Appendix for Lifelong Compression Mixture Model via Knowledge Relationship Graph

December 1, 2022

Contents

A Additional information for the discard mechanism

We have discussed the implementation of the proposed mixture component discarding mechanism, which is explained in Section 3.4 of the paper. In this Appendix we introduce two approaches that implement the proposed component discarding mechanism into a greedy algorithm framework which removes successively additional components from the DEM's architecture

First, given a DEM $\mathcal{G} = \{S_1, \cdots, S_t\}$ that has already trained t components, we calculate the graph relationship matrix $\mathbf{E}$ for these components using the FID score. Then we define a limit $n \in [1, 100]$ on the desired number of the mixture components to be retained after the discarding procedure. A greedy algorithm is implemented for discarding components from DEM, described as follows :

Step 1 (Check the number of components). If $|\mathcal{G}| > n$, then we perform the second step, otherwise, we quit the component discarding procedure, where $|\mathcal{G}|$ denotes the number of components in $\mathcal{G}$.

Step 2 (Discard the component). We select a pair of components according to :

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

where $c^\star$ and $g^\star$ are the indices of the selected components. Then we remove one component, according to the diversity evaluation (Eq. (15) and Eq. (16) from the the paper). Once the selected component is removed from DEM's architecture, we would also remove all edge values of the removed component from $\mathbf{E}$ and repeat this procedure. We call this first approach as "discard".

In the following, we introduce a second approach for implementing the component discarding procedure into a greedy algorithm framework. The approach consists of two steps, expressed as follows :

Step 1 (Check the discard process). First, we calculate the maximal edge value of $\mathbf{E}$ as :

$$E^\star = \max_{c,g=1,\cdots,t} E_{(c,g)}. \tag{2}$$

Then if $E^\star > \lambda_2$, where threshold $\lambda_2 \in [0, 3]$ is used to control the condition when components are discarded from the DEM mixture and then we proceed the second step. Otherwise, we quit the discarding procedure.

Step 2 (Discarding the component). We select a pair of components according to Eq. (1). Then we remove one of them according the diversity evaluation (Eq. (15) and Eq. (16)

Algorithm 1 Algorithm for "DivSS + Discard"

(**Input**:The data stream);
for $i < n$ **do**
  for $index < batchCount$ **do**
    $\mathcal{D}_i \sim \mathcal{D}$
    if $|\mathcal{M}_i| < |\mathcal{M}_i|^{Max}$ **then**
      $\mathcal{M}_i = \mathcal{M}_i \cup \mathcal{D}_i$
    end if
    Train S_t on $\mathcal{M}_i$
  end for
  Check the model's expansion
  if $|\mathcal{M}_i| > |\mathcal{M}_i|^{Max}$ **then**
    Estimate the MMD using Eq.(8) of the paper
    if $\min\left\{\mathcal{L}^e_{\mathrm{MMD}^2}\left(\mathbb{P}_{\tilde{\mathbf{z}}_1}, \mathbb{P}_{\mathbf{z}_{\mathcal{M}_c}}\right), \cdots, \mathcal{L}^e_{\mathrm{MMD}^2}\left(\mathbb{P}_{\tilde{\mathbf{z}}_{t-1}}, \mathbb{P}_{\mathbf{z}_{\mathcal{M}_c}}\right)\right\} \geq \lambda$ **then**
      Build a new component $\mathcal{G} = \mathcal{G} \bigcup S_{t+1}$
    else
      Perform the sample selection
      Calculate $\mathcal{L}_d(\mathbf{x}_s) = \frac{1}{t-1}\sum_{j=1}^{t-1}\Big\{ - \mathcal{L}_V(S_j, \mathbf{x}_s)\Big\}$
      $\mathbf{x}'_s = \arg\min_{\mathbf{x}_s \in \mathcal{M}_i}(\mathcal{L}_d(\mathbf{x}_s)$
      We repeat recursively the exclusion of $\mathbf{x}'_s$'es until $|\mathcal{M}_i| \leq |\mathcal{M}_i|^{Max}$
    end if
  end if
end for
Perform the discard mechanism
Calculate the matrix e using FID criterion
for $j < k$, (k is the number of the desired removed components) **do**
  Search for a pair of components by $c^\star, g^\star = \arg\max_{c,g=1,\cdots,t} E_{(c,g)}$
  Diversity evaluation $\mathcal{L}_{diversity}(S_a) = \frac{1}{t-1}\sum_{j=1}^{t}\left\{\frac{1}{E_{(j,a)})}\right\}, j \neq a$
  Remove one selected component from $\mathcal{G}$
  Remove all edge values of the deleted component from $\mathbf{E}$
end for

from the the paper) and also remove all edge values of the removed component from $\mathbf{E}$.

We call the first and second approaches for implementing the proposed component discarding mechanism as 'discard' and 'discard2'. The advantage of discard2 over discard is that it is not necessary to define the number of remaining components in DEM. We provide the pseudo code of the proposed "DivSS + Discard" in Algorithm 1.

Algorithm 2 Algorithm for "DivSS + Discard2"

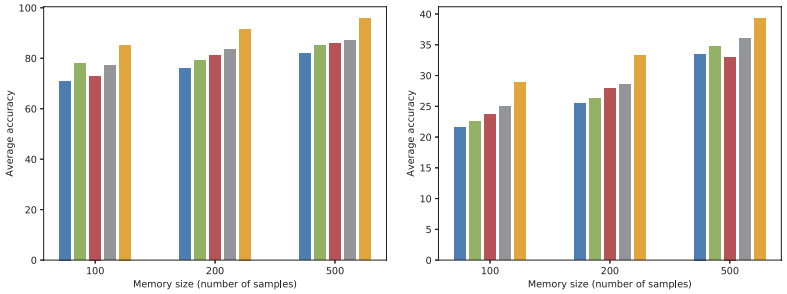
(Input:The data stream);
for $i < n$ **do**
  for $index < batchCount$ **do**
    $\mathcal{D}_i \sim \mathcal{D}$
    if $|\mathcal{M}_i| < |\mathcal{M}_i|^{Max}$ **then**
      $\mathcal{M}_i = \mathcal{M}_i \cup \mathcal{D}_i$
    end if
    Train S_t on $\mathcal{M}_i$
  end for
  Check the model's expansion
  if $|\mathcal{M}_i| > |\mathcal{M}_i|^{Max}$ **then**
    Estimate the MMD using Eq.(8) of the paper
    if $\min\left\{\mathcal{L}^e_{\mathrm{MMD}^2}\left(\mathbb{P}_{\tilde{\mathbf{z}}_1}, \mathbb{P}_{\mathbf{z}_{\mathcal{M}_c}}\right), \cdots, \mathcal{L}^e_{\mathrm{MMD}^2}\left(\mathbb{P}_{\tilde{\mathbf{z}}_{t-1}}, \mathbb{P}_{\mathbf{z}_{\mathcal{M}_c}}\right)\right\} \geq \lambda$ **then**
      Build a new component $\mathcal{G} = \mathcal{G} \bigcup S_{t+1}$
    else
      Perform the sample selection
      Calculate $\mathcal{L}_d(\mathbf{x}_s) = \frac{1}{t-1}\sum_{j=1}^{t-1}\Big\{ - \mathcal{L}_V(S_j, \mathbf{x}_s)\Big\}$
      $\mathbf{x}'_s = \arg\min_{\mathbf{x}_s \in \mathcal{M}_i}(\mathcal{L}_d(\mathbf{x}_s)$
      We repeat recursively the exclusion of $\mathbf{x}'_s$'es until $|\mathcal{M}_i| \leq |\mathcal{M}_i|^{Max}$
    end if
  end if
  Check the discard procedure
  Calculate the matrix $\mathbf{E}$ using FID criterion
  for $j < |\mathcal{G}|$ **do**
    $E^\star = \max_{c,g=1,\cdots,t} E_{(c,g)}$
    if $|\mathcal{G}| == 5$ **then**
      break
    end if
    if $E^\star > \lambda_2$ **then**
      Perform the discard procedure
      Search for a pair of components by $c^\star, g^\star = \arg\max_{c,g=1,\cdots,t} E_{(c,g)}$
      Diversity evaluation $\mathcal{L}_{diversity}(S_a) = \frac{1}{t-1}\sum_{j=1}^{t}\Big\{\frac{1}{E_{(j,a)}}\Big\}, j \neq a$
      Remove one selected component from $\mathcal{G}$
      Remove all edge values of the deleted component from $\mathbf{E}$
    else
      break
    end if
  end for
end for

B Discarding components during the training

The approaches for component discarding mechanisms, described above in Appendix A, can significantly compress the DEM model by selectively removing superfluous components. However, the algorithms 'discard' and 'discard1' are employed after the learning process is finished which is complicated when several data streams have to be learned during the lifelong learning training. In the following, we show how to include discard2 into our framework to compress the DEM model during training.

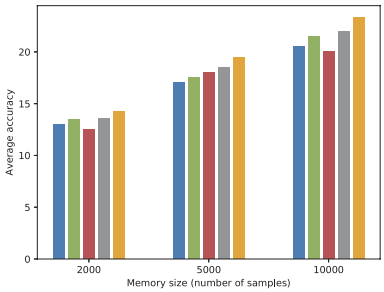
We provide the pseudo code of the proposed "DivSS + Discard2" in Algorithm 2, which can be summarized as five steps :

Step 1 (The training of the current component). We assume that a dynamic expansion model has learnt t components. Then we train the current component S_t on $\mathcal{M}_i$ at $\mathcal{T}_i$

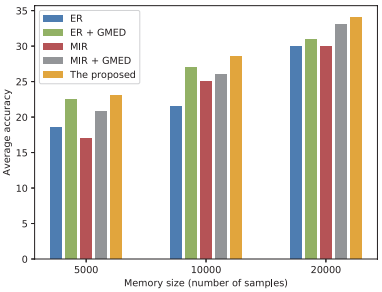


(a) Split MNIST.

(b) Split CIFAR10.



(c) Split CIFAR100.



(d) Split MINI-ImageNet.

Figure 1: The performance change of various models when changing the memory size.

using $\mathcal{L}_C(S_t, \mathcal{M}_i)$ and $\mathcal{L}_V(S_t, \mathcal{M}_i)$ from Eq. (1) and (2) in the paper.

Step 2 (Check the expansion of the model). If the current memory $\mathcal{M}_i$ is full $|\mathcal{M}_i| \geq |\mathcal{M}_i|^{Max}$, then we check the expansion of the model using Eq. (10) of the paper. If the expansion criterion is satisfied, we create a new component S_{t+1} added to $\mathbf{G}$ and clean up $\mathcal{M}_i$ in order to learn samples which are non-overlapping with the existing ones in the memory. We also require to check whether we perform the component discard procedure.

Step 3 (Check the discarding procedure.) If $|G| > 5$, then we calculate the maximal edge value $E^\star$ of $\mathbf{E}$ using Eq. (2). Then if $E^\star > \lambda_2$, we perform the fourth step, otherwise, we quit the discard procedure.

Step 4 (Discarding components). We select a pair of components according to Eq. (1). Then we remove one of them according the diversity evaluation (Eq.(16) of the paper) and also remove all edge values of the removed component from $\mathbf{E}$. Then, we continue with performing Step 3.

Step 5 (Sample selection). If $|\mathcal{M}_i| \geq |\mathcal{M}_i|^{Max}$, then we estimate the sample log-likelihood for each stored sample using Eq.(12) of the paper. We perform the sample

selection for the current memory $\mathcal{M}_i$ using Eq.(13) of the paper.

C Additional results for the experiments

In this section, we provide additional results for the experimental results. We have provided the detailed implementation of the proposed model. We also provide the source code in the supplemental material. In addition, We will organize the source code of the proposed model for the sake of easy understanding and for facilitating the re-implementation and we will release it publicly on https://github.com/ if the paper is accepted.

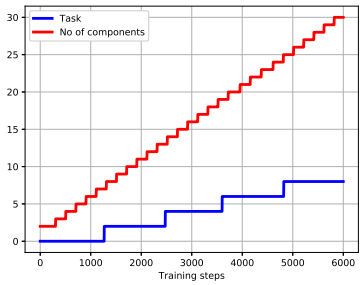
GPU and operating system : We employ the similar experiment setting from Ye and Bors (2022a,b,c, 2021a,b); Aljundi et al. (2019); Ye and Bors (2022d, 2021c, 2020a,b, 2022e, 2021d,e, 2020c).

C.1 The performance of the model when changing the memory buffer size

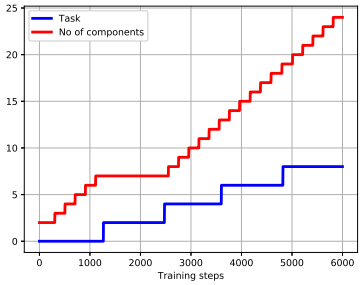
In Fig. 1 We provide the classification accuracy of the proposed approach "DivSS + Discard" with different memory configurations. These results show that the proposed approach outperforms other baselines on four datasets, Split MNIST, Split CIFA10, Split CIFAR100 and Split MINI-ImageNet with different memory buffer sizes.

C.2 The change of the model's complexity during the training

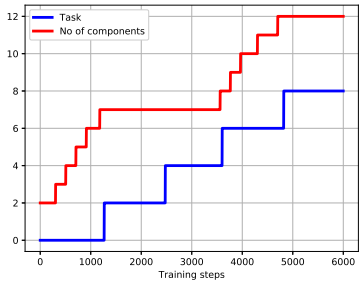
In this section we examine the change in the number of components for the proposed approach during training. We train "DivSS" under split MNIST, and record the number of components and the number of tasks seen in each training step. We plot the results in Fig. 2, where we can observe that a large threshold λ encourages the proposed model to use fewer components, resulting that a single component would learn two tasks. A small threshold λ, on the other hand, leads to training more components while multiple components model sections of a single task (See also the relationship matrix in Fig. 3a of the paper). With the proposed discard mechanism, we can compress the proposed model significantly by selectively removing unnecessary components (See results in Fig. 5b of the paper).



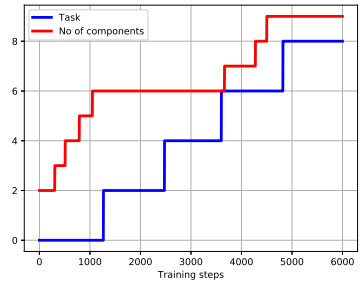
(a) $\lambda = 0.009$.



(b) $\lambda = 0.013$.



(c) $\lambda = 0.015$.



(d) $\lambda = 0.022$.

Figure 2: The number of components used by the proposed approach when increasing the number of given tasks.

C.3 Discarding the components during the training

In this section we examine the effectiveness of the proposed 'Discard2' used in our system. We train the proposed "DivSS + Discard2" under split MNIST, split CIFAR10 and split CIFAR100. The threshold values for λ_2 used for Split MNIST, Split CIFAR10 and Split CIFAR100 are 0.05, 0.03 and 0.06, respectively. The results can be found in Table 1, where the proposed "DivSS + Discard2" achieves similar performance compared to "DivSS + Discard". Moreover, the number of components of the proposed model with 'Dsicard2' for Split MNIST, Split CIFAR10 and Split CIFAR100 are 12, 9 and 8, respectively, showing that an appropriate λ_2 can enable the proposed model to achieve good performance with a suitable network architecture.

Table 1: Classification accuracy of five independent runs for various models on three datasets. '*' and '†' denote the results cited from De Lange and Tuytelaars (2021) and Jin et al. (2021), respectively.

Methods	**Split MNIST**	**Split CIFAR10**	**Split CIFAR100**
finetune*	19.75 ± 0.05	18.55 ± 0.34	3.53 ± 0.04
GEM*	93.25 ± 0.36	24.13 ± 2.46	11.12 ± 2.48
iCARL*	83.95 ± 0.21	37.32 ± 2.66	10.80 ± 0.37
reservoir*	92.16 ± 0.75	42.48 ± 3.04	19.57 ± 1.79
MIR*	93.20 ± 0.36	42.80 ± 2.22	20.00 ± 0.57
GSS*	92.47 ± 0.92	38.45 ± 1.41	13.10 ± 0.94
CoPE-CE*	91.77 ± 0.87	39.73 ± 2.26	18.33 ± 1.52
CoPE*	93.94 ± 0.20	48.92 ± 1.32	21.62 ± 0.69
ER + GMED†	82.67 ± 1.90	34.84 ± 2.20	20.93 ± 1.60
ER_a + GMED†	82.21 ± 2.90	47.47 ± 3.20	19.60 ± 1.50
CURL*	92.59 ± 0.66	-	-
CNDPM*	93.23 ± 0.09	45.21 ± 0.18	20.10 ± 0.12
DivSS + Discard	96.16 ± 0.11	50.12 ± 0.23	25.24 ± 0.17
DivSS	**96.95** ± 0.13	**53.71** ± 0.19	**26.03** ± 0.16
SW	96.81 ± 0.12	50.91 ± 0.25	25.65 ± 0.16
SW + Discard	95.93 ± 0.15	49.93 ± 0.23	24.18 ± 0.18
DivSS + Discard2	96.36 ± 0.12	50.15 ± 0.21	24.79 ± 0.18

Methods	**Split MNIST**	**Split CIFAR10**	**Split MImageNet**
DivSS + Discard	10	10	6
DivSS	25	32	12

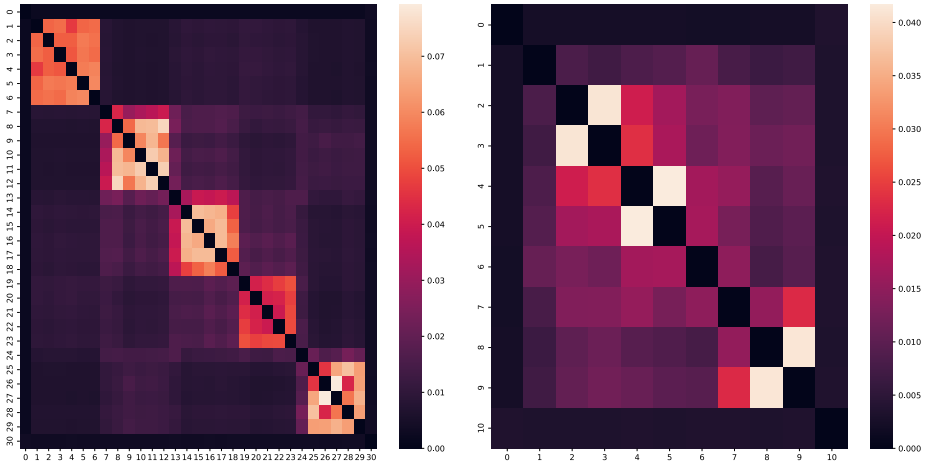
Table 2: The number of components from the proposed model for Split MNIST, Split CIFAR10 and Split MImageNet under the fuzzy task boundaries.

C.4 The number of components

We provide the number of components of the proposed model in Tab. 2, corresponding to the results of Table. 4 of the paper. These results show that by using the proposed discarding mechanism, the proposed model still achieves good performance while maintaining a compact model structure.

C.5 Graph relationship matrix

To investigate how the graph relationship matrix drives the discarding procedure, we train the proposed model under Split MNIST and present the matrix $\mathbf{E}$ in Figs. 3-a and 3-b before and after applying the component discarding procedure, respectively. In Fig. 3-a the mixture components are grouped into five clusters. The edges between the components from each cluster tend to have large values (light colour) and thus learn similar knowledge. After the component discard procedure, we have significantly



(a) (b)

Figure 3: The representation of the graph relationship matrix $\mathbf{E}$. (a) : The initial relationships. (b) : After compressing the model.

reduced the total number of components from 30 to 10, while we can observe from Figs. 3-(b) that the number of components in each cluster was reduced as well. This shows that the proposed component discarding mechanism can compress the model while preserving the diversity of information.

C.6 The performance when changing the batch size

In this section, we investigate whether changing the batch size can significantly affect the performance of the proposed model. We consider to use the different batch size configurations for training the proposed model under Split MNIST and the results are reported in Fig. 4. It observes that the proposed model does not change its performance too much when varying the batch size.

C.7 The computational cost

In this section, we compare the proposed DivSS + Discard with Dynamic-OCM in terms of computational costs. The training time (minute) of various models is reported in Tab. 3. It observes that the proposed model still requires less training time than Dynamic-OCM.

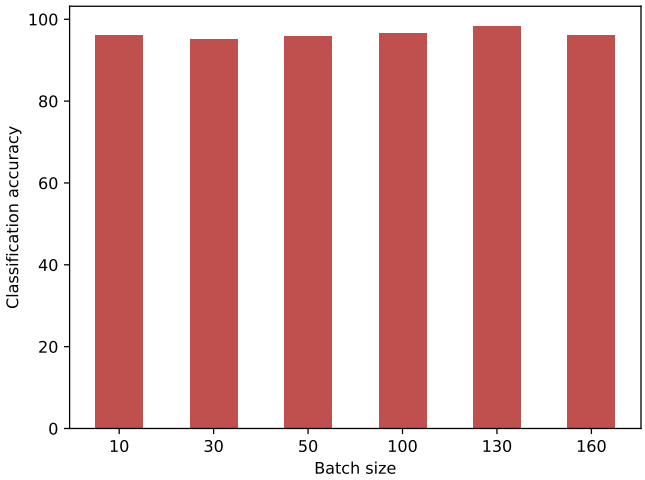


Figure 4: The performance of the proposed model on Split MNIST when changing the batch size.

Methods	**Split MNIST**	**Split CIFAR10**	**Split CIFAR100**
Dynamic-OCM	10.2	42.3	47.8
DivSS + Discard	8.9	39.6	45.2

Table 3: The training time of various models.

C.8 Comparison to the other sample selection approach

In this section, we investigate whether the proposed sample selection approach can improve the performance of the proposed DEM model when compared with the other sample selection approach. We replace the proposed sample selection approach by employing the existing sample selection approach, resulting in MMD-CoPE, MMD-MIR, and MMD-reservoir. We report the classification results in Tab. 4. It observes that the proposed sample selection approach outperforms the other existing sample selection methods on all datasets, demonstrating that the proposed sample selection can further improve the performance of the proposed DEM model when compared with the other sample selection methods.

References

Fei Ye and Adrian G Bors. Continual variational autoencoder learning via online cooperative memorization. In *European Conference on Computer Vision*, pages 531–549.

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

Table 4: Assessing the proposed data selection in the proposed model.

Springer, 2022a. 6

Fei Ye and Adrian G Bors. Task-free continual learning via online discrepancy distance learning. *arXiv preprint arXiv:2210.06579*, 2022b. 6

Fei Ye and Adrian G Bors. Dynamic self-supervised teacher-student network learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022c. 6

Fei Ye and Adrian G. Bors. Lifelong twin generative adversarial networks. In *Proc. IEEE Int. Conf. on Image Processing (ICIP)*, pages 1289–1293, 2021a. 6

Fei Ye and Adrian G. Bors. Lifelong mixture of variational autoencoders. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2021b. doi: 10.1109/TNNLS.2021.3096457. 6

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proc. of IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 11254–11263, 2019. 6

Fei Ye and Adrian G Bors. Learning an evolved mixture model for task-free continual learning. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 1936–1940. IEEE, 2022d. 6

Fei Ye and Adrian G. Bors. Lifelong teacher-student network learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021c. 6

Fei Ye and Adrian G. Bors. Learning latent representations across multiple data domains using lifelong VAEGAN. In *Proc. European Conf. on Computer Vision (ECCV), vol. LNCS 12365*, pages 777–795, 2020a. 6

Fei Ye and Adrian G. Bors. Lifelong learning of interpretable image representations. In *Proc. Int. Conf. on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6, 2020b. 6

Fei Ye and Adrian G Bors. Lifelong generative modelling using dynamic expansion graph model. In *AAAI on Artificial Intelligence*. AAAI Press, 2022e. 6

Fei Ye and Adrian G. Bors. Deep mixture generative autoencoders. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2021d. doi: 10.1109/TNNLS.2021.3071401. 6

Fei Ye and Adrian G Bors. Lifelong infinite mixture model based on knowledge-driven dirichlet process. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10695–10704, 2021e. 6

Fei Ye and Adrian G Bors. Mixtures of variational autoencoders. In *2020 Tenth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6. IEEE, 2020c. 6

Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *Proc. of the IEEE/CVF International Conference on Computer Vision*, pages 8250–8259, 2021. 8

Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient-based editing of memory examples for online task-free continual learning. In *Advances in Neural Information Processing Systems (NeurIPS), arXiv preprint arXiv:2006.15294*, 2021. 8