Cognitively Inspired Learning of Incremental Drifting Concepts

Mohammad Rostami, Aram Galstyan
University of Southern California
\{mrostami,galstyan\}@isi.edu

Abstract
Humans continually expand their learned knowledge to new domains and learn new concepts without any interference with past learned experiences. In contrast, machine learning models perform poorly in a continual learning setting, where input data distribution changes over time. Inspired by the nervous system learning mechanisms, we develop a computational model that enables a deep neural network to learn new concepts and expand its learned knowledge to new domains incrementally in a continual learning setting. We rely on the Parallel Distributed Processing theory to encode abstract concepts in an embedding space in terms of a multimodal distribution. This embedding space is modeled by internal data representations in a hidden network layer. We also leverage the Complementary Learning Systems theory to equip the model with a memory mechanism to overcome catastrophic forgetting through implementing pseudo-rehearsal. Our model can generate pseudo-data points for experience replay and accumulate new experiences to past learned experiences without causing cross-task interference.

1 Introduction
Humans continually abstract concept classes from their input sensory data to build semantic descriptions, and then update and expand these concepts as more experiences are accumulated [Widmer and Kubat, 1996], and use them to express their ideas and communicate with each other [Gennari et al., 1989; Lake et al., 2015]. For example, “cat” and “dog” are one of the first concept classes that many children learn to identify. Most humans expand these concepts as concept drift occurs, e.g., incorporating many atypical dog breeds into the “dog” concept, and also incrementally learn new concept classes, e.g. “horse” and “sheep,” as they acquire more experiences. Although this concept learning procedure occurs continually in humans, continual and incremental learning of concept classes remains a major challenge in artificial intelligence (AI). AI models are usually trained on a fixed number of classes and the data distribution is assumed to be stationary during model execution. Hence, when an AI model is trained or updated on sequentially observed tasks with diverse distributions or is trained on new classes, we generally need new annotated data points from the new classes [Rostami et al., 2018] and the model also would tend to forget what has been learned before due to cross-task interference, known as the phenomenon of catastrophic forgetting [French, 1991].

Inspired by the Parallel Distributed Processing (PDP) paradigm [McClelland et al., 1986; McClelland and Rogers, 2003], our goal is to enable a deep neural network to learn drifting concept classes [Gama et al., 2014; Rostami and Galstyan, 2023] incrementally and continually in a sequential learning setting. PDP hypothesizes that abstract concepts are encoded in higher layers of the nervous system [McClelland and Rogers, 2003; Saxe et al., 2019]. Similarly, and based on behavioral similarities between artificial deep neural networks and the nervous system [Morgenstern et al., 2014], we can assume that the data representations in hidden layers of a deep network encode semantic concepts with different levels of abstractions. We model these representations as an embedding space in which semantic similarities between input data points are encoded in terms of geometric distances [Jiang and Conrath, 1997], i.e., data points that belong to the same concept class are mapped into separable clusters in the embedding space. When a new concept is abstracted, a new distinct cluster should be formed in the embedding space to encode that new class. Incremental concepts learning is feasible by tracking and remembering the representation clusters that are formed in the embedding space and by considering their dynamics as more experiences are accumulated in new domains.

We benefit from the Complementary Learning Systems (CLS) theory [McClelland et al., 1995] to mitigate catastrophic forgetting. CLS is based on empirical evidences that suggest experience replay of recently observed patterns during sleeping and waking periods in the human brain helps to accumulate the new experiences to the past learned experiences without causing interference [McClelland et al., 1995; Robins, 1995]. According to this theory, hippocampus plays the role of a short-term memory buffer that stores samples of recent experiences and catastrophic forgetting is prevented by replaying samples from the hippocampal storage to implement pseudo-rehearsal in the neocortex during sleeping periods through enhancing past learned knowledge. Unlike AI memory buffers that store raw input data point, hippocampal storage can only store encoded abstract representations.
Inspired by the above two theories, we expand a base neural classifier with a decoder network, which is amended from a hidden layer, to form an autoencoder with the hidden layer as its bottleneck. The bottleneck is used to model the discriminative embedding space. As a result of supervised learning, the embedding space becomes discriminative, i.e., a data cluster is formed for each concept class in the embedding space [McClelland and Rogers, 2003; Rostami, 2021b]. These clusters can be considered analogous to neocortical representations in the brain, where the learned abstract concepts are encoded [McClelland et al., 1986]. We use a multi-modal distribution to estimate this distribution [Stan and Rostami, 2021; Rostami, 2021a]. We update this parametric distribution to accumulate new experiences to past learned experiences consistently. Since our model is generative, we can implement the offline memory replay process to prevent catastrophic forgetting [McClelland et al., 1995; Rasch and Born, 2013]. When a new task arrives, we draw random samples from the multi-modal distribution and feed them into the decoder to generate representative pseudo-data points. These pseudo-data points are then used to implement pseudo-rehearsal for experience replay [Robins, 1995].

2 Related Work

Continual learning: the major challenge of continual learning is tackling catastrophic forgetting. Previous works in the literature mainly rely on experience replay [Li and Hoiem, 2018]. The core idea of experience replay is to implement pseudo-rehearsal by replaying representative samples of past tasks along with the current task data to retain the learned distributions. Since storing these samples requires a memory buffer, the challenge is selecting the representative samples to meet the buffer size limit. For example, selecting uncommon samples that led to maximum effect in past experiences has been found to be effective [Schaul et al., 2016]. However, as more tasks are learned, selecting the effective samples becomes more complex. The alternative approach is to use generative models that behave more similar to humans [French, 1999]. Shin et al. ([Shin et al., 2017]) use a generative adversarial structure to mix the distributions of all tasks. It is also feasible to couple the distributions of all tasks in the bottleneck of an autoencoder [Rostami et al., 2019; Rostami et al., 2020b]. The shared distribution then can be used to generate pseudo-samples [Rannen et al., 2017]. Weight consolidation using structural plasticity [Lamprecht and LeDoux, 2004; Zenke et al., 2017; Kirkpatrick et al., 2017] is another approach to approximate experience replay. The idea is to identify important weights that retain knowledge about a task and then consolidate them according to their relative importance for past tasks. Continual learning of sequential tasks can be improved used high-level tasks descriptors to compensate for data scarcity [Rostami et al., 2020a].

Incremental learning: forgetting in incremental learning systems from updating the model when new classes are incorporated, rather concept drifts in a fixed number of learned classes. Hence, the goal is to learn new classes such that knowledge about the past learned classes is not overwritten. A simple approach is to expand the base network as new classes are observed. Tree-CNN [Roy et al., 2020] proposes a hierarchical structure that grows like a tree when new classes are observed. The idea is to group new classes into feature-driven super-classes and find the exact label by limiting the search space. As the network grows, the new data can be used to train the expanded network. Sarwar et al. [Sarwar et al., 2019] add new convolutional filters in all layers to learn the new classes through new parameters. The alternative approach is to retain the knowledge about old classes in an embedding feature space. Rebuffi et al. [Rebuffi et al., 2017] proposed iCarl which maps images into a feature space that remains discriminative as more classes are learned incrementally. A fixed memory buffer is used to store exemplar images for each observed class. Each time a new class is observed, these images are used to learn a class-level vector in the feature space such that the testing images can be classified using nearest neighbor with respect to these vectors.

Gaussian mixture model: are useful for modeling distributions that exhibit multiple modes or clusters. GMMs assume that the data is generated by a mixture of several Gaussian distributions, each representing a different cluster or mode in the data. The model is trained by estimating the parameters of the component Gaussians, including their means and variances, as well as the mixture weights that determine the relative contribution of each Gaussian to the overall distribution. GMMs are widely used in a variety of applications, including continual learning [Rostami et al., 2019].

Contributions: We develop a unified framework that addresses challenges of both incremental learning and lifelong learning. Our idea is based on tracking and consolidating the multimodal distribution that is formed by the internal data representations of sequential tasks in hidden layers of a neural network. We model this distribution as a Gaussian mixture model (GMM) with time-dependent number of components. Concept drifts are learned by updating the corresponding GMM component for a particular class and new concepts are learned by adding new GMM components. We also make the model generative to implement experience replay.

3 Problem Statement

Consider a learning agent which observes a sequence of observed tasks \( \{ Z(t) \}_{t=1}^{T} \) [Chen and Liu, 2016] and after learning each task moves forward to learn the next task. Each task is a classification problem in a particular domain and each class represents a concept. The classes for each task can be new unobserved classes, i.e., necessitating incremental learning [Rebuffi et al., 2017], or drifted forms of the past learned classes, i.e., necessitating lifelong learning [Chen and Liu, 2016], or potentially a mixture of both cases. Formally, a task is characterized by a dataset \( D(t) = (X(t), Y(t)) \), where \( X(t) = [x_1^{(t)}, \ldots, x_n^{(t)}] \in \mathbb{R}^{d \times n_t} \) and \( Y(t) \in \mathbb{R}^{k_t \times n_t} \) are the data points and one-hot labels, respectively. The goal is to train a time-dependent classifier function \( f(t)(\cdot) : \mathbb{R}^{d} \rightarrow \mathbb{R}^{k_t} \), where \( k_t \) is the number of classes for the \( t \)-th task and is fixed for each task, such that the classifier continually generalizes on the tasks seen so far. The data points \( x_i^{(t)} \sim q_i^{(t)}(x) \) are assumed to be drawn i.i.d. from an unknown task distri-
distribution \( q^{(1)}(x) \). Figure 1 visualizes a block-diagram of this continual and dynamic learning procedure. The agent needs to expand its knowledge about all the observed concepts such that it can perform well on all the previous learned domains.

Learning each task in isolation is a standard supervised learning problem. After selecting a suitable parameterized family of functions \( f^{(1)}_\theta : \mathbb{R}^d \to \mathbb{R}^{k_1} \), we can solve for the optimal parameters using the empirical risk minimization (ERM): \( \hat{\theta}^{(1)} = \arg\min_{\theta} E_{\theta}^{c^{(1)}}(\theta) = \arg\min_{\theta} \sum_i L_d(f^{(1)}_\theta(x_i), y_i^{(1)}) \), where \( L_d(\cdot) \) is a proper loss function. If \( n_t \) is large enough, the empirical risk expectation would be a good approximation of the real expected risk function \( e^{(1)}(\theta) = E_{x \sim q^{(1)}(x)}(L_d(f^{(1)}_\theta(x), f(x))) \). As a result, if the base parameteric family is rich and complex enough for learning the task function, then the ERM optimal model generalizes well on unseen test samples that are drawn from \( q^{(1)}(x) \).

For the rest of the paper, we consider the base model \( f^{(1)}_\theta \) to be a deep neural network with an increasing output size to encode incrementally observed classes. As stated, we rely on the PDP paradigm. Hence, we decompose the deep network into an encoder sub-network \( \phi_v(\cdot) : \mathbb{R}^d \to \mathcal{Z} \subseteq \mathbb{R}^j \) with learnable parameter \( v \), e.g., convolutional layers of a CNN, and a classifier sub-network \( h_w(\cdot) : \mathbb{R}^j \to \mathbb{R}^{k_1} \) with learnable parameters \( w \), e.g., fully connected layers of a CNN, where \( \mathcal{Z} \) denotes the embedding space in which the concepts will be formed as separable clusters.

The concepts for each task are known a priori and hence new nodes are added to the classifier sub-network output to incorporate the new classes at time \( t \). We use a softmax layer as the last layer of the classifier subnetwork. Hence, we can consider the classifier to be a a maximum \textit{a posteriori} (MAP) estimator after training. This means that the encoder network transforms the input data distribution into an internal multi-modal distribution with \( k_1 \) modes in the embedding space because the embedding space \( \mathcal{Z} \) should be concept-discriminative for good generalization. Each concept class is represented by a single mode of this distribution. We use a Gaussian mixture model (GMM) to model and approximate this distribution (see Figure 1, middle panel). Catastrophic forgetting is the result of changes in this internal distribution when changes in the input distribution leads to updating the internal distribution heuristically. Our idea is to track changes in the data distribution and update and consolidate the internal distribution such that the acquired knowledge from past experiences is not overwritten when learning new tasks.

The main challenge is to adapt the network \( f^{(1)}_\theta(\cdot) \) and the standard ERM such that we can track the internal distribution continually and accumulate the new acquired knowledge consistently to the past learned knowledge with minimum interference. For this purpose, we form a generative model by amending the base model with a decoder \( \psi_u : \mathcal{Z} \to \mathbb{R}^d \), with learnable parameters \( u \). This decoder maps back the internal representations to reconstruct the input data point in the input space such that the pair \( (\phi_u, \psi_u) \) forms an autoencoder. According to our previous discussion, a multi-modal distribution would be formed in the bottleneck of the autoencoder upon learning each task. This distribution encodes the learned knowledge about the concepts that have been learned from past experiences so far. If we approximate this distribution with a GMM, we can generate pseudo-data points that represent the previously learned concepts and use them for pseudo-rehearsal. For this purpose, we can simply draw samples from all modes of the GMM and feed these samples into the decoder subnetwork to generate a pseudo-dataset (see Figure 1). After learning each task, we can update the GMM estimate such that the new knowledge acquired is accumulated to the past gained knowledge consistently to avoid interference. By doing this procedure continually, our model is able to learn drifting concepts incrementally. Figure 1 visualizes this repetitive procedure in our setting.

### 4 Proposed Algorithm

When the first task is learned, there is no prior experience and hence learning reduces the following:

\[
\min_{W, W', M} \mathcal{L}_c(X^{(1)}, Y^{(1)}) = \min_{W, W', M} \frac{1}{n_1} \sum_{t=1}^{n_1} \mathcal{L}_d \left( h_w(\phi_v(x^{(1)}_i)), y^{(1)}_i \right) + \gamma \mathcal{L}_r \left( \psi_u(\phi_v(x^{(1)}_i)), x^{(1)}_i \right),
\]

where \( \mathcal{L}_d \) is the discrimination loss, e.g., cross-entropy loss, \( \mathcal{L}_r \) is the reconstruction loss for the autoencoder, e.g., \( \ell_2 \)-norm, \( \mathcal{L}_c \) is the combined loss, and \( \gamma \) is a trade-off parameter between the terms. When the first task is learned, also any future task, according to the PDP hypothesis, a multi-modal distribution \( p^{(1)}(z) = \sum_{j=1}^{k_1} \alpha_j \mathcal{N}(z|\mu_j, \Sigma_j) \) with \( k_1 \) components is formed in the embedding space. We assume that
this distribution can be modeled with a GMM. Since the labels for the input task data samples are known, we use MAP estimation to recover the GMM parameters (see Appendix for details). Let \( \hat{p}(z) \) denotes the estimated distribution.

As subsequent tasks are learned, the internal distribution should be updated continually to accumulate the new acquired knowledge. Let \( k_t = k_{old} + k_{new} \), where \( k_{old} \) denotes the number of the previously learned concepts that exist in the current task and \( k_{new} \) denotes the number of the new observed classes. Hence, the total number of learned concepts until \( t = T \) is \( k_{Tot} = \sum_{t=1}^{T} k_t \). Also, let the index set \( \mathbb{N}_{Tot}^t \) denotes an order on the classes \( C_j \), with \( j \in \mathbb{N}_{Tot}^t \), that are observed until \( t = T \). Let \( \mathbb{N}_T = \mathbb{N}_{Tot}^T \) contains the \( k_T \) indices of the existing concepts in \( \mathcal{Z}(T) \). To update the internal distribution after learning \( \mathcal{Z}(T) \), the number of distribution modes should be updated to \( k_{Tot}^t \). Additionally, catastrophic forgetting must be mitigated using experience replay. We can draw random samples from the GMM distribution \( z_i \sim \hat{p}^{(t-1)}(z) \) and then pass each sample through the decoder \( \psi(z_i) \) to generate pseudo-data points for pseudo-rehearsal. Since each particular concept is represented by exactly one mode of the internal GMM distribution, the corresponding pseudo-labels for the generated pseudo-data points are known. Moreover, the confidence levels for these labels are also known from the classifier softmax layer. To generate a clean pseudo-dataset, we can set a threshold \( \tau \) and only pick the pseudo-data points for which the model confidence level is more than \( \tau \). We also generate a balanced pseudo-dataset with respect to the learned classes. Doing so, we ensure suitability of a GMM with \( k_{Tot}^t \) components to estimate the empirical distribution accurately after learning the next tasks.

Let \( \mathcal{D}(t) = (\psi(\mathcal{Z}(t)), \mathcal{Y}(t)) \) denotes the pseudo-dataset, generated at time \( t \) after learning the tasks \( \{\mathcal{Z}(s)\}_{s=1}^{t-1} \). We form the following objective to learn the task \( \mathcal{Z}(t) \), \( \forall t \geq 2 \):

\[
\min_{w,w'} \mathcal{L}_c(X(t), Y(t)) + \mathcal{L}_c(\hat{X}(t), \hat{Y}(t)) + \lambda \sum_{j \in \mathbb{N}_{old}} D\left(\phi_w(q^{(t)}(X(t))|C_j), \hat{p}^{(t-1)}(\mathcal{Z}(t)|C_j)\right), \tag{2}
\]

where \( D(\cdot, \cdot) \) is a probability metric and \( \lambda \) is a parameter.

The first and the second terms in Eq. (2) are combined loss terms for the current task training dataset and the generated pseudo-dataset that represent the past tasks, defined similar to Eq. (1). The second term in Eq. (2) mitigates catastrophic forgetting through pseudo-rehearsal process. The third term is a crucial term to guarantee that our method will work in a lifelong learning setting. This term enforces that each concept is encoded in one mode of the internal distribution across all tasks. This term is computed on the subset of the concept classes that are shared between the current task and the pseudo-dataset, i.e. \( \mathbb{N}_{old}^t \), to enforce consistent knowledge accumulation. Minimizing the probability metric \( D(\cdot, \cdot) \) enforces that the internal conditional distribution for the current task \( \phi_w(q^{(t)}(\cdot|C_j)) \), conditioned on a particular shared concept \( C_j \), to be close to the conditional shared distribution \( \hat{p}^{(t-1)}(\cdot|C_j) \). Hence, both form a single mode of the internal distribution and concept drifting is mitigated. Conditional matching of the two distributions is feasible as we have access to pseudo-labels. Adding this term guarantees that we can continually use a GMM with exactly \( k_{Tot}^t \) components to capture the internal distribution in this lifelong learning setting. The remaining task is to select a suitable probability metric \( D(\cdot, \cdot) \) for solving Eq. (2). Wasserstein Distance (WD) metric has been found to be an effective choice for deep learning due to its applicability for gradient-based optimization [Courty et al., 2017]. To reduce the computational burden of computing WD, we use the Sliced Wasserstein Distance (SWD) [Bonnet et al., 2015]. (for details on the SWD, refer to the Appendix). Our Incremental Concept Learning Algorithm (ICLA) method is summarized in Algorithm 1.

### Algorithm 1 ICLA (λ, γ, τ)

1. **Input:** labeled training datasets in a sequence
2. \( D(t) = (X(t), X(t)) \) for \( t \geq 1 \)
3. **Initial Learning:** learn the first task via Eq. (1)
4. **Fitting GMM:**
   - estimate \( \hat{p}_j^{(t)}(\cdot) \) using \( \{\phi_w(x_i^{(t)}), y_i^{(t)}\}_{i=1}^{n_t} \)
5. **For \( t \geq 2 \)**
6. **Generate the pseudo-dataset:**
   - \( \mathcal{D}(t) = (\tilde{x}_i^{(t)}, \psi(z_i^{(t)}), \tilde{y}_i^{(t)}) \)
   - \( \tilde{z}_i^{(t)}(\cdot), \tilde{y}_i^{(t)} \sim \hat{p}^{(t-1)}(\cdot) \)
7. **Task learning:**
   - learnable parameters are updated via Eq. (2)
8. **Estimating the internal distribution:**
   - update \( \hat{p}_j^{(t)}(\cdot) \) with \( k_{Tot}^t \) components via the combined samples \( \{\phi_w(x_i^{(t)}), \phi_w(x_i^{(t)})\}_{i=1}^{n_t} \)
9. **EndFor**

### 5 Theoretical Analysis

We demonstrate that ICLA minimizes an upperbound for the expected risk of the learned concept classes across all the previous tasks for all \( t \). We perform our analysis in the embedding space as an input space and consider the hypothesis class \( \mathcal{H} = \{h_w(\cdot) | h_w(\cdot) : \mathcal{Z} \rightarrow \mathbb{R}^k, w \in \mathbb{R}^{H}\} \). Let \( e_\tau(w) \) denote the real risk for a given function \( h_w(\cdot) \in \mathcal{H} \) when used on task \( \mathcal{Z}(t) \) data representations in the embedding space. Similarly, \( \tilde{e}_\tau(w) \) denotes the observed risk for the function \( h_w(\cdot) \) when used on the pseudo-task, generated by sampling the learned GMM distribution \( \hat{p}^{(t-1)} \). Finally, let \( e_{t,s}(w) \) denote the risk of the model \( h_{bmw(\cdot)}(\cdot) \) when used only on the concept classes in the set \( \mathbb{N}_t \subseteq \mathbb{N}_{Tot}^t \), for \( s \leq t \), i.e., task specific classes, after learning the task \( \mathcal{Z}(t) \).

**Theorem 1:** Consider two tasks \( \mathcal{Z}(t) \) and \( \mathcal{Z}(s) \), where \( s \leq t \). Let \( h_{bmw(\cdot)} \) be an optimal classifier trained for the \( \mathcal{Z}(t) \) using the ICLA algorithm. Then for any \( d' > d \) and \( \zeta < \sqrt{2} \), there exists a constant number \( N_0 \) depending on \( d' \) such that for any \( \xi > 0 \) and \( \min(n(s), n_t) \geq \max(\xi^{-d'+2}), 1 \) with probability at least \( 1 - \xi \) for \( h_{bmw(\cdot)} \in \mathcal{H} \), then:

\[
e_{t,s}(w) \leq e_{t-1,s}(w) + W(\hat{p}^{(t-1)}, \phi(q^{(s)}) + \tilde{e}_\tau(w) + \sqrt{2 \log(\frac{1}{\xi})/\zeta} \left( \frac{1}{\sqrt{n(s)}} + \sqrt{\frac{1}{n_t}} \right), \tag{3}
\]

3061
where $W(\cdot, \cdot)$ denotes the WD metric, $\tilde{n}_t(N)$ denotes the pseudo-task samples that belong to the classes in $N$, $\phi(q(\cdot))$ denotes the empirical marginal distribution for $Z^{(t)}$ in the embedding, $\hat{p}(t-1)$ is the conditional empirical shared distribution when the distribution $\hat{p}(t-1)(\cdot)$ is conditioned to the classes in $N$, and $e_C(w^*)$ denotes the optimal model learned for the combined risk of the tasks on the shared classes in $N$, i.e., $w^* = \arg \min_w e_C(\theta) = \arg \min_w (\epsilon_{t,s}(w) + \epsilon_s(w))$. This is a model with the best performance if the tasks could be learned simultaneously.

**Proof:** included in the Appendix due to page limit.

We then use Theorem 1 to conclude the following lemma:

**Lemma 1** : Consider the ICLA algorithm after learning $Z^{(T)}$. Then all tasks $t < T$ and under the conditions of Theorem 1, we can conclude the following inequality:

$$e_t(w) \leq e_{T-1,t}(w) + W(\phi(q^{(t)}), \hat{p}^{(t)}) + e_C(w^*) + \sum_{s=t}^{T-2} W(\hat{p}^{(s)}, \hat{p}^{(s+1)}) + \sqrt{\frac{2\log(1/\xi)}{\xi}} \left( \frac{1}{n_{t,s}} + \frac{1}{n_{t}} \right),$$  \hspace{1cm} (4)

**Proof:** included in the Appendix due to page limit.

Lemma 1 concludes that when a new task is learned at time $t = T$, ICLA updates the model parameters conditioned on minimizing the upper bound of $e_t$ for all $t < T$ in Eq. 4. The last term in Eq. 4 is a small constant term when the number of training data points is large. If the network is complex enough so that the PDP hypothesis holds, then the classes would be separable in the embedding space and in the presence of enough labeled samples, the terms $e_{T-1,t}(w)$ would be small because $e_{T-1}(w)$ is minimized using ERM. The term $W(\phi(q^{(t)}), \hat{p}^{(t)})$ would be small because we deliberately fit the GMM distribution $\hat{p}^{(t)}$ to the distribution $\phi(q^{(t)})$ in the embedding space when learning the task $Z^{(t)}$. Existence of this term indicates that our algorithm requires that internal distribution can be fit with a GMM distribution with high accuracy and this limits applicability of our algorithm. Note however, all parametric learning algorithms face this limitation. The term $e_C(w^*)$ is small because we continually match the distributions in the embedding space class-conditionally. Hence, if the model is trained on task $Z^{(t)}$ and the pseudo-task at $t - T$, it will perform well on both tasks. Note that this is not trivial because if the wrong classes are matched across the domains in the embedding space, the term $e_C(w^*)$ will not be minimal. Finally, the sum term in Eq. 4 indicates the effect of experience replay. Each task in this sum is minimized at $s = t+1$ because we draw random samples from $\hat{p}^{(t)}$ and then train the autoencoder to enforce $\hat{p}^{(t)} \approx \psi(\phi(q^{(t)}))$. Since all the terms in the upperbound of $e_t(w)$ in Eq. 4 are minimized when a new task is learned, catastrophic forgetting of the previous tasks will be mitigated. Another important intuition from Eq. 4 is that as more tasks are learned after learning a task, the upperbound becomes looser as more terms are accumulated in the sum which enhances forgetting. This observation accords with our intuition about forgetting as more time passes after initial learning time of a task or concept.

### 6 Experimental Validation

We validate our method on two sequential task learning settings: incremental learning and continual incremental learning. Incremental learning is a special case of our learning setting when each concept class is observed only in one task and concept drift does not exist. We use this special case to compare our method against existing incremental learning approaches. Our implementation is available as a supplement.

**Evaluation Methodology** : We use the same network structure for all the methods for fair comparison. To visualize the results, we generate learning curves by plotting the model performance on the testing split of datasets versus the training epochs, i.e., to model time. We report the average performance of five runs. Visualizing learning curves allows studying temporal aspects of learning. For comparison, we provide learning curves for: (a) full experience replay (FR) which stores the whole training data for all the previous tasks and (b) experience replay using a memory buffer (MB) with a fixed size, similar to Li et al. ([Li and Hoiem, 2018]). At each time-step, the buffer stores an equal number of samples per concept from the previous tasks. When a new task is learned, a portion of old stored samples is discarded and replaced with samples from the new task to keep the buffer size fixed. FR serves as a best achievable upperbound to measure the effectiveness of our method against the upperbound. For more details about the experimental setup and all parameteric values, please refer to the Appendix and the provided code.

#### 6.1 Incremental Learning

The classes are encountered only at one task in incremental learning. We design two incremental learning experiments using the MNIST and the Fashion-MNIST datasets. Both datasets are classification datasets with ten classes. MNIST dataset consists of gray scale images of handwritten digits and Fashion-MNIST consists of images of common fashion products. We consider an incremental learning setting with nine tasks for the MNIST dataset. The first task is a binary classification of digits 0 and 1 and each subsequent task involves learning a new digit. The setup for Fashion-MNIST dataset is similar, but we considered four tasks and each task involves learning two fashion classes. We use a memory buffer with the fixed size of 100 for MB. We build an autoencoder by expanding a VGG-based classifier by mirroring the layers.

Figure 2 presents results for the designed experiments. For simplicity, we have provided condensed results for all tasks in a single curve. Each task is learned in 100 epochs and at each epoch, the model performance is computed as the average classification rate over all the classes, observed before. We report performance on the standard testing split of each dataset for the observed classes. Figure 2a and present the learning curves for the MNIST experiments. Similarly, Figure 2b present learning curves for the Fashion-MNIST experiments. We can see in both figures that FR (dashed blue curves) leads to superior performance. This is according to expectation but as we discussed, the challenge is the requirement for a memory buffer with an unlimited size. The buffer cannot have a fixed size as the number of data points grows when more tasks are learned. MB (solid yellow curves) is
initially somewhat effective and comparable with ICLA, but as more tasks are learned, forgetting effect becomes more severe. This is because fewer data points per task can be stored in the buffer with fixed size as more tasks are learned. As a result, the stored samples would not be sufficiently representative of the past learned tasks. In comparison, we can generate as many pseudo-data points as desired.

We can also see in Figure 2a and Figure 2b that ICLA (dotted green curves) is able to mitigate catastrophic forgetting considerably better than MB and the performance difference between ICLA and MB increases as more tasks are learned. We also observe that ICLA is more effective for MNIST dataset. This is because FMNIST data points are more diverse. As a result, generating pseudo-data points that look more similar to the original data points is easier for the MNIST dataset given that we are using the same network structure for both tasks. Another observation is that the major performance degradation for ICLA occurs each time the network starts to learn a new concept class as initial sudden drops. This degradation occurs due to the existing distance between the distributions \( p_i(T^{-1}) \) and \( q_i(0^s) \) at \( t = T \) for \( s < T \). Although ICLA minimizes this distance, the autoencoder is not ideal and this distance is non-zero in practice.

For comparison purpose, we have listed our performance and a number of methods for incremental learning on MNIST in Table 1. Two sets of incremental learning tasks have been designed using MNIST in the literature: 5 tasks (5T) setting and 2 tasks (2T) setting. In the 2T setting, two tasks are defined involving digits \((0-4)\) and \((5-9)\). In the 5T setting, five binary classification tasks are defined involving digits \((0, 1)\) to \((8, 9)\). We have compared our performance against several methods, representative of prior works: CAB [He and Jaeger, 2018], IMM [Lee et al., 2017], OWM [Zeng et al., 2019], GEM [Lopez-Paz and Ranzato, 2017], iCarl [Rebuffi et al., 2017], GSS [Aljundi et al., 2019], DGR [Shin et al., 2017], and MeRGAN [Wu et al., 2018]. The CAB, IMM, and OWM methods are based on regularizing the network weights. The GEM, iCarl, and GSS methods use a memory buffer to store selected samples. Finally, DGR and MeRGAN methods are based on generative replay similar to ICLA but use adversarial learning. We have reported the classification accuracy on the ten digit classes after learning the last task in Table 1. A memory buffer with a fixed size of 100 is used for GEM, iCarl, and GSS. Following these works, an MLP with two layers is used as the base model for fair comparison.

We observe in Table 1 that when the buffer size is small, buffer-based methods perform poorly. Methods based on weight regularization perform quite well but note that these methods limit the network learning capacity. As a result, when the number of tasks grow, the network cannot be used to learn new tasks. Generative methods, including ICLA, perform better compared to buffer-based methods and at the same time do not limit the network learning capacity because the network weights can change after generating the pseudo-dataset. Although ICLA has the state-of-the-art performance for these tasks, there is no superior method for all conditions, because by changing the experimental setup, e.g., network structure, dataset, hyper-parameters such as memory buffer, etc, a different method may have the best performance result. However, we can conclude that ICLA has a superior performance when the network size is small and using a memory buffer is not possible, i.e., we have limited learning resources.

### 6.2 Continual Incremental Learning

Permuted MNIST task is a common supervised learning benchmark for sequential task learning [Kirkpatrick et al., 2017]. The sequential tasks are generated using the MNIST dataset. Each task \( Z^{(t)} \) is generated by rearranging the pixels of all images in the dataset using a fixed random predetermined permutation transform and keeping the labels as their original value. As a result, we can generate many tasks that are diverse, yet equally difficult. As a result, these tasks are suitable for performing controlled experiments. Since no prior work has addressed incremental learning of drifting concepts, we should design a suitable set of tasks.

We design continual incremental learning tasks that share common concepts using five permuted MNIST tasks. The first task is a binary classification of digits 0 and 1 for the MNIST dataset. For each subsequent task, we generate a permuted MNIST task but include only the previously seen digits plus two new digits in the natural number order, e.g., the third task includes permuted versions of digit 0–5. This means that at each task, new forms of all the previously learned concepts are encountered, i.e., we need to learn drifting concepts, in addition to new tasks. Hence, the model needs to expand its knowledge about the previously learned concepts while learning new concepts. We use a memory buffer with size of 30000

![Learning curves for the incremental learning experiments](image)

**Table 1: Classification accuracy for MNIST.**

<table>
<thead>
<tr>
<th>Method</th>
<th>2T</th>
<th>5T</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAB [He and Jaeger, 2018]</td>
<td>94.9±0.3</td>
<td>-</td>
</tr>
<tr>
<td>IMM [Lee et al., 2017]</td>
<td>94.1±0.3</td>
<td>-</td>
</tr>
<tr>
<td>OWM [Zeng et al., 2019]</td>
<td>96.3±0.1</td>
<td>-</td>
</tr>
<tr>
<td>GEM [Lopez-Paz and Ranzato, 2017]</td>
<td>-</td>
<td>78.0</td>
</tr>
<tr>
<td>iCarl [Rebuffi et al., 2017]</td>
<td>-</td>
<td>81.0</td>
</tr>
<tr>
<td>GSS [Aljundi et al., 2019]</td>
<td>-</td>
<td>61.0</td>
</tr>
<tr>
<td>DGR [Shin et al., 2017]</td>
<td>88.7±2.6</td>
<td>-</td>
</tr>
<tr>
<td>MeRGAN [Wu et al., 2018]</td>
<td>97.0</td>
<td>-</td>
</tr>
<tr>
<td>ICLA</td>
<td>97.2±0.2</td>
<td>91.6±0.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>2T</th>
<th>5T</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAB [He and Jaeger, 2018]</td>
<td>94.9±0.3</td>
<td>-</td>
</tr>
<tr>
<td>IMM [Lee et al., 2017]</td>
<td>94.1±0.3</td>
<td>-</td>
</tr>
<tr>
<td>OWM [Zeng et al., 2019]</td>
<td>96.3±0.1</td>
<td>-</td>
</tr>
<tr>
<td>GEM [Lopez-Paz and Ranzato, 2017]</td>
<td>-</td>
<td>78.0</td>
</tr>
<tr>
<td>iCarl [Rebuffi et al., 2017]</td>
<td>-</td>
<td>81.0</td>
</tr>
<tr>
<td>GSS [Aljundi et al., 2019]</td>
<td>-</td>
<td>61.0</td>
</tr>
<tr>
<td>DGR [Shin et al., 2017]</td>
<td>88.7±2.6</td>
<td>-</td>
</tr>
<tr>
<td>MeRGAN [Wu et al., 2018]</td>
<td>97.0</td>
<td>-</td>
</tr>
<tr>
<td>ICLA</td>
<td>97.2±0.2</td>
<td>91.6±0.4</td>
</tr>
</tbody>
</table>
We developed an algorithm for continual incremental learning of concepts based on modeling the internal distribution of input data as a GMM and then updating the GMM as new experiences are acquired. We track this distribution to accumulate the new learned knowledge to the past learned knowledge consistently. We expand the base classifier model to make a generative model to allow for generating a pseudo-dataset for pseudo-rehearsal and experience replay. We provided theoretical and empirical result to validate our algorithm.

7 Conclusions
We developed an algorithm for continual incremental learning of concepts based on modeling the internal distribution of input data as a GMM and then updating the GMM as new experiences are acquired. We track this distribution to accumulate the new learned knowledge to the past learned knowledge consistently. We expand the base classifier model to make a generative model to allow for generating a pseudo-dataset for pseudo-rehearsal and experience replay. We provided theoretical and empirical result to validate our algorithm.

Ethical Statement
We foresee no significant ethical issues for our work.
References


