma_{stu}}\left(\mathbf{z}\,|\,\mathbf{x}\right)\,||\,p\left(\mathbf{z}\right)\right]]\big\}\,.\end{aligned} \tag{6.5}$$

Together with the KD loss Eq. (6.5), and using the current memory buffer $\mathcal{M}_t$ for training the model at the t-th training step, we design a unified objective function for training the student module as:

$$\begin{aligned}\mathcal{L}_{\mathrm{Stu}}^{\mathrm{CGKD}}(\mathcal{M}_t,\theta_1,\theta_2,\cdots,\theta_K,\theta_{stu},\varsigma_{stu}) = \mathbb{E}_{\mathbf{x}\sim \mathrm{P}_{\mathcal{M}_t}}[-\mathbb{E}_{q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x})}\left[\log p_{\theta_{stu}}\left(\mathbf{x}\,|\,\mathbf{z}\right)\right] \\ + D_{KL}\left[q_{\varsigma_{stu}}\left(\mathbf{z}\,|\,\mathbf{x}\right)\,||\,p\left(\mathbf{z}\right)\right]] \\ + \mathcal{L}_{\mathrm{KD}}^{\mathrm{CGKD}}(\theta_1,\theta_2,\cdots,\theta_K,\theta_{stu},\varsigma_{stu})\,,\end{aligned} \tag{6.6}$$

where $\mathcal{L}_{\mathrm{KD}}^{\mathrm{CGKD}}(\mathcal{M}_t,\theta_1,\theta_2,\cdots,\theta_K,\theta_{stu},\varsigma_{stu})$ is provided in Eq. (6.5) and $\mathrm{P}_{\mathcal{M}_t}$ is the data distribution of the samples from the memory buffer $\mathcal{M}_t$ at the t-th training step. The first term encourages the student module to learn samples obtained from the memory buffer $\mathcal{M}_t$ that preserves the short-term information, while the second term $\mathcal{L}_{\mathrm{KD}}^{\mathrm{CGKD}}(\mathcal{M}_t,\theta_1,\theta_2,\cdots,\theta_K,\theta_{stu},\varsigma_{stu})$ transfers the teacher's knowledge, representing the long-term information, to the student module. We train the student module using Eq. (6.6) in the mini-batch learning manner [92], which has been widely used in the deep learning field [61, 92]. The network architecture of the decoder (generator) and encoder of the student module is provided in Fig. 6.1 and Fig. 6.4, respectively. Compared to the D-TS framework described in Section 5.2.6 and the proposed KD approach (Eq. (6.6)), there are several differences: (1) Eq. (6.6) is used to transfer the knowledge at each training step under TFCL, while the D-TS can access the whole training dataset from the current task learning; (2) Eq. (6.6) employs a memory buffer $\mathcal{M}_t$ to store more recent samples, which enables the student module to learn both short- and long-term information during the training. In contrast, the D-TS framework does not have a memory buffer and its learning relies on the task identity. (3) The teacher module in the D-TS framework always increases its model size/capacity when learning a growing number of tasks. In contrast, the proposed framework introduces an expert pruning approach that can maintain a fixed teacher's model size without sacrificing much performance, which is introduced in the next section.

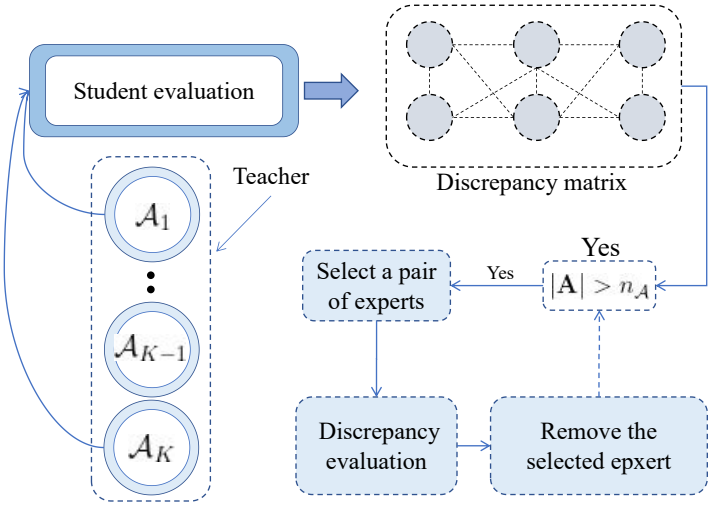


Figure 6.2: The removal of unnecessary experts, where the given criterion controls the number of teacher experts.

6.2.4 Expert Pruning Approach

The teacher's ensembles cannot grow forever, and in order to keep the number of parameters in check and the model architecture compact while preserving the statistical representation diversity of the teacher, we consider a component expert pruning approach. The primary idea of the proposed expert pruning approach is that we can find and remove certain teacher experts that share overlapping knowledge with other components. However, directly searching multiple teacher components which represent similar information is computationally intractable. Instead, we formulate such a searching process as a simple problem that first finds a pair of teacher experts and then we remove one of them. Then, we can continually perform this search process to remove more unnecessary teacher experts until a certain criterion is met.

Suppose that the teacher module has already trained K experts at the t-th training step, $\mathbf{A} = \{\mathcal{A}_1, \cdots, \mathcal{A}_K\}$. Let $\mathbf{Q} \in \mathbf{R}^{K \times K}$ be a knowledge discrepancy matrix, where each $\mathbf{Q}[a, g]$ represents the discrepancy score between experts $\mathcal{A}_a$ and $\mathcal{A}_g$, where a and g are teacher expert indexes. Given that the student module is already knowledgeable about the informational content of all experts, it can be used for identifying whether two experts represent statistically overlapping knowledge. We evaluate $\mathbf{Q}[a, g]$ by calculating the square loss $\|\cdot\|^2$ on the latent variables, inferred by the inference model $q_{\zeta_{stu}}(\mathbf{z} \,|\, \mathbf{x})$ of the student module:

$$\mathcal{L}_{ks}(\mathcal{A}_a, \mathcal{A}_g) = \frac{1}{m'} \sum_{j=1}^{m'} \|\mathbf{z}_{a,j} - \mathbf{z}_{g,j}\|^2 , \tag{6.7}$$

where $\mathbf{z}_{a,j}$ and $\mathbf{z}_{g,j}$ are latent variables, drawn from $q_{\varsigma_{stu}}(\mathbf{z}\,|\,\mathbf{x})$ when considering the data samples $\mathbf{x}'_{a,j}$ and $\mathbf{x}'_{g,j}$ generated by $\mathcal{A}_a$ and $\mathcal{A}_g$, respectively. $m' = 1000$ is the total number of generated samples. Eq. (6.7) is computationally efficient since it is evaluated on the low-dimensional latent space. Although the FID can be used to evaluate the distance onto the data generated by a pair of teacher experts, it still requires additional computation resources when compared to Eq. (6.7). If the two experts, $\mathcal{A}_a$ and $\mathcal{A}_b$ share significant amounts of information, their corresponding latent variables $\mathbf{z}_a$ and $\mathbf{z}_b$ tend to be similar, resulting in a small $\mathcal{L}_{ks}(\mathcal{A}_a, \mathcal{A}_g)$ in Eq. (6.7). We use the square loss to evaluate the distance between the latent variables due to being computationally efficient while other choices are discussed in Section 6.4. Once the discrepancy matrix $\mathbf{Q}$ is evaluated, we identify a pair of experts containing overlapping information by searching for the minimal discrepancy score in $\mathbf{Q}$:

$$F_{\text{find}}(\mathbf{Q}) = \arg\min_{\{\{a,g\}\,|\,a=1,\cdots,K-1,b=1,\cdots,K-1,a\neq g\}} \left\{\mathbf{Q}[a,g]\right\}, \tag{6.8}$$

where $\{a^\star, g^\star\} = F_{\text{find}}(\mathbf{Q})$. $a^\star$ and $g^\star$ are the indices of the selected experts. We only employ Eq. (6.8) to find a pair of frozen teacher experts since the current teacher expert ($\mathcal{A}_K$) is still activated and can not be removed. We then evaluate the discrepancy score between each other expert from the teacher's ensemble and either $\mathcal{A}_{a^\star}$ or $\mathcal{A}_{g^\star}$:

$$F_{\text{diversity}}(a^\star, \mathbf{Q}) = \sum_{j=1,j\neq a^\star}^{K-1} \left\{\mathbf{Q}[a^\star, j]\right\}. \tag{6.9}$$

We employ Eq. (6.9) to estimate the discrepancy score for $\mathcal{A}_{a^\star}$ and $\mathcal{A}_{g^\star}$, denoted as $s_{a^\star} = F_{\text{diversity}}(a^\star, \mathbf{Q})$ and $s_{g^\star} = F_{\text{diversity}}(g^\star, \mathbf{Q})$. A higher discrepancy score, calculated using Eq. (6.9), indicates that the selected component represents distinct knowledge from the other components and should be kept. The main goal for Eq. (6.9) is that we keep the expert with the largest discrepancy scores in the teacher module to ensure the knowledge diversity between the existing teacher's experts. If $s_{a^\star} > s_{g^\star}$, then we decide to remove $\mathcal{A}_{g^\star}$, while otherwise we remove $\mathcal{A}_{a^\star}$. In order to eliminate more experts from the teacher module, after removing the deleted expert from $\mathbf{Q}$ we continue with identifying other redundant experts from the teacher module using Eq. (6.8) and (6.9) and designate them for removal. We iteratively remove experts from the teacher module until the number of experts is equal to a predefined $n_{\mathcal{A}} \in [3, 10]$, as illustrated in Fig. 6.2.

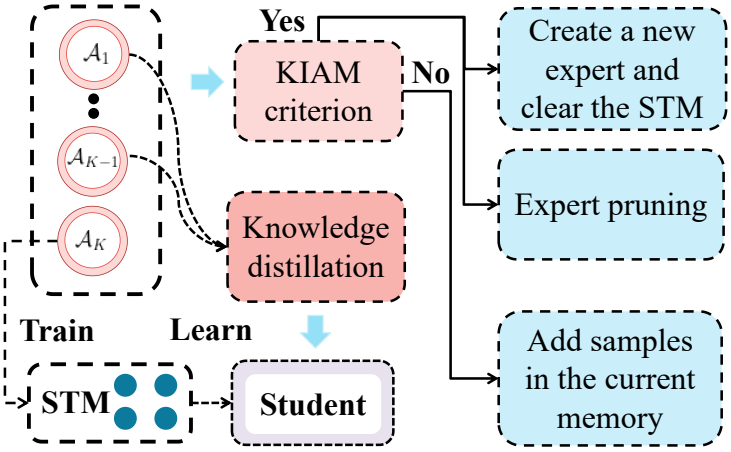


Figure 6.3: The learning procedure of the proposed framework where we omit the updating of the memory for the sake of simplification.

6.2.5 Implementation

Since the teacher module can dynamically build several components to capture the information from a data stream during the training, one appropriate approach for training the teacher module is to update the newly created component (current component) while freezing all previously learnt components to preserve past information. Such an approach allows the teacher module to gradually store new knowledge without forgetting previously learnt information. Unlike the D-TS framework described in Chapter 5, which considers the task information as being known and thus allows the teacher module to reuse an existing component to learn a new task, the CGKD does not perform the expert selection and constantly updates the current expert during training. Suppose that the CGKD has learnt K experts $\{\mathcal{A}_1, \cdots, \mathcal{A}_K\}$, while $\mathcal{A}_K$ is the current expert, we provide the main objective function for the teacher module in the following.

GAN-based teacher module. We have two approaches: implementing each expert of the teacher module by using either a VAE or a GAN. For the GAN-based teacher module, the current expert $\mathcal{A}_K$ has a discriminator network $\mathcal{D}_\eta$ parameterized by η and a generator $\mathcal{G}_{\theta_K}$ parameterized by θ_K. We consider the WGAN-GP loss [55] for training the current K-th expert $\mathcal{A}_K$ at the t-th training step:

$$\mathcal{L}_{\mathcal{G}^{\text{CGKD}}}(\theta_K, \eta) = \frac{1}{m}\sum_{j=1}^{m}\Big\{ - \mathcal{D}_\eta(\mathcal{G}_{\theta_K}(\mathbf{z}_j))\Big\}, \tag{6.10}$$

$$\mathcal{L}_{\mathcal{D}^{\text{CGKD}}}(\mathbf{X}_{batch}, \theta_K, \eta) = \frac{1}{m}\sum_{j=1}^{m}\left\{\mathcal{D}_\eta(\mathcal{G}_{\theta_K}(\mathbf{z}_j) - \mathcal{D}_\eta(\mathbf{x}_j) + \lambda(\|\nabla_{\tilde{\mathbf{x}}_j}\mathcal{D}_\eta(\tilde{\mathbf{x}}_j)\|_2 - 1)^2\right\}, \tag{6.11}$$

where $m = 64$ is the batch size and $\mathbf{z}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a random vector. $\mathbf{x}_j$ is the j-th real sample from the data batch $\mathbf{X}_{batch}$ obtained from the memory buffer $\mathcal{M}_t$, where we employ the subscript t to denote that the memory buffer $\mathcal{M}_t$ was updated at the t-th training step/time. λ is a hyperparameter, and the last term is used to ensure the discriminator's Lipschitz constraint, [55]. $\tilde{\mathbf{x}}_j$ is the interpolated image produced by $\tilde{\mathbf{x}}_j = s\mathbf{x}_j + (1-s)\mathbf{x}'_j$ where s is drawn from a uniform distribution $U(0,1)$ and $\mathbf{x}'_j$ is a generated image. We only need a single discriminator during the training to reduce the whole model size. We provide the detailed network architecture of the generator and discriminator in Fig. 6.1.

VAE-based teacher module. For the VAE-based teacher module, the current expert $\mathcal{A}_K$ is implemented as a VAE model consisting of a decoder $p_{\theta_K}(\mathbf{x}\,|\,\mathbf{z})$ and an encoding distribution $q_\varsigma(\mathbf{z}\,|\,\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$ whose hyperparameters $\{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$ are given by the encoder. A latent variable $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \bigodot \boldsymbol{\tau}$ is then drawn from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$ using the reparameterization trick [85], where $\boldsymbol{\tau} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\bigodot$ is the element-wise product. We provide the detailed network architecture of the encoder of the VAE-based expert in Fig. 6.4 while the decoder uses the same network architecture from the GAN-based expert. We employ the VAE objective function for training the current expert $\mathcal{A}_K$, expressed as:

$$\mathcal{L}_{\text{VAE}}^{\text{CGKD}}(\mathbf{X}_{batch}, \theta_K, \varsigma) = \frac{1}{m}\sum_{j=1}^{m}\Big\{ - \mathbb{E}_{q_\varsigma(\mathbf{z}\,|\,\mathbf{x}_j)}\left[\log p_{\theta_K}\left(\mathbf{x}\,|\,\mathbf{z}\right)\right] + D_{KL}\left[q_\varsigma\left(\mathbf{z}\,|\,\mathbf{x}_j\right) \,||\, p\left(\mathbf{z}\right)\right]\Big\}, \tag{6.12}$$

where $\mathbf{x}_j$ is the j-th sample from the data batch $\mathbf{X}_{batch}$. Similar to GAN-based experts, we only need a single encoding distribution $q_\varsigma(\mathbf{z}\,|\,\mathbf{x})$ for the teacher module during the training because all previously frozen experts $\{\mathcal{A}_1, \cdots, \mathcal{A}_{K-1}\}$ do not need to be updated in the subsequent learning. We summarize several differences for the teacher module between the D-TS and the CGKD: (1) The teacher module in D-TS can select an existing component to learn a new task, which requires accessing the task information. The proposed CGKD always updates the current expert to learn new knowledge while freezing all previously learnt experts, which can be used in TFCL; (2) The teacher's model size in D-TS would grow as the number of tasks increases. In contrast, we can fix the teacher's model size in the CGKD by employing the

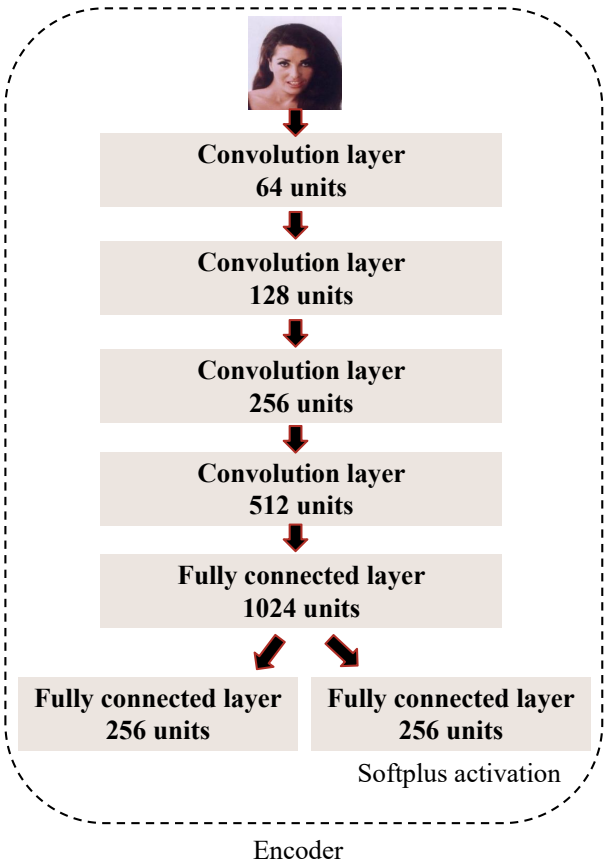


Figure 6.4: The network architecture of the encoder.

expert pruning approach for removing the redundant teacher's mixture modules; (3) The D-TS only considers employing the GAN-based teacher module. In contrast, the CGKD investigates the performance of both GAN- and VAE-based teacher experts.

Algorithm. We provide the pseudo-code in **Algorithm 7**, and we summarize the training algorithm in five steps.

- **(Updating the memory buffer $\mathcal{M}_t$).** We update $\mathcal{M}_t$ at the t-th training time/step by adding a batch of samples $\mathbf{X}_{batch}^t$ obtained from the data stream $\mathcal{S}$ into its buffer if the memory is not full $|\mathcal{M}_t| < |\mathcal{M}_t|_{max}$, otherwise, we remove the earliest batch of samples included in $\mathcal{M}_t$ and add $\mathbf{X}_{batch}^t$, where $|\mathcal{M}_t|$ and $|\mathcal{M}_t|_{max}$ denote the current and maximum number of samples for the memory buffer $\mathcal{M}_t$.

- **(Teacher learning).** If the teacher module has only a single expert at the initial training phase, we automatically build a new expert $\mathcal{A}_2$ at the (100)-th training step while freezing $\mathcal{A}_1$. Such a building process can allow us to perform the dynamic expansion mechanism during the subsequent learning because Eq. (6.3) requires at least a frozen expert. We update the current expert on the memory buffer $\mathcal{M}_t$ using Eq. (6.10) and Eq. (6.11)

Algorithm 7: Training algorithm of CGKD*-GAN

Input: All training databases
Output: The model's parameters

for $t < N'$ **do**
  $\mathbf{X}_{batch}^{t}$ obtained from $\mathcal{S}$;
  Updating of the memory ;
  if *(*$|\mathcal{M}_t| \geq |\mathcal{M}_t|_{max}$*)* **then**
    Remove earliest samples from $\mathcal{M}_t$;
  end
  $\mathcal{M}_t = \mathcal{M}_t \bigcup \mathbf{X}_{batch}^{t}$;
  Teacher learning ;
  if *(t == 100)* **then**
    Build a new expert $\mathcal{A}_2$ while fixing $\mathcal{A}_1$;
  end
  else
    for *j < iterations (Each model is only updated once in each iteration)* **do**
      $\mathbf{X}_{batch}$ randomly obtained from $\mathcal{M}_t$;
      Update the generator of the current expert on $\mathbf{X}_{batch}$ using Eq. (6.10) ;
      Update the discriminator on $\mathbf{X}_{batch}$ using Eq. (6.11) ;
    end
  end
  Checking the expansion ;
  if *(*$|\mathcal{M}_t| \geq |\mathcal{M}_t|_{max}$*)* **then**
    if *Eq.* (6.3) *is stratified* **then**
      Build a new expert ;
      Cleaning up $\mathcal{M}_t$;
    end
  end
  Expert pruning ;
  Perform the expert pruning (See details in **Algorithm 8**) ;
  Student learning; ;
  Perform the student learning (See details in **Algorithm 9**) ;
end

in the mini-batch learning manner in which 'iterations' in **Algorithm 7** is determined by $iterations = (|\mathcal{M}_t|/batchSize)epoch$ where $batchSize = 64$ is the batch size, and $epoch = 1$ is the number of training epochs.

- **(Checking the expansion).** To avoid frequent evaluation, we check the expansion when the memory is full $|\mathcal{M}_t| = |\mathcal{M}|_{max}$. When the expansion criterion is satisfied, according to Eq. (6.3), we add a new expert $\mathcal{A}_{K+1}$ to the teacher module while cleaning up $\mathcal{M}_t$ to avoid learning probabilistically overlapping data, in subsequent learning;

- **(Expert pruning for KD).** We remove the non-essential experts from the teacher module using the proposed expert pruning approach until the number of experts ($|\mathbf{A}|$) in the teacher module $\mathbf{A}$ matches $n_{\mathcal{A}}$. We provide the detailed pruning process in **Algorithm 8** ;

Algorithm 8: Expert pruning.

while $|\mathbf{A}| > n_{\mathcal{A}}$ **do**
  Calculate $\mathbf{Q}$ using Eq. (6.7) ;
  Find the indexes $\{a^\star, g^\star\}$ of a pair of experts by $F_{\text{find}}(\mathbf{Q})$;
  $s_{a^\star} = F_{\text{diversity}}(a^\star, \mathbf{Q})$, $s_{g^\star} = F_{\text{diversity}}(g^\star, \mathbf{Q})$;
  if $s_{a^\star} > s_{g^\star}$ **then**
    Remove the expert $\mathcal{A}_{g^\star}$ from the teacher module ;
  end
  else
    Remove the expert $\mathcal{A}_{a^\star}$ from the teacher module ;
  end
end

Algorithm 9: Student learning.

for $j <$ *iterations (Each model is only updated once in each iteration)* **do**
  $\mathbf{X}_{batch}$ randomly obtained from $\mathcal{M}_t$;
  for $j' < K - 1$ **do**
    Generate the data batch $\mathbf{X}'_{j'}$ using $\mathcal{A}_{j'}$;
  end
  Update the student module on $\{\mathbf{X}_{batch}, \mathbf{X}'_1, \cdots, \mathbf{X}'_{K-1}\}$ using Eq. (6.6) ;
end

- **(Student learning).** We distil the generative information from the teacher module to the student module while simultaneously learning the information from the current memory $\mathcal{M}_t$ using Eq. (6.6). The student module is updated in a mini-batch learning manner and the detailed updating process is provided in **Algorithm 9**. Then we return to **Step 1** for the next training step (the $(t+1)$-th training step).

6.3 Experiments

6.3.1 Settings and Baselines

Setting: We consider a series of six data domains including MNIST [93], SVHN [118], Fashion [177], IFashion, RMNIST and CIFAR10 [87]. We create a data stream $\mathcal{S}$ from all training sets of these data domains, namely MSFIRC. We also consider the class-incremental setting where we split each data domain into five parts, and each part consists of images from two different classes [30]. We then create a data stream $\mathcal{S}$ by combining all parts of the six data domains, namely Class Incremental (CL)-MSFIRC. The batch size and the number of epochs for each training step are 64 and 1, respectively. The maximum memory size (STM) for MSFIRC and CI-MSFIRC is 5000. Since this chapter focuses on lifelong generative modelling under TFCL,

Table 6.1: FID for various models under the MSFIRC setting.

Methods	MNIST	SVHN	Fashion	IFashion	RMNIST	CIFAR10	Average	No
finetune	174.1	148.3	237.0	229.1	159.2	216.4	194.0	1
Reservoir [164]	127.2	159.3	213.4	201.6	110.2	113.3	154.2	1
LTS [180]	44.8	62,9	92.9	83.1	41.8	80.3	67.7	1
LGM [132]	104.8	134.3	194.3	168.1	94.8	91.5	131.3	1
CN-DPM [95]	118.7	73.4	120.7	120.3	97.9	97.6	104.8	18
CGKD-GAN	11.6	70.6	101.9	29.9	11.41	68.6	49.0	16
CGKD-VAE	122.9	73.6	109.2	104.3	119.1	86.4	102.6	11
CGKD*-GAN	12.0	74.6	69.8	22.3	11.4	68.5	**43.1**	7
CGKD*-VAE	82.6	82.5	127.0	132.9	88.8	86.3	100.0	7

we follow the settings from [181], which adopts the Fréchet Inception Distance (FID) [63] to evaluate the performance of various models. We employ a Tesla V100-SXM2 (32GB) GPU and RHEL 8 operating system for the experiment.

Baselines: The majority of the existing lifelong learning approaches do not focus on generative modelling or require accessing the task information during training. Therefore, we compare the proposed approach with more related methods such as: Reservoir [164], Lifelong Teacher Student (LTS) [180], Lifelong Generative Modelling (LGM) [132], and CN-DPM [95], respectively. For a fair comparison with CN-DPM, we also train a student module to learn generated data by CN-DPM together with that from the memory buffer. In addition, we also assign a short-term memory buffer with a maximum size of 5,000 for LGM and LTS to support TFCL. Specifically, when the short-term memory buffer is full, LTS and LGM perform the GRM process to produce 5,000 samples while the memory buffer is empty and is ready to store new samples during subsequent learning. All models are trained using the Adam algorithm [82] with a learning rate of 0.0002.

6.3.2 Generative Modelling Tasks Under TFCL

In this section, we investigate the performance of the student module on a data stream with multiple data distributions. The FID results for MSFIRC are shown in Tab. 6.1, where 'No' in the last column stands for the number of experts required following the proposed lifelong learning methodology. CGKD-GAN and CGKD-VAE represent using GAN and VAE as experts in the teacher module and '*' denotes the use of the proposed Expert Pruning, as explained in Section 6.2.4. We can observe that GAN-based approaches significantly outperform VAE-based methods in terms of the FID criterion, by achieving results of 67.7 by LTS versus 131.3 by LGM.

The proposed CGKD-GAN outperforms CN-DPM, despite the latter using many more experts in its teacher module. Overall, GAN-based approaches can provide high-quality generative replay data when compared to the VAE-based models and thus achieve better lifelong learning performance.

In addition, compared to CGKD-GAN, CGKD*-GAN can reduce the number of experts from 16 to 7 while still maintaining good performance. We can also observe that CGKD-VAE and CGKD*-VAE achieve similar performance, while the CGKD*-VAE reduces the number of components from 11 to 7. These results show that the proposed expert pruning approach can further reduce the teacher model size without sacrificing performance. Furthermore, the CGKD-GAN significantly outperforms CGKD-VAE by a large margin, demonstrating that the GAN used as the teacher module can lead to better performance. The image reconstruction results by CGKD-GAN are sharper than other baselines, as we can observe in Fig. 6.5.

In the following, we investigate the performance of various models in the class-incremental setting. Under this setting, each dataset is divided into five parts, where each part consists of samples from two classes. This setting is more challenging since the model requires dealing with both class and domain shifts. We report the performance of the student module in Table. 6.2. We can observe that static models, including finetune, Reservoir, LTS and LGM, perform worse on the class-incremental setting when compared with the domain-incremental setting. This is because the static model has a fixed capacity and can not capture more classes during the training. From the results of Table. 6.2, we can observe that the CGKD-GAN outperforms CGKD*-GAN, which shows that learning more components increases the performance for the class-incremental setting. The reason behind this case is that more components can potentially capture additional category information and thus are suitable for the class-incremental setting in which the class changes over time. In addition, the GAN-based models still achieve a better performance than the VAE-based methods in the class-incremental setting, showing that the high-quality generative replay samples play a critical role in lifelong generative modelling. Furthermore, the proposed CGKD-GAN outperforms other baselines in the class-incremental setting.

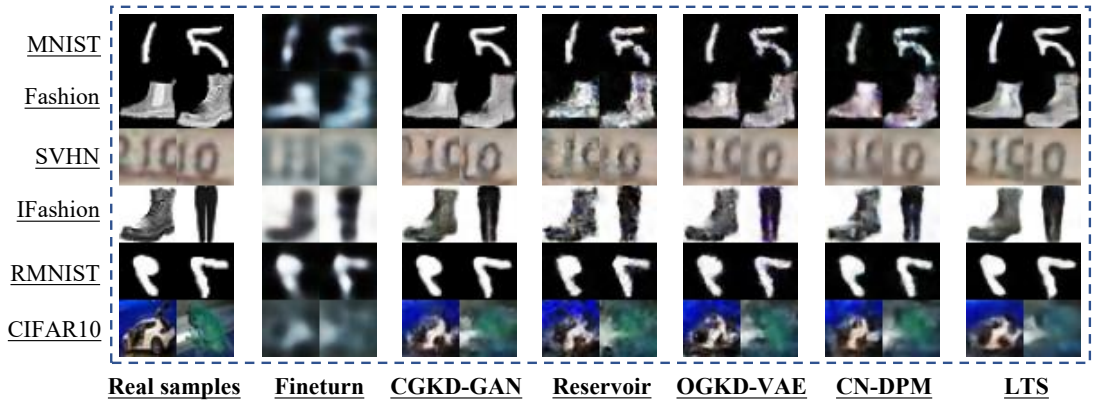


Figure 6.5: The cross-domain reconstruction results of various models under MSFIRC setting.

Table 6.2: The FID of various models under the CI-MSFIRC setting.

Methods	MNIST	SVHN	Fashion	IFashion	RMNIST	CIFAR10	Average	No
finetune	158.1	167.6	246.2	233.3	138.6	229.4	195.6	1
Reservoir [164]	141.7	163.6	220.0	200.1	127.1	115.5	161.3	1
LTS [180]	101.9	99.4	140.6	139.5	99.9	95.5	112.8	1
LGM [132]	108.5	122.1	189.5	175.9	96.6	92.4	130.9	1
CN-DPM [95]	90.9	62.0	109.0	95.0	77.9	95.5	88.4	18
CGKD-GAN	16.7	65.1	44.5	43.9	27.9	85.2	**47.2**	11
CGKD-VAE	102.6	69.9	117.1	99.5	113.0	82.7	97.5	11
CGKD*-GAN	13.5	72.7	89.9	52.1	12.4	71.9	52.1	7
CGKD*-VAE	131.0	70.3	106.7	92.2	126.5	87.7	102.4	7

6.3.3 Learning Complex Data Streams Under TFCL

We examine the performance of the student module for the datasets consisting of complex images. Following from [180], we consider 5000 samples for testing from each of the datasets, CelebA [105] and 3D-chair [14], and we create a data stream named CelebA-Chair consisting of these training samples. All samples in CelebA and 3D-Chair are resized as $64 \times 64 \times 3$ pixels. We adopt the setting of MSFIRC for CelebA-Chair, and the results of the student module are provided in Tab. 6.3. We can observe that the static model can achieve good performance on the last dataset being learnt (3D-Chair) while performing worse on the first dataset (CelebA) in the data stream being learnt, which shows that the static model suffers from forgetting. In contrast, the dynamic expansion models can perform well on the first and second datasets. In addition, the proposed CGKD-GAN outperforms other baselines while employing fewer components when compared with CN-DPM. Furthermore, compared with CGKD-VAE, CN-DPM achieves slightly better performance but employs many more experts than CGKD-VAE. Since both CGKD-GAN and CGKD-VAE employ fewer components after CelebA-Chair

lifelong learning, we do not further reduce the model size for them using the proposed expert pruning.

Table 6.3: The FID for the results generated by various models under the CelebA-Chair learning setting.

Methods	CelebA	3D-Chair	Average	No
finetune	35.79	9.90	22.85	1
Reservoir [164]	20.82	11.04	15.93	1
LTS [180]	20.68	11.47	16.07	1
LGM [132]	21.58	11.84	16.71	1
CN-DPM [95]	20.19	11.45	15.82	11
CGKD-GAN	18.05	11.32	**14.68**	3
CGKD-VAE	20.63	11.79	16.21	5

We also investigate the ability of the student module to learn cross-domain interpolations in a single latent space. After the lifelong learning, we perform interpolations on the latent space and the visual results are shown in Fig. 6.6. We observe that a 3D chair can be seamlessly transformed into a human face, with the outline of the chair gradually becoming the eyes of the human. These results show that the student module can learn cross-domain latent representations under TFCL and would implicitly model the correlations between different regions of two data domains into a single latent space.

Figure 6.6: Image interpolation results of CGKD-GAN under CelebA-Chair setting.

6.3.4 Ablation Study for Defining the Number of Components

In this section, we investigate the performance of the proposed CGKD-GAN when varying the threshold λ^{CGKD} in Eq. (6.3). We train the proposed CGKD-GAN under MSFIRC lifelong learning and the average FID score on all six testing datasets is provided in Fig. 6.7. We can observe that the proposed CGKD-GAN does not lead to a significant change in the performance

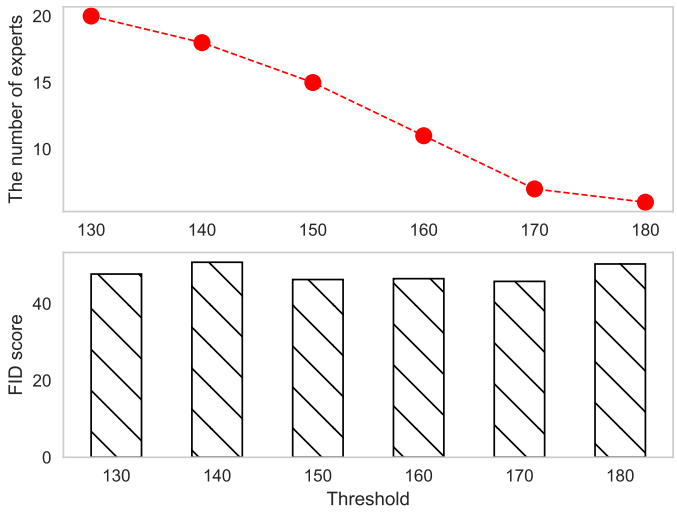


Figure 6.7: The effect of varying λ^{CGKD}.

when considering different values for λ^{CGKD}, as observed in the plot from the top of Fig. 6.7. A small λ^{CGKD} tends to result in adding more experts for the teacher module. This result shows that more experts do not lead to greater performance gains and an appropriate λ^{CGKD} would represent a trade-off between model complexity and performance.

6.4 Conclusion and Limitation

In this chapter, we propose a new framework for task-agnostic lifelong generative modelling from several different data domains without forgetting. Unlike the existing memory-based methods, which employ a single memory buffer [7], the key idea of the proposed framework is to manage two memory systems to store short- and long-term information. To implement this goal, this study considers using a memory buffer with a fixed capacity to store more recent data samples and implement the long-term memory system using a dynamic expansion model. In order to learn data representation successively, we treat the long-term memory as a teacher module into a unified framework to transfer its knowledge to a student module. However, expanding the teacher's capacity remains challenging under task-free continual learning due to the lack of task information. To solve this issue, this study introduces the Knowledge Incremental Assimilation Mechanism (KIAM) to progressively increase the teacher's knowledge through a dynamic expansion mechanism, resulting in a model with a minimal number of parameters. In addition, this study aims to further optimize the teacher module size by introducing an expert pruning approach that automatically removes those experts deemed unimportant from the teacher module. This mechanism aims to promote knowledge diversity among experts and

reduces computational costs for knowledge distillation. Furthermore, we introduce a new data-free approach that transfers the teacher's knowledge to the student module without accessing any task information, encouraging the student module to learn cross-domain representations without forgetting.

In the following, we summarise several limitations of the proposed teacher-student framework (CGKD) as follows. The proposed lifelong learning framework can only address a sequence of datasets that belong to the same kind of applications. For instance, an artificial intelligence system should be able to continually deal with different applications, including classification, object detection, generation and language processing tasks. Moreover, such a system should perform the inference in any task category following lifelong learning. However, the main challenge for this system is that different applications usually require different network architecture designs, while existing methods fail to address this challenge. We believe that by developing such a system can allow the machine to continually learn knowledge from different domains, sources and devices, which benefits a wide range of real-world applications.

In addition, the computational cost for the proposed CGKD remains challenging since it requires frequently checking the model expansion (Eq. (6.3)) and performing the expert pruning process. One possible way to further reduce the computational costs is to have a more efficient expansion criterion. In addition, instead of checking the model expansion at each training step, we can perform the dynamic expansion mechanisms after performing a pre-defined number of training steps. The other option to reduce the computational costs is to compare the distance between the newly seen data batch and the already-learnt knowledge. Such an approach can accelerate the dynamic expansion mechanism by generating fewer data samples.

On the other hand, the proposed CGKD would leak privileged information from the neural network-based evaluation into the training regime since the proposed expansion criterion (Eq. (6.3)) involves the generation process. To address this issue, we can replace the FID-based criterion with a new criterion that does not need the generation process. For instance, we can compare the loss change in each training step using the student module, which is used as the expansion signal for the teacher module. We will investigate this idea in a future study.

Furthermore, as shown in Section 6.2.4, we only consider employing the square loss to calculate the distance between two latent variables in Eq. (6.7) for reducing the teacher model size. However, the square loss is not a unique choice, and there are several other options, such as probability measures. Firstly, Maximum Mean Discrepancy (MMD) [97] is a well-

known approach consisting of a distance on the space of probability measures, which has been widely used in deep and machine learning research [42, 42]. Specifically, the MMD measure can be empirically estimated using data samples from the empirical data distributions [42], which is suitable for evaluating Eq. (6.7). Secondly, instead of calculating the distance between samples in Eq. (6.7), we can evaluate the probability distance such as the Jensen–Shannon (JS) divergence between variational distributions $q_{\varsigma_{stu}}(\mathbf{z} \mid \mathbf{x}_{a,j})$ and $q_{\varsigma_{stu}}(\mathbf{z} \mid \mathbf{x}_{g,j})$ modelled by the student module, where $\mathbf{x}_{a,j}$ and $\mathbf{x}_{g,j}$ are data samples generated by $\mathcal{A}_a$ and $\mathcal{A}_g$, respectively. Moreover, although the proposed expert pruning approach can maintain a compact teacher module by appropriately selecting a hyperparameter $n_{\mathcal{A}}$, it still can not learn an infinite data stream due to the fixed maximum capacity. In a future study, we will address this limitation by proposing two solutions. Firstly, we can gradually increase the hyperparameter $n_{\mathcal{A}}$ over time to increase the model's capacity. Such a solution requires carefully designing a schedule to update $n_{\mathcal{A}}$. The second solution is to introduce a new hyperparameter λ_2 to replace $n_{\mathcal{A}}$ and we continually remove overlapped experts until a new criterion $\lambda_2 < \min\{\mathbf{Q}\}$ is met. Compared to the hyperparameter $n_{\mathcal{A}}$, which has a pre-defined maximum model size, this solution can allow the teacher module to reduce the model size appropriately while having no model capacity limitations. We will investigate these ideas in our future study.

On the other hand, TFCL is still a new and challenging learning paradigm. In this chapter, we have only investigated the performance of the proposed framework for unsupervised image reconstruction and interpolation tasks. However, the proposed framework can be extended to other downstream tasks, including supervised and disentangled representation learning, which will be investigated in future work.

Chapter 7

Conclusion and Future Work

This thesis studies the deep generative model under the context of lifelong learning. Deep generative models have shown promising results on individual tasks, especially when all training samples are accessible. However, these models suffer from serious forgetting when training on a sequence of tasks. To relieve forgetting, we first propose a novel hybrid model that combines advantages from both GAN and VAE models, in which a GAN is trained to provide high-quality replay samples while the VAE-based inference model is trained to capture meaningful latent representations across multiple domains over time. However, this hybrid approach can not deal with a growing number of tasks due to the fixed model capacity. This thesis addresses this issue by proposing a growing mixture model (GMM) that can deal with a long sequence of tasks. GMM dynamically expands the model capacity to address the learning of a novel task while reusing the existing component to learn a related task, which reduces the whole model size. A GMM can be used to train a weak student module by combining data generated by the GMM and new data, through knowledge distillation. However, a student module, after being trained by means of knowledge distillation, would perform worse on the image interpolation and cross-domain image reconstruction tasks. To address this issue, we propose a new lifelong teacher-student framework which dynamically adds new GAN components to learn novel concepts. Meanwhile, we implement the student module as a generative latent variable model and propose a new knowledge distillation loss that dynamically transfers generative factors from multiple teacher components to the student module without forgetting. We further extend the proposed teacher-student framework to task-free continual learning, which does not require accessing task information and can be used for learning infinite data streams. In the following, we briefly summarise the contributions of each chapter.

7.1 Summary of Contributions

In Chapter 3, we propose a new hybrid model for lifelong generative modelling, which integrates the generative ability of GANs and the inference capacity of VAEs into a unified optimisation framework. We extend the proposed hybrid framework to deal with various applications, including unsupervised generative modelling, semi-supervised classification, image interpolation and disentangled representation learning. The empirical results show that the proposed LGAA achieves better performance than other approaches while providing high-quality image representation and disentangled representation results. In addition, we also study the transfer ability of the LGAA when learning a new task. The empirical results demonstrate that when a new task shares similar visual concepts with respect to the previously learnt datasets, LGAA can converge quickly during the training phase when compared to the baseline that learns a new task using randomly initialized parameters.

In Chapter 4, we propose a growing mixture model aiming to dynamically build new components to learn novel tasks while freezing all previously trained components. Such a mechanism can reduce the number of generative replay processes. In addition, we share many parameters among all components for GMM to further reduce the whole model size. Furthermore, the proposed GMM computes the correlation between the already accumulated knowledge and a new task, which can guide to reuse an existing component for learning a related task. We perform a series of experiments, and the empirical results demonstrate that the proposed GMM achieves better performance than other baselines in both unsupervised and supervised learning tasks.

In Chapter 5, we discuss the limitation of the proposed GMM, which learns a student module that tends to produce blurred image reconstructions. This is caused by using a VAE-based teacher module that usually gives lower-quality generative replay samples for training the student module. We address this limitation by developing a novel lifelong teacher-student system that learns a dynamically expandable mixture GAN model as a teacher module. The primary advantage of the GAN-based teacher module is to provide high-quality knowledge transfer for student learning, enabling the student module to capture high-quality latent representations across multiple data domains over time. In addition, we introduce a prior distribution to regulate the student optimisation by incorporating the identity information of each teacher expert in the objective function, which further improves the image interpolation performance. Finally, the proposed Dynamic Teacher-Student (D-TS) can be extended to various applications with

only small modifications. We perform a series of experiments, and the empirical results demonstrate that the proposed D-TS framework performs better than other related methods on both supervised and unsupervised learning. In addition, the analysis results show that the proposed D-TS can reuse a certain teacher expert to learn several similar tasks, benefitting the positive knowledge transfer and the reduction of the model size.

In Chapter 6, we address a more realistic and challenging learning scenario (TFCL) in which the task information and boundaries are unavailable during the training. The existing teacher-student frameworks fail to address TFCL since their dynamic expansion process relies on knowing the task identity. To address this issue, this study first introduces a short-term memory buffer to store novel data samples, which are then used to provide enough information for training the current teacher expert. Then, we introduce a new dynamic expansion mechanism to evaluate the distance between the current memory and the already-learnt knowledge as the expansion signal, which appropriately expands the teacher architecture without the need to know any task information. Furthermore, we introduce an expert pruning approach to remove those teacher experts, which contain similar (overlapping) information, which further reduces the model size. The empirical results show that the proposed framework is scalable to a more realistic continual learning scenario with several different data domains and performs better than other methods.

7.2 Future Work

The Denoising Diffusion Probabilistic Models (DDPM) is a rapidly developed generative model [152, 66, 107, 154], which has been been successfully applied to many applications beyond image synthesis, including image super-resolution [67, 98], image inpainting [155], graph generation [120], shape generation [21] and text-to-image generation [54, 79]. Unlike other popular generative models such as GANs and VAEs, which consider a unique data pass for the generation process, generating an image using the diffusion-based model is implemented through an iterative process in which noise is generated and gradually refined to a clear image through hundreds of diffusion steps [66]. However, such a generation procedure requires significant computational costs. Although many studies have proposed several solutions to accelerate the generation process of DDPM by shortening the reverse diffusion steps [31] or by performing the diffusion process in the low-dimensional latent space [140], the computational cost for the diffusion

model is still very high compared to that for the GANs or VAEs. Due to the DDPM's powerful image generation ability, we can implement each teacher module using a diffusion model in the proposed teacher-student system, which can provide much better generative replay samples for student learning when compared with VAE- and GAN-based teacher modules. To reduce the overall computational costs, one can develop a new knowledge distillation approach that transfers the teacher's knowledge to the student module when all tasks are finished, aiming to reduce the generation process of the teacher module. As a result, the computational costs for such a knowledge distillation approach rely on the teacher model size instead of the number of tasks to be learnt. Furthermore, we can employ the expert pruning approach to remove those experts considered unimportant to the teacher module, which would accelerate the knowledge distillation process.

The proposed lifelong learning teacher-student framework can be developed as a continuation of the research presented in this thesis, in which the generators are treated as the parameterized memories (teacher), which provide generative replay samples to relieve forgetting. Then a mixture of solvers can be developed to improve the student learning capabilities in the proposed framework, where each solver can be implemented using different network architectures to deal with different applications such as classification, image reconstruction and object detection. In addition, a knowledge distillation approach can be proposed to selectively transfer the knowledge from each selected teacher component to a related student's solver. Therefore, the proposed framework's component expansion and selection mechanism would evaluate the relevance between old knowledge and the incoming task and determine the application type of the incoming task. Finally, such a lifelong learning framework can address the learning of a sequence of tasks with different applications over time.

Moreover, learning from data riddled with noisy labels is a more realistic scenario that has been recently investigated in several works [78, 169, 187]. Such noisy data streams represent more challenging experimental settings since a deep learning system can be intentionally "infected" or "attacked" intentionally with wrong data in a security breach attempt to derail its training. To address this challenging scenario, a correction mechanism can be proposed, which continually corrects the noisy samples in order to achieve a good generalization performance. The other solution is to develop a noisy sample detection mechanism, which identifies the correct data samples from a noisy data stream and stores them in a memory buffer. In the subsequent learning, the corrected supervised signals can be used from the memory buffer to

detect incoming samples while the model is optimized using the memorized samples.

Moreover, few-shot learning [170] and continual learning [191] are two different but important tasks for machine intelligence. Although these two tasks have been studied intensively, merging them within a unified learning paradigm has scarcely been investigated. A recent study has proposed the first benchmark for continual few-shot learning [10], which attempts to promote the development of this research topic. In future studies, we can apply the proposed teacher-student framework to address continual few-shot learning.

Furthermore, the proposed frameworks described in Chapters 3, 4, 5 and 6 mainly employ the convolutional network as the backbone. In our future works, we would like to employ the Vision Transformer (ViT) [36], which employs a self-attention mechanism, which proved its effectiveness in many deep learning applications. Due to its good generalisation performance, the ViT has recently been explored for continual learning [38, 172, 178]. However, ViT-based approaches require access to the task information to generate specific attention masks for each learning task [38, 178], which cannot be considered in a realistic continual learning scenario with no explicit task boundaries [173]. Moreover, these approaches are based on either static networks [172, 178] or on expansion models, where the number of their model parameters grows infinitely when learning an increasing number of tasks [38], which makes them intractable for learning long-term data streams. In a future study, we will develop a new ViT-based model to address TFCL.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016.

[2] A. Achille, T. Eccles, L. Matthey, C. Burgess, N. Watters, A. Lerchner, and I. Higgins. Life-long disentangled representation learning with cross-domain latent homologies. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 9873–9883, 2018.

[3] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *In Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 4394–4404, 2019.

[4] E Akagünduz, A. G. Bors, and K. Evans. Defining image memorability using visual memory schema. *IEEE Trans. on Pattern Anal. and Machine Intelligence*, 42(9):2165–2178, 2020.

[5] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *In Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 1–9, 2019.

[6] R. Aljundi, P. Chakravarty, and T. Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3366–3375, 2017.

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

[8] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio. Gradient based sample selection for online continual learning. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 11816–11825, 2019.

[9] Mark Gales Andrey Malinin, Bruno Mlodozeniec. Ensemble distribution distillation. In *Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2020.

[10] Antreas Antoniou, Massimiliano Patacchiola, Mateusz Ochal, and Amos Storkey. Defining benchmarks for continual few-shot learning. *arXiv preprint arXiv:2004.11967*, pages 1–9, 2020.

[11] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proc. Int. Conf. on Machine Learning (ICML)*, volume 70, pages 214–223. PMLR, 2017.

[12] Marcus Rohrbach Arslan Chaudhry, Marc'Aurelio Ranzato and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2019.

[13] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic. Seeing 3D chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3762–3769, 2014.

[14] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3762–3769, 2014.

[15] Mohammadamin Banayeeanzade, Rasoul Mirzaiezadeh, Hosein Hasani, and Mahdieh Soleymani. Generative vs. discriminative: Rethinking the meta-continual learning. In *In Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, volume 34, pages 21592–21604, 2021.

[16] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 8218–8227, 2021.

[17] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *Proc. Inter. Conference on Machine Learning (ICML), vol. PMLR 80*, pages 531–540, 2018.

[18] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, pages 1–8, 2015.

[19] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in *beta*-vae. *arXiv preprint arXiv:1804.03599*, pages 1–8, 2018.

[20] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *In Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 1–9, 2020.

[21] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 364–381, 2020.

[22] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. Dokania, P. H. S. Torr, and M.'A. Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, pages 1–9, 2019.

[23] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, pages 1–9, 2016.

[24] B.-C. Chen, C.-S. Chen, and W. H. Hsu. Cross-age reference coding for age-invariant face recognition and retrieval. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 768–783, 2014.

[25] Liqun Chen, Shuyang Dai, Yunchen Pu, Erjin Zhou, Chunyuan Li, Qinliang Su, Changyou Chen, and Lawrence Carin. Symmetric variational autoencoder and connections to adversarial learning. In *Proc. Int. Conf. on Artificial Intel. and Statistics (AISTATS)*, volume 84, pages 661–669. PMLR, 2018.

[26] T. Q. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 2615–2625, 2018.

[27] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 2172–2180, 2016.

[28] Jaewoong Choi, Geonho Hwang, and Myungjoo Kang. Disentangling the correlated continuous and discrete generative factors of data. *Pattern Recognition*, 133:109055, 2023.

[29] Yulai Cong, Miaoyun Zhao, Jianqiao Li, Sijia Wang, and Lawrence Carin. GAN memory with no forgetting. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *In Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 1–9, 2020.

[30] Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, pages 8250–8259, 2021.

[31] Kamil Deja, Anna Kuzina, Tomasz Trzcinski, and Jakub Mikolaj Tomczak. On analyzing generative and denoising capabilities of diffusion-based deep generative models. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 1–9, 2022.

[32] Mohammad Mahdi Derakhshani, Xiantong Zhen, Ling Shao, and Cees Snoek. Kernel continual learning. In *Proc. Int Conf. on Machine Learning (ICML)*, volume 139, pages 2621–2631. PMLR, 2021.

[33] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5138–5146, 2019.

[34] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, pages 1–21, 2016.

[35] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2017.

[36] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2021.

[37] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 86–102, 2020.

[38] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 9285–9295, 2022.

[39] DC Dowson and BV Landau. The Fréchet distance between multivariate Normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455, 1982.

[40] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially learned inference. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2017.

[41] E. Dupont. Learning disentangled joint continuous and discrete representations. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 710–720, 2018.

[42] Gintare Karolina Dziugaite, Daniel M. Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI*, pages 258–267, 2015.

[43] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning. In *Proc. Proc. European Conf. on Computer Vision (ECCV) (ECCV), vol. LNCS 12356*, pages 386–402, 2020.

[44] J. Fagot and R. G. Cook. Evidence for large long-term memory capacities in baboons and pigeons and its implications for learning and the evolution of cognition. *Proc. of the National Academy of Sciences (PNAS)*, 103(46):17564–17567, 2006.

[45] Enrico Fini, Stéphane Lathuilière, Enver Sangineto, Moin Nabi, and Elisa Ricci. Online continual learning under extreme memory constraints. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 720–735, 2020.

[46] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. Int Conf. on Machine Learning (ICML)*, volume 70, pages 1126–1135. PMLR, 2017.

[47] Anna Floyer-Lea and Paul M Matthews. Distinguishable brain activation networks for short-and long-term motor skill learning. *Journal of neurophysiology*, 94(1):512–518, 2005.

[48] S. Gao, R. Brekelmans, G. ver Steeg, and A. Galstyan. Auto-encoding total correlation explanation. In *Proc. Int. Conf. on Artificial Intel. and Statistics (AISTATS)*, volume 89, pages 1157–1166. PMLR, 2019.

[49] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016.

[50] Benoit Gaujac, Ilya Feige, and David Barber. Improving gaussian mixture latent variable model convergence with optimal transport. In *Asian Conference on Machine Learning*, pages 737–752. PMLR, 2021.

[51] Jacob Goldberger, Shiri Gordon, Hayit Greenspan, et al. An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures. In *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, pages 487–493, 2003.

[52] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 2672–2680, 2014.

[53] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

[54] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10696–10706, 2022.

[55] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of Wasserstein GANs. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 5767–5777, 2017.

[56] Emil Julius Gumbel. Statistical theory of extreme values and some practical applications. *NBS Applied Mathematics Series*, 33, 1954.

[57] Jaehong Kim Hanul Shin, Jung Kwon Lee and Jiwon Kim. Continual learning with deep generative replay. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 2990–2999, 2017.

[58] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

[59] Jiangpeng He, Runyu Mao, Zeman Shao, and Fengqing Zhu. Incremental learning in online scenario. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 13923–13932, 2020.

[60] Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. In *Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2019.

[61] 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.

[62] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge distillation with adversarial samples supporting decision boundary. In *Proc. AAAI Conf. on Artificial Intelligence*, volume 33, pages 3771–3778, 2019.

[63] 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 Neural Inf. Proc. Systems (NeurIPS)*, pages 6626–6637, 2017.

[64] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. β-VAE: Learning basic visual concepts with a constrained variational framework. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2017.

[65] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, pages 1–9, 2015.

[66] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, 33:6840–6851, 2020.

[67] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *J. Mach. Learn. Res.*, 23(47):1–33, 2022.

[68] Alain Hore and Djemel Ziou. Image quality metrics: PSNR vs. SSIM. In *Proc. Int. Conf. on Pattern Recognition (ICPR)*, pages 2366–2369, 2010.

[69] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 831–839, 2019.

[70] Chin-Wei Huang, Kris Sankaran, Eeshan Dhekane, Alexandre Lacoste, and Aaron Courville. Hierarchical importance weighted autoencoders. In *Int. Conf. on Machine Learning (ICML)*, volume 97, pages 2869–2878, 2019.

[71] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1125–1134, 2017.

[72] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with Gumbel-Softmax. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2017.

[73] Khurram Javed and Martha White. Meta-learning representations for continual learning. *Advances in neural information processing systems*, 32, 2019.

[74] Ghassen Jerfel, Erin Grant, Tom Griffiths, and Katherine A Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 9122–9133, 2019.

[75] Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient-based editing of memory examples for online task-free continual learning. In *In Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 1–9, 2021.

[76] Zixuan Ke, Bing Liu, and Xingchang Huang. Continual learning of a mixed sequence of similar and dissimilar tasks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *In Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 1–9, 2020.

[77] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proc. AAAI Conf. on Artificial Intelligence*, pages 3390–3398, 2018.

[78] Chris Dongjoo Kim, Jinseo Jeong, Sangwoo Moon, and Gunhee Kim. Continual learning on noisy data streams via self-purified replay. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, pages 537–547, 2021.

[79] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2426–2435, 2022.

[80] H. Kim and A. Mnih. Disentangling by factorising. In *Proc. of Int. Conf. on Machine Learning (ICML), vol. PMLR 80*, pages 2649–2658, 2018.

[81] Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1317–1327, 2016.

[82] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2015.

[83] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 3581–3589, 2014.

[84] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 4743–4751, 2016.

[85] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, pages 1–9, 2013.

[86] 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, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proc. of the National Academy of Sciences (PNAS)*, 114(13):3521–3526, 2017.

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

[88] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 1097–1105, 2012.

[89] Anna Kuzina, Evgenii Egorov, and Evgeny Burnaev. Boovae: Boosting approach for continual learning of vae. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 17889–17901, 2021.

[90] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[91] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proc. Int Conf. on Machine Learning (ICML)*, volume 48, pages 1558–1566, 2015.

[92] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[93] 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.

[94] Seungwon Lee, Sima Behpour, and Eric Eaton. Sharing less is more: Lifelong learning in deep networks with selective layer transfer. In Marina Meila and Tong Zhang, editors, *Proc. Int. Conf. on Machine Learning (ICML)*, volume 139, pages 6065–6075. PMLR, 2021.

[95] Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural Dirichlet process mixture model for task-free continual learning. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2020.

[96] C. Li, H. Liu, C. Chen, Y. Pu, L. Chen, R. Henao, and L. Carin. Alice: Towards understanding adversarial learning for joint distribution matching. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 5495–5503, 2017.

[97] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: Towards deeper understanding of moment matching network. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 2203–2213, 2017.

[98] Haoying Li, Yifan Yang, Meng Chang, Shiqi Chen, Huajun Feng, Zhihai Xu, Qi Li, and Yueting Chen. Srdiff: Single image super-resolution with diffusion probabilistic models. *Neurocomputing*, 479:47–59, 2022.

[99] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.

[100] Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. TRGP: Trust region gradient projection for continual learning. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2022.

[101] Huidong Liu, Xianfeng Gu, and Dimitris Samaras. Wasserstein gan with quadratic transport cost. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4832–4841, 2019.

[102] Hao Liu and Huaping Liu. Continual learning with recursive gradient optimization. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2022.

[103] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 700–708, 2017.

[104] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan C. Raducanu, Andrew D. Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 915–924, 2020.

[105] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proc. of IEEE Int. Conf. on Computer Vision (ICCV)*, pages 3730–3738, 2015.

[106] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 6467–6476, 2017.

[107] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, pages 1–9, 2021.

[108] Divyam Madaan, Jaehong Yoon, Yuanchun Li, Yunxin Liu, and Sung Ju Hwang. Rethinking the representational continuity: Towards unsupervised continual learning. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2022.

[109] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2016.

[110] C. J Maddison, D. Tarlow, and T. Minka. A* sampling. *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, page 1–10, 2014.

[111] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022.

[112] Alireza Makhzani and Brendan J Frey. Pixelgan autoencoders. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 1975–1985, 2017.

[113] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2016.

[114] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *Proc. Conf. on Learning Theory (COLT), arXiv preprint arXiv:2002.06715*, 2009.

[115] L. Mescheder, S. Nowozin, and A. Geiger. Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. In *Proc. Int. Conf. on Machine Learning (ICML)*, volume 70, pages 2391–2400. PMLR, 2017.

[116] Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, pages 1–9, 2016.

[117] S. Narayanaswamy, T. B. Paige, J.-W. Van de Meent, A. Desmaison, N. Goodman, P. Kohli, F. Wood, and P. Torr. Learning disentangled representations with semi-supervised deep generative models. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 5925–5935, 2017.

[118] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS) Workshop*, pages 1–9, 2011.

[119] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. In *Proc. of Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2018.

[120] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *Proc. Proc. Int. Conf. on Artificial Intel. and Statistics (AISTATS)*, pages 4474–4484. PMLR 108, 2020.

[121] S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 271–279, 2016.

[122] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *Int Conf. on Machine Learning (ICML)*, volume 70, pages 2642–2651. JMLR, 2017.

[123] Alon Oring, Zohar Yakhini, and Yacov Hel-Or. Autoencoder image interpolation by shaping the latent space. In *Proc. of Int. Conf. on Machine Learning (ICML)*, volume 139, pages 8281–8290. PMLR, 2021.

[124] Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard Turner, and Mohammad Emtiyaz E Khan. Continual deep learning by functional regularisation of memorable past. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, volume 33, pages 4453–4464, 2020.

[125] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[126] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3967–3976, 2019.

[127] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proc. AAAI Conf. on Artificial Intelligence*, pages 3942–3951. AAAI Press, 2018.

[128] Mary Phuong and Christoph Lampert. Towards understanding knowledge distillation. In *Proc. Int Conf. on Machine Learning (ICML)*, volume 97, pages 5142–5151. PMLR, 2019.

[129] R. Polikar, L. Upda, S. S. Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Trans. on Systems Man and Cybernetics, Part C*, 31(4):497–508, 2001.

[130] Mozhgan PourKeshavarzi, Guoying Zhao, and Mohammad Sabokrou. Looking back on learned experiences for class/task incremental learning. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2022.

[131] Radim Šára Radim Tyleček. Spatial pattern templates for recognition of objects with regular structure. In *Proc. German Conf. on Pattern Recognition, vol. LNCS 8142*, pages 364–374, 2013.

[132] Jason Ramapuram, Magda Gregorova, and Alexandros Kalousis. Lifelong generative modeling. *Neurocomputing*, 404:381–400, 2020.

[133] Dushyant Rao, Francesco Visin, Andrei A. Rusu, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Continual unsupervised representation learning. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 7645–7655, 2019.

[134] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2001–2010, 2017.

[135] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[136] B. Ren, H. Wang, J. Li, and H. Gao. Life-long learning based on dynamic combination model. *Applied Soft Computing*, 56:398–404, 2017.

[137] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *Proc. Int. Conf. on Machine Learning (ICML)*, volume 37, pages 1530–1538. JMLR, 2015.

[138] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proc. Int. Conf. on Machine Learning (ICML), vol. PMLR 32*, pages 1278–1286, 2014.

[139] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[140] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022.

[141] Mohammad Rostami, Soheil Kolouri, Praveen K Pilly, and James McClelland. Generative continual concept learning. In *Proc. AAAI Conf. on Artificial Intelligence*, pages 5545–5552, 2020.

[142] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, pages 1–9, 2016.

[143] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 2234–2242, 2016.

[144] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 2234–2242, 2016.

[145] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *Proc. Int Conf. on Machine Learning (ICML)*, volume 80, pages 4535–4544. PMLR, 2018.

[146] Ari Seff, Alex Beatson, Daniel Suo, and Han Liu. Continual learning in generative adversarial nets. *arXiv preprint arXiv:1705.08395*, 2017.

[147] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision.*, 128(2):336–359, 2020.

[148] Huajie Shao, Shuochao Yao, Dachun Sun, Aston Zhang, Shengzhong Liu, Dongxin Liu, Jun Wang, and Tarek Abdelzaher. Controlvae: Controllable variational autoencoder. In *Proc. Int Conf. on Machine Learning (ICML)*, volume 119, pages 8655–8664. PMLR, 2020.

[149] Christian Simon, Piotr Koniusz, and Mehrtash Harandi. On learning the geodesic path for incremental learning. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1591–1600, 2021.

[150] Alex J Smola, SVN Vishwanathan, and Eleazar Eskin. Laplace propagation. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 441–448, 2004.

[151] Artem Sobolev and Dmitry Vetrov. Importance weighted hierarchical variational inference. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, volume 32, pages 601–613, 2019.

[152] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proc. Int Conf. on Machine Learning (ICML)*, volume 37, pages 2256–2265. JMLR, 2015.

[153] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 3483–3491, 2015.

[154] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2021.

[155] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2021.

[156] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 3308–3318, 2017.

[157] Shengyang Sun, Daniele Calandriello, Huiyi Hu, Ang Li, and Michalis Titsias. Information-theoretic online memory selection for continual learning. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2022.

[158] Siddharth Swaroop, Cuong V Nguyen, Thang D Bui, and Richard E Turner. Improving and understanding variational continual learning. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS), workshop*, pages 1–9, 2018.

[159] Hiroshi Takahashi, Tomoharu Iwata, Yuki Yamanaka, Masanori Yamada, and Satoshi Yagi. Variational autoencoder with implicit optimal priors. In *Proc. AAAI Conf. on Artificial Intelligence*, volume 33, pages 5066–5073, 2019.

[160] Jeff Tang. *Intelligent Mobile Projects with TensorFlow: Build 10+ Artificial Intelligence Apps Using TensorFlow Mobile and Lite for IOS, Android, and Raspberry Pi*. Packt Publishing Ltd, 2018.

[161] Shixiang Tang, Dapeng Chen, Jinguo Zhu, Shijie Yu, and Wanli Ouyang. Layerwise optimization by gradient decomposition for continual learning. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 9634–9643, 2021.

[162] Michalis K Titsias, Jonathan Schwarz, Alexander G de G Matthews, Razvan Pascanu, and Yee Whye Teh. Functional regularisation for continual learning with Gaussian processes. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2019.

[163] Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, pages 1–9, 2019.

[164] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[165] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[166] Liyuan Wang, Xingxing Zhang, Kuo Yang, Longhui Yu, Chongxuan Li, Lanqing HONG, Shifeng Zhang, Zhenguo Li, Yi Zhong, and Jun Zhu. Memory replay with data compression for continual learning. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2022.

[167] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 184–193, 2021.

[168] X. Wang, R. Zhang, Y. Sun, and J. Qi. KDGAN: knowledge distillation with generative adversarial networks. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 775–786, 2018.

[169] Yisen Wang, Weiyang Liu, Xingjun Ma, James Bailey, Hongyuan Zha, Le Song, and Shu-Tao Xia. Iterative learning with open-set noisy labels. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 8688–8696, 2018.

[170] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[171] Yu-Chun Wang, Chien-Yi Wang, and Shang-Hong Lai. Disentangled representation with dual-stage feature learning for face anti-spoofing. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1955–1964, 2022.

[172] Zhen Wang, Liu Liu, Yiqun Duan, Yajing Kong, and Dacheng Tao. Continual learning with lifelong vision transformer. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 171–181, 2022.

[173] Zhenyi Wang, Li Shen, Le Fang, Qiuling Suo, Tiehang Duan, and Mingchen Gao. Improving task-free continual learning by distributionally robust memory evolution. In *Proc. Int Conf. on Machine Learning (ICML)*, volume 162, pages 22985–22998. PMLR, 2022.

[174] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2020.

[175] Chenshen Wu, Luis Herranz, Xialei Liu, Joost van de Weijer, and Bogdan Raducanu. Memory replay GANs: Learning to generate new categories without forgetting. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 5962–5972, 2018.

[176] C. Wu, L. Herranz, X. Liu, J. van de Weijer, and B. Raducanu. Memory replay GANs: Learning to generate new categories without forgetting. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 5962–5972, 2018.

[177] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, pages 1–9, 2017.

[178] Mengqi Xue, Haofei Zhang, Jie Song, and Mingli Song. Meta-attention for vit-backed continual learning. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 150–159, 2022.

[179] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3014–3023, 2021.

[180] Fei Ye and Adrian Bors. Lifelong teacher-student network learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[181] Fei Ye and Adrian G. Bors. Learning latent representations across multiple data domains using lifelong VAEGAN. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 777–795, 2020.

[182] Fei Ye and Adrian G. Bors. Deep mixture generative autoencoders. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2021.

[183] Fei Ye and Adrian G Bors. Learning joint latent representations based on information maximization. *Information Sciences*, 567:216–236, 2021.

[184] Fei Ye and Adrian G Bors. Lifelong generative modelling using dynamic expansion graph model. *arXiv preprint arXiv:2112.08370*, pages 1–9, 2021.

[185] Fei Ye and Adrian G. Bors. Lifelong mixture of variational autoencoders. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2021.

[186] Fei Ye and Adrian G Bors. Dynamic self-supervised teacher-student network learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5731–5748, 2022.

[187] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 7017–7025, 2019.

[188] Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2022.

[189] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 3320–3328, 2014.

[190] Aron Yu and Kristen Grauman. Semantic jitter: Dense supervision for visual comparisons via synthetic images. In *Proc. of IEEE Int. Conf. on Computer Vision (ICCV)*, pages 5571–5580, 2017.

[191] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *Proc. of Int. Conf. on Machine Learning (ICML)*, volume 70, pages 3987–3995. PMLR, 2017.

[192] Mengyao Zhai, Lei Chen, Jiawei He, Megha Nawhal, Frederick Tung, and Greg Mori. Piggyback gan: Efficient lifelong learning for image conditioned generation. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 397–413. Springer, 2020.

[193] M. Zhai, L. Chen, F. Tung, J He, M. Nawhal, and G. Mori. Lifelong GAN: Continual learning for conditional image generation. *arXiv preprint arXiv:1907.10107*, pages 1–9, 2019.

[194] Mengyao Zhai, Lei Chen, Frederick Tung, Jiawei He, Megha Nawhal, and Greg Mori. Lifelong gan: Continual learning for conditional image generation. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, pages 2759–2768, 2019.

[195] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2018.

[196] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 13205–13214, 2020.

[197] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. In *Proc. Int. Conf. on Learning Representations (ICLR)*, pages 1–9, 2017.

[198] Miaoyun Zhao, Yulai Cong, and Lawrence Carin. On leveraging pretrained gans for generation with limited data. In *Proc. Int Conf. on Machine Learning (ICML)*, volume 119, pages 11340–11351. PMLR, 2020.

[199] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Multimodal image-to-image translation by enforcing bi-cycle consistency. In *Proc. Advances Neural Inf. Proc. Systems (NeurIPS)*, pages 465–476, 2017.