Supplementary Materials for Lifelong Infinite Mixture Model Based on Knowledge-Driven Dirichlet Process

Contents

A. The proof of Theorem 1

Theorem 1 *Let $\tilde{S}_i^{(t-i+1)}$ and S_i be two joint distributions over $\mathcal{X} \times \mathcal{Y}$. Let $\tau : \mathcal{Y} \times \mathcal{Y} \to [0,1]$ be a symmetric and bounded loss function $\forall (y, y') \in \mathcal{Y}^2, \tau(y, y') \leq M$ that obeys the triangle inequality, where M is a positive number. Let $h_i = \arg\min_{h \in \mathcal{H}} \mathrm{R}(h, S_i)$ and $\tilde{h}_i^{(t-i+1)} = \arg\min_{h \in \mathcal{H}} \mathrm{R}(h, \tilde{S}_i^{(t-i+1)})$ represent the ideal classifiers for S_i and $\tilde{S}_i^{(t-i+1)}$, we have:*

$$\mathrm{R}\left(h, S_i\right) \leq \mathrm{R}'\left(h, \tilde{h}_i^{(t-i+1)}, \tilde{S}_i^{(t-i+1)}\right) + \Psi\left(S_{i,X}, \tilde{S}_{i,X}^{(t-i+1)}\right) + \sigma(S_i, \tilde{S}_i^{(t-i+1)}) \tag{1}$$

where

$$\sigma(S_i, \tilde{S}_i^{(t-i+1)}) = \mathrm{R}'(h_i^*, h_i, S_i) + \mathrm{R}'(h_i, \tilde{h}_i^{(t-i+1)}, \tilde{S}_i^{(t-i+1)}) \tag{2}$$

and

$$\mathrm{R}'(h_i, \tilde{h}_i^{(t-i+1)}, \tilde{S}_i^{(t-i+1)}) = \mathbb{E}_{\mathbf{x} \sim \tilde{S}_{i,X}^{(t-i+1)}} \tau(h_i(\mathbf{x}), \tilde{h}_i^{(t-i+1)}(\mathbf{x})) \tag{3}$$

where h_i^* is the true labeling function for S_i.

Proof *Firstly, we consider $\mathrm{R}(h, S_i) \equiv \mathrm{R}'(h, h_i^*, S_i)$. We adopt similar derivations to those used for Domain Adaptation theory [10]. We consider fixing the classifier $h \in \mathcal{H}$. According to the triangle inequality property of $\mathrm{R}(\cdot)$ and $\mathrm{R}'(\cdot)$ and* Definition 3 *from the paper, see the discrepancy distance $\Psi(\cdot, \cdot)$ from equation (14) from page 5 of the paper, we have:*

$$\begin{aligned} \mathrm{R}'(h, h_i^*, S_i) &\leq \mathrm{R}'(h, \tilde{h}_i^{(t-i+1)}, S_i) + \mathrm{R}'(h_i, \tilde{h}_i^{(t-i+1)}, S_i) + \mathrm{R}'(h_i, h_i^*, S_i) \\ &\leq \mathrm{R}'(h, \tilde{h}_i^{(t-i+1)}, \tilde{S}_i^{(t-i+1)}) + \mathrm{R}'(h_i, \tilde{h}_i^{(t-i+1)}, \tilde{S}_i^{(t-i+1)}) + \mathrm{R}'(h_i, h_i^*, S_i) + \Psi(S_{i,X}, \tilde{S}_{i,X}^{(t-i+1)}) \end{aligned} \tag{4}$$

□

This proves Theorem 1.

B. The proof of Theorem 2

Theorem 2 *Let $\tilde{S}_i^{(t-i+1)}$ be the joint distribution over $\mathcal{X} \times \mathcal{Y}$. Let $\tau : \mathcal{Y} \times \mathcal{Y} \to [0,1]$ be a symmetric loss function, which obeys the triangle inequality and $\forall (y, y') \in \mathcal{Y}^2, \tau(y, y') \leq A$, where A is a positive number. The accumulated errors in the knowledge associated with a previously learnt i-th task, after learning a newly given t-th task, can be defined as :*

$$\mathrm{R}(h, S_i) \leq \mathrm{R}'(h, \tilde{h}_i^{(t-i+1)}, \tilde{S}_i^{(t-i+1)}) + \sum_{k=0}^{t-i} \left(\Psi(\tilde{S}_{i,X}^{(k)}, \tilde{S}_{i,X}^{(k+1)}) + \sigma(\tilde{S}_i^{(k)}, \tilde{S}_i^{(k+1)}) \right) \tag{5}$$

where

$$\sigma(\tilde{S}_i^{(k)}, \tilde{S}_i^{(k+1)}) = \mathrm{R}'(\tilde{h}_i^{(k)}, \tilde{h}_i^{*(k)}, \tilde{S}_i^{(k)}) + \mathrm{R}'(\tilde{h}_i^{(k)}, \tilde{h}_i^{(k+1)}, \tilde{S}_i^{(k+1)})$$

.

It notes that $\tilde{h}_i^{*(k)}$ is the true labeling function for $\tilde{S}_i^{(k)}$, which outputs label y for a giving sample $\mathbf{x} \sim \tilde{S}_{i,X}^{(k)}$.

Proof *Firstly, we take $\tilde{S}_i^{(t-i+1)}$ and $\tilde{S}_i^{(t-i)}$ to be the source and the target domain and we then have a bound, according to* Theorem 1*:*

$$\mathrm{R}'(h, \tilde{h}_i^{(t-i)}, \tilde{S}_i^{(t-i)}) \leq \mathrm{R}'(h, \tilde{h}_i^{(t-i+1)}, \tilde{S}_i^{(t-i+1)}) + \Psi(\tilde{S}_{i,X}^{(t-i+1)}, \tilde{S}_{i,X}^{(t-i)}) + \sigma(\tilde{S}_i^{(t-i)}, \tilde{S}_i^{(t-i+1)}) \tag{6}$$

Furthermore, we take $\tilde{S}_i^{(t-i)}$ and $\tilde{S}_i^{(t-i-1)}$ to be the source and the target domain and we then have a bound:

$$\mathrm{R}'(h, \tilde{h}_i^{(t-i-1)}, \tilde{S}_i^{(t-i-1)}) \leq \mathrm{R}'(h, \tilde{h}_i^{(t-i)}, \tilde{S}_i^{(t-i)}) + \Psi(\tilde{S}_{i,X}^{(t-i)}, \tilde{S}_{i,X}^{(t-i-1)}) + \sigma(\tilde{S}_i^{(t-i)}, \tilde{S}_i^{(t-i-1)}) \tag{7}$$

It notes that $\tilde{S}_i^{(t-i+1)}$ and $\tilde{S}_i^{(t-i)}$ denote the model which was trained on the i-th task for $(t-i+1)$ and $(t-i)$ times, respectively. So the risk bound is reasonable for Eq (6)*. We also know that* $\mathrm{R}'(h, \tilde{h}_i^{(t-i)}, \tilde{S}_i^{(t-i)}) = \mathbb{E}_{\mathbf{x} \sim \tilde{S}_i^{(t-i)}} \tau(h(\mathbf{x}), \tilde{h}_i^{(t-i)}(\mathbf{x}))$

where $\tilde{h}_i^{(t-i)}$ represents the classifier to be trained on the i-th dataset for $(t-i)$ times. After summing up the left-hand side and the right hand side of Eq. (6) and Eq. (7), we have :

$$\begin{aligned}\mathrm{R}'(h,\tilde{h}_i^{(t-i)},\tilde{S}_i^{(t-i)})+\mathrm{R}'(h,\tilde{h}_i^{(t-i-1)},\tilde{S}_i^{(t-i-1)}) \leq \mathrm{R}'(h,\tilde{h}_i^{(t-i)},\tilde{S}_i^{(t-i)})+\mathrm{R}'(h,\tilde{h}_i^{(t-i+1)},\tilde{S}_i^{(t-i+1)}) \\ +\Psi(\tilde{S}_{i,X}^{(t-i)},\tilde{S}_{i,X}^{(t-i-1)})+\sigma(\tilde{S}_i^{(t-i)},\tilde{S}_i^{(t-i-1)}) \\ +\Psi(\tilde{S}_{i,X}^{(t-i+1)},\tilde{S}_{i,X}^{(t-i)})+\sigma(\tilde{S}_i^{(t-i)},\tilde{S}_i^{(t-i+1)})\end{aligned} \tag{8}$$

Then, when moving the first term of the left hand side to the right-hand side in Eq 8, results in:

$$\begin{aligned}\mathrm{R}'(h,\tilde{h}_i^{(t-i-1)},\tilde{S}_i^{(t-i-1)}) \leq \mathrm{R}'(h,\tilde{h}_i^{(t-i)},\tilde{S}_i^{(t-i)})-\mathrm{R}'(h,\tilde{h}_i^{(t-i)},\tilde{S}_i^{(t-i)}) \\ +\mathrm{R}'(h,\tilde{h}_i^{(t-i+1)},\tilde{S}_i^{(t-i+1)})+\Psi(\tilde{S}_{i,X}^{(t-i)},\tilde{S}_{i,X}^{(t-i-1)}) \\ +\sigma(\tilde{S}_i^{(t-i)},\tilde{S}_i^{(t-i-1)})+\Psi(\tilde{S}_{i,X}^{(t-i+1)},\tilde{S}_{i,X}^{(t-i)})+\sigma(\tilde{S}_i^{(t-i)},\tilde{S}_i^{(t-i+1)})\end{aligned} \tag{9}$$

We rewrite the above equation as :

$$\begin{aligned}\mathrm{R}'(h,\tilde{h}_i^{(t-i-1)},\tilde{S}_i^{(t-i-1)}) \leq \mathrm{R}'(h,\tilde{h}_i^{(t-i+1)},\tilde{S}_i^{(t-i+1)})+\Psi(\tilde{S}_{i,X}^{(t-i)},\tilde{S}_{i,X}^{(t-i-1)}) \\ +\sigma(\tilde{S}_i^{(t-i)},\tilde{S}_i^{(t-i-1)})+\Psi(\tilde{S}_{i,X}^{(t-i+1)},\tilde{S}_{i,X}^{(t-i)})+\sigma(\tilde{S}_i^{(t-i)},\tilde{S}_i^{(t-i+1)})\end{aligned} \tag{10}$$

Similarly with the above derivations, we can consider $\tilde{S}_i^{(t-i-1)}$ and $\tilde{S}_i^{(t-i-2)}$ to be the source and the target domain and then we have a bound:

$$\mathrm{R}'(h,\tilde{h}_i^{(t-i-2)},\tilde{S}_i^{(t-i-2)}) \leq \mathrm{R}'(h,\tilde{h}_i^{(t-i-1)},\tilde{S}_i^{(t-i-1)})+\Psi(\tilde{S}_{i,X}^{(t-i-1)},\tilde{S}_{i,X}^{(t-i-2)})+\sigma(\tilde{S}_i^{(t-i-1)},\tilde{S}_i^{(t-i-2)}) \tag{11}$$

We then sum up the left hand side and the right hand side of Eq. (11) and Eq. (10), resulting in:

$$\begin{aligned}\mathrm{R}'(h,\tilde{h}_i^{(t-i-2)},\tilde{S}_i^{(t-i-2)})+\mathrm{R}'(h,\tilde{h}_i^{(t-i-1)},\tilde{S}_i^{(t-i-1)}) \leq \mathrm{R}'(h,\tilde{h}_i^{(t-i+1)},\tilde{S}_i^{(t-i+1)})+\mathrm{R}'(h,\tilde{h}_i^{(t-i-1)},\tilde{S}_i^{(t-i-1)}) \\ +\Psi(\tilde{S}_{i,X}^{(t-i)},\tilde{S}_{i,X}^{(t-i-1)})+\sigma(\tilde{S}_i^{(t-i)},\tilde{S}_i^{(t-i-1)}) \\ +\Psi(\tilde{S}_{i,X}^{(t-i+1)},\tilde{S}_{i,X}^{(t-i)})+\sigma(\tilde{S}_i^{(t-i)},\tilde{S}_i^{(t-i+1)}) \\ +\Psi(\tilde{S}_{i,X}^{(t-i-1)},\tilde{S}_{i,X}^{(t-i-2)})+\sigma(\tilde{S}_i^{(t-i-1)},\tilde{S}_i^{(t-i-2)})\end{aligned} \tag{12}$$

The second term of the left-hand side and the second term from the left-hand side in Eq. (12) are equal and would cancel each other. We observe that the term $\mathrm{R}'(h,\tilde{h}_i^{(t-i+1)},\tilde{S}_i^{(t-i+1)})$ remains in the right hand side. We can continue the derivation by using mathematical induction.

From Theorem 1 we have :

$$\mathrm{R}\left(h,S_i\right) \leq \mathrm{R}'\left(h,\tilde{h}_i^1,\tilde{S}_i^1\right)+\Psi\left(S_{i,X},\tilde{S}_{i,X}^1\right)+\sigma(S_i,\tilde{S}_{i,X}^1) \tag{13}$$

Eventually, by considering the derivation through induction by repeating Eq. (12) and replacing into Eq. (13) we have :

$$\mathrm{R}(h,S_i) \leq \mathrm{R}'(h,\tilde{h}_i^{(t-i+1)},\tilde{S}_i^{(t-i+1)})+\Psi(S_{i,X},\tilde{S}_{i,X}^{(1)})+\sigma(S_i,\tilde{S}_{i,X}^{(1)})+\sum_{k=1}^{t-i}\left(\Psi(\tilde{S}_{i,X}^{(k)},\tilde{S}_{i,X}^{(k+1)})+\sigma(\tilde{S}_i^{(k)},\tilde{S}_i^{(k+1)})\right) \tag{14}$$

In order to simplify the expression (Eq. (14)), we allow $S_{i,X}$ and S_i to be denoted by $\tilde{S}_{i,X}^{(0)}$ and $\tilde{S}_i^{(0)}$, respectively. Then Eq. (14) can be rewritten as :

$$\mathrm{R}(h,S_i) \leq \mathrm{R}'(h,\tilde{h}_i^{(t-i+1)},\tilde{S}_i^{(t-i+1)})+\sum_{k=0}^{t-i}\left(\Psi(\tilde{S}_{i,X}^{(k)},\tilde{S}_{i,X}^{(k+1)})+\sigma(\tilde{S}_i^{(k)},\tilde{S}_i^{(k+1)})\right) \tag{15}$$

C. The proof of Lemma 1

Lemma 1 *Let us consider that we satisfy the Assumption 1 and the accumulated error after learning the probabilistic representations of all databases after the t-th task learning is defined as:*

$$\sum_{i=1}^{t} \mathrm{R}(h, S_i) \leq \sum_{i=1}^{t} \left(\mathrm{R}'(h, \tilde{h}_i^{(t-i+1)}, \tilde{S}_i^{(t-i+1)}) + \sum_{k=0}^{t-i} \left(\Psi(\tilde{S}_{i,X}^{(k)}, \tilde{S}_{i,X}^{(k+1)}) + \sigma(\tilde{S}_i^{(k)}, \tilde{S}_i^{(k+1)}) \right) \right). \tag{16}$$

We consider the results of Theorem 2 and sum up the accumulated errors from Eq. (5) for learning t tasks and this results in Eq. (16). From Lemma 1, we observe that if i is small, indicating that a task was learnt in one of the initial training stages of the model, then after the training with a series of new tasks, the terms $\sum_{k=0}^{t-i} \tau(\tilde{S}_{i,X}^{(k)}, \tilde{S}_{i,X}^{(k+1)}) + \sigma(\tilde{S}_i^{(k)}, \tilde{S}_i^{(k+1)})$ accumulate likely leading to large errors. This explains that $\mathcal{M}(\theta^t, \varsigma^t, \varphi^t)$ would tend to forget the tasks learnt earlier during its lifelong learning process. In order to investigate this process, we train a single model with GRMs under the MNIST, SVHN, Fashion and CIFAR10 (MSFC) where we only evaluate the target accumulated risk of the model on MNIST. We also train the model under the SVHN, Fashion, MNIST and CIFAR10 (SFMC). We present the results in Figure 1, where $i = 1$ and $i = 3$ represent the risk on MNIST for the learning settings MSFC and SFMC, respectively. The results show that the model tends to forget earlier learnt tasks than those tasks which have been learnt more recently.

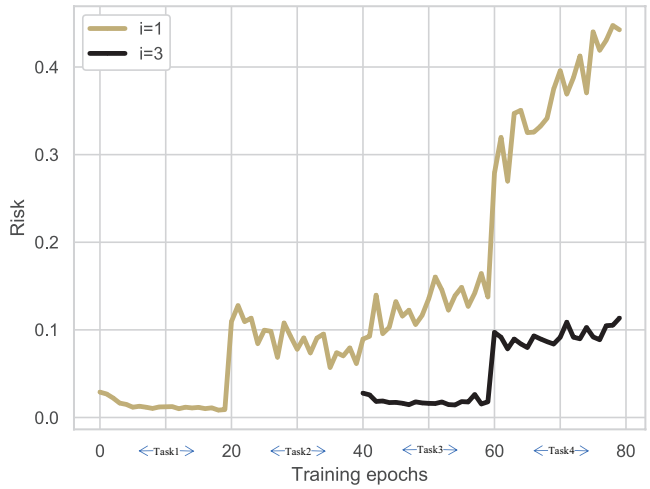


Figure 1. The target risk of a single model with GRMs, evaluated on MNIST during lifelong learning.

D. The proof of Lemma 2

Lemma 2 *Let $\tau : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ be a symmetric loss function, which obeys the triangle inequality and $\forall (y, y') \in \mathcal{Y}^2, \tau(y, y') \leq A$. Let $\mathcal{H}$ be the classifier space. We assume that we have learned an infinite mixture model which contains K number of components after the t-th task learning. If $K = t$, then the accumulated errors of the infinite mixture model for all tasks is defined as:*

$$\sum_{i=1}^{t} \mathrm{R}\left(h, S_i\right) \leq \sum_{i=1}^{t} \left(\mathrm{R}'(h, \tilde{h}_i^1, \tilde{S}_i^1) + \Psi(S_{i,X}, \tilde{S}_{i,X}^1) + \sigma(S_i, \tilde{S}_i^1) \right) \tag{17}$$

Proof *Since our mixture model has K number of components, where each component is only trained once on a certain database during the lifelong learning. So the accumulated error of a certain component on the i-th database is, according to Theorem 1, given by:*

$$\mathrm{R}(h, S_i) \leq \mathrm{R}'(h, \tilde{h}_i^1, \tilde{S}_i^1) + \Psi(S_{i,X}, \tilde{S}_{i,X}^1) + \sigma(S_i, \tilde{S}_i^1) \tag{18}$$

where $\tilde{S}_i^1$ represent the training set of the i-th task. We then sum up the errors of all components on all databases and this results in the relationship from (17), which proves Lemma 2. We can observe that h in Eq. (17) *is implemented as K classifiers $\{h_{\varsigma_1}, \ldots, h_{\varsigma_K}\}, h_{\varsigma_1} \in \mathcal{H}$. So we can rewrite Eq.* (17) *as :*

$$\sum_{i=1}^{t} \mathrm{R}\left(h_{\varsigma_i}, S_i\right) \leq \sum_{i=1}^{t} \Big(\mathrm{R}'(h_{\varsigma_i}, \tilde{h}_i^1, \tilde{S}_i^1) + \Psi(S_{i,X}, \tilde{S}_{i,X}^1) + \sigma(S_i, \tilde{S}_i^1) \Big) \tag{19}$$

From Eq. (19)*, we can observe that the performance of LIMix on the target distribution is relying on the generalization of each classifier h_{ς_i} on the corresponding target distribution.*

In practice, the number of components is smaller than the number of learning tasks. We assume that the mixture model adds a new component in order to learn a new task, where each component models a unique task. Then, a certain mixing component can be reused to learn the following task, if this task is similar to one of the previously learnt tasks. The accumulated error of the this mixture model across all tasks is defined as:

$$\begin{aligned} &\sum_{i=1}^{t} \mathrm{R}(h, S_i) \leq \\ &\sum_{i=1}^{m} \Big(\mathrm{R}'(h, \tilde{h}_i^1, \tilde{S}_i^1) + \Psi(S_{i,X}, \tilde{S}_{i,X}^1) + \sigma(S_i, \tilde{S}_i^1) \Big) + \\ &\sum_{i=m+1}^{t} \left(\begin{array}{l} \mathrm{R}'(h, \tilde{h}_i^{(t-i+1)}, \tilde{S}_i^{(t-i+1)}) + \\ \sum\limits_{k=0}^{t-i} \Big(\Psi(\tilde{S}_{i,X}^k, \tilde{S}_{i,X}^{(k+1)}) + \sigma(\tilde{S}_i^k, \tilde{S}_i^{(k+1)}) \Big) \end{array} \right) \end{aligned} \tag{20}$$

where m is the number of components in the mixture model and determines the degree of the gap on the risk bound and where we indicated in the last expression from Eq. (20) that certain mixture components are retrained with the new given tasks. If m is large, then the resulting gap would be small, otherwise the gap would be larger.

E. The proof of Lemma 3

Lemma 3 *Let $B = \{b_1, \ldots, b_j\}$ represent the labels for the tasks that the corresponding distributions $\{\tilde{S}_{b_1}^{(1)}, .., \tilde{S}_{b_j}^{(1)}\}$ is accessed only once after lifelong learning. Let $B' = \{b'_1, \ldots, b'_n\}$ indicate which task was used for refined training, more than once. We also define a set $\hat{B} = \{\hat{b}_1, \ldots, \hat{b}_n\}$ that records how many times each task was used for refined, where $\hat{b}_i > 1$ represents that the b'_i-th task has been refined for $\hat{b}_i$ times $\tilde{S}_{b'_i}^{(1)} \rightarrow \tilde{S}_{b'_i}^{(\hat{b}_i)}$. After a task has been learnt by the model, we can access its corresponding probabilistic representation defined by $\tilde{S}_i^{(k)}, k = 1, \ldots, (\mathrm{t} - \mathrm{i} + 1)$, where $t > 1$ represents the number of tasks. All sets satisfy $\{B, B', \hat{B}\} \subsetneqq \mathbf{N}^*$ where $\mathbf{N}^*$ is the set of all positive integers. Let $\mathrm{card}(\cdot)$ be the cardinal number in the set which satisfies $\mathrm{card}(B) + \mathrm{card}(B') = K$ and $0 \leq \mathrm{card}(B) \leq K, 0 \leq \mathrm{card}(B') \leq K, \mathrm{card}(B') = \mathrm{card}(\hat{B})$, where K is the number of components used for training the mixture model. We define the risk for a single learned model $\mathcal{M}(\theta^t, \omega^t, \psi^t)$ as $\mathrm{R}_{\mathrm{single}}$, as in (16)), while the risk for the mixture model, $\mathrm{R}_{\mathrm{mixture}}$ is defined as:*

$$\begin{aligned} \sum_{i=1}^{t} \mathrm{R}\left(h, S_i\right) \leq & \sum_{i=1}^{\mathrm{card}(B)} \Big(\mathrm{R}'(h, \tilde{h}_{b_i}^1, \tilde{S}_{b_i}^1) + \Psi(S_{b_i,X}, \tilde{S}_{b_i,X}^1) + \sigma(S_{b_i}, \tilde{S}_{b_i}^1) \Big) + \\ & \sum_{i=1}^{\mathrm{card}(B')} \left(\begin{array}{l} \mathrm{R}'(h, \tilde{h}_{b'_i}^{(t-\hat{b}_i+1)}, \tilde{S}_{b'_i}^{(t-\hat{b}_i+1)}) + \\ \sum\limits_{k=0}^{\hat{b}_i-1} \Big(\Psi(\tilde{S}_{b'_i,X}^k, \tilde{S}_{b'_i,X}^{(k+1)}) + \sigma(\tilde{S}_{b'_{i\,i}}^k, \tilde{S}_{b'_i}^{(k+1)}) \Big) \end{array} \right) = \mathrm{R}_{\mathrm{mixture}} \end{aligned} \tag{21}$$

From the above definitions, we have $\mathrm{R}_{\mathrm{single}} \geq \mathrm{R}_{\mathrm{mixture}}$.

We can observe that h in Eq. (21) is implemented by several components in LIMix and we omit the component index for h for the sake of simplification. According to Lemma 3 we do not explicitly indicate which task is associated to which specific component, but we provide an explicit way to analyze the risk bound for the mixture model in a more practical way, where the number of components and the number of times each task was trained on, is varying according to different learning settings (by considering the order of learning the tasks or their complexity). If $B' = \emptyset$, so there is no database used twice

for training, then the right-hand side of Eq. (21) is reduced to Eq. (17), meaning a small gap on the risk bound but requiring more memory. On the other hand, if $\mathrm{card}(B) = 1 \Rightarrow B = \{t\}$, the right hand side of Eq. (21) is equal to Eq. (16) meaning a large gap on the risk bound. we can define the ratio $v = (K - Count(B))/(K - Count(B'))$ as a trade-off index which explains the trade-off between the model complexity and performance. When v increases, the model performs better, while also having an increased complexity. In contrast, when v is small, then the model has accumulated more error terms but has a smaller complexity.

In the following, we also provide an expression of Eq. (21) by considering to indicate the component index. Let $C = \{c_1, \ldots, c_j\}$ be a set where each c_i represents the component index for each b_i-th task. Let $C' = \{c'_1, \ldots, c'_n\}$ be a set where each c'_i represents the component index for each b'_i-th task. we can observe that c'_i could be equal to c'_j since a single component would learn more than one task. Based on these definitions, we provide an expression for Eq. (21) by considering the component index :

$$\sum_{i=1}^{\mathrm{card}(B)} \mathrm{R}\left(h_{\zeta_{c_i}}, S_{b_i}\right) + \sum_{i=1}^{\mathrm{card}\left(B'\right)} \mathrm{R}\left(h_{\zeta_{c'_i}}, S_{b'_i}\right) \leq \sum_{i=1}^{\mathrm{card}(B)} \left(\mathrm{R}'(h_{\zeta_{c_i}}, \tilde{h}^1_{b_i}, \tilde{S}^1_{b_i}) + \Psi(S_{b_i,X}, \tilde{S}^1_{b_i,X}) + \sigma(S_{b_i}, \tilde{S}^1_{b_i})\right) + \sum_{i=1}^{\mathrm{card}(B')} \left(\begin{array}{l} \mathrm{R}'(h_{\zeta_{c'_i}}, \tilde{h}^{(t-\hat{b}_i+1)}_{b'_i}, \tilde{S}^{(t-\hat{b}_i+1)}_{b'_i}) + \\ \sum_{k=0}^{\hat{b}_i-1} \left(\Psi(\tilde{S}^k_{b'_i,X}, \tilde{S}^{(k+1)}_{b'_i,X}) + \sigma(\tilde{S}^k_{b'_{i_i}}, \tilde{S}^{(k+1)}_{b'_i})\right) \end{array} \right) = \mathrm{R}_{\mathrm{mixture}} \tag{22}$$

where each $h_{\zeta_{c_i}} \in \mathcal{H}$ is a component in LIMix.

Proof *Firstly, we consider the risk bound for the tasks that are only trained only once, we have :*

$$\sum_{i=1}^{\mathrm{card}(B)} \mathrm{R}\left(h, S_{b_i}\right) \leq \sum_{i=1}^{\mathrm{card}(B)} \left(\mathrm{R}'(h_{\zeta_{c_i}}, \tilde{h}^1_{b_i}, \tilde{S}^1_{b_i}) + \Psi(S_{b_i,X}, \tilde{S}^1_{b_i,X}) + \sigma(S_{b_i}, \tilde{S}^1_{b_i})\right) \tag{23}$$

Then we derive the risk bound for the tasks that are trained more than once, we have :

$$\sum_{i=1}^{\mathrm{card}\left(B'\right)} \mathrm{R}\left(h, S_{b'_i}\right) \leq \sum_{i=1}^{\mathrm{card}(B')} \left(\begin{array}{l} \mathrm{R}'(h, \tilde{h}^{(t-\hat{b}_i+1)}_{b'_i}, \tilde{S}^{(t-\hat{b}_i+1)}_{b'_i}) + \\ \sum_{k=0}^{\hat{b}_i-1} \left(\Psi(\tilde{S}^k_{b'_i,X}, \tilde{S}^{(k+1)}_{b'_i,X}) + \sigma(\tilde{S}^k_{b'_i}, \tilde{S}^{(k+1)}_{b'_i})\right) \end{array} \right) \tag{24}$$

Then we sum up Eq. 23 and Eq. 24, resulting in Eq. 21.

In the following, we prove $\mathrm{R}_{\mathrm{single}} \geq \mathrm{R}_{\mathrm{mixture}}$. *We can derive the risk bound for a single model as :*

$$\mathrm{R}_{\mathrm{single}} = \sum_{i=1}^{t} \left(\mathrm{R}'(h, \tilde{h}^{(t-i+1)}_i, \tilde{S}^{(t-i+1)}_i) + \sum_{k=0}^{t-i} \left(\Psi(\tilde{S}^{(k)}_{i,X}, \tilde{S}^{(k+1)}_{i,X}) + \sigma(\tilde{S}^{(k)}_i, \tilde{S}^{(k+1)}_i)\right) \right). \tag{25}$$

Then we consider an extreme case for $\mathrm{R}_{\mathrm{mixture}}$ *in which the number of components is only two. The first component learns the first task and is fixed in the following task learning. We then derive* $\mathrm{R}_{\mathrm{mixture}}$ *as :*

$$\begin{aligned} \mathrm{R}_{\mathrm{mixture}} =& \mathrm{R}'\left(h, \tilde{h}^1_1, \tilde{S}^1_1\right) + \Psi\left(S_{1,X}, \tilde{S}^1_{1,X}\right) + \sigma(S_1, \tilde{S}^1_1) \\ &+ \sum_{i=2}^{t} \left(\mathrm{R}'(h, \tilde{h}^{(t-i+1)}_i, \tilde{S}^{(t-i+1)}_i) + \sum_{k=0}^{t-i} \left(\Psi(\tilde{S}^{(k)}_{i,X}, \tilde{S}^{(k+1)}_{i,X}) + \sigma(\tilde{S}^{(k)}_i, \tilde{S}^{(k+1)}_i)\right) \right). \end{aligned} \tag{26}$$

where the second component is used to learn the following tasks. It clearly see that Eq. (25) $\geq$ *Eq.* (26) *since the first task in Eq.* (25) *has more accumulated errors and the source risk usually keeps stable during GRMs process (See empirical results in Fig.2c of the paper). Additionally, If* $card(B) = 1 \Rightarrow B = \{t\}$, *then* $\mathrm{R}_{\mathrm{mixture}}$ *is equal to* $\mathrm{R}_{\mathrm{single}}$. *As* $card(B)$ *increase,* $\mathrm{R}_{\mathrm{mixture}}$ *is decreased. This proves* $\mathrm{R}_{\mathrm{single}} \geq \mathrm{R}_{\mathrm{mixture}}$.

F. Lemma 4

Lemma 4 *Let us define the log-likelihood of the proposed lifelong infinite mixture model. We can evaluate the model log-likelihood for arbitrary testing data samples during both training and testing stages.*

Proof *Suppose that we have learned K components at the t-th task learning. Estimating the log-likelihood*

$$\log p\left(\mathbf{x} \mid \Theta, \Omega, \pi_1, \ldots, \pi_K\right) = \log \sum_{j=1}^{K} \pi_j p_{\theta_j}\left(\mathbf{x} \mid \mathbf{z}\right) p\left(\mathbf{z}\right) \tag{27}$$

can be done by firstly inferring each component weight π_j by using :

$$v_i = \frac{\exp\left(1/(-\log p_{\theta_i}(\mathbf{x}))\right)}{\sum_{j=1}^{K} \exp\left(1/(-\log p_{\theta_j}(\mathbf{x}))\right)} \tag{28}$$

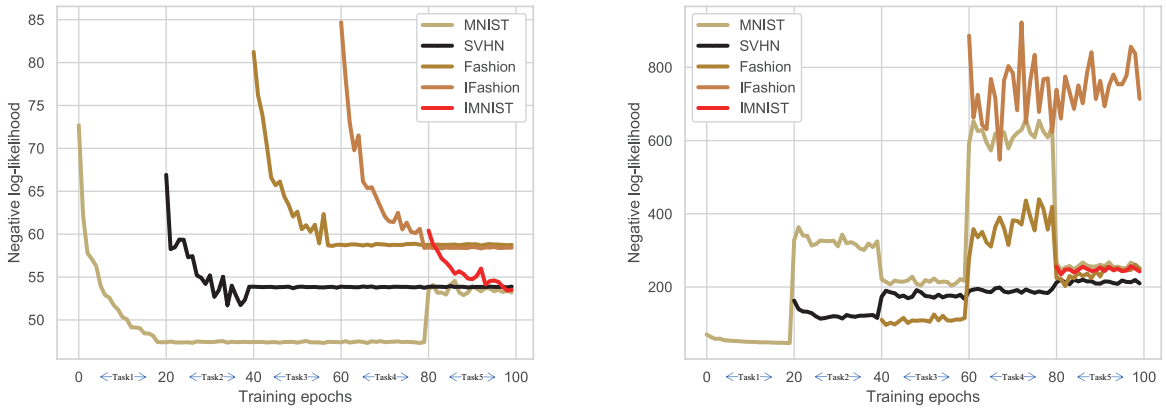
$$\pi \sim Cat\left(v_1, \ldots, v_K\right) \tag{29}$$

It notes that $-\log p_{\theta_i}(\mathbf{x})$ is estimated by ELBO using the i-th component. Cat is a Categorical distribution. Therefore, the down-stream tasks such as the sample log-likelihood, classification and reconstruction are performed by a selected component that has the highest sample log-likelihood $\log p_{\theta_s}(\mathbf{x})$:

$$\log p_{\theta_s}(\mathbf{x}) > \log p_{\theta_i}(\mathbf{x}), \forall i \neq s \tag{30}$$

□

In the following, we investigate how the proposed LIMix model, described in the paper, can evaluate the sample log-likelihood across domains during the lifelong learning. We train LIMix under the MNIST, SVHN, Fashion, IFashion and IMNIST lifelong learning and evaluate the negative log-likelihood (we use the ELBO to estimate the log-likelihood and use the reconstruction error as the first term in ELBO) by using the Teacher module for testing each dataset. We consider 20 training epochs for each task and provide the results in Figure 2a. We can observe that the Teacher module can maintain the sample log-likelihood of all prior tasks during the lifelong learning. We also evaluate the sample log-likelihood estimated by using the Student module and show the results in Figure 2b.



(a) Results provided by the Teacher module.

(b) Results provided by the Student module.

Figure 2. Negative log-likelihood evaluation by the Teacher and the Student modules in (a) and (b), respectively, for the proposed LIMix model, under the MNIST, SVHN, Fashion, IFashion, IMNIST lifelong learning.

G. Theoretical analysis for existing lifelong learning models

A single model with GRM: Firstly, we apply the proposed theoretical tool to derive the risk bound for a single model with GRM [15]. Let $M(\theta^t, \omega^t, \psi^t)$ represent this model after learning t tasks, where we also assume that $U_\varphi : \mathcal{X} \to \mathcal{T}$ is a perfect task-inference network that can allow us to form several joint distributions $\{\tilde{S}_1^t, \ldots, \tilde{S}_t^1\}$. From Theorem 2, we can define

the risk bound for $M(\theta^t, \omega^t, \psi^t)$ after learning t tasks :

$$\mathrm{R}(h, S_i) \leq \mathrm{R}'(h, \tilde{h}_i^{(t-i+1)}, \tilde{S}_i^{(t-i+1)}) + \sum_{k=0}^{t-i} \left(\Psi(\tilde{S}_{i,X}^{k}, \tilde{S}_{i,X}^{(k+1)}) + \sigma(\tilde{S}_i^{k}, \tilde{S}_i^{(k+1)}) \right), \tag{31}$$

where each $\tilde{S}_i^{(k+1)}$ is approximated by $M(\theta^{(k+1)}, \omega^{(k+1)}, \psi^{(k+1)})$ considering the optimal task-inference network where $k = \{0, \ldots, (t-i+1)\}$. We find that while increasing the number of tasks, the model accumulates more error terms. This can lead to a large gap on the risk bound. This explains why a single model with only GRMs has limitations when aiming to learn a long sequence of tasks. We can observe that the approximation distribution $\tilde{S}_{i,X}^{k}$ can be implemented by any generative models such as VAEs and GANS. Therefore, this analysis can also explain most of existing lifelong learning approaches that use GRMs with a single model [2, 3, 13, 15, 18, 21].

CURL: In the following, we apply this theoretical tool to derive the risk bound for a mixture model enabled with the GRM and expansion mechanisms [14]. By using the Lemma 3, we can derive a flexible risk bound for the model after learning t tasks :

$$\begin{aligned} \sum_{i=1}^{t} \mathrm{R}\left(h, S_i\right) \leq & \sum_{i=1}^{\mathrm{card}(B)} \Big(\mathrm{R}'(h, \tilde{h}_{b_i}^{1}, \tilde{S}_{b_i}^{1}) + \Psi(S_{b_i,X}, \tilde{S}_{b_i,X}^{1}) \\ & + \sigma(S_{b_i}, \tilde{S}_{b_i}^{1}) \Big) + \sum_{i=1}^{\mathrm{card}(B')} \Big(\mathrm{R}'(h, \tilde{h}_{b'_i}^{(t-\hat{b}_i+1)}, \tilde{S}_{b'_i}^{(t-\hat{b}_i+1)}) \\ & + \sum_{k=0}^{\hat{b}_i-1} \Big(\Psi(\tilde{S}_{b'_i,X}^{k}, \tilde{S}_{b'_i,X}^{(k+1)}) + \sigma(\tilde{S}_{b'_i}^{k}, \tilde{S}_{b'_i}^{(k+1)}) \Big) \Bigg) = \mathrm{R}_{\mathrm{mixture}} \end{aligned} \tag{32}$$

This risk bound can explain the learning conditions when using the model. Especially, when the model does not expand its network architecture, the risk bound becomes the one from (31). When increasing the number of components in the mixture model, the model would get a tight bound due to the reduction of the accumulated error terms. This analysis is similar to the one explaining the proposed LIMix model. When comparing to the CURL model [14], the proposed LIMix model not only that it expands its network architecture for the inference network but also for the decoder, which benefits overcoming the forgetfulness issue for some down-stream tasks such as data reconstruction and data log-likelihood estimation. While CURL continuously updates its decoder when learning a new task, it would lead to a drop when CURL reuses a component to learn a new task using GRM. For instance, when reusing the b'_i-th component for a new task, there would be a large gap between $\tilde{S}_{b'_i,X}^{n}, n > 1$ and the true marginal distribution $S_{b'_i,X}$ when the number of tasks is large. However, this does not happen in the LIMix model, given that LIMix updates only a limited set of parameters (those of a sub-decoder), which is part of a decoder in each task learning while the other parameters of the decoder do not change.

Ensemble: We also apply the proposed theoretical tool to analysis the risk bound for the Ensemble model [17]. With Lemma 2, we can derive the risk bound for the Ensemble model after the t-th task learning :

$$\sum_{i=1}^{t} \mathrm{R}\left(h, S_i\right) \leq \sum_{i=1}^{t} \left(\mathrm{R}'(h, \tilde{h}_i^{1}, \tilde{S}_i^{1}) + \Psi(S_{i,X}, \tilde{S}_{i,X}^{1}) + \sigma(S_i, \tilde{S}_i^{1}) \right). \tag{33}$$

In this case, the ensemble model does not require the task-inference network or the generator. Each S_i can be simply formed by the data samples from the i-th testing set D_T^i and each $\tilde{S}_i^1$ can be formed by samples drawn from the i-th training set D_S^i and therefore $\tilde{h}_i^1$ is the true labelling function for D_S^i. This ensemble model would achieve a tight bound when comparing with other methods. However, when comparing to the proposed LIMix model, the Ensemble model has to know the task labels during both the training and testing stages. Furthermore, the Ensemble model also requires defining the number of experts to be used before the training, which has limitations on real-time applications.

Regularisation based approaches with episodic memory. Gradient Episodic Memory (GEM) [9] is a typical regularization approach that introduces a memory M storing the same number of data samples for each task. These data samples are usually collected from the training set from each task and we can define the empirical distribution by using these samples for the source domain of the i-th task, denoted as m_i. Then we can derive the risk bound for this model at the t-th task learning :

$$\sum_{i=1}^{t} \mathrm{R}\left(h, S_i\right) \leq \sum_{i=1}^{t} \left(\mathrm{R}'(h, \tilde{h}_i^{1}, m_i) + \Psi(S_{i,X}, m_{i,X}) + \sigma(S_i, m_i) \right), \tag{34}$$

where $\tilde{h}_i^1$ is the true labelling function for m_i. From Eq. (34), the lifelong learning is transformed into a multiple source-target domain domain adaptation problem. We also find that the selection of past data samples for each task is crucial for the improvement of the performance since the appropriate m_i can allow the discrepancy term to be small. Additionally, existing methods using episodic memory [4, 5] can be also explained by using Eq. (34) through which the algorithm finds the optimal updated direction in the parameter space by taking into account all m_i, $i = 1, \ldots, t$. The other method [11], considers identifying the influential examples that are used to form the influential examples m_i corresponding to each task. These influential examples can reduce the computational complexity while improving the generalization performance on the target domains.

Dirichlet process mixture model (DP-Mix) [8] and comparison with LIMix. DP-Mix is designed for the task-free learning manner and is only applied for learning many tasks within one domain. Additionally, DP-Mix only has the expansion mechanism and therefore would lead to catastrophic forgetting in the cross-domain lifelong learning in which one component models more than one task. However, the proposed LIMix can be applied in both cross-domain and single-domain (also called the class incremental learning or continuous learning [22]) lifelong learning (Please see results in Table 2 and Table 3 of the paper). Besides, DP-Mix method does not provide theoretical guarantees. According to our theoretical analysis, DP-Mix can guarantee the optimal performance if and only if the number of components matches the number of tasks, as in Eq. (33), which would require more parameters than LIMix. Furthermore, DP-Mix still requires to use Short-Term Memory (STM) to evaluate the selection and expansion of the mixture model. However, the proposed LIMix does not store any past samples and we evaluate the selection and expansion by comparing the knowledge generated by each component and the data samples from a newly given task.

This paper is the first to provide theoretical insights into various approaches of lifelong learning models. We believe that the proposed LIMix can be used in a task-free learning manner with few modifications, which will be studied in our future work.

H. Theoretical analysis when changing the order for learning the tasks during the lifelong learning

The performance in existing lifelong learning models will be affected when changing the order of the tasks used for training during the lifelong learning. However, there is no theoretical study for the order of tasks and this paper is the first study to propose a theoretical insight into this problem. From Theorem 2, we know that a single model with GRMs tends to forget the tasks learnt earlier during the lifelong learning process. The model will consequently retrain with those tasks several times. However, the ability of the model to learn different tasks vary according to the complexity of the given tasks and according to what tasks have been learnt previously. For instance, the model learns a hard task at the beginning and tends to lose its performance during the lifelong learning when comparing to learning an easy task at the beginning. In the following, we provide the theoretical framework for the learning order of the tasks and their associated learning complexity during the lifelong learning.

H.1. Preliminaries

Definition 1 *(The complexity of a given task). Let $\mathcal{M}(\theta, \varsigma, \varphi)$ represent a single model, enabled with Generative Replay Mechanisms (GRMs). We define the complexity of a task by evaluating $\mathcal{M}(\theta, \varsigma, \varphi)$ on this task, corresponding to learning the probabilistic representation of a dataset. We consider the error of a model when learning a task as the learning complexity measure for that task. When considering two different datasets, H and E, if the empirical risk $\mathrm{R}(h, H) > \mathrm{R}(h, E)$ then the complexity for learning the task associated with database H is larger than for E, where $R(\cdot)$ is the empirical risk from Definition 4, equation (15) in the paper. The assumption is that the same model is used under identical training conditions, learning rate and model initialization for training with either database H or E.*

Assumption 1 *We assume $\tau : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ be an symmetric and bounded loss function $\forall (y, y') \in \mathcal{Y}^2, \tau(y, y') \leq M$ and $\tau(\cdot, \cdot)$ obeys the triangle inequality, where M is a positive number.*

Assumption 2 *Let $\mathcal{A} = \{D_T^1, \ldots, D_T^t\}$ be a set of testing databases and $\widetilde{\mathcal{A}} = \{D_S^1, \ldots, D_S^t\}$ be a set of training databases, and we assume that the complexity of each dataset can be indicated by the error provided at the convergence by the model trained on $\widetilde{\mathcal{A}}$. The large error means that the dataset is more complex and a smaller error is characteristic of a model with lower complexity. However, the complexity is measured relative to the deep learning structure and its parameters. We use $\mathbf{Q} \in \mathbf{R}^{t \times t}$ to represent a matrix where each item $q_{i,k}$ denotes the accumulated error estimated for the i-th task (database) between the (k)-th learning time and $(k+1)$-th learning time. For instance, $q_{i,k}$ can be represented by $q_{i,k} = \Psi(\widetilde{\mathcal{A}}(i)_X^{(k)}, \widetilde{\mathcal{A}}(i)_X^{(k+1)}) + \sigma(\widetilde{\mathcal{A}}(i)^{(k)}, \widetilde{\mathcal{A}}(i)^{(k+1)})$. We can observe that we only access a task once during lifelong learning, so*

we actually access the pseudo distributions, represented by $\widetilde{\mathcal{A}}(i)^{(k)}), k = 2, \ldots, t-i+1$, where the superscript k denotes that $\widetilde{\mathcal{A}}(i)^{(1)}$ is refined for (k) times, resulting in $\widetilde{\mathcal{A}}(i)^{(k)}$ through the subsequent GRMs process. We use $\tilde{A}(i)^{(0)}$ and $\tilde{A}(i)^{(1)}$ to represent D_T^i and D_S^i for simplicity. We use $\tilde{A}(i)_{\mathcal{X}}^{(k)}$ to represent the marginal distribution along $\mathcal{X}$. We assume that $q_{i,k} \neq q_{j,k}, \forall_{k \in \{1,\ldots,t-1\}, i \neq j}$. This assumption is reasonable in practice since the accumulated error by a model for different datasets is different due to the difference in the complexities of the databases used for training.

H.2. Theoretical analysis and empirical results

Lemma 5 *Let $\tilde{S}_i^{(t-i+1)}$ be the joint distribution over $\mathcal{X} \times \mathcal{Y}$. Let us consider the Assumption 1 from above, and we also assume that we have two different databases, the more complex dataset H and the simpler dataset E, where 'complex' and 'simple' represent the complexity of the data (images) contained in the database. We use $\tilde{H}_i^{(t-i+1)}$ and $\tilde{E}_i^{(t-i+1)}$ to represent that H and E are assigned to the i-th task which is used for training the model $(t-i+1)$ times. We assume that we have a single model $\mathcal{M}(\theta, \varsigma, \varphi)$, enabled with GRMs, which is trained using H and E databases, respectively. The accumulated error terms for H and E would be different and we assume that the error obtained for learning each task is performed under identical training conditions and model initialization. We use the notation $\Delta_H = \Psi(\tilde{H}_{i,X}^{(k)}, \tilde{H}_{i,X}^{(k+1)}) + \sigma(\tilde{H}_i^{(k)}, \tilde{H}_i^{(k+1)})$ and $\Delta_E = \Psi(\tilde{E}_{i,X}^{(k)}, \tilde{E}_{i,X}^{(k+1)}) + \sigma(\tilde{E}_i^{(k)}, \tilde{E}_i^{(k+1)})$. The model $\mathcal{M}(\theta, \varsigma, \varphi)$ tends to have a lower performance on H when compared to that on E, while learning one of these databases as a new task. Consequently, the model tends to have a larger risk on H than E, corresponding to the complexity degree of the data contained in the respective database.*

Proof *We can have the risk bound for both H and E after the t-th task learning, derived from Theorem 2, as:*

$$\mathrm{R}(h, H) \leq \mathrm{R}'(h, \tilde{h}_i^{(t-i+1)}, \tilde{H}_i^{(t-i+1)}) + \sum_{k=0}^{t-i+1} \Delta_H = \mathrm{R}_{\tilde{H}_i^{(t-i+1)}} \tag{35}$$

$$\mathrm{R}(h, E) \leq \mathrm{R}'(h, \tilde{h}_i^{(t-i+1)}, \tilde{E}_i^{(t-i+1)}) + \sum_{k=0}^{t-i+1} \Delta_E = \mathrm{R}_{\tilde{E}_i^{(t-i+1)}} \tag{36}$$

$\mathrm{R}_{\tilde{\mathrm{H}}_{\mathrm{i}}^{(\mathrm{t-i+1})}} > \mathrm{R}_{\tilde{\mathrm{E}}_{\mathrm{i}}^{(\mathrm{t-i+1})}}$ *since each Δ_H is larger than Δ_E.*

In the following, we investigate the theoretical results for Lemma 5 through experiments. We train a single model, enabled with GRMs, under the MNIST, SVHN and SVHN (MSS sequence) lifelong learning and evaluate the target risk on the MNIST. We also train a single model with GRMs under the CIFAR10, SVHN and SVHN (CSS sequence) lifelong learning and evaluate the target risk on CIFAR10 database. We consider that MNIST, showing grey-level images of handwritten digits, is an easy dataset, corresponding to E from the Lemma 5 above, while CIFAR10 which contains complex images is considered as a dataset, corresponding to H from the Lemma 5 above. We present the results in Figure 3 where we can observe that the target risk on the CIFAR10 is an upper bound on the MNIST when learning every new task.

Theorem 3 *The performance of a single model with GRMs would be influenced when changing of order in which the tasks are learnt during the lifelong learning.*

Proof *Let us consider the Assumptions 1 and 2 from above, in Appendix H.1. We have the accumulated errors for a model across all tasks, by considering Theorem 1 and Lemma 1, we obtain a relationship similar to Eq. (21), corresponding to* Lemma 1 *from the paper :*

$$\sum_{i=1}^{t} \mathrm{R}(h, \mathcal{A}(i)) \leq \sum_{i=1}^{t} \left(\mathrm{R}'(h, \tilde{h}_i^{(t-i+1)}, \tilde{\mathcal{A}}(i)^{(t-i+1)}) + \sum_{k=0}^{t-i} \left(\Psi(\tilde{\mathcal{A}}(i)_X^{(k)}, \tilde{\mathcal{A}}(i)_X^{(k+1)}) + \sigma(\tilde{\mathcal{A}}(i)^{(k)}, \tilde{\mathcal{A}}(i)^{(k+1)}) \right) \right) \tag{37}$$

We can further simplify the relationship Eq. (37), by using $q_{i,k} \in \mathbf{Q}$, as notation for the last expression from the right side, resulting in :

$$\sum_{i=1}^{t} \mathrm{R}(h, \mathcal{A}(i)) \leq \sum_{i=1}^{t} \left(\mathrm{R}'(h, \tilde{h}_i^{(t-i+1)}, \tilde{\mathcal{A}}(i)^{(t-i+1)}) + \sum_{k=0}^{t-i} q_{i,k} \right) \tag{38}$$

Theorem 3 provides an explicit way to analyse the risk bound when changing the learning order of tasks during the lifelong learning. Firstly, we consider an extreme case that the complexity of each training dataset in $\tilde{\mathcal{A}}$ is introduced in the

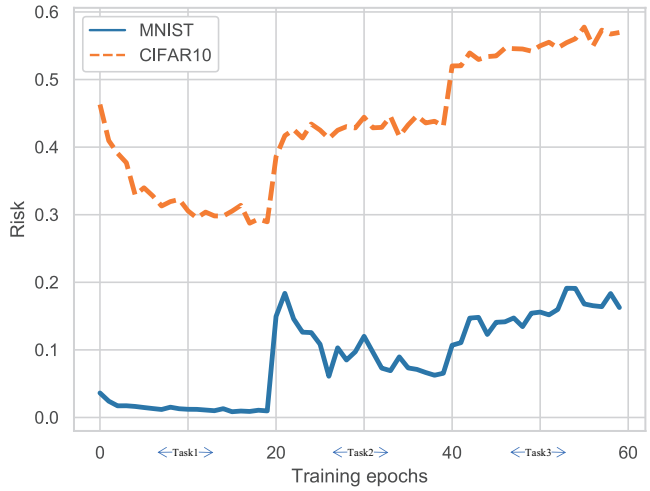


Figure 3. The accumulated target risk on MNIST and CIFAR10, estimated by a single VAE component model with GRMs, with a classifier used for evaluating the risk, under MSS and CSS lifelong learning, respectively.

increasing order of their complexity $q_{i,1} > q_{j,1}$, $i > j$ during the lifelong learning. This means that the model starts with being trained using an easy task and then gradually starts learning more complex tasks. We also consider the inverse case when we consider the training datasets in $\tilde{\mathcal{A}}$ according to their decreasing complexity order $q_{i,1} < q_{j,1}$, $i > j$, during the lifelong learning, meaning that the model starts learning a complex task and then gradually learns simpler tasks. Nevertheless, $q_{j,1} > q_{i,1}$ can not guarantee that $q_{j,k} > q_{i,k}$, $k > 1$, because the complexity for a database is relative to the parameters learnt by the network, and this can change after learning the same database several times. From the relationship (38), we can see that if we change the order of tasks, some of the tasks would be reused for training more times than others. From Assumption 2, we have known that the complexity for each dataset is different. When we change the order of tasks, we actually change the order within $\mathbf{Q}$, resulting in learning variations within the risk bound for a single model with GRMs. In order to investigate Theorem 3, we train a single model with GRMs under the lifelong learning of MSC and CMS database sequences, respectively. We evaluate the accumulated target risk $\sum_{i=1}^{t} \mathrm{R}(h, \mathcal{A}(i))$ across all tasks. The results are shown in Figure 4, where we observe that the performance of the model is influenced by changing the learning order of tasks during the lifelong learning.

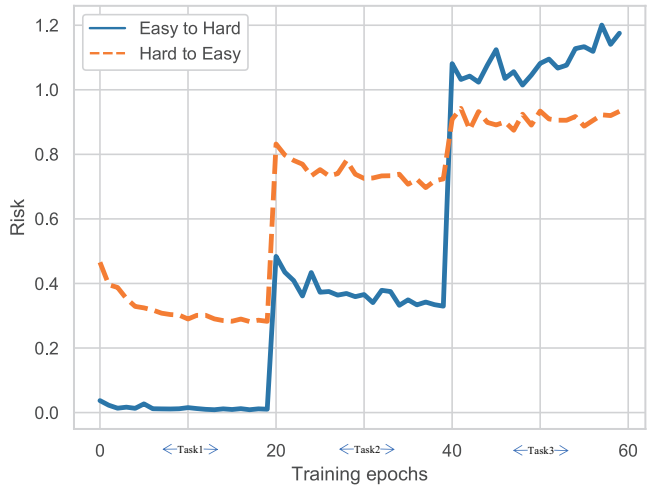


Figure 4. The accumulated target risk evaluated on all datasets, estimated by a single model with GRMs under the MNIST, SVHN and CIFAR10 lifelong learning and when learning the probabilistic representations of the same databases, but in reversed order. Easy-to-Hard and Hard-to-Easy denote that the model is trained under MSC and CMS lifelong learning, respectively.

In the following we show how can we make the LIMix model robust to changes in the learning order of tasks.

Lemma 6 *If we have a mixture model with fewer components than the number of tasks being learnt, then the result depends on the order for learning the tasks during the lifelong learning.*

Proof *Based on Assumptions 1 and 2, we can replace each approximation distribution such as* $\tilde{S}_{b_i'}^{(t-\hat{b}_i+1)}$ *in Eq.* (21) *by using* $\mathcal{A}$ *and* $\widetilde{\mathcal{A}}$*, resulting in :*

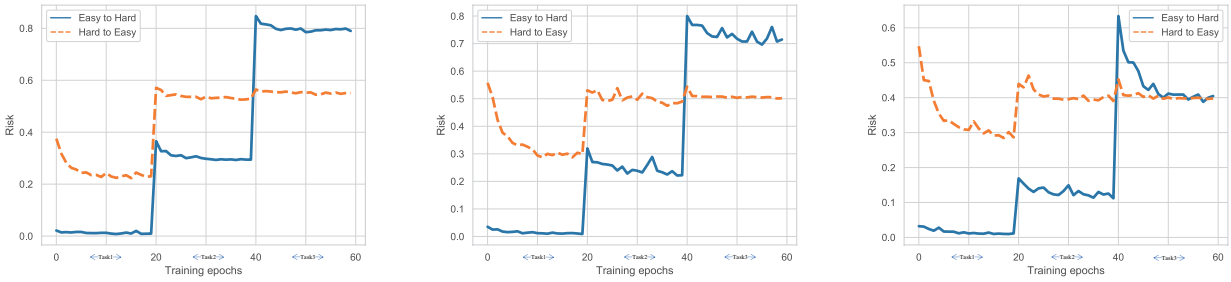
$$\begin{aligned}\sum_{i=1}^{t}\mathrm{R}\left(h,\mathrm{A}(i)\right) \leq & \sum_{i=1}^{\mathrm{card}(B)}\left(\mathrm{R}'(h,\tilde{h}_{b_i}^{1},\tilde{\mathrm{A}}(b_i)_{b_i}^{1})+\Psi(\tilde{\mathrm{A}}(b_i)_{b_i,X},\tilde{\mathrm{A}}(b_i)_{b_i,X}^{1})\ +\sigma(S_{b_i},\tilde{S}_{b_i}^{1})\right)\\ &+\sum_{i=1}^{\mathrm{card}(B')}\left(\mathrm{R}'(h,\tilde{h}_{b_i'}^{(t-\hat{b}_i+1)},\tilde{\mathrm{A}}(b_i')_{b_i'}^{(t-\hat{b}_i+1)})+\sum_{k=0}^{\hat{b}_i-1}\left(\Psi(\tilde{\mathrm{A}}(b_i')_{b_i',X}^{k},\tilde{\mathrm{A}}(b_i')_{b_i',X}^{(k+1)})+\sigma(\tilde{\mathrm{A}}(b_i')_{b_i'}^{k},\tilde{\mathrm{A}}(b_i')_{b_i'}^{(k+1)})\right)\right)\\ &=\mathrm{R}_{\mathrm{mixture}}\end{aligned} \tag{39}$$

We can further simplify expression (39) by using the simplifying notations $\mathbf{Q}$ *:*

$$\sum_{i=1}^{t}\mathrm{R}\left(h,\mathrm{A}(i)\right) \leq \sum_{i=1}^{\mathrm{card}(B)}\left(\mathrm{R}'(h,\tilde{h}_{b_i}^{1},\tilde{\mathrm{A}}(b_i)_{b_i}^{1})+q_{b_i}\right)+\sum_{i=1}^{\mathrm{card}(B')}\left(\mathrm{R}'(h,\tilde{h}_{b_i'}^{(t-\hat{b}_i+1)},\tilde{\mathrm{A}}(b_i')_{b_i'}^{(t-\hat{b}_i+1)})+\sum_{k=0}^{\hat{b}_i-1}\left(q_{b_i',k}\right)\right)=\mathrm{R}_{\mathrm{mixture}} \tag{40}$$

It notes that if the number of components is less than the number of tasks, $\mathrm{card}(B')\neq\varnothing$*. If we change the order of tasks, which corresponding to the change of orders in* $\mathbf{Q}$*, then* $\mathrm{R}_{\mathrm{mixture}}$ *would be changed, caused by the change of* $\sum_{k=0}^{\hat{b}_i-1}\left(q_{b_i',k}\right)$ *when we change* b_i' *(see details in Assumption 2). This conclusion leads to the fact that for a mixture model, if the number of components is less than the number of tasks, a certain component must learn more than one task in which GRMs is used for retraining the prior samples in order to overcome forgetting. If we change the order of tasks, this component would firstly learn the later task of the original order, which would lead to a different accumulated error during the GRMs process when comparing with the original order of learning the tasks.*

In the expression (Eq. (40)) we find that if the number of components in the mixture model is equal to the number of tasks, $B'=\emptyset$, then the performance in the mixture does not change when changing the order of tasks since the last term of the



(a) BatchEnsemble. (b) Components in LIMix share part of sets of parameters. (c) Each component has independent parameters from each other, in LIMix.

Figure 5. The accumulated target on all datasets, estimated by LIMix under the MNIST, SVHN and CIFAR10 lifelong learning and their learning in inverse order during the lifelong learning. Easy-to-Hard and Hard-to-Easy denote that the model is trained under MSC and CMS lifelong learning, respectively.

right-hand side in Eq. (39) is empty. As $\mathrm{card}(B')$ increases, the performance in the mixture model tends to be influenced by changing the order for learning the given tasks because the term $\sum_{k=0}^{\hat{b}_i-1}\left(q_{b'_i,k}\right)$ can be different when we change the order of $\mathbf{Q}$.

During the experiments we find that the performance of LIMix and Batch Ensemble is affected by the change of orders of tasks. We show the empirical results in Figures 5-a and b. This is because the components in LIMix and BatchEnsemble share most of the parameters with each other. Once the learning of the first task was finished, the parameters of the shared models will be fixed during the learning of the following task. From the results provided in Figure 5-c, we can observe that when CIFAR10 database is used as the first learning task, LIMix would achieve a smaller target risk across all tasks than the case when the probabilistic representation of the MNIST database is used as the first learnt task. In order to allow LIMix to be robust to the change of orders of tasks in the following we do not share the parameters between components in LIMix and each component has an independent parameter set. We train this mixture model using the same setting and present the accumulated target risk in Figure 5-b, where we can observe that the performance of LIMix does not change when changing the order of tasks. This result also proves Lemma 6.

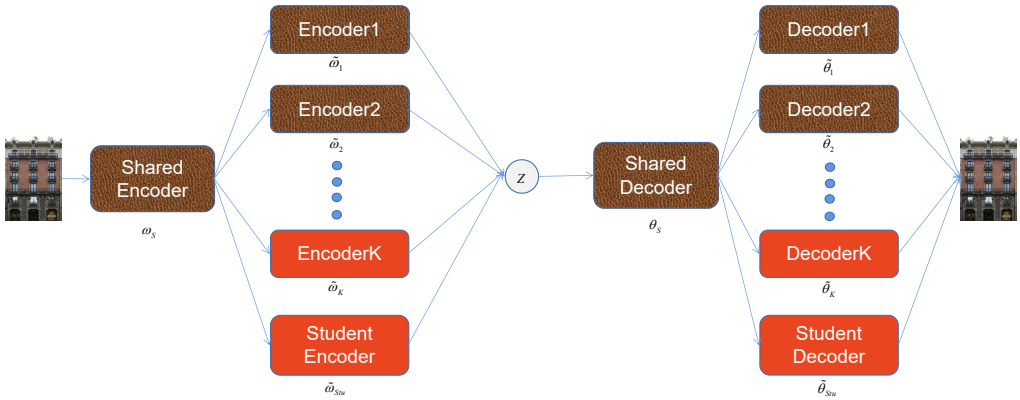


Figure 6. Diagram showing the learning structure for the proposed LIMix mixture model. Only red components update parameters during lifelong learning.

I. Learning a compressed Student model and implementation

I.1. The derivation of the loss function for the Student

In order to reduce the complexity of the proposed LIMix mixture model, we propose to share most parameters of the generator and the inference models, where parameters $\theta_i = \{\theta_S, \tilde{\theta}_i\}$ and $\omega_i = \{\omega_S, \tilde{\omega}_i\}$ of each component consists of the shared part $\{\theta_S, \omega_S\}$ and the individual part $\{\tilde{\theta}_i, \tilde{\omega}_i\}$. The network corresponding to each component is built on the top of the shared component. Once the learning of the first task is finished, the parameters of the shared module will be

fixed, and we only update the parameters of a certain selected component during the following learning. We also learn a compressed Student model which shares the same network architecture with the component. The structure of the LIMix model is presented in Figure 6. In order to transfer the knowledge from the Teacher (represented by the mixture model) to the Student module, we introduce a new knowledge distillation loss, defined as :

$$D_{KL}\left(\mathbb{P}_{\theta_1}, \mathbb{P}_{\theta_2}, \ldots, \mathbb{P}_{\theta_K} \mid\mid \mathbb{P}_{\theta_{stu}}\right) \tag{41}$$

where each $\mathbb{P}_{\theta_i}$ is the distribution approximated by the i-th component in the Teacher module. $\mathbb{P}_{\theta_{stu}}$ is the distribution approximated by the Student module. However, solving Eq. (41) is intractable through optimization and we rewrite it as :

$$D_{KL}\left(\mathbb{P}_{\theta_1}, \mathbb{P}_{\theta_2}, \ldots, \mathbb{P}_{\theta_K} \mid\mid \mathbb{P}_{\theta_{stu}}\right) \approx \sum_{i=1}^{K} D_{KL}\left(\mathbb{P}_{\theta_i} \mid\mid \mathbb{P}_{\theta_{stu}}\right) \tag{42}$$

where each $D_{KL}(\mathbb{P}_{\theta_i} \mid\mid \mathbb{P}_{\theta_{stu}})$ can be expressed as :

$$D_{KL}\left(\mathbb{P}_{\theta_i} \mid\mid \mathbb{P}_{\theta_{stu}}\right) = \mathbb{E}_{\mathbb{P}_{\theta_i}}\left[\log p_{\theta_i}(\mathbf{x}) - \log p_{\theta_{stu}}(\mathbf{x})\right] \tag{43}$$

where $p_{\theta_i}(\mathbf{x})$ and $p_{\theta_{stu}}(\mathbf{x})$ are the density functions for $\mathbb{P}_{\theta_i}$ and $\mathbb{P}_{\theta_{stu}}$, respectively. We can omit the first term since we only update the student's parameters θ_{stu} during the knowledge distillation. Then the optimization problem is redefined by maximizing :

$$\mathbb{E}_{\mathbb{P}_{\theta_i}} \log p_{\theta_{stu}}(\mathbf{x}) \tag{44}$$

The knowledge distillation loss is defined as :

$$\sum_{i=1}^{K} \mathbb{E}_{\mathbb{P}_{\theta_i}} \log p_{\theta_{stu}}(\mathbf{x}) \tag{45}$$

where $\log p_{\theta_{stu}}(\mathbf{x})$ can be optimized by maximizing the Evidence Lower Bound (ELBO). Eventually, the loss function for the Student module contains two terms :

$$\mathcal{L}_{stu} = \mathbb{E}_{\mathbf{x}\sim S_{t,X}} \log p_{\theta_{stu}}(\mathbf{x}) + \sum_{i=1}^{K} \mathbb{E}_{\mathbb{P}_{\theta_i}} \log p_{\theta_{stu}}(\mathbf{x}), \tag{46}$$

where we decompose each log-likelihood function as ELBO, resulting in :

$$\begin{aligned} &\underbrace{\mathbb{E}_{q_{\omega_{Stu}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\theta_{Stu}}(\mathbf{x} \mid \mathbf{z})\right] - D_{KL}\left[q_{\omega_{Stu}}(\mathbf{z} \mid \mathbf{x}) \mid\mid p(\mathbf{z})\right]}_{\text{ELBO on the t-th task}} \\ &+ \underbrace{\sum_{i=1}^{K} \mathbb{E}_{\mathbf{x}'\sim\mathbb{P}_{\theta_i}}\left(\mathbb{E}_{q_{\omega_{Stu}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\theta_{Stu}}(\mathbf{x} \mid \mathbf{z})\right] - D_{KL}\left[q_{\omega_{Stu}}(\mathbf{z} \mid \mathbf{x}) \mid\mid p(\mathbf{z})\right]\right)}_{\text{Knowledge distillation optimization}} \end{aligned} \tag{47}$$

It notes that we only train a Student model under the unsupervised learning setting. In the future work, we would like to extend the LIMix to train a Student module under the supervised learning setting. We also provide the pseudocode in Algorithm 1 and the learning process of the Teacher from LIMix under unsupervised learning in Figure 7, where the Student learning is omitted for the sake of simplification. After the learning of the $(t-1)$-th task is finished, we calculate $\{K_{i,1}, \ldots, K_{n_G,K}\}$ by Eq. (7) from the paper, where K is the number of components. Then $p(c_i^t = j \mid c_{-i}^t, a)$ evaluates the probability of the i-th data belonging to the j-th component using Eq. (4) from the paper for $j \leq K$ or Eq. (7) when generating a new component, $j = K+1$. We use a group of samples from the t-th task to calculate the average probability $p(c^t = j) = \frac{1}{n_G}\sum_{s=1}^{n_G} p(c_s^t = j \mid c_{-s}^t, a), j = 1, \ldots, K+1$. If $c^t = K+1$, then we add a new component to the mixture, otherwise, we choose a component according to c^t for the t-th task learning.

I.2. Knowledge assimilation by the Student

In this section, we theoretically explain the weakness in the learning by the Student by using the proposed theoretical framework. Although knowledge distillation (KD) has been used as a training procedure in lifelong learning [21], there is no theoretical analysis of the forgetting behaviour in the model when using KD for knowledge transfer. This paper is the

Algorithm 1: LIMix lifelong unsupervised learning

Input: All training databases
Output: The model's parameters
for $i < taskCount$ **do**
  for $index < batchCount$ **do**
    Teacher learning: ;
    if *isAdd* == *True* **then**
      Generate samples $\mathbf{x}$ from the selected component ;
      Train the selected component with $\mathbf{x}$ and samples drawn from the i-th task by maximizing the log-likelihood function $F(\cdot \mid c^i, \theta_{c^i}, \omega_{c^i})$;
    end
    else
      Train the selected component with samples drawn from the i-th task by maximizing the log-likelihood function $F(\cdot \mid c^i, \theta_{c^i}, \omega_{c^i})$;
    end
    Knowledge distillation ;
    Transfer knowledge from the teacher and the current task to the student by $\mathcal{L}_{stu}$;
    Gating mechanism based Dirichelt process for the selection and expansion: ;
    $p(c^{(i+1)} = j) = \frac{1}{n_G} \sum_{s=1}^{n_G} p(c_s^{(i+1)} = j \mid c_{-s}^{(i+1)}, a)$;
    $c^{(i+1)} = j^*$ where $\frac{1}{n_G} \sum_{s=1}^{n_G} p(c_s^{(i+1)} = j^* \mid c_{-s}^{(i+1)}, a) > \frac{1}{n_G} \sum_{s=1}^{n_G} p(c_s^{(i+1)} = k \mid c_{-s}^{(i+1)}, a), k \neq j^*$;
  end
end

first research study to provide insights into how a Student loses knowledge (information) from a strong Teacher module. Firstly, from Eq. (46), we observe that the Student learns the prior knowledge from each expert from the Teacher module in LIMix and its performance on the target set of prior tasks depends inevitably on the quality of the approximation distribution modelled by each expert in LIMix. In the following we provide the risk bound for the Student module.

Lemma 7 *For a given optimal LIMix, where the number of experts matches the number of tasks and a classifier, the Student module, $h_s \in \mathcal{H}$ is trained along with LIMix during the lifelong learning. The risk bound for the Student at the t-th task learning is defined as :*

$$\begin{aligned}\sum_{i=1}^{t} \mathrm{R}\left(h_s, S_i\right) \leq \sum_{i=1}^{t-1} \Big(\mathrm{R}'(h_s, \tilde{h}_i^2, \tilde{S}_i^2) + \Psi(S_{i,X}, \tilde{S}_{i,X}^2) + \sigma(S_i, \tilde{S}_i^2)\Big) \\ + \mathrm{R}'(h_s, \tilde{h}_t^1, \tilde{S}_t^1) + \Psi(S_{t,X}, \tilde{S}_{t,X}^1) + \sigma(S_t, \tilde{S}_t^1)\end{aligned} \tag{48}$$

Although LIMix can lead to a tight bound for the risk, according to Lemma 2, the gap on the risk bound, given by Eq. (48) for the Student is still large. This is because the Student learns the knowledge of prior tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_{t-1}\}$ from the degenerated distribution $\{\tilde{S}_{1,X}^2, \ldots, \tilde{S}_{(t-1),X}^2\}$. Additionally, since the Student has only a single network as a classifier which is trained on the multiple domains. This can also lead to a degenerated performance on the target risk due to the negative transfer. One solution to improve the generalization of the Student in the optimal LIMix is to use regularization approaches [9], which would regularize the network optimization to relieve the negative transfer.

In the following, we analyze the performance of the Student when the Teacher module of the LIMix dynamically changes its network architectures.

Lemma 8 *Let $h_s \in \mathcal{H}$ be a Student which is trained along with LIMix that dynamically changes its network architectures during the lifelong learning. The risk bound for h_s at the t-th task learning is defined as :*

$$\begin{aligned}\sum_{i=1}^{t} \mathrm{R}\left(h_s, S_i\right) \leq \sum_{i=1}^{\mathrm{card}(B)} \Big(\mathrm{R}'(h_s, \tilde{h}_{b_i}^2, \tilde{S}_{b_i}^2) + \Psi(S_{b_i,X}, \tilde{S}_{b_i,X}^2) + \sigma(S_{b_i}, \tilde{S}_{b_i}^2)\Big) + \sum_{i=1}^{\mathrm{card}(B')} \Big(\mathrm{R}'(h_s, \tilde{h}_{b'_i}^{(t-\hat{b}_i+1)}, \tilde{S}_{b'_i}^{(t-\hat{b}_i+1)}) \\ + \sum_{k=0}^{\hat{b}_i-1} \Big(\Psi(\tilde{S}_{b'_i,X}^k, \tilde{S}_{b'_i,X}^{(k+1)}) + \sigma(\tilde{S}_{b'_i}^k, \tilde{S}_{b'_i}^{(k+1)})\Big)\Big) + \mathrm{R}'(h_s, \tilde{h}_t^1, \tilde{S}_t^1) + \Psi(S_{t,X}, \tilde{S}_{t,X}^1) + \sigma(S_t, \tilde{S}_t^1) = \mathrm{R}_s^{\mathrm{mixture}}\end{aligned} \tag{49}$$

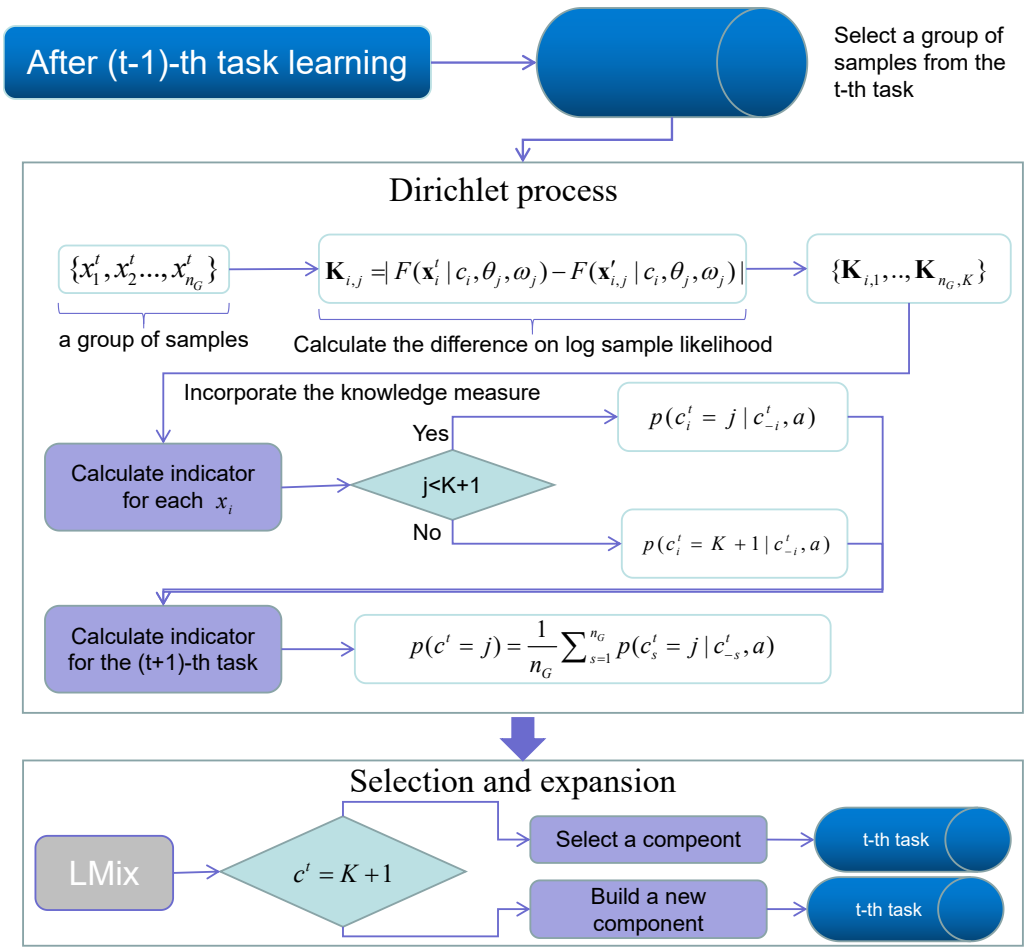


Figure 7. The learning process of the Teacher of LIMix under the unsupervised learning.

It notes that B in Eq. (49) does not contain the t-th task. According to Lemma 3, see Appendix E, we have $\mathrm{R}_s^{\text{mixture}} \geq \mathrm{R}^{\text{mixture}}$. Lemma 8 is easily proved since the Student's classifier h_s is trained on the degenerated distributions $\hat{S}_{b_i}^2$ even if the b_i-th task is only trained once by a certain expert in LIMix.

Eventually, we implement the Student only for unsupervised learning and we find that the Student performs worse than the Teacher in LIMix. This can also be explained by Eq, (48) and Eq, (49), although these derivations are used only for the classification tasks.

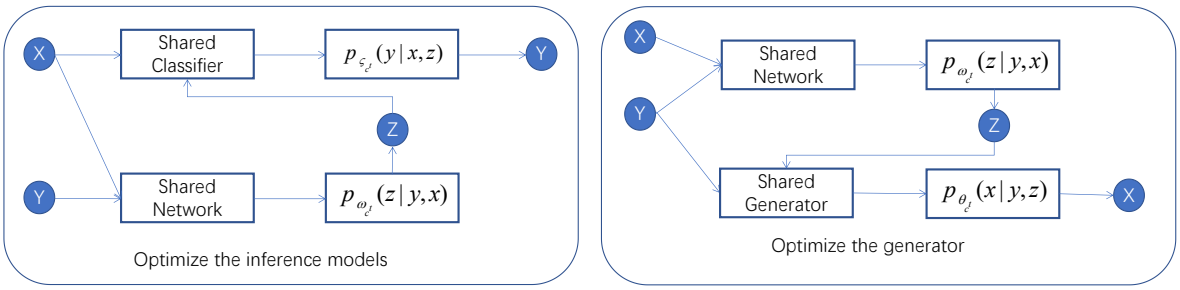
I.3. The implementation of LIMix in classification tasks

During the prediction tasks, the inference models in each component are built on top of the shared inference model. In order to overcome forgetting the learnt knowledge, we do not update the shared parameters after the first task is finished. Instead, we create a new component in the mixture model by initializing its parameters with those corresponding to the existing mixture' component, which has the highest sample log-likelihood for the incoming task. Then we train the newly added component considering just a few iterations in order to learn the new task. The optimization of the inference model and the generator during the training of the LIMix model with a certain task, is illustrated in the diagrams from Figures 8.

J. Learning prediction tasks and image-to-image translation

In the following we consider two applications: image classification and image-to-image translation (IToI). Conditional VAEs (CVAEs) is one of the most popular generative models [16] used in predictive tasks, defined by :

$$\log p_{\theta_i}(y \mid \mathbf{x}) \geq \mathbb{E}_{q_{\omega_i}(\mathbf{z}|\mathbf{x},y)}[\log p_{\theta_i}(y \mid \mathbf{x}, \mathbf{z})] - D_{KL}(q_{\omega_i}(\mathbf{z} \mid \mathbf{x}, y) \mid\mid p_{\theta_i}(\mathbf{z} \mid \mathbf{x})). \tag{50}$$



(a) Optimization of the inference models. (b) Optimization of the generator.

Figure 8. Diagrams showing the supervised learning structure when optimizing a single component for the proposed LIMix mixture model.

However, CVAE requires an additional prior network $p_{\theta_i}(\mathbf{z} \mid \mathbf{x})$ and the recognition network $q_{\omega_i}(\mathbf{z} \mid \mathbf{x}, y)$ is conditioned on two variables $\mathbf{x}, y$, which uses more parameters and increases the computing time during both training and testing. The conditional distribution $p_{\theta_i}(y \mid \mathbf{x}, \mathbf{z})$, where y is conditioned on $\mathbf{x}, \mathbf{z}$, so we can compress the network $q_{\omega_i}(\mathbf{z} \mid \mathbf{x}, y)$ into $q_{\omega_i}(\mathbf{z} \mid y)$ given that $\mathbf{z}$ contains information about y is enough for modelling $p_{\theta_i}(y|\mathbf{x}, \mathbf{z})$. We also consider to replace the prior network $p_{\theta_i}(\mathbf{z} \mid \mathbf{x})$ by using a simple normal distribution $\mathcal{N}(0, I)$ which does not depend on $\mathbf{x}$, benefiting on reducing the whole model complexity. Then, we rewrite Eq. (50) to be used as the loss function for Image-to-Image translation, $\mathcal{L}_{\text{IToI}}(\mathbf{x}, y, \theta_{c^t}, \omega_{c^t})$:

$$\mathcal{L}_{\text{IToI}}(\mathbf{x}, y, \theta_{c^t}, \omega_{c^t}) = \mathbb{E}_{q_{\omega_{c^t}}(\mathbf{z}|y)}\left[\log p_{\theta_{c^t}}(y \mid \mathbf{x}, \mathbf{z})\right] - D_{KL}(q_{\omega_{c^t}}(\mathbf{z}|y) \,||\, p_{\theta_{c^t}}(\mathbf{z})). \tag{51}$$

This objective function can also benefit on the efficient inference process at the testing phase $y \sim p_{\theta_{c^t}}(y \mid \mathbf{x}, \mathbf{z})$ and $\mathbf{z} \sim p_{\theta_{c^t}}(\mathbf{z})$. In the Image-to-Image translation (IToI) task where y belongs to the image domain, we use a large threshold in order to allow growing a component to adapt the incoming task for each task switch since we can not generate previously learned knowledge from $p_{\theta_{c^t}}(y \mid \mathbf{x}, \mathbf{z}), \forall i = 1, \ldots, t-1$ when learning the t-th task.

In the classification task, y belongs to the discrete domain (one-hot vector) and $p_{\varsigma_{c^t}}(y \mid \mathbf{x}, \mathbf{z})$ is implemented as a classifier, we replace $p_{\varsigma_{c^t}}(\mathbf{z} \mid \mathbf{x})$ by $p(\mathbf{z}) = \mathcal{N}(0, I)$ in Eq. (50), used as the objective function for training the classifier:

$$\mathcal{L}_{\text{P}}(\mathbf{x}, y, \varsigma_{c^t}, \omega_{c^t}) = \mathbb{E}_{q_{\omega_{c^t}}(\mathbf{z}|\mathbf{x},y)}\left[\log p_{\varsigma_{c^t}}(y \mid \mathbf{x}, \mathbf{z})\right] - D_{KL}(q_{\omega_{c^t}}(\mathbf{z} \mid \mathbf{x}, y) \,||\, p(\mathbf{z})). \tag{52}$$

However, Eq. (52) is only used to train the classifier. In order to learn the density functions, for each component we incorporate the class label, which can be rewritten as $p_{\theta_{c^t}}(\mathbf{x} \mid y)$. Its ELBO can be defined as :

$$\mathcal{L}_{\text{G}}(\mathbf{x}, y, \theta_{c^t}, \omega_{c^t}) = \mathbb{E}_{q_{\omega_{c^t}}(\mathbf{z}|\mathbf{x},y)}\left[\log p_{\theta_{c^t}}(\mathbf{x}|y, \mathbf{z})\right] - D_{KL}(q_{\omega_{c^t}}(\mathbf{z} \mid \mathbf{x}, y) \,||\, p(\mathbf{z})). \tag{53}$$

In practice, we do not optimize $\mathcal{L}_{\text{G}}$ and $\mathcal{L}_{\text{P}}$ by using the same optimizer, but separately, considering the same mini-batch of data samples.

The derivation of VAE objective function. Please see the main derivation of the lower bound to the conditional log-likelihood in Appendix-S1 from the Supplementary Material in [16].

K. Experiment settings for the ablation study and theoretical results

K.1. Hyperparameter setting and network architecture

All models are implemented by using TensorFlow [1]. We use Adam optimization algorithm [6], considering a learning rate of 0.0002 and $\beta = 0.5$. We consider images of input size of 32×32 and use kernel size of $3\tilde{3}$, the shared encoder ω_S is a CNN which has four layers of units $\{128, 256, 512, 1024\}$ while $\tilde{\omega}_i$ is a fully connected network with two layers $\{1024, 100\}$. For the decoder, the shared part θ_S is a CNN which has three layers consisting of a fully connected layer with 256 * 8 * 8 units and other two convolution layers $\{256, 256\}$. $\tilde{\theta}_i$ is a CNN consisting of three layers $\{256, 256, 3\}$. For the input size of 64×64, we implement the shared encoder ω_S as a CNN $\{64, 128, 256\}$ and $\tilde{\omega}_i$ is a network consisting of a convolution layer 512 and three connected layers $\{1024, 256\}$. For the decoder, the shared part θ_S is a network consisting of a fully connected layer with 256 * 8 * 8 units and three convolution layers $\{256, 256, 256\}$. $\tilde{\theta}_i$ is a CNN with $\{256, 128, 3\}$ processing units.

K.2. The lifelong learning of complex datasets

Sub-ImageNet. We create the Sub-ImageNet by randomly selecting 60,000 samples from the ImageNet dataset [7] and assign 50,000 and 10,000 samples to be used as the training and testing sets, respectively.

We train various models for learning a sequence of tasks defined by complex data, such as CCCOS, consisting of CelebA, CADS, 3D-Chair, Omniglot and Sub-ImageNet databases, and the results are provided in Table 1 where all images are resized as $64 \times 64 \times 3$. The results of the proposed LIMix outperform the other methods when learning either MSFIR or CCCOS sequence of tasks, according to the results from Table 1 from page 7 of the paper, and Tab. 1, respectively.

	MSE					**SSMI**					**PSNR**				
Datasets	LGM	CURL	BE	LIMix	Stud	LGM	CURL	BE	LIMix	Stud	LGM	CURL	BE	LIMix	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-ImageNet	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

Table 1. The performance of various models under the CCCOS learning setting.

K.3. The estimation of the source risk and discrepancy

Initially, we investigate the theoretical results for Theorem 2 from the paper. Firstly, in order to calculate the discrepancy, we train a task-inference network that is used to infer the task label for the given data samples. Also we train an auxiliary classifier which is used to infer the prediction for class labels and is trained on the MNIST training set. For calculating the discrepancy on MNIST, we use the task-inference network to choose the images (images are sampled from the testing set or the generator distribution) that belong to MNIST. Then we have the source distribution (generated images are belong to MNIST) and the target distribution (the testing set of MNIST). Then we calculate the discrepancy between the source and target distribution using equation (14) from the paper, by using the model and the auxiliary classifier. For the source risk on MNIST, we generate the images and the associated labels inferred by the model before the current task learning, then we use the task-inference network to choose the images that belong to MNIST. Finally, we can calculate the source risk on the selected images with associated class labels by using the model which is training on the current task learning. This estimation process is used for the results presented in Figures 2c and 2d of the paper.

Additionally, we evaluate the discrepancy distance between different target distributions. We train two classifiers where the first classifier is only trained on MNIST, and the second classifier is trained on the joint training set of MNIST and a certain selected dataset (let us call this dataset as A). Then we calculate the discrepancy distance between MNIST and A by using these two classifiers. We report the results in Figure 9 where "$(MNIST, Fashion)$" represents the estimation of the discrepancy distance between MNIST and Fashion databases. We can observe from Figure 9 that the discrepancy distance is small when two domains are very similar (for example MNIST and RMNIST). This indicates that the proposed selection process can help LIMix to choose an appropriate component that has a small discrepancy to the new task since LIMix reuses the component that was trained on MNIST, to learn a similar dataset, such as RMNIST for example.

K.4. The settings for the continuous learning benchmark

This section refers to the second paragraph from Section 5.3 "Classification tasks" and to the results Table 3 from the paper. Permuted MNIST defines a series of tasks as parts of the same database. In this paper, we set 10 tasks where each task has a fixed permutation of pixels to the MNIST dataset. Split MNIST [20] contains five tasks where each task is built based on the data samples collected from two classes. For Permuted MNIST, we implement the classifier for each component by using a fully connected network consisting of two layers and each layer has 100 hidden units. For Split MNIST, we implement the classifier by using a network of two hidden layers and each layer has 256 hidden units. For the hyperparameter setting, we search the threshold from 80 to 120 for Permuted MNISTT and between 30 and 60 for Split MNIST.

Split CIFAR. In the following, we also evaluate our LIMix on a more challenging continuous learning benchmark, Split CIFAR [20]. As the same process from [11], we take CIFAR10 as the first task and create the following 5 tasks where each task collects samples from 10 consecutive classes from CIFAR100. We also adopt the same network architecture consisting of four convolutional layers and two hidden layers. It notes that we only build a shared classifier which only consists of four convolutional layers. When a new component is built, we only create a sub-classifier which has only two hidden layers and

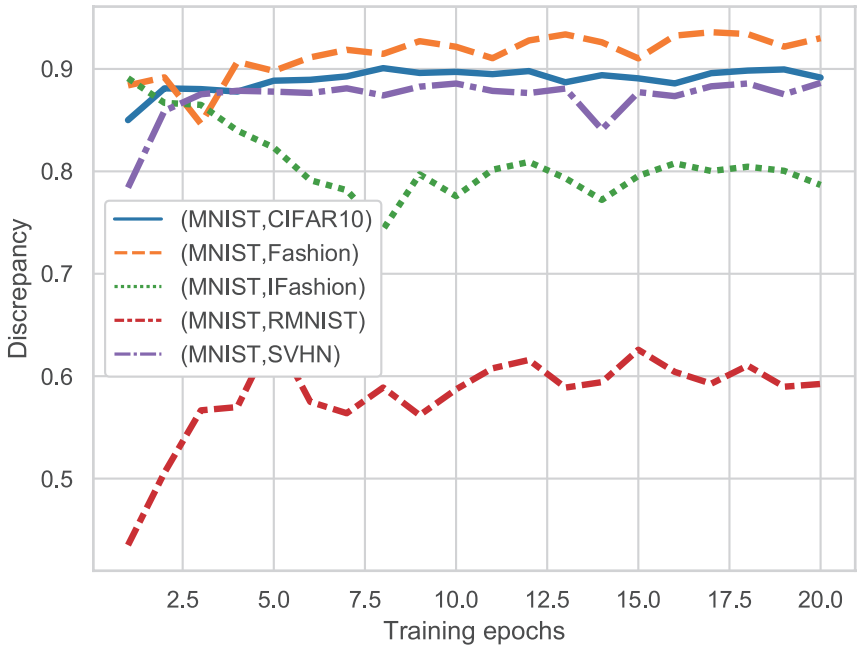


Figure 9. The estimation of the discrepancy distance between different domains.

reuses parameters from this shared classifier. The parameters of the shared classifier are only updated at the first task learning and will be frozen at the following task learning. We search the threshold for LIMix from 80 to 10 in the experiments. We perform five independent runs for the proposed LIMix. The results are reported in Table 2 where all results are cited from [11] except the proposed LIMix. "6 C" represents that LIMix has learnt 6 components. LIMix can achieve the optimal results when the number of components matches the number of tasks and there have no accumulated errors (See details in Lemma 2 of the paper). Then the performance on the target set of each individual tasks is relying on the generalization ability of the associated component, as discussed in Lemma 2 of the paper.

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

Table 2. Results of Split CIFAR.

K.5. Learning a long sequence of tasks under multiple domain setting

In this section, we evaluate the performance on a long sequence of tasks: MNIST, SVHN, Fashion, IFashion, RMNIST, rated Fashion (RFashion) and CIFAR10 (MSFIRRC). We train LIMix with a threshold of 180 for the lifelong learning of MSFIRRC. After training, LIMix has five components for the Teacher module, where the first component learns MNIST and RMNIST. The third component learns Fashion and RFashion. We report the results in Table 3, where LIMix achieves a better average result than BE. Also LIMix obtains better results for six out of seven individual databases, with BE having better results only for MNIST database, which contains the simplest images from all the databases under consideration in these experiments.

L. The complexity evaluation for the LIMix model

In the following we provide the total number of parameters required by each method. In the Tables 4, 5, 6 'Stu' represents the number of parameters for the Student module in LIMix, compared with the number of parameters required by other methods.

Dataset	LIMix	BE [17]
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

Table 3. Classification accuracy of various models after the MSFIRRC's lifelong learning.

Model	LGM [13]	CURL [14]	BE [17]	LIMix	Stu
Parameters	3.3×10^8	2.3×10^8	3.6×10^8	2.1×10^8	1.4×10^8

Table 4. The number of parameters of various models under MSFIR unsupervised learning.

Model	LGM [13]	CURL [14]	BE [17]	LIMix	Stu
Parameters	1.9×10^9	2.0×10^9	2.0×10^9	7.2×10^8	1.7×10^8

Table 5. The number of parameters of various models under the CelebA, CACD, 3D-Chair, Omniglot and Sub-ImageNet (CCCOS) lifelong learning setting.

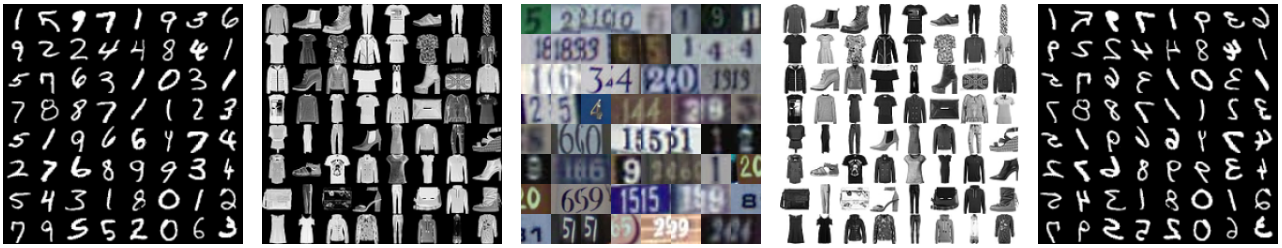
Model	LGM	CURL	BE [17]	The proposed	MRGANs [18]
Parameters	5.9×10^8	3.3×10^8	3.9×10^8	3.4×10^8	3.3×10^8

Table 6. The number of parameters of various models under the lifelong supervised learning.

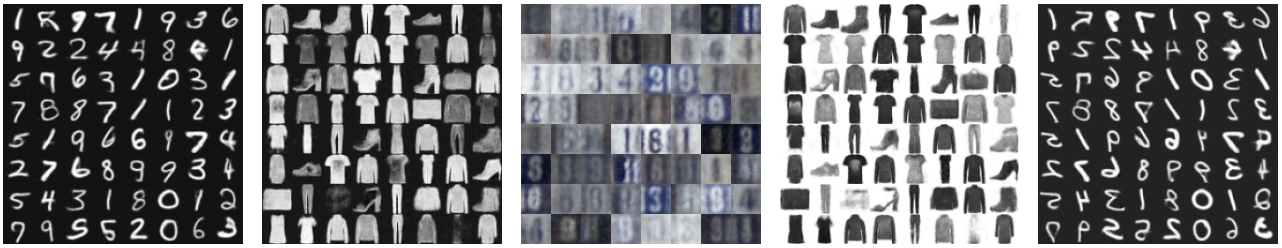
M. Visual results

In this section, we provide visual results for image reconstruction and generation. We train the proposed LIMix model under the MSFIR lifelong learning. The results obtained by the Teacher module as well as by the Student module are shown in Figures 10 and 11, respectively. We also train LIMix under CCCOS lifelong learning. The results obtained by the Teacher module and by the Student module are shown in Figures 12 and 13, respectively. From these results we observe that both the Teacher and Student modules can yield high-quality reconstructions. In Figure 14 we also provide the interpolation results achieved by the Student module from LIMix, after the CCCOS lifelong learning.

In the following, we train LIMix for the image to image translation tasks. The model is trained considering the lifelong learning of Map, CMP [12] and Shoe [19] datasets. We provide the visual results in Figure 15, where we observe that the proposed LIMix can be successfully applied for image-to-image translation tasks.



(a) MNIST. (b) Fashion. (c) SVHN. (d) IFasion. (e) RMNIST.



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

Figure 10. Image reconstructions when using LIMix under the MSFIR lifelong learning. The first row represents testing data samples and the second row are their reconstructions using LIMix.

(a) Testing samples. (b) Reconstructions.

Figure 11. Reconstructions when using the Student module component of LIMix after MSFIR lifelong learning.

(a) CelebA. (b) CACD. (c) 3D-chair. (d) Omniglot. (e) CIFAR10.

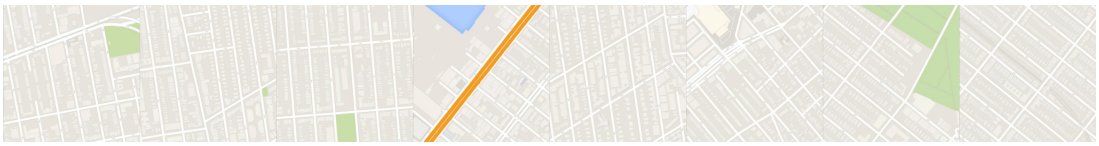
(f) Task 1. (g) Task 2. (h) Task 3. (i) Task 4. (j) Task 5.

Figure 12. Reconstructions when using LIMix after CCCOS lifelong learning. The first row represents testing images and the second row are their reconstructions using LIMix.

(a) Testing samples. (b) Reconstructions.

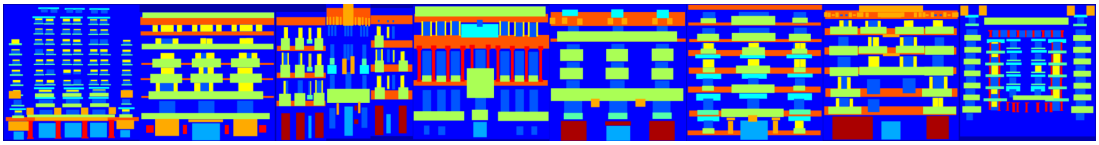
Figure 13. Reconstructions when using the Student module of LIMix after CCCOS lifelong learning.

Figure 14. Interpolation results obtained by the Student module of LIMix, after CCCOS lifelong learning.



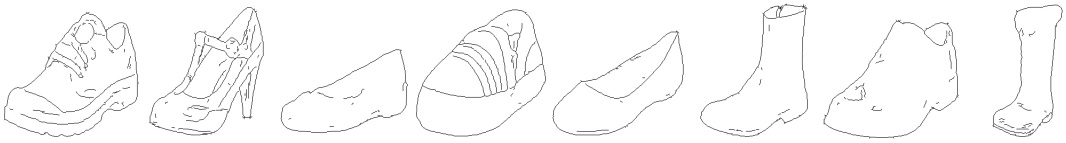
(a) Maps.

(b) Reconstructions.



(c) testing samples from CMP [12]

(d) Reconstructions.



(e) Testing samples from [19].

(f) Reconstructions.

Figure 15. Image to Image translation results when learning three different tasks under the lifelong learning.

References

[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 ({OSDI} 16)*, pages 265–283, 2016. 17

[2] Alessandro Achille, Tom Eccles, Loic Matthey, Christopher Burgess, Nick Watters, Alexander Lerchner, and Irina Higgins. Life-long disentangled representation learning with cross-domain latent homologies. In *Advances in Neural Inf. Proc. Systems (NIPS)*, pages 9873–9883, 2018. 8

[3] Ali Ayub and Alan Wagner. {EEC}: Learning to encode and regenerate images for continual learning. In *International Conference on Learning Representations*, 2021. 8

[4] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018. 9

[5] Yunhui Guo, Mingrui Liu, Tianbao Yang, and Tajana Rosing. Improved schemes for episodic memory-based lifelong learning. In *Advances in Neural Information Processing Systems*, pages 1023–1035, 2020. 9

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. Int. Conf. on Learning Representations (ICLR), arXiv preprint arXiv:1412.6980*, 2015. 17

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. Advances in Neural Inf. Proc. Systems (NIPS)*, pages 1097–1105, 2012. 18

[8] Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural Dirichlet process mixture model for task-free continual learning. *Proc. International Conference of Learning Representations (ICLR), arXiv preprint arXiv:2001.00689*, 2020. 9

[9] 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, 15

[10] 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. 2

[11] 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 *Advances in Neural Information Processing Systems*, volume 33, pages 4453–4464, 2020. 9, 18, 19

[12] 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. 20, 24

[13] Jason Ramapuram, Magda Gregorova, and Alexandros Kalousis. Lifelong generative modeling. *Neurocomputing*, 404:381–400, 2020. 8, 20

[14] Dushyant Rao, Francesco Visin, Andrei A. Rusu, Yee W. Teh, Razvan Pascanu, and Raia Hadsell. Continual unsupervised representation learning. In *Advances in Neural Information Proc. Systems (NeurIPS)*, pages 7645–7655, 2019. 8, 20

[15] Hanul Shin, Jung K. Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Proc. Advances in Neural Inf. Proc. Systems (NIPS)*, pages 2990–2999, 2017. 7, 8

[16] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Inf. Proc. Systems (NIPS)*, pages 3483–3491, 2015. 16, 17

[17] 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), arXiv preprint arXiv:2002.06715*, 2020. 8, 20

[18] Chenshen Wu, Luis Herranz, Xialei Liu, Joost van de Weijer, and Bogdan Raducanu. Memory replay GANs: Learning to generate new categories without forgetting. In *Advances In Neural Inf. Proc. Systems (NIPS)*, pages 5962–5972, 2018. 8, 20

[19] 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. 20, 24

[20] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *Proc. of Int. Conf. on Machine Learning, vol. PLMR 70*, pages 3987–3995, 2017. 18

[21] Mengyao Zhai, Lei Chen, Frederick Tung, Jiawei He, Megha Nawhal, and Greg Mori. Lifelong GAN: Continual learning for conditional image generation. In *Proc. of IEEE Int. Conf. on Computer Vision (ICCV)*, pages 2759–2768, 2019. 8, 14

[22] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13208–13217, 2020. 9