Appendix for Learning Dynamic Latent Spaces for Task-Free Continual Generative Modelling

December 1, 2022

Contents

A Additional information for the proposed ORVAE

We provide the pseudo-code for ORVAE in Algorithm 1, which can be summarized into five steps including the testing phase :

Step 1 (Updating the memory): We add a new data batch $\mathbf{X}_{batch}^{t+1}$ to the memory buffer $\mathcal{S}_t$, resulting in $\mathcal{S}_{(t+1)}$. If the memory buffer is overloaded, we randomly remove the samples from $\mathcal{S}_{t+1}$ until its size is equal to $|\mathcal{S}_{(t+1)}|_{max}$.

Step 2 (Checking expansion): If $|\mathcal{S}_{(t+1)}| = |\mathcal{S}_{(t+1)}|_{max}$, then we evaluate the novelty of the incoming batch of samples $\mathbf{X}_{batch}^{t+1}$. We use a measure d^t, representing the absolute difference when evaluating the objective function from Eq.(9) of the paper, calculated by the current model, on the memorized samples from $\mathcal{S}_t$ and the new batch of data $\mathbf{X}_{batch}^{t+1}$:

$$\begin{aligned} d^t = \big| & \mathbb{E}_{\mathbf{x} \sim \mathcal{S}_t} [\mathcal{L}_{\mathrm{ORVAE}}(\mathbf{x}; \theta_K^t, \omega_K^t)] \\ & - \mathbb{E}_{\mathbf{x} \sim \mathbf{X}_{batch}^{t+1}} [\mathcal{L}_{\mathrm{ORVAE}}(\mathbf{x}; \theta_K^t, \omega_K^t)] \big| \,, \end{aligned} \tag{1}$$

where $\{\theta_K^t, \omega_K^t\}$ are the parameters of the K-th mixture model trained at the training step $\mathcal{T}_t$.

Step 3 (Expansion): If $d^t > \lambda$, where λ is a threshold defined empirically, ORVAE builds a new mixture component $\mathcal{M}^{(K+1)}$, in a recursive way, using Eq.(7) and Eq.(5) of the paper, while $\mathcal{S}_{(t+1)}$ is the set to contain $\mathbf{X}_{batch}^{t+1}$ only at $\mathcal{T}_{(t+1)}$ in order to learn novel samples, otherwise, we randomly remove samples from $\mathcal{S}_{(t+1)}$ until its size small than $|\mathcal{S}_{(t+1)}|_{max}$.

Step 4 (Learning): We train ORVAE and attention parameters are updated based on $\mathcal{S}_{(t+1)}$ using Eq.(9) of the paper.

Step 5 (Evaluation): For a given sample, we estimate the sample log-likelihood by using each component of OVAE. We then choose a component with the highest sample log-likelihood for evaluation.

B The proof of Theorem 1

In this section, we show that the proposed loss function used for training ORVAE is a strict lower bound on the marginal log-likelihood. As shown in Fig.2 of the paper, the base decoder is given several latent variables drawn from the previously learned distributions $\{Q_{\omega_1}(\mathbf{z}), Q_{\omega_2}(\mathbf{z}), \cdots, Q_{\omega_k}(\mathbf{z})\}$, and the outputs of the base decoder are combined to be an input to the component-specific decoder for reconstruction. This

Algorithm 1 The training algorithm for ORVAE

```
Input:$\mathcal{D}^S$ (Training dataset);
for $t < trainingSteps$ do
    Updating the memory
    $\mathcal{S}_t = \mathcal{S}_{(t-1)} \bigcup \mathbf{X}^t_{batch}$ Add a new batch of images
    if $|\mathcal{S}_{(t)}| > |\mathcal{S}_{(t)}|_{max}$ then
        Randomly remove samples from $\mathcal{S}_t$
    end if
    The process of the dynamic memory
    if $|\mathcal{S}_{(t)}| > |\mathcal{S}_{(t)}|_{max}$ then
        $d^{t-1} = |\mathbb{E}_{\mathbf{x}\sim\mathcal{S}_{t-1}}[\mathcal{L}_{\text{ORVAE}}(\mathbf{x};\theta_K^{t-1},\omega_K^{t-1})]-\mathbb{E}_{\mathbf{x}\sim\mathbf{X}^t_{batch}}[\mathcal{L}_{\text{ORVAE}}(\mathbf{x};\theta_K^{t-1},\omega_K^{t-1})]|$;
        if $d^{t-1} > \lambda$ then
            Clear the memory $\mathcal{S}_t$
            Add $\mathbf{X}^t_{batch}$ in $\mathcal{S}_t$
            Build a new component in ORVAE
        else
            Randomly remove samples frome $\mathcal{S}_t$
        end if
    end if
    Train ORVAE to fit the memory
    for $i < batchCount$ do
        $\mathbf{X} \sim \mathcal{S}_t$
        Train ORVAE with $\mathbf{X}$ using Eq. (9) of the paper
    end for
end for
```

process can reuse all the previously learned information in the inference and decoding processes to learn a novel sample. Before proving Theorem 1, we first make the following assumption.

Assumption 1 *In learning a certain component (K-th component), we assume that we treat the base decoder in conjunction with the component-specific decoder "Decoder3" as a single decoder for the K-th component. It notes that the representation $\mathbf{X}^K_S$ involves the information of all previously learnt variables $\{\mathbf{z}_1, \cdots, \mathbf{z}_{K-1}\}$. We assume that the base decoder only receives a single latent variable $\mathbf{z}_K$ since $\mathbf{z}_K$ has already involved $\{\mathbf{z}_1, \cdots, \mathbf{z}_{K-1}\}$. Under this assumption, we only have a single variational distribution $Q_{\omega_K}(\mathbf{z})$ for the K-th component.*

Theorem 1 *Based on Assumption 1, the loss function (Eq.(9) of the paper) is a lower* *bound to the marginal log-likelihood for a given model* $\mathcal{M}^K$*,*

$$\log p_{\theta^\star_K}(\mathbf{x}) \geq \mathcal{L}_{\mathrm{ORVAE}}(\mathbf{x};\theta^\star_K,\omega^\star_K) \tag{2}$$

Proof. Based on Assumption 1, we consider only a single variational distribution $Q_{\omega^\star_K}(\mathbf{z})$ when deriving ELBO. Firstly, we consider the KL divergence between the variational distribution $Q_{\omega^\star_K}(\mathbf{z})$ and posterior $p(\mathbf{z} \mid \mathbf{x})$ Doersch (2016) :

$$D_{KL}[Q_{\omega^\star_K}(\mathbf{z}) \mid\mid p(\mathbf{z} \mid \mathbf{x})] = \mathbb{E}_{\mathbf{z}\sim Q_{\omega^\star_K}(\mathbf{z})}[\log Q_{\omega^\star_K}(\mathbf{z}) - \log p(\mathbf{z} \mid \mathbf{x})] \tag{3}$$

By applying Bayes rule to $p(\mathbf{z} \mid \mathbf{x})$, Eq. (3) is rewritten as :

$$\begin{aligned} D_{KL}[Q_{\omega^\star_K}(\mathbf{z}) \mid\mid p(\mathbf{z} \mid \mathbf{x})] &= \mathbb{E}_{\mathbf{z}\sim Q_{\omega^\star_K}(\mathbf{z})}[\log Q_{\omega^\star_K}(\mathbf{z}) - \log p(\mathbf{x} \mid \mathbf{z}) - \log p(\mathbf{z})] \\ &\quad + \log p(\mathbf{x}) \end{aligned} \tag{4}$$

We then rearrange Eq. (4), resulting in :

$$\begin{aligned} \log p(\mathbf{x}) - D_{KL}[Q_{\omega^\star_K}(\mathbf{z}) \mid\mid p(\mathbf{z} \mid \mathbf{x})] &= \mathbb{E}_{\mathbf{z}\sim Q_{\omega^\star_K}(\mathbf{z})}[\log p(\mathbf{x} \mid \mathbf{z})] \\ &\quad - D_{KL}[Q_{\omega^\star_K}(\mathbf{z}) \mid\mid p(\mathbf{z})] \end{aligned} \tag{5}$$

where RHS of Eq. (5) is called Evidence Lower Bound (ELBO). Since the variational distribution $Q_{\omega^\star_K}(\mathbf{z})$ is the mixture distribution, it has the density function form :

$$q_{\omega^\star_K}(\mathbf{z}) = \sum_{i=1}^{K-1} \left\{\pi_{(i,K)} q_{\omega^\star_i}(\mathbf{z})\right\} + \pi_{(K,K)} q'_{\omega^\star_K}(\mathbf{z}) \tag{6}$$

where we use $q'_{\omega^\star_K}(\mathbf{z})$ represent the density for the distribution $q_{\omega_K}(\mathbf{z} \mid \mathbf{z}_S) q_{\omega_S}(\mathbf{z}_S \mid \mathbf{x})$

Then $D_{KL}[Q_{\omega^\star_K}(\mathbf{z}) \mid\mid p(\mathbf{z})]$ can be expressed by the expectation form :

$$\begin{aligned} D_{KL}[Q_{\omega^\star_K}(\mathbf{z}) \mid\mid p(\mathbf{z})] &= \int \left\{ \sum_{i=1}^{K-1} \left\{\pi_{(i,K)} q_{\omega^\star_i}(\mathbf{z})\right\} + \pi_{(K,K)} q'_{\omega^\star_K}(\mathbf{z}) \right\} \log \frac{q_{\omega^\star_K}(\mathbf{z})}{p(\mathbf{z})} \mathrm{d}\mathbf{z} \\ &= \sum_{i=1}^{K-1} \left\{ \int \pi_{(i,K)} q_{\omega^\star_i}(\mathbf{z}) \log \frac{q_{\omega^\star_K}(\mathbf{z})}{p(\mathbf{z})} \mathrm{d}\mathbf{z} \right\} \\ &\quad + \int \pi_{(K,K)} q'_{\omega^\star_K}(\mathbf{z}) \log \frac{q_{\omega^\star_K}(\mathbf{z})}{p(\mathbf{z})} \mathrm{d}\mathbf{z} \end{aligned} \tag{7}$$

We then add $q_{\omega_i^\star}(\mathbf{z})/q_{\omega_i^\star}(\mathbf{z})$ in the first term of RHS and add $q'_{\omega_K^\star}(\mathbf{z})/q'_{\omega_K^\star}(\mathbf{z})$ in the second term of RHS in Eq. (7), resulting in :

$$\begin{aligned}
D_{KL}[Q_{\omega_K^\star}(\mathbf{z}) \mid\mid p(\mathbf{z})] &= \sum_{i=1}^{K-1} \left\{ \int \pi_{(i,K)} q_{\omega_i^\star}(\mathbf{z}) \log \frac{q_{\omega_K^\star}(\mathbf{z})}{p(\mathbf{z})} \frac{q_{\omega_i^\star}(\mathbf{z})}{q_{\omega_i^\star}(\mathbf{z})} \mathrm{d}\mathbf{z} \right\} \\
&+ \int \pi_{(K,K)} q'_{\omega_K^\star}(\mathbf{z}) \log \frac{q_{\omega_K^\star}(\mathbf{z})}{p(\mathbf{z})} \frac{q'_{\omega_K^\star}(\mathbf{z})}{q'_{\omega_K^\star}(\mathbf{z})} \mathrm{d}\mathbf{z} \\
&= \sum_{i=1}^{K-1} \left\{ \int \pi_{(i,K)} q_{\omega_i^\star}(\mathbf{z}) \log \frac{q_{\omega_i^\star}(\mathbf{z})}{p(\mathbf{z})} \mathrm{d}\mathbf{z} + \int \pi_{(i,K)} q_{\omega_i^\star}(\mathbf{z}) \log \frac{q_{\omega_K^\star}(\mathbf{z})}{q_{\omega_i^\star}(\mathbf{z})} \mathrm{d}\mathbf{z} \right\} \\
&+ \int \pi_{(K,K)} q'_{\omega_K^\star}(\mathbf{z}) \log \frac{q‘_{\omega_K^\star}(\mathbf{z})}{p(\mathbf{z})} \mathrm{d}\mathbf{z} + \int \pi_{(K,K)} q'_{\omega_K^\star}(\mathbf{z}) \log \frac{q_{\omega_K^\star}(\mathbf{z})}{q'_{\omega_K^\star}(\mathbf{z})} \mathrm{d}\mathbf{z}
\end{aligned} \tag{8}$$

Then Eq. (8) can be expressed by the KL divergence form :

$$\begin{aligned}
D_{KL}[Q_{\omega_K^\star}(\mathbf{z}) \mid\mid p(\mathbf{z})] = &\sum_{i=1}^{K-1} \left\{ \pi_{(i,K)} D_{KL} \left[Q_{\omega_i^\star}(\mathbf{z}) \mid\mid p(\mathbf{z}) \right] \right\} \\
- &\sum_{i=1}^{K-1} \left\{ \pi_{(i,K)} D_{KL} \left[Q_{\omega_i^\star}(\mathbf{z}) \mid\mid Q_{\omega_K^\star}(\mathbf{z}) \right] \right\} \\
+ &\, \pi_{(K,K)} D_{KL} \left[Q'_{\omega_K^\star}(\mathbf{z}) \mid\mid p(\mathbf{z}) \right] \\
- &\, \pi_{(K,K)} D_{KL} \left[Q'_{\omega_K^\star}(\mathbf{z}) \mid\mid Q_{\omega_K^\star}(\mathbf{z}) \right]
\end{aligned} \tag{9}$$

We replace the expression of $D_{KL}[Q_{\omega_K^\star}(\mathbf{z}) \mid\mid p(\mathbf{z})]$ from Eq. (9) into Eq. (4), resulting in :

$$\begin{aligned}
\log p(\mathbf{x}) - D_{KL}[Q_{\omega_K^\star}(\mathbf{z}) \mid\mid p(\mathbf{z} \mid \mathbf{x})] = &\, \mathbb{E}_{\mathbf{z} \sim Q_{\omega_K^\star}(\mathbf{z})}[\log p(\mathbf{x} \mid \mathbf{z})] \\
- &\sum_{i=1}^{K-1} \left\{ \pi_{(i,K)} D_{KL} \left[Q_{\omega_i^\star}(\mathbf{z}) \mid\mid p(\mathbf{z}) \right] \right\} \\
+ &\sum_{i=1}^{K-1} \left\{ \pi_{(i,K)} D_{KL} \left[Q_{\omega_i^\star}(\mathbf{z}) \mid\mid Q_{\omega_K^\star}(\mathbf{z}) \right] \right\} \\
- &\, \pi_{(K,K)} D_{KL} \left[Q'_{\omega_K^\star}(\mathbf{z}) \mid\mid p(\mathbf{z}) \right] \\
+ &\, \pi_{(K,K)} D_{KL} \left[Q'_{\omega_K^\star}(\mathbf{z}) \mid\mid Q_{\omega_K^\star}(\mathbf{z}) \right]
\end{aligned} \tag{10}$$

We rearrange Eq. (10), resulting in :

$$\begin{aligned} \log p(\mathbf{x}) - D_{KL}[Q_{\omega_K^\star}(\mathbf{z}) \mid\mid p(\mathbf{z} \mid \mathbf{x})] - \sum_{i=1}^{K-1} &\left\{ \pi_{(i,K)} D_{KL}\left[Q_{\omega_i^\star}(\mathbf{z}) \mid\mid Q_{\omega_K^\star}(\mathbf{z})\right]\right\} \\ - \pi_{(K,K)} D_{KL}\left[Q'_{\omega_K^\star}(\mathbf{z}) \mid\mid Q_{\omega_K^\star}(\mathbf{z})\right] &= \mathbb{E}_{\mathbf{z}\sim Q_{\omega_K^\star}(\mathbf{z})}[\log p(\mathbf{x} \mid \mathbf{z})] \\ - \sum_{i=1}^{K-1} &\left\{ \pi_{(i,K)} D_{KL}\left[Q_{\omega_i^\star}(\mathbf{z}) \mid\mid p(\mathbf{z})\right]\right\} - \pi_{(K,K)} D_{KL}\left[Q'_{\omega_K^\star}(\mathbf{z}) \mid\mid p(\mathbf{z})\right] \end{aligned} \tag{11}$$

Since KL divergence term is equal or large than 0, we have :

$$\begin{aligned} \log p(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim Q_{\omega_K^\star}(\mathbf{z})}[\log p(\mathbf{x} \mid \mathbf{z})] - \sum_{i=1}^{K-1} &\left\{ \pi_{(i,K)} D_{KL}\left[Q_{\omega_i^\star}(\mathbf{z}) \mid\mid p(\mathbf{z})\right]\right\} \\ - \pi_{(K,K)} D_{KL}\left[Q'_{\omega_K^\star}(\mathbf{z}) \mid\mid p(\mathbf{z})\right] & \end{aligned} \tag{12}$$

It notes that each augmented variational distribution $Q_{\omega_i^\star}(\mathbf{z})$ is still a Gaussian distribution.

The gap between $\log p_{\theta_K}(\mathbf{x})$ and $\mathcal{L}_{\text{ORVAE}}(\mathbf{x}; \theta_K, \omega_K)$ is :

$$\begin{aligned} \mathcal{L}_{GAP} = &D_{KL}[Q_{\omega_K^\star}(\mathbf{z}) \mid\mid p(\mathbf{z} \mid \mathbf{x})] \\ &+ \sum_{i=1}^{K-1} \left\{ \pi_{(i,K)} D_{KL}\left[Q_{\omega_i^\star}(\mathbf{z}) \mid\mid Q_{\omega_K^\star}(\mathbf{z})\right]\right\} \\ &+ \pi_{(K,K)} D_{KL}\left[Q'_{\omega_K^\star}(\mathbf{z}) \mid\mid Q_{\omega_K^\star}(\mathbf{z})\right] . \end{aligned} \tag{13}$$

Eq. (12) proves Theorem 1.

C The proof of Theorem 2

In this section, our main goal is to provide insights into the forgetting behaviour of the model under OCL from a probabilistic aspect. Firstly, we demonstrate that the ELBO on the source domain with several negative terms is a strict lower bound on the marginal log-likelihood.

Theorem 2 *Let $p_{\theta^t}(\mathbf{x})$ be a single generative latent variable model trained on $\mathcal{S}_t$ and let $\mathbb{P}_{\theta^t}$ be the generator distribution of this model. We can derive an upper bound to* *the model's log-likelihood as :*

$$\begin{aligned}\mathbb{E}_{\mathbf{x}\sim\tau_i}\big[\log p_{\theta^t}(\mathbf{x})\big] \geq\ & \mathbb{E}_{\mathbf{x}\sim\tau'_t}\big[\log p_{\theta^t}(\mathbf{x})\big] - D_{KL}(\tau_i \,||\, \tau'_t) \\ & - |D_{KL}(\tau'_t \,||\, P_{\theta^t}) - D_{KL}(\tau_i \,||\, P_{\theta^t})| \\ & - \mathbb{E}_{\mathbf{x}\sim\tau'_t}\big[p_{\tau'_t}(\mathbf{x})\log p_{\tau'_t}(\mathbf{x})\big] + \mathbb{E}_{\mathbf{x}\sim\tau_i}\big[p_{\tau_i}(\mathbf{x})\log p_{\tau_i}(\mathbf{x})\big]\end{aligned} \tag{14}$$

We call RHS of Eq. (14) *as* $\mathcal{L}_{OELBO}(\mathbf{x};\theta^t,\omega^t)$*, which can be recovered up to the standard ELBO when the source and target distributions are equal,* $\tau_i = \tau'_t$ *:*

$$\begin{aligned}\mathbb{E}_{\mathbf{x}\sim\tau_i}\big[\log p_{\theta^t}(\mathbf{x})\big] &\geq \mathbb{E}_{\mathbf{x}\sim\tau_i}\big[\mathcal{L}_{ELBO}(\mathbf{x};\theta^t,\omega^t)\big] \\ &= \mathbb{E}_{\mathbf{x}\sim\tau_i}\big[\mathcal{L}_{OELBO}(\mathbf{x};\theta^t,\omega^t)\big]\,.\end{aligned} \tag{15}$$

Eq. (15) *holds since* $C(\tau_i, \tau'_t, \mathbb{P}_{\theta^t})$ *and* $D_{Log}(\tau_i, \tau'_t)$ *are equal to zero for* $\tau_i = \tau'_t$*.*

Proof. Firstly, we consider several KL divergences, $D_{KL}(\tau_i \,||\, \tau'{}_t)$, $D_{KL}(\tau'{}_t \,||\, \mathbb{P}_{\theta^t})$ and $D_{KL}(\tau_i \,||\, \mathbb{P}_{\theta^t})$. And we have :

$$D_{KL}(\tau_i \,||\, \mathbb{P}_{\theta^t}) \leq D_{KL}(\tau_i \,||\, \tau'_t) + |D_{KL}(\tau'_t \,||\, \mathbb{P}_{\theta^t}) - D_{KL}(\tau_i \,||\, \mathbb{P}_{\theta^t})| + D_{KL}(\tau'_t \,||\, \mathbb{P}_{\theta^t}) \tag{16}$$

Eq. (16) holds because the sum of last two terms in RHS is large than LHS while $D_{KL}(\tau_i \,||\, \tau'_t) \geq 0$. In the following, we can rewrite $D_{KL}(\tau_i \,||\, \mathbb{P}_{\theta^t})$ and $D_{KL}(\tau'_t \,||\, \mathbb{P}_{\theta_t})$ as :

$$D_{KL}(\tau_i \,||\, \mathbb{P}_{\theta^t}) = \mathbb{E}_{\tau_i}\big[p_{\tau_i}(\mathbf{x})\log p_{\tau_i}(x)\big] - \mathbb{E}_{\tau_i}\big[\log p_{\theta^t}(x)\big] \tag{17}$$

$$D_{KL}(\tau'_t \,||\, \mathbb{P}_{\theta_t}) = \mathbb{E}_{\mathbf{x}\sim\tau'_t}\big[p_{\tau'_t}(\mathbf{x})\log p_{\tau'{}_t}(\mathbf{x})\big] - \mathbb{E}_{\mathbf{x}\ \sim\tau'_t}\big[\log p_{\theta_t}(\mathbf{x})\big] \tag{18}$$

Then we take Eq. (17) and Eq. (18) into Eq. (16), resulting in :

$$\begin{aligned}&\mathbb{E}_{\tau_i}\big[p_{\tau_i}(\mathbf{x})\log p_{\tau_i}(x)\big] - \mathbb{E}_{\tau_i}\big[\log p_{\theta^t}(x)\big] \leq \mathbb{E}_{\mathbf{x}\sim\tau'_t}\big[p_{\tau'_t}(\mathbf{x})\log p_{\tau'{}_t}(\mathbf{x})\big] \\ &- \mathbb{E}_{\mathbf{x}\ \sim\tau'_t}\big[\log p_{\theta_t}(\mathbf{x})\big] + |D_{KL}(\tau'_t \,||\, \mathbb{P}_{\theta^t}) - D_{KL}(\tau_i \,||\, \mathbb{P}_{\theta^t})| \\ &+ D_{KL}(\tau'_t \,||\, \mathbb{P}_{\theta^t})\end{aligned} \tag{19}$$

We then rearrange Eq. (19), resulting in :

$$\begin{aligned}-\mathbb{E}_{\tau_i}\big[\log p_{\theta^t}(x)\big] \leq\; & \mathbb{E}_{\mathbf{x}\sim\tau'_t}\big[p_{\tau'_t}(\mathbf{x})\log p_{\tau'{}_t}(\mathbf{x})\big] - \mathbb{E}_{\tau_i}\big[p_{\tau_i}(\mathbf{x})\log p_{\tau_i}(x)\big] \\ & - \mathbb{E}_{\mathbf{x}\ \sim\tau'_t}\big[\log p_{\theta_t}(\mathbf{x})\big] + |D_{KL}(\tau'_t \,||\, \mathbb{P}_{\theta^t}) - D_{KL}(\tau_i \,||\, \mathbb{P}_{\theta^t})| \\ & + D_{KL}(\tau'_t \,||\, \mathbb{P}_{\theta^t})\end{aligned} \tag{20}$$

We then multiple -1 for both LHS and RHS of Eq. (20), resulting in :

$$\begin{aligned}\mathbb{E}_{\mathbf{x}\sim\tau_i}\big[\log p_{\theta^t}(\mathbf{x})\big] \geq\; & \mathbb{E}_{\mathbf{x}\sim\tau'_t}\big[\log p_{\theta^t}(\mathbf{x})\big] - D_{KL}(\tau_i \,||\, \tau'_t) \\ & - |D_{KL}(\tau'_t \,||\, P_{\theta^t}) - D_{KL}(\tau_i \,||\, P_{\theta^t})| \\ & - \mathbb{E}_{\mathbf{x}\sim\tau'_t}\big[p_{\tau'_t}(\mathbf{x})\log p_{\tau'_t}(\mathbf{x})\big] + \mathbb{E}_{\mathbf{x}\sim\tau_i}\big[p_{\tau_i}(\mathbf{x})\log p_{\tau_i}(\mathbf{x})\big]\end{aligned} \tag{21}$$

This proves Theorem 2.

D The proof of Lemma 1

Lemma 1 *The OELBO for* $\{\mathbf{X}_1^{(T,c)}, \cdots, \mathbf{X}_{N^c}^{(T,c)}\}$ *at* $\mathcal{T}_t$ *is derived as :*

$$\begin{aligned}\sum_{i=1}^{N^c}\big\{\mathbb{E}_{\mathbf{x}\sim\tau_i}\big[\log p_{\theta^t}(\mathbf{x})\big]\big\} \geq N^c\mathbb{E}_{\mathbf{x}\sim\tau'_t}\big[\mathcal{L}_{ELBO}(\mathbf{x};\theta^t,\omega^t)\big] \\ + \sum_{i=1}^{N^c}\big\{-C(\tau_i,\tau'_t,\mathbb{P}_{\theta^t}) - D_{Log}(\tau'_t,\tau_i)\big\}\,.\end{aligned} \tag{22}$$

Remark. The proof is provided in Appendix-D from SM. We have several observations from **Lemma 1** :

- The generalization performance of a single model on $\{\tau_1, \cdots, \tau_{N^c}\}$ is relying on $\sum_{i=1}^{N^c}\{-C(\tau_i,\tau'_t,\mathbb{P}_{\theta^t}) - D_{Log}(\tau'_t,\tau_i)\}$, the last two terms in the RHS of Eq. (22).
- If the stored samples in $\mathcal{S}_t$ does not capture the underlying data distribution of the i-th target domain, the term $-C(\tau_i,\tau'_t,\mathbb{P}_{\theta^t})$ is decreased, which leads to the large gap between ELBO of τ'_t and the data likelihood of τ_i. In this case, the model forgets the knowledge from the i-th target domain.
- The negative backward transfer also degenerates the performance Lopez-Paz and Ranzato (2017), even if the sample in $\mathcal{S}_t$ capture the underlying distribution for each target set.

Proof. From Theorem 2, we can derive OEBLO for N^c tasks :

$$\begin{aligned}\sum_{i=1}^{N^c}\big\{\mathbb{E}_{\mathbf{x}\sim\tau_i}\big[\log p_{\theta^t}(\mathbf{x})\big]\big\} \geq \sum_{i=1}^{N^c}\big\{&\mathbb{E}_{\mathbf{x}\sim\tau_t'}\big[\mathcal{L}_{ELBO}(\mathbf{x};\theta^t,\omega^t)\big]\\ &- C(\tau_i,\tau_t',\mathbb{P}_{\theta^t}) - D_{Log}(\tau_t',\tau_i)\big\}\end{aligned} \tag{23}$$

where the first term in RHS of Eq. (23) is independent from index i. Therefore, we can rewrite Eq. (23) as :

$$\begin{aligned}&\sum_{i=1}^{N^c}\big\{\mathbb{E}_{\mathbf{x}\sim\tau_i}\big[\log p_{\theta^t}(\mathbf{x})\big]\big\} \geq N^c\mathbb{E}_{\mathbf{x}\sim\tau_t'}\big[\mathcal{L}_{ELBO}(\mathbf{x};\theta^t,\omega^t)\big]\\ &+\sum_{i=1}^{N^c}\big\{- C(\tau_i,\tau_t',\mathbb{P}_{\theta^t}) - D_{Log}(\tau_t',\tau_i)\big\}\end{aligned} \tag{24}$$

This proves Lemma 1.

E The proof of Lemma 2

Lemma 2 *The online ELBO of* $\mathbf{M}^K$ *for multiple target sets* $\{\mathbf{X}_1^{(T,c)},\cdots,\mathbf{X}_{N^c}^{(T,c)}\}$ *is defined as :*

$$\begin{aligned}&\sum_{i=1}^{N^c}\big\{\mathbb{E}_{\mathbf{x}\sim\tau_i}\big[\log p_{\theta}(\mathbf{x})\big]\big\} \geq \sum_{i=1}^{N^c}\Big\{\max_{\mathcal{M}\in\mathbf{M}^K}\big\{- C(\tau_i,\tau',\mathbb{P}_{\theta})\\ &+\mathbb{E}_{\mathbf{x}\sim\tau'}\big[\mathcal{L}_{\mathrm{ORVAE}}(\mathbf{x};\theta,\omega)\big] - D_{Log}(\tau_i,\tau')\big\}\Big\}\,.\end{aligned} \tag{25}$$

where $\{\theta,\omega\}$ *are the parameters of* $\mathcal{M}$ *and* τ' *is the distribution of the memorized samples that* $\mathcal{M}$ *was converged on.* $\mathbb{P}_\theta$ *represents the generator distribution of* $\mathcal{M}$.

Proof. Firstly, according to Theorem 2, we have :

$$\begin{aligned}\mathbb{E}_{\mathbf{x}\sim\tau_i}\big[\log p_{\theta^t}(\mathbf{x})\big] \geq\ &\mathbb{E}_{\mathbf{x}\sim\tau_t'}\big[\log p_{\theta^t}(\mathbf{x})\big] - D_{KL}(\tau_i\ ||\ \tau_t')\\ &- |D_{KL}(\tau_t'\ ||\ P_{\theta^t}) - D_{KL}(\tau_i\ ||\ P_{\theta^t})|\\ &- \mathbb{E}_{\mathbf{x}\sim\tau_t'}\big[p_{\tau_t'}(\mathbf{x})\log p_{\tau_t'}(\mathbf{x})\big] + \mathbb{E}_{\mathbf{x}\sim\tau_i}\big[p_{\tau_i}(\mathbf{x})\log p_{\tau_i}(\mathbf{x})\big]\end{aligned} \tag{26}$$

Since we have know that $\log p_\theta(\mathbf{x}) \geq \mathcal{L}_{\mathrm{ORVAE}}(\mathbf{x};\theta,\omega)$ (See Theorem 1 of the paper), we can rewrite Eq. (26) as :

$$\begin{aligned}\mathbb{E}_{\mathbf{x}\sim\tau_i}\big[\log p_{\theta^t}(\mathbf{x})\big] \geq \mathbb{E}_{\mathbf{x}\sim\tau'_t}\big[\mathcal{L}_{\mathrm{ORVAE}}(\mathbf{x};\theta^t,\omega^t)\big] - D_{KL}(\tau_i \mid\mid \tau'_t) \\ - |D_{KL}(\tau'_t \mid\mid P_{\theta^t}) - D_{KL}(\tau_i \mid\mid P_{\theta^t})| \\ - \mathbb{E}_{\mathbf{x}\sim\tau'_t}\big[p_{\tau'_t}(\mathbf{x})\log p_{\tau'_t}(\mathbf{x})\big] + \mathbb{E}_{\mathbf{x}\sim\tau_i}\big[p_{\tau_i}(\mathbf{x})\log p_{\tau_i}(\mathbf{x})\big]\end{aligned} \tag{27}$$

Eq (27) describes the forgetting behaviour of a single component in ORVAE. Since ORVAE would learn several components during OCL, the best ELBO estimation for a certain target set is usually to choose a component that has the highest likelihood of modelling this target set. We express this process as the selection process :

$$\mathrm{Select}(\tau_i) = \max_{\mathcal{M}\in\mathbf{M}^K} \big\{ - C(\tau_i,\tau',\mathbb{P}_\theta) + \mathbb{E}_{\mathbf{x}\sim\tau'}\big[\mathcal{L}_{\mathrm{ORVAE}}(\mathbf{x};\theta,\omega)\big] - D_{Log}(\tau_i,\tau')\big\}\,, \tag{28}$$

where τ_i is used as the input for the selection function $\mathrm{Select}(\cdot)$ which returns the highest ELBO estimation. For the multiple target sets, we can perform the selection process for each target set, expressed as :

$$\sum_{i=1}^{N^c} \big\{\mathbb{E}_{\mathbf{x}\sim\tau_i}\big[\log p_\theta(\mathbf{x})\big]\big\} \geq \sum_{i=1}^{N^c} \Big\{\mathrm{Select}(\tau_i)\Big\}. \tag{29}$$

This proofs Lemma 2.

F The detailed configuration for the experiment

The release of the code. We have provided the detailed implementation of the proposed ORVAE. We have provided the detailed implementation of the proposed ORVAE model. 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.

Hyperparameter setting and GPUs. For the density estimation task, the SGD optimization algorithm for each model uses Adam Kingma and Ba (2015), with a learning rate of 0.0001 while the other hyperparameters are set to their default values. For the generative modelling task, we consider Adam with a learning rate of 0.00005 for Split CIFAR10 and Split Tiny-ImageNe and 0.0001 for Split MNIST and Split Fashion. The batch size for all tasks is 64 and the training epoch for each training step is 100. Following from Ye and Bors (2022a,b,c, 2021a,b); Aljundi et al. (2019a); Ye and Bors

(2022d, 2021c, 2020a,b, 2022e, 2021d,e, 2020c), the GPU used for the experiments is GeForce GTX 1080. The operating system is Ubuntu 18.04.5.

The network architecture and hyperameters for the density task: We use the network architecture from Burda et al. (2015) where the inference and generator models are implemented by two fully connected layers. Each layer has 200 hidden units. For ORVAE, the shared module in both the inference and generator models is implemented by a single fully connected layer with 200 hidden units. When building a new component in ORVAE, we add a new fully connected layer with 200 hidden units on the top of the shared module of the inference and generator models.

The network architecture and hyperameters for the generative modelling task: We implement the shared encoder by using a fully connected network with three layers of processing units $[2000, 1500, 1000]$, and the component encoder by using a fully connected network with three layers $[600, 300, 200]$. The shared decoder is implemented by a fully connected network with three layers $[200, 300, 600]$ and the component encoder is implemented by a fully connected network with three layers $[1000, 1500, 2000]$. The dimension of the latent variable is 200.

The baseline (ORVAE*) We consider training a VAE model on 120,000 images which have been randomly sampled from ImageNet Krizhevsky et al. (2012). Then we extract $\{\omega_S, \theta_S\}$ from this trained VAE model as the parameter of the shared module in ORVAE. During the online learning, the shared module in ORVAE is frozen and is used to provide fundamental representations for multiple domains.

F.1 Evaluation of generative modelling using other criteria

For the image reconstruction, we use the structural similarity index measure (SSIM) Hore and Ziou (2010) and the Peak-Signal-to-Noise Ratio (PSNR) Hore and Ziou (2010). Additionally, we train a simple classifier on the reconstruction of the training set and calculate the classification accuracy on the testing set, denoted as ”Acc” which is also used for the evaluation of the reconstruction quality.

We aim to evaluate the generative abilities of various models. All models use ELBO with a small weight $\beta = 0.01$ on the KL divergence term to avoid the over-regularized issue. We set the maximum size of the dynamic memory as 512 for Split MNIST and Split Fashion. The threshold of ORVAE for Split MNIST and Split Fashion is 20. we report the results in Table 1 where the proposed ORVAE outperforms other baselines under three criteria, SSMI, PSNR and ACC, respectively. The number of trained components is provided in Table 2.

	SSMI			
Datasets	ORVAE	OnlineBE	OVAE	CNDPM
Split MNIST	**0.82**	0.69	0.53	0.77
Split Fashion	**0.72**	0.38	0.29	0.66
	PSNR			
Split MNIST	**18.18**	14.89	12.88	16.90
Split Fashion	**18.24**	12.05	10.50	15.35
	Acc			
Split MNIST	**0.94**	0.86	0.20	0.85
Split Fashion	**0.83**	0.38	0.13	0.78

Table 1: The quality of the reconstruction results for various models under CLA.

	Number of components			
Datasets	ORVAE	OnlineBE	OVAE	CNDPM
Split MNIST	14	20	1	26
Split Fashion	11	20	1	14

Table 2: The number of components trained by various models under CLA.

In the following, we also investigate the generative modelling capability of various models under GSSMD. We create a data stream $\mathbf{X}^S$ by collecting samples from MNIST and Fashion without changing the order. The hyperparameters for GSSMD is the same for the experiment of Split MNIST. We report the results in Table 3 where ORVAE still achieves the state of the art results on the challenging task.

	SSMI			
Datasets	ORVAE	OnlineBE	OVAE	CNDPM
MNIST-Fashion	**0.71**	0.56	0.42	0.70
	PSNR			
MNIST-Fashion	**16.57**	14.52	12.69	16.44
	Acc			
MNIST-Fashion	**0.83**	0.60	0.40	0.82
	Number of components			
Split MNIST	23	20	1	30

Table 3: The quality of the reconstruction results for various models under GSSMD.

F.2 Visual results

The reconstruction on CIFAR10, achieved by ORVAE and OVAE under Split CIFAR10 is presented in Fig. 1. From these results we can observe that ORVAE can produce more high-quality reconstructions than OVAE.

The number of components when learning Split CIFAR10 and Split Tiny-ImageNe

	The number of components				
Datasets	ORVAE	OnlineBE	OVAE	CNDPM	ORVAE*
SC	26	20	1	30	22
STI	24	20	1	30	17

Table 4: The number of components when learning natural images under CLA.

We report the number of components of various models under Split CIFAR10 and Split Tiny-ImageNe in Table 4.

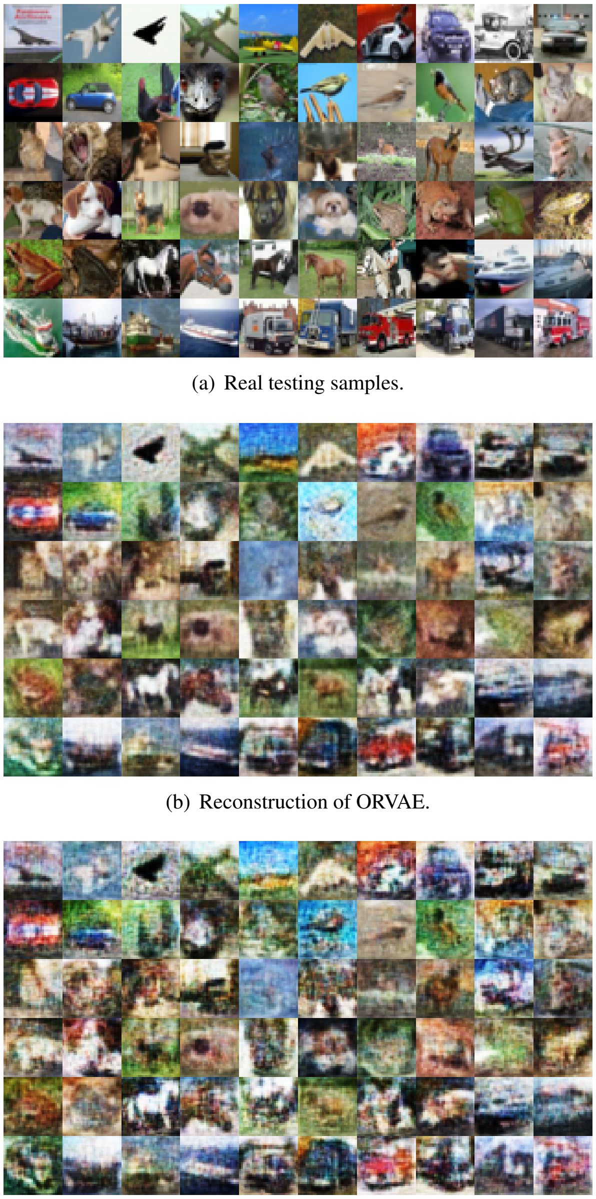
G More results for the ablation study

G.1 The importance of the proposed attention mechanism

In this section, we investigate the effectiveness of the proposed attention mechanism used in ORVAE. We create a baseline that does not update the attention parameters (Eq,(10) of the paper) during the training, namely ORVAE-ELBO-Baseline. The weights in each component of ORVAE-ELBO-Baseline are set to identical values. We train ORVAE-ELBO and ORVAE-ELBO-Baseline under Split MNIST and report the result in Fig. 2. We observe that the proposed attention mechanism used in ORVAE can significantly improve the performance.

G.2 The benefits from the recursive expansion mechanism

ORVAE dynamically expands its network architecture on both the inference and generator while also incorporating all previously learnt representations during the inference and decoding processes. In here, we investigate the effectiveness when reusing previous knowledge for learning novel data samples. We create a baseline which would not use previously learnt representations when adding a new component to ORVAE, namely ORVAE-ELBO-NoAdaptiveWeight. We train both models under Split MNIST and report the result in Fig. 3. From the bar-plots we can observe that by reusing pre-



(a) Real testing samples.

(b) Reconstruction of ORVAE.

(c) Reconstruction of OVAE.

Figure 1: The testing and reconstructed images under Split CIFAR10..

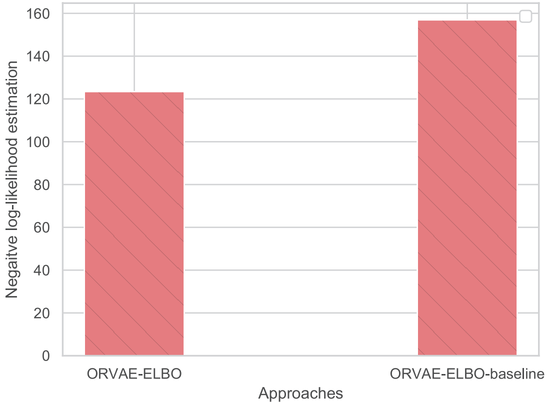


Figure 2: The negative log-likelihood estimated by ORVAE-ELBO and ORVAE-ELBO-Baseline under Split MNIST.

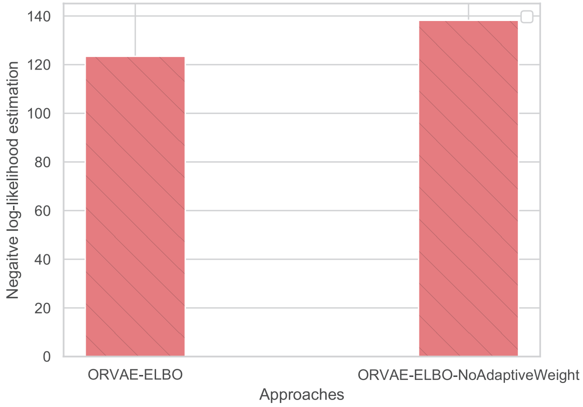


Figure 3: The negative log-likelihood estimated by ORVAE-ELBO and ORVAE-ELBO-NoAdaptiveWeight under Split MNIST.

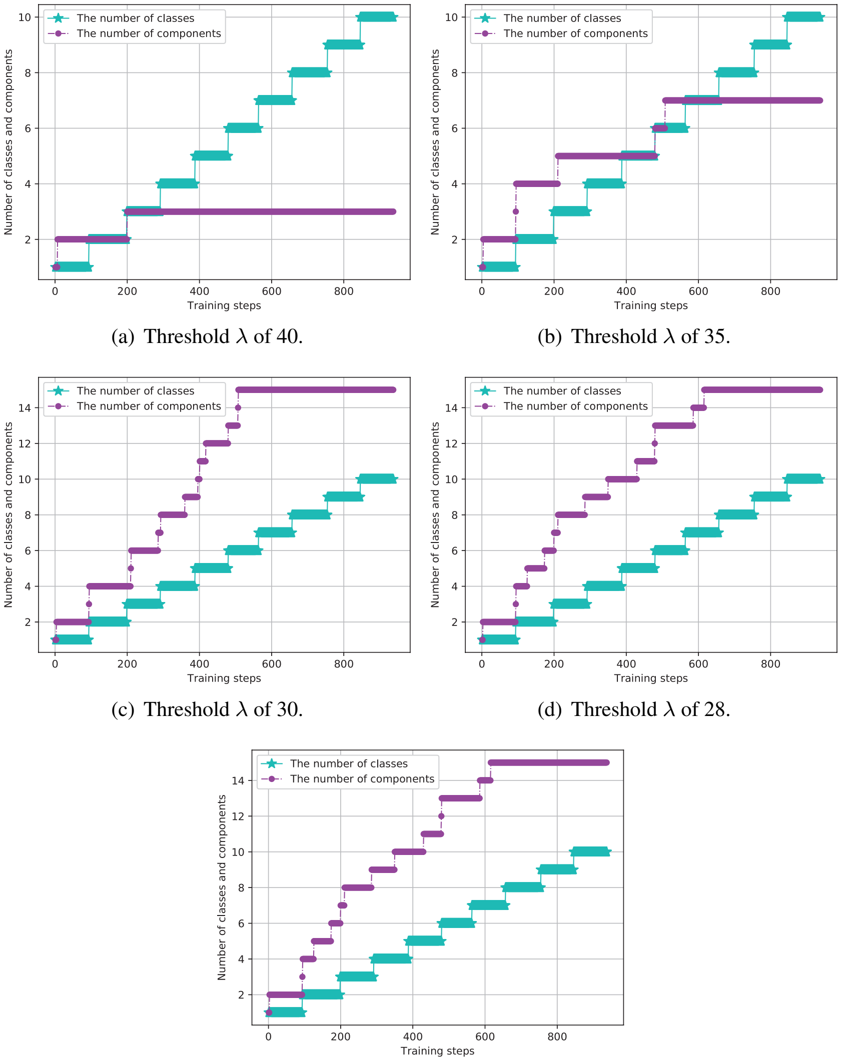
vious knowledge when learning novel samples we can improve the performance of the model.

G.3 The dynamic memory change in ORVAE

In this section, we investigate when the old memory is replaced by the new one during the online continual learning. We train ORVAE under Split MNIST where we record the number of classes and components in each training step. Since the new memory and new component are built at the same time, so we can use the number of components to describe the dynamic of the memory. We show the result in Fig. 4. From this figure we can observe that by changing the threshold λ we have a different dynamic memory process, as defined by the change in the number of model components and data classes being learnt. The memory tends to keep stable during the last training steps which would indicate that ORVAE expands its network architecture slowly when accumulating more prior knowledge. A large threshold would degenerate the performance of ORVAE since the memory only captures the distribution shift at the initial training phases and would not be efficient for the following ones.

G.4 ELBO on the target and source sets

In order to investigate how ORVAE models the underlying data distribution during the online continual learning we train ORVAE-ELBO under Split MNIST. ELBO estimated on the source set (the memory) and the target set (all testing samples) are reported in Fig. 5. We observe that ORVAE has a small ELBO on all testing sets at the beginning and then gradually achieves a high ELBO as the number of training steps is increasing. This demonstrates that ORVAE is able to continually model the underlying data distribution during online continual learning. Additionally, Theorem 2 (Eq. (14) in the paper) shows that the episodic memory plays an important role in the generalization performance of ORVAE. The gap on the ELBO between the source set and target set is gradually reduced as increasing the number of training steps, as shown in Fig. 5. This demonstrates that the proposed episodic memory can well capture the underlying data distribution of target sets.



(a) Threshold λ of 40.

(b) Threshold λ of 35.

(c) Threshold λ of 30.

(d) Threshold λ of 28.

(e) Threshold of 25.

Figure 4: The dynamic process of the memory system during the training.

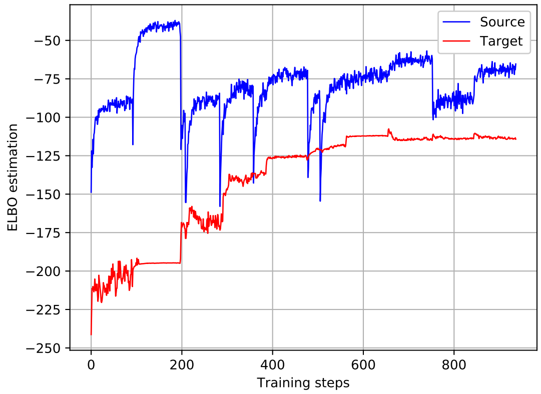


Figure 5: The estimation of ELBO on the target and source sets under Split MNIST.

G.5 Additional information for the classification task

We introduce the datasets used in the classification task as follows.

Split MNIST. We divide MNIST that contains 60k training samples into five tasks according to pairs of incrementing digits De Lange and Tuytelaars (2021).

Split CIFAR10. We split CIFAR10 into five tasks where each task consists of samples from two different classes De Lange and Tuytelaars (2021).

Split CIFAR100. We split CIFAR100 into 20 tasks where each task has 2500 examples from five different classes Lopez-Paz and Ranzato (2017).

Follow from De Lange and Tuytelaars (2021), we adapt an MLP network with 2 fully connected layers of 400 units for Split MNIST. For Split CIFAR10 and Split 100, We adapt ResNet 18 He et al. (2016) as the classifier. The maximize memory size for Split MNIST, Split CIFAR10, Split CIFAR100 are 2K, 1K and 5K, respectively. Since ORVAE is a dynamic expansion model, we build a new classifier associated with the newly created component for learning novel samples. The classifier is trained on the labelled samples stored in memory at the current training session.

After learning, the final number of components in ORVAE is 7 and 22 for Split MNIST and Split CIFAR10.

We introduce the baselines used in the classification task but not mentioned in the paper.

Gradient Episodic Memory (GEM) Lopez-Paz and Ranzato (2017) introduces to use a memory buffer to store past samples in order to relieve forgetting. However, GEM requires knowing both the task information and the class label during the training.

Incremental Classifier and Representation Learning (iCARL) Rebuffi et al. (2017) is also a memory-based approach that aims to store the balanced samples, which is mainly used in an incremental class setup.

MIR Aljundi et al. (2019b) introduces a retrieval strategy for the sample selection in the memory during the online continual learning. However, the retrieval strategy in MIR requires evaluating the loss values in each training session. This means that MIR requires modifying the retrieval strategy for different tasks such as the classification and generation tasks. The proposed OCM does not change the sample selection strategy for different tasks since we evaluate the sample similarity on the feature space using the kernel function.

GSS Aljundi et al. (2019c) formulates the sample selection process as the constraint reduction problem. GSS stores samples to the buffer based on the gradient information which requires to access the class labels and can not be applied in the unsupervised learning setting.

G.6 The model's complexity analysis

We report the number of parameters used for the classification task in Table. 5. It notes that only CN-DPM reported the model's complexity in the classification task.

Methods	Split MNIST	Split CIFAR10	Split CIFAR100
CN-DPM Lee et al. (2020)	524K	4.60M	19.2M
ORVAE	493K	4.32M	13.3M

Table 5: The number of parameters for the classification task. The number of parameters for CN-DPM is reported in Lee et al. (2020).

Time complexity analysis. Since the proposed ORVAE only optimizes a certain component in a mixture model, it does not require huge computational costs. Additionally, the optimization of the proposed attention mechanism can be optimized jointly with the update of the model's parameters, which can benefit two aspects. Firstly, it makes the easy way for the implementation of ORVAE. Secondly, it can help find the optimal parameters configuration for both the attention mechanism and the model.

G.7 The criteria for the knowledge transfer under OCL

The forward and backward transfer was firstly introduce in Lopez-Paz and Ranzato (2017) for the continual supervised learning. There are three criteria, which are defined as :

$$\mathrm{accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{Acc}_{(N,i)}\,. \tag{30}$$

$$\mathrm{BWT} = \frac{1}{N-1}\sum_{i=1}^{N-1} \left(\mathrm{Acc}_{(N,i)} - \mathrm{Acc}_{(i,i)}\right). \tag{31}$$

$$\mathrm{FWT} = \frac{1}{N-1}\sum_{i=2}^{N} \left(\mathrm{Acc}_{(i,i)} - \mathrm{Acc}'_{i}\right). \tag{32}$$

where $\mathrm{accuracy}$ is calculated by the model on all testing datasets. $\mathrm{Acc}_{(N,i)}$ is the classification accuracy evaluated on the i-th testing set, predicted by the model that was trained on the N-th task. BWT and FWT are the criteria for the backward and forward transfer, respectively. Acc'_i is the classification accuracy evaluated on the i-th task, predicted by a model with the random initialization, These criteria, however, are only used in the class-incremental learning setting in which the task information is also given.

Since the main task in our paper is the unsupervised learning under OCL where the task boundaries are not provided, we divide the whole training set into N' parts where each part has the same number of samples and is seen as a task. Then we introduce several new criteria for the forward/backward transfer. Firstly, we define the backward transfer criterion as :

$$\mathrm{BWT}' = \frac{1}{N'-1}\sum_{i=1}^{N'-1} \left(\mathrm{ELBO}_{(N',i)} - \mathrm{ELBO}'_{i}\right). \tag{33}$$

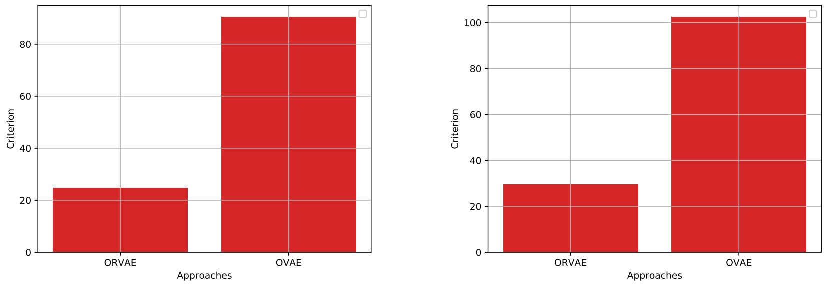
where ELBO'_i is the ELBO evaluated on the i-th task (batch), achieved by an auxiliary VAE trained on previous tasks (batches) $\{\mathcal{T}_1, \cdots, \mathcal{T}_{i-1}\}$. $\mathrm{ELBO}_{(N',i)}$ is the ELBO evaluated on the i-th task (batch), achieved by the proposed model trained on the memory at $\mathcal{T}_{N'}$.

Then we define forward transfer criterion as :

$$\mathrm{FWT}' = \frac{1}{N'-1}\sum_{i=2}^{N'} \left(\mathrm{ELBO}_{(i-1,i)} - \mathrm{ELBO}'_{i}\right). \tag{34}$$

It notes that the auxiliary VAE model can access all previous samples in each training step. Therefore, we desire to achieve the small value in Eq. (33) and Eq. (34) where the gap in the performance between the proposed model and the auxiliary VAE model should be small.

In order to evaluate how the proposed ORVAE benefit the knowledge transfer, we train ORVAE and OVAE under Split MNIST where we calculate Eq. (33) and Eq. (34). Finally, we report the results in Fig. 6, which demonstrates that the proposed ORVAE significantly outperforms OVAE in both the forward and backward transfer criteria.



(a) Forward transfer.

(b) Backward transfer.

Figure 6: The results for the forward/backward transfer.

G.8 The change of batch size

In this section, we investigate the performance of ORVAE when changing the batch size. We train ORVAE under Split MNIST with the batch size of 30, 64, 80, 100, 130, and 150. We report the results in Fig. 7, which indicates that the performance and model size of the ORVAE keeps stable when changing the batch size. However, a very small batch size would increase the model's size while improving the performance further.

G.9 Imbalanced Benchmarks.

In this section, we also investigate the performance of the proposed ORVAE for the imbalanced data stream De Lange and Tuytelaars (2021). We following the setting from De Lange and Tuytelaars (2021) and train ORVAE under Split MNIST, Split CIFAR10 and Split CIFAR100, respectively.

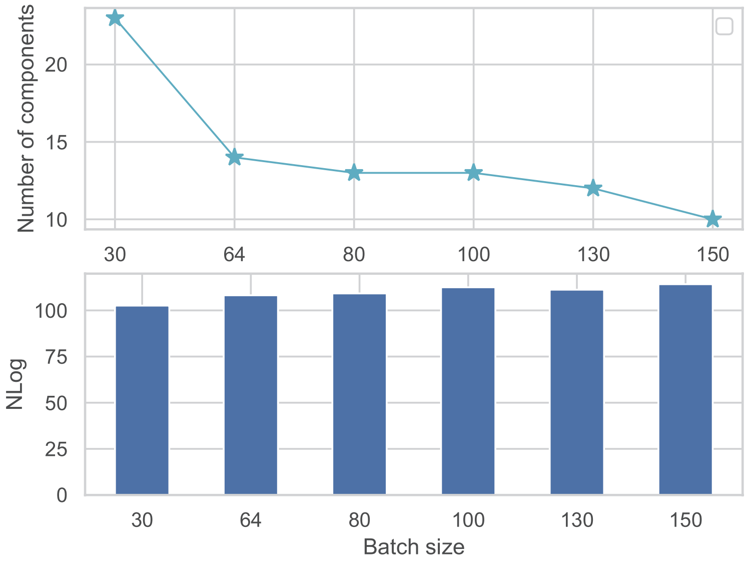
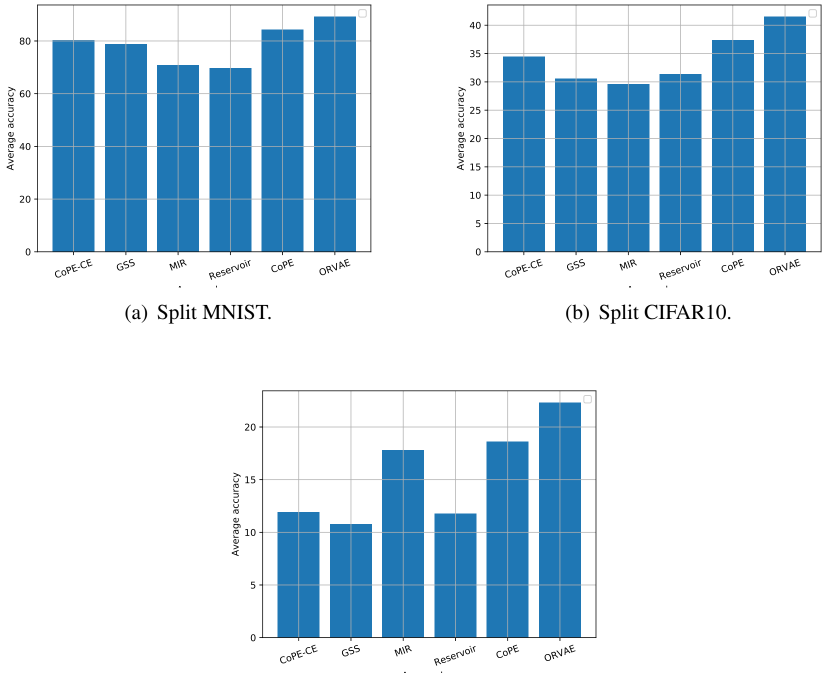


Figure 7: The performance of ORVAE when changing the batch size under Split MNIST.

We report our results in Fig. 8 where the number of components of ORVAE is 5, 8, 16 for Split MNIST, Split CIFAR10, and Split CIFAR100, respectively. These results show that the proposed ORVAE still achieves the state of the art performance for the imbalanced benchmark.

G.10 Limitation of the proposed ORVAE

The main limitation of the proposed ORVAE is that, like other dynamic expansion approaches such as CN-DPM, the model would not be scalable when learning an infinite number of data streams. However, to compare with CN-DPM, the proposed ORVAE grows its network architectures slowly when learning more samples while CN-DPM tends to expand its network architectures regularly because ORVAE can accumulate prior knowledge for learning novel samples and thus does not require to build more components, as shown in Fig.5 of the paper.



(a) Split MNIST.

(b) Split CIFAR10.

(c) Split CIFAR100.

Figure 8: The results for the imbalanced benchmark where the results of baselines are cited by De Lange and Tuytelaars (2021).

H Exploring the sample selection in the dynamic episodic memory

In this section, we explore developing a simple sample selection approach for the memory of ORVAE. The diversity of the memory plays an important role in the generalization performance of the model, which motivates us to consider storing the samples that are outside the current memory distribution. In order to achieve this goal, we want to store the samples for which the current model has a small likelihood. We modify the original process of the proposed dynamic memory as in the following :

1) $\mathcal{S}_t$ adds $\mathcal{X}_{batch}^{t+1}$ to its memory, resulting in $\mathcal{S}_{(t+1)}$.

2) If $|\mathcal{S}_{(t+1)}| > |\mathcal{S}_{(t+1)}|_{max}$, then we evaluate the novelty of the incoming batch of samples $\mathbf{X}_{batch}^{t+1}$ by comparing the loss value between the new samples and the memory:

$$\begin{aligned} d^t = \Big| & \mathbb{E}_{\mathbf{x} \sim \mathcal{S}_t}[\mathcal{L}_{\text{ORVAE}}(\mathbf{x}; \theta_K^t, \omega_K^t)] \\ & - \mathbb{E}_{\mathbf{x} \sim \mathbf{X}_{batch}^{t+1}}[\mathcal{L}_{\text{ORVAE}}(\mathbf{x}; \theta_K^t, \omega_K^t)] \Big| \ , \end{aligned} \tag{35}$$

where $\{\theta_K^t, \omega_K^t\}$ are the parameters of the K-th mixture model trained at the training step $\mathcal{T}_t$. d^t is the absolute difference in the loss function from Eq. (7) from the paper, between the memorized samples from $\mathcal{S}_t$ and the new batch of data $\mathbf{X}_{batch}^{t+1}$, calculated by the current model. We then perform steps 3 and 4.

3) If $d^t > \lambda$, where λ is a predefined threshold, then ORVAE builds a new mixture model $\mathcal{M}^{(K+1)}$ in a recursive way, while $\mathcal{S}_{(t+1)}$ is the set to contain $\mathbf{X}_{batch}^{t+1}$ only at $\mathcal{T}_{(t+1)}$ in order to learn novel samples.

4) If $d^t \leq \lambda$, we consider two different criteria. **First criterion.** We call the first criterion with ORVAE as ORVAE-ELBO-SampleSelection1. We calculate the sample log-likelihood of each sample in the memory by using Eq. (7) of the paper :

$$s^i = \mathcal{L}_{\text{ORVAE}}(\mathbf{x}_i; \theta_K^t, \omega_K^t), i = 1, \cdot, |\mathcal{S}_{(t+1)}| \tag{36}$$

Then we remove samples corresponding to small s^i from $\mathcal{S}_{(t+1)}$ until the memory capacity would match the maximum memory size.

Second criterion. We call the second criterion with ORVAE as ORVAE-ELBO-SampleSelection2. We evaluate the sample log-likelihood of each sample by considering the current component and all previously learnt components :

$$s^i = \sum_{j=1}^{K-1} \{\mathcal{L}_{\mathrm{ORVAE}}(\mathbf{x}_i; \theta_j^\star, \omega_j^\star)\} + \mathcal{L}_{\mathrm{ORVAE}}(\mathbf{x}_i; \theta_K^t, \omega_K^t), i = 1, \cdot, |\mathcal{S}_{(t+1)}| \tag{37}$$

Then we remove samples corresponding to small s^i from $\mathcal{S}_{(t+1)}$ until the memory capacity would match the maximum memory size.

Comparison results using the sample selection and not. In the following, we investigate the results of ORVAE when the sample selection is used in the memory. We report the results in Table 6 where ”ORVAE-ELBO-SampleSelection1” represents ORVAE using the sample selection. It observes that ORVAE-ELBO-SampleSelection1 achieves the best result when using 30 components. However, the sample selection used in the memory would encourage ORVAE to frequently building components and therefore we use the large threshold for ORVAE-ELBO-SampleSelection1 in order to control the network architecture expansion. The results from Table 6 show that randomly removing samples in the memory performs well for ORVAE.

Methods	MNIST	N	Threshold
ORVAE-ELBO	-108.21	14	30
ORVAE-ELBO-SampleSelection1	-101.55	30	90
ORVAE-ELBO-SampleSelection1	-115.03	14	100

Table 6: Comparison results of the density task for Split MNIST under CLA.

I The details of the sampling procedure for ORVAE

In this section, we provide the additional information for the sampling procedure of ORVAE. For each individual variational distribution $Q_{\omega_1^\star}(\mathbf{z}), \cdots, Q'_{\omega_K^\star}(\mathbf{z})$, we can draw samples from each of them by using the reparameterization trick Kingma and Welling (2013) in order to ensure the differentiable optimization.

In the following, we show how we draw a sample from a certain augmented variational distribution. First, we consider $Q_{\omega_2^\star}(\mathbf{z})$:

$$Q_{\omega_2^\star}(\mathbf{z}) = \pi_{(1,2)} Q_{\omega_1^*}(\mathbf{z}) + \pi_{(2,2)} Q'_{\omega_2^\star}(\mathbf{z}), \tag{38}$$

where $Q_{\omega_1^\star}(\mathbf{z}) = \mathcal{N}(\mu_1, {\gamma_1}^2)$ and $Q'_{\omega_2^\star}(\mathbf{z}) = \mathcal{N}(\mu'_2, {\gamma'_2}^2)$ are Gaussian distributions.

Therefore, $Q_{\omega_2^\star}(\mathbf{z})$ is also a Gaussian distribution, expressed by :

$$Q_{\omega_2^\star}(\mathbf{z}) = \mathcal{N}(\pi_{(1,2)}\mu_1 + \pi_{(2,2)}\mu'_2, \pi^2_{(1,2)}{\omega_1}^2 + \pi^2_{(2,2)}{\omega_2}^2) \tag{39}$$

Then we can draw a latent variable $\mathbf{z}_2$ from $Q_{\omega_2^\star}(\mathbf{z})$ using the reparameterization trick $\mathbf{z} = \pi_{(1,2)}\mu_1 + \pi_{(2,2)}\mu'_2 + (\pi^2_{(1,2)}{\omega_1}^2 + \pi^2_{(2,2)}{\omega_2}^2) * \epsilon$ where ϵ is a random noise drawn from $\mathcal{N}(0,1)$.

For the augmented variational distribution $Q_{\omega_3^\star}(\mathbf{z})$, we express it as :

$$Q_{\omega_3^\star}(\mathbf{z}) = \pi_{(1,3)}Q_{\omega_1^*}(\mathbf{z}) + \pi_{(2,3)}Q_{\omega_2^\star}(\mathbf{z}) + \pi_{(3,3)}Q_{\omega_3^\star}(\mathbf{z}), \tag{40}$$

We can rewrite Eq. (40) according to Eq. (39) :

$$\begin{aligned} Q_{\omega_3^\star}(\mathbf{z}) = \mathcal{N}(&\pi_{(1,3)}\mu_1 + \pi_{(2,3)}(\pi_{(1,2)}\mu_1 + \pi_{(2,2)}\mu'_2) + \pi_{(2,3)}\mu'_3, \\ &\pi^2_{(1,3)}{\omega_1}^2 + \pi^2_{(2,3)}(\pi^2_{(1,2)}{\omega_1}^2 + \pi^2_{(2,2)}{\omega_2}^2) + \pi^2_{(3,3)}{\omega'_3}^2)\,. \end{aligned} \tag{41}$$

Then we can draw a latent variable $\mathbf{z}_3$ from $Q_{\omega_3^\star}(\mathbf{z})$ using the reparameterization trick :

$$\begin{aligned} \mathbf{z}_3 = {} & \pi_{(1,3)}\mu_1 + \pi_{(2,3)}(\pi_{(1,2)}\mu_1 + \pi_{(2,2)}\mu'_2) + \pi_{(2,3)}\mu'_3 \\ & + (\pi^2_{(1,3)}{\omega_1}^2 + \pi^2_{(2,3)}(\pi^2_{(1,2)}{\omega_1}^2 + \pi^2_{(2,2)}{\omega_2}^2) + \pi^2_{(3,3)}{\omega'_3}^2) * \epsilon\,. \end{aligned} \tag{42}$$

By inductive summary, we can use the similar sampling process from Eq. (42) for other augmented variational distributions $\{Q_{\omega_4^\star}(\mathbf{z}), \cdots, Q_{\omega_K^\star}(\mathbf{z})\}$.

J The additional information for the construction and learning of the attention mechanism

We have described the construction of the attentional weights $\{\zeta^e_{(1,2)}, \zeta^e_{(2,2)}, \zeta^d_{(1,2)}, \zeta^d_{(2,2)}\}$ for the second component of ORVAE in section 3.4 of the paper. In the following, we describe how we extend the attention region to fit the extension of ORVAE. When ORVAE builds the third component, the proposed attentional mechanism generates a new set of attentional weights $\{\zeta^e_{(1,3)}, \zeta^e_{(2,3)}, \zeta^e_{(3,3)}, \zeta^d_{(1,3)}, \zeta^d_{(2,3)}, \zeta^d_{(3,3)}\}$ used to calculate the component weights $\{\pi_{(1,3)}, \pi_{(2,3)}, \pi_{(3,3)}, \alpha_{(1,3)}, \alpha_{(2,3)}, \alpha_{(3,3)}\}$ using Eq.(11) of the paper. To avoid forgetting, we update only the newly created attentional weights during training, while all previously learned attentional weights are frozen to preserve the previously learned latent structure.

References

Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016. 4

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017. 8, 18, 20

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. Int. Conf. on Learning Representations (ICLR), arXiv preprint arXiv:1412.6980*, 2015. 10

Fei Ye and Adrian G Bors. Continual variational autoencoder learning via online cooperative memorization. In *European Conference on Computer Vision*, pages 531–549. Springer, 2022a. 10

Fei Ye and Adrian G Bors. Task-free continual learning via online discrepancy distance learning. *arXiv preprint arXiv:2210.06579*, 2022b. 10

Fei Ye and Adrian G Bors. Dynamic self-supervised teacher-student network learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022c. 10

Fei Ye and Adrian G. Bors. Lifelong twin generative adversarial networks. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 1289–1293, 2021a. doi: 10.1109/ICIP42928.2021.9506116. 10

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. 10

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11254–11263, 2019a. 10

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. 10

Fei Ye and Adrian Bors. Lifelong teacher-student network learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021c. 11

Fei Ye and Adrian G.Bors Bors. Learning latent representations across multiple data domains using lifelong vaegan. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 777–795, Cham, 2020a. Springer International Publishing. ISBN 978-3-030-58565-5. 11

Fei Ye and Adrian G. Bors. Lifelong learning of interpretable image representations. In *2020 Tenth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6, 2020b. doi: 10.1109/IPTA50016.2020.9286663. 11

Fei Ye and Adrian G Bors. Lifelong generative modelling using dynamic expansion graph model. In *AAAI on Artificial Intelligence*. AAAI Press, 2022e. 11

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. 11

Fei Ye and Adrian G. Bors. Lifelong infinite mixture model based on knowledge-driven dirichlet process, 2021e. 11

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. 11

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015. 11

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Inf. Proc. Systems (NIPS)*, pages 1097–1105, 2012. 11

Alain Hore and Djemel Ziou. Image quality metrics: PSNR vs. SSIM. In *Proc. Int. Conf. on Pattern Recognition (ICPR)*, pages 2366–2369, 2010. 11

Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8250–8259, 2021. 18, 21, 23

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. 18

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 19

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019b. URL https://proceedings.neurips.cc/paper/2019/file/15825aee15eb335cc13f9b559f166ee8-Paper.pdf. 19

R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio. Gradient based sample selection for online continual learning. In *Proc. Neural Inf. Proc. Systems (NIPS), arXiv preprint arXiv:1903.08671*, 2019c. 19

Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural dirichlet process mixture model for task-free continual learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SJxSOJStPr. 19

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013. 25