Deep Bayesian Unsupervised Lifelong Learning

Tingting Zhao[a], Zifeng Wang[a], Aria Masoomi[a], Jennifer Dy[a]

[a]*Department of Electrical and Computer Engineering, Northeastern University*

Abstract

Lifelong Learning (LL) refers to the ability to continually learn and solve new problems with incremental available information over time while retaining previous knowledge. Much attention has been given lately to Supervised Lifelong Learning (SLL) with a stream of labelled data. In contrast, we focus on resolving challenges in Unsupervised Lifelong Learning (ULL) with streaming unlabelled data when the data distribution and the unknown class labels evolve over time. Bayesian framework is natural to incorporate past knowledge and sequentially update the belief with new data. We develop a fully Bayesian inference framework for ULL with a novel end-to-end Deep Bayesian Unsupervised Lifelong Learning (DBULL) algorithm, which can progressively discover new clusters without forgetting the past with unlabelled data while learning latent representations. To efficiently maintain past knowledge, we develop a novel knowledge preservation mechanism via sufficient statistics of the latent representation for raw data. To detect the potential new clusters on the fly, we develop an automatic cluster discovery and redundancy removal strategy in our inference inspired by Nonparametric Bayesian statistics techniques. We demonstrate the effectiveness of our approach using image and text corpora benchmark datasets in both LL and batch settings.

Keywords: Unsupervised Lifelong Learning, Bayesian Learning, Deep Generative Models, Deep Neural Networks, Sufficient Statistics

1. Introduction

With exposure to a continuous stream of information, human beings are able to learn and discover novel clusters continually by incorporating past knowledge; however, traditional machine learning algorithms mainly focus on static data distributions. Training a model with new information often interferes with previously learned knowledge, which typically compromises performance on previous datasets [1]. In order to empower algorithms with the ability of adapting to emerging data while preserving the performance on seen data, a new machine learning paradigm called Lifelong Learning (LL) has recently gained some attention.

LL, also known as continual learning, was first proposed in [2]. It provides a paradigm to exploit past knowledge and learn continually by transferring previously learned knowledge to solve similar but new problems in a dynamic environment, without performance degradation on old tasks. LL is still an emerging field and most existing research work [3, 4, 5] has focused on Supervised Lifelong Learning (SLL), where the boundaries between different tasks are known and each task refers to a supervised learning problem with output labels provided. The term *task* can have different meanings under various contexts. For example, different tasks can represent different subsets of classes or labels in the same supervised learning problem [6] and can also represent supervised learning problems in different fields, where researchers target to perform continual learning across different domains [7] or lifelong transfer learning [8, 9].

While most research on LL has focused on resolving challenges in SLL problems with class labels provided, we instead consider Unsupervised Lifelong Learning (ULL) problems, where the learning system is interacting with a non-stationary stream of unlabelled data and the cluster labels are unknown. One objective of ULL is to discover new clusters by interacting with the environment dynamically and adapting to the changes in the unlabeled data without external supervision or knowledge. To avoid confusion, it is worth pointing out that our work assumes that the non-stationary streaming unlabelled data come from a single domain; and we target to develop a single dynamic model that can perform well on all sequential data at the end of each training stage without forgetting previous knowledge. This setting can serve as a reasonable starting point for ULL. We leave the challenges in ULL across different problem domains for future work.

To retain knowledge from past data and learn new clusters from new data continually, good representations of the raw data make it easier to extract information and, in turn, support effective learning. Thus, it is more computationally appealing if we can discover new clusters in a low-dimensional latent space instead of the complex original data space while performing representation learning. To achieve this, we propose Deep Bayesian Unsupervised Lifelong Learning (DBULL), which is a flexible probabilistic generative model that can adapt to new data and expand with new clusters while seamlessly learning deep representations in a Bayesian framework.

A critical objective of LL is to achieve consistently good performance incrementally as new data arrive in a streaming fashion, without performance decrease on previous data, even if the data may have been completely overwritten. Compared with a traditional batch learning setting, there are additional challenges to resolve in a LL setting. One important question is how to design a knowledge preservation scheme to efficiently maintain previously learned information. To discover the new clusters automatically with streaming, unlabelled data, another challenge is how to design a dynamic model that can expand with incoming data to perform unsupervised learning. The last challenge is how to design an end-to-end inference algorithm to obtain good performance in an incremental learning way. To answer these questions, we make the following contributions:

- To solve the challenges in ULL, we provide a fully Bayesian formulation that performs representation learning, clustering and automatic new cluster discovery simultaneously via our end-to-end novel variational inference strategy DBULL.

- To efficiently extract and maintain knowledge seen in earlier data, we provide innovation in our incremental inference strategy by first using sufficient statistics in the latent space in an LL context.
- To discover new clusters in the emerging data, we choose a nonparametric Bayesian prior to allow the model to grow dynamically. We develop a sequential Bayesian inference strategy to perform representation learning simultaneously with our proposed Cluster Expansion and Redundancy Removal (CERR) trick to discover new clusters on the fly without imposing bounds on the number of clusters, unlike most existing algorithms, using a truncated Dirichlet Process (DP) [10].
- To show the effectiveness of DBULL, we conduct experiments on image and text benchmarks. DBULL can achieve superior performance compared with state-of-the-art methods in both ULL and classical batch settings.

2. Related Work

2.1. Alleviating Catastrophic Forgetting in Lifelong Learning

Research on LL aims to learn knowledge in a continual fashion without performance degradation on previous tasks when trained for new data. Reference [11] have provided a comprehensive review on LL with neural networks. The main challenge of LL using Deep Neural Networks (DNNs) is that they often suffer from a phenomenon called *catastrophic forgetting* or *catastrophic interference*, where a model's performance on previous tasks may decrease abruptly due to the interference of training with new information [1, 12]. Recent work aims at adapting a learned model to new information while ensuring the performance on previous data does not decrease. Currently, there are no universal past knowledge preservation schemes for different algorithms and settings [13] in LL. Regularization methods target to reduce interference of new learning by minimizing changes to certain parameters that are important to previous learning tasks [14, 15]. Alternative approaches based on rehearsal have also been proposed to alleviate catastrophic forgetting while training DNNs sequentially. Rehearsal methods use either past data [16, 17], coreset data summarization [18] or a generative model [19] to capture the data distribution of previously seen data.

However, most of the existing LL methods focus on supervised learning tasks [14, 18, 7, 19], where each learning task performs supervised learning with output labels provided. In comparison, we propose a novel alternative knowledge preservation scheme in an unsupervised learning context via sufficient statistics. This is in contrast to existing work which uses previous model parameters [5, 20], representative items exacted from previous models [21] or past raw data, or coreset data summarization [16, 17, 18] as previous knowledge for new tasks. Our proposal to use sufficient statistics is novel and has the advantage of preserving past knowledge while allowing incremental updates as new data arrive, due to the additive property of sufficient statistics; without the need of storing previous data, which are often unavailable as new data arrive in LL.

2.2. Comparable Methods in Unsupervised Lifelong Learning

Recently, [22] proposed Continual Unsupervised Representation Learning (CURL) to deal with a fully ULL setting with unknown cluster labels. We have developed our idea independently in parallel with CURL but in a fully Bayesian framework. CURL is the most related and comparable method to ours in the literature. CURL focuses on learning representations and discovering new clusters using a threshold method. One major drawback of CURL is that it has over-clustering issues as shown in their real data experiment. We also show this empirically and demonstrate the improvement of our method over CURL in our experiment section. In contrast to CURL, we provide a novel probabilistic framework with a nonparametric Bayesian prior to allow the model to expand without bound automatically instead of using an ad hoc threshold method as in CURL. We develop a novel end-to-end variational inference strategy for learning deep representations and detecting novel clusters in ULL simultaneously.

2.3. Bayesian Lifelong Learning

A Bayesian formulation is a natural choice for LL since it provides a systematic way to incorporate previously learnt information in the prior distribution and obtain a posterior distribution that combines both prior belief and new information. The sequential nature of Bayes theorem also paves the way to recursively update an approximation of the posterior distribution and then use it as a new prior to guide the learning for new data in LL.

The most related work to ours with a Bayesian formulation in LL is [18], which provides a variational online inference framework for deep discriminative models and deep generative models, where they studied the approximate posterior distribution of the parameters in DNNs in a continual fashion. However, our work differentiates from [18] due to different objectives, inference strategies and knowledge preservation techniques. For deep generative models, [18] focuses on learning the approximate posterior distribution of the parameters for Variational Autoencoders (VAE) [23] in LL but are not able to find out the latent clustering structure of the data or detect new clusters for emerging data. Although both under a Bayesian framework, our work is different from [18] since we develop a novel Bayesian framework for representation learning and discovering latent clustering structure and new clusters on the fly together with a novel end-to-end variational inference strategy in an ULL context.

2.4. Deep Generative Unsupervised Learning Methods in a Batch Setting

Recent research has focused on combining deep generative models to learn good representations of the original data and conduct clustering analysis in an unsupervised learning context [23, 24, 25, 26, 27]. However, the latest existing methods are designed for an independent and identically distributed (i.i.d.) batch training mode instead of a LL context. The majority of these methods are in a static unsupervised learning setting, where the number of clusters are fixed in advance. Thus, these methods cannot detect potential new clusters when new data arrive or the data distribution changes. These methods cannot adapt to a LL setting.

To summarize, our work fills the gap by providing a fully Bayesian framework for ULL, which has the unique capacity to use a deep generative model for representation learning while performing new cluster discovery on the fly with a nonparametric Bayesian prior and our proposed CERR technique. To alleviate catastrophic forgetting challenge in LL, we propose to use sufficient statistics to maintain knowledge as a novel alternative to existing methods. We further develop an end-to-end Bayesian inference strategy DBULL to achieve our goal.

3. Model

3.1. Problem Formulation

In our ULL setting, a sequence of datasets $D_1, D_2, \ldots, D_N$ arrive in a streaming order. When a new dataset D_N arrives in memory, the previous dataset D_{N-1} is no longer available. Our goal is to automatically learn the clusters (unlabeled classes) in each dataset.

Let $\boldsymbol{x} \in X$ represent the unlabeled observation of the current dataset in memory, where X can be a high-dimensional data space. We assume that a low-dimensional latent representation $\boldsymbol{z}$ can be learned from $\boldsymbol{x}$ and in turn can be used to reconstruct $\boldsymbol{x}$. We assume that the variation among observations $\boldsymbol{x}$ can be captured by its latent representation $\boldsymbol{z}$. Thus, we let $\boldsymbol{y}$ represent the unknown cluster membership of $\boldsymbol{z}$ for observation $\boldsymbol{x}$.

We target to find: (1) a good low-dimensional latent representation $\boldsymbol{z}$ from $\boldsymbol{x}$ to efficiently extract knowledge from the original data; (2) the clustering structure within the new dataset with the capacity to discover potentially novel clusters without forgetting the previously learned clusters of the seen datasets; and, (3) an incremental learning strategy to optimize the cluster learning performance for a new dataset without dramatically degrading the clustering performance in seen datasets.

We summarize our work in a flow chart in Fig. 1 and describe the generative process of our graphical model for DBULL in the next section.

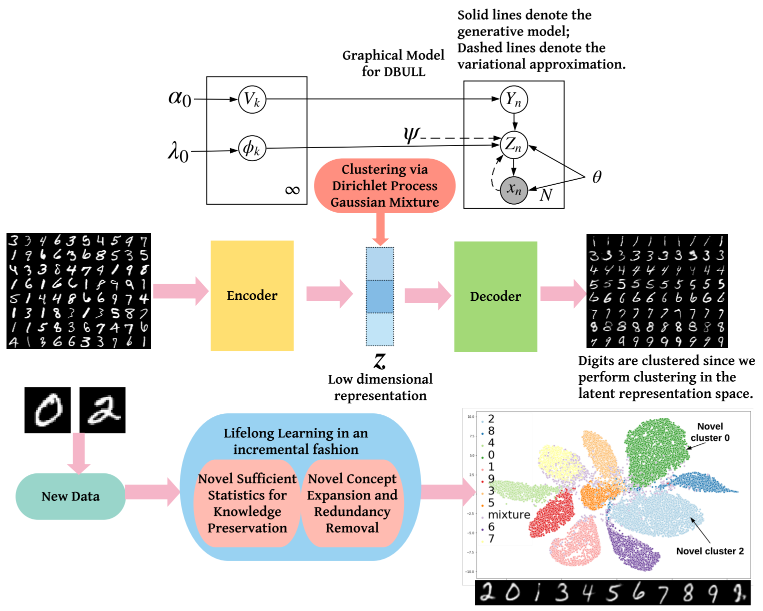
Solid lines denote the generative model; Dashed lines denote the variational approximation.
Graphical Model for DBULL
α_0
λ_0
V_k
ϕ_k
∞
ψ
Y_n
Z_n
x_n
N
θ
Clustering via Dirichlet Process Gaussian Mixture
Encoder
Decoder
z
Low dimensional representation
Digits are clustered since we perform clustering in the latent representation space.
New Data
Lifelong Learning in an incremental fashion
Novel Sufficient Statistics for Knowledge Preservation
Novel Concept Expansion and Redundancy Removal
Novel cluster 0
Novel cluster 2
2
8
4
0
1
9
3
5
mixture
6
7

Figure 1: Flow chart of our work. This figure is best viewed in color.

3.2. Generative Process of DBULL

The generative process for DBULL is as follows.

(a) Draw a latent cluster membership $\boldsymbol{y} \sim \text{Cat}(\pi(\boldsymbol{v}))$, where the vector $\boldsymbol{v}$ comes from the stick-breaking construction of a Dirichlet Process (DP).

(b) Draw a latent representation vector $\boldsymbol{z}|\boldsymbol{y} = k \sim \mathcal{N}\left(\boldsymbol{\mu}_k^*, \boldsymbol{\sigma}_k^{*2}\boldsymbol{I}\right)$, where k is the cluster membership sampled from (a).

(c) Generate data $\boldsymbol{x}$ from $\boldsymbol{x}|\boldsymbol{z} = \boldsymbol{z} \sim \mathcal{N}\left(\boldsymbol{\mu}(\boldsymbol{z};\theta), \text{diag}(\boldsymbol{\sigma}^2(\boldsymbol{z};\theta))\right)$ in the original data space.

In (a), $\pi(\boldsymbol{v})$ represents the probabilities for each cluster and the value of $\pi(\boldsymbol{v})$ depends on a vector of scalars $\boldsymbol{v}$ coming from the stick-breaking construction of a DP [28]. CURL uses a latent mixture of Gaussian components to capture the clustering structure in an unsupervised learning context. In comparison, we adopt the DP mixture model in the latent space with the advantages that the number of mixture components can be random and grow without bound as new data arrive, which is an appealing property desired by LL. We further explain in Section 4.2 why DP is an appropriate prior for our problem in details. In (b), $\boldsymbol{z}$ is considered a low-dimensional latent representation of the original data $\boldsymbol{x}$. We describe in Section 4.2 that a DP Gaussian mixture is used for modelling $\boldsymbol{z}$ since it is often assumed that the variation in $\boldsymbol{z}$ is able to reflect the variation within $\boldsymbol{x}$. The current representation in (b) is for easy understanding. In (c), we assume that the generative model $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is parameterized by $g_\theta : Z \to X$ and $g_\theta(\boldsymbol{z}) = \big(\boldsymbol{\mu}(\boldsymbol{z};\theta), \boldsymbol{\sigma}^2(\boldsymbol{z};\theta)\big)$, where g_θ is chosen as DNNs due to its powerful function approximation and good feature learning capabilities [29, 30, 31].

Under this generative process, the joint probability density function can be factorized as

$$p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\phi}, \boldsymbol{v}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}|\boldsymbol{y})p(\boldsymbol{y}|\pi(\boldsymbol{v}))p(\boldsymbol{v})p(\boldsymbol{\phi}), \tag{1}$$

where $\boldsymbol{\phi} = (\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_k)$ and $\boldsymbol{\phi}_k = (\boldsymbol{\mu}_k^*, \boldsymbol{\sigma}_k^{*2})$ represents the parameters of the kth mixture component (or cluster), $p(\boldsymbol{v})$ and $p(\boldsymbol{\phi})$ represent the prior distribution for $\boldsymbol{v}$ and $\boldsymbol{\phi}$.

Next, we discuss how to choose appropriate $p(\boldsymbol{v})$ and $p(\boldsymbol{\phi})$ to endow our model with the flexibility to grow the number of mixture components without bound with new data in a LL setting.

4. Why Bayesian for DBULL

In this section, we illustrate why Bayesian framework is a natural choice for our ULL setting. Recall that we have a sequence of datasets $D_1, D_2, \ldots, D_N$ from a single domain arriving in a streaming order. To mimic a LL setting, we assume each time only one dataset can fit in memory. One key question in LL is how to efficiently maintain past knowledge to guide future learning.

4.1. Bayesian Reasoning for Lifelong Learning

Bayesian framework is a suitable solution to this type of learning since it learns a posterior distribution, or an approximation of a posterior distribution, that takes advantage of both the prior belief and the additional information in the new dataset. The sequential nature of Bayes theorem ensures valid recursive updates on an approximation of the posterior distribution given the observations. Later, the approximation of the posterior distribution serves as the new prior to guide future learning for new data in LL. Before describing our inference strategy, we first explain why utilizing the Bayesian updating rule is valid for our problem.

Given $(N-1)$ datasets $\mathcal{D}_i$, where $i = 1, 2, \ldots, (N-1)$, the posterior after considering the Nth dataset is

$$\begin{aligned} &P(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\phi}, \boldsymbol{v} | \mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_N) \\ &\propto P(\mathcal{D}_N | \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\phi}, \boldsymbol{v}) P(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\phi}, \boldsymbol{v} | \mathcal{D}_1, \ldots, \mathcal{D}_{N-1}), \end{aligned} \tag{2}$$

which reflects that the posterior of $(N-1)$ tasks and datasets can be considered as the prior for the next task and dataset. If we know exactly the normalizing constant for $P(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\phi}, \boldsymbol{v} | \mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_N)$ and $P(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\phi}, \boldsymbol{v} | \mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_{N-1})$, repeatedly updating (2) is streaming without the need of reusing past data. However, it is often intractable to compute the normalizing constant exactly. Thus, an approximation of the posterior distribution is necessary to update (2) since the exact posterior is infeasible to obtain.

4.2. Dirichlet Process Prior

The DP is often used as a nonparametric prior for partitioning exchangeable observations into discrete clusters. The DP mixture is a flexible mixture model where the number of mixture components can be random and grow without bound as more data arrive. These properties make it a natural choice for our LL setting. In practice, we show in our inference how we expand and merge the number of mixture components as new data arrive by starting from only one cluster in Section 5.6. Next, we briefly review DP and introduce our DP Gaussian mixture model to derive the joint probability density $p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\phi}, \boldsymbol{v})$ defined in (1).

A DP is characterized by a base distribution G_0 and a parameter α denoted as $\mathrm{DP}(G_0, \alpha)$. A constructive definition of DP via a stick-breaking process is of the form $G(\cdot) = \sum_{k=1}^{\infty} \pi_k \delta_{\boldsymbol{\phi}_k}$, where $\delta_{\boldsymbol{\phi}_k}$ is a discrete measure concentrated at $\boldsymbol{\phi}_k \sim G_0$, which is a random sample from the base distribution G_0 with mixing proportion π_k [32]. In DP, the π_ks are random weights independent of G_0 but satisfy $0 \leqslant \pi_k \leqslant 1$ and $\sum_{k=1}^{\infty} \pi_k = 1$. The weights π_k can be drawn through an iterative process:

$$\pi_k = \begin{cases} v_1, & \text{if} \quad k = 1, \\ v_k \prod_{j=1}^{k-1}(1 - v_j), & \text{for} \quad k > 1, \end{cases}$$

where $v_k \sim \mathrm{Beta}(1, \alpha)$.

Under the generative process of DBULL in Section 3.2, these π_ks represent the probabilities for each cluster (mixture component) used in step (a) and $\boldsymbol{\phi}_k$ can be seen as the parameters of the Gaussian mixture for $\boldsymbol{z}$ in step (b). Thus, given our generative process, the corresponding joint probability density for our model is

$$\begin{aligned} p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\phi}, \boldsymbol{v}) &= p(\boldsymbol{x}|\boldsymbol{z}; \theta) p(\boldsymbol{z}|\boldsymbol{y}) p(\boldsymbol{y}|\boldsymbol{v}) p(\boldsymbol{v}) G_0(\boldsymbol{\phi}|\boldsymbol{\lambda}_0) \\ &= \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}(\boldsymbol{z}_n; \theta), \mathrm{diag}(\boldsymbol{\sigma}^2(\boldsymbol{z}_n, \theta))) \\ &\quad \prod_{k=1}^{\infty} \mathcal{N}(\boldsymbol{z}_n | \boldsymbol{\mu}_k^*, \boldsymbol{\sigma}_k^{*2} \boldsymbol{I}) P(\boldsymbol{y}_n = k | \pi(\boldsymbol{v})) \\ &\quad \mathrm{Beta}(v_k | 1, \alpha_0) G_0(\boldsymbol{\phi}_k | \lambda_0). \end{aligned} \tag{3}$$

For a Gaussian mixture model, the base distribution is often chosen as the Normal-Wishart (NW) denoted as $G_0 = \mathrm{NW}(\boldsymbol{\lambda}_0)$ to generate the mixture parameters $\boldsymbol{\phi}_k = (\boldsymbol{\mu}_k^*, \boldsymbol{\sigma}_k^*) \sim \mathrm{NW}(\boldsymbol{\lambda}_0)$, where $\boldsymbol{\lambda}_0 = (m_0, \beta_0, \nu_0, W_0) = (\mathbf{0}, 0.2, D + 2, I_{D \times D})$, and D is the dimension of the latent vector $\boldsymbol{z}$. The values of the hyper-parameter $\boldsymbol{\lambda}_0$ are conventional choices in the Bayesian nonparameteric literature for Gaussian mixture. Moreover, the performance of our method is robust to the hyper-parameter values.

5. Inference for DBULL

There are several new challenges to develop an end-to-end inference algorithm for our problem under the ULL setting that do not exist in the batch setting: one has to deal with catastrophic forgetting, mechanisms for past knowledge preservation, and dynamic model expansion capacity for novel cluster discovery. For pedagogical reasons, we first describe our general parameter learning strategy via variational inference for DBULL in a standard batch setting. We then further describe how we resolve the additional challenges in the lifelong (streaming) learning setting. We describe our novel components in the inference algorithm in terms of a new knowledge preservation scheme via sufficient statistics in Sections 5.4 and an automatic CERR strategy in Section 5.6. To help understand the role of the sufficient statistics in preserving past knowledge and its contribution to the probabilistic density function of our problem, we explain this in Section 5.5.

5.1. Variational Inference and ELBO Derivation

In practice, it is often infeasible to obtain the exact posterior distribution since the normalizing constant in the posterior distribution is intractable. Markov Chains Monte Carlo (MCMC) methods are a family of algorithms that provide a systematic way to sample from the posterior distribution but is often slow in a high-dimensional parameter space. Thus, effective alternative methods are needed. Variational inference is a promising alternative, which approximates the posterior distribution by casting inference as an optimization problem. It aims to find a surrogate distribution that is the most similar to the distribution of interest over a class of tractable distributions that can minimize the Kullback-Leibler (KL) divergence to the exact posterior distribution. Minimizing the KL divergence between $q(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\phi}, \boldsymbol{v}|\boldsymbol{x})$ and $p(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\phi}, \boldsymbol{v}|\boldsymbol{x})$ in our setting is equivalent to maximizing the Evidence Lower Bound (ELBO), where $q(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\phi}, \boldsymbol{v}|\boldsymbol{x})$ is the variational posterior distribution used to approximate the true posterior distribution. To make it easier for the readers to understand the core idea, we provide a high-level explanation of variational inference here instead of the mathematical derivation details.

Given the generative process in Section 3.2 and using Jensen's inequality, we obtain

$$\begin{aligned}\log p(\boldsymbol{x}) &\geqslant \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left\{\log \frac{p(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v})}{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\right\} \\ &= \mathcal{L}_{\mathrm{ELBO}}(\boldsymbol{x}),\end{aligned} \tag{4}$$

For simplicity, we assume that $q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x}) = q_\psi(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{y})q(\boldsymbol{v})q(\boldsymbol{\phi})$. Thus, the ELBO is

$$\begin{aligned}&\mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}|\boldsymbol{y},\boldsymbol{\phi})p(\boldsymbol{y}|\boldsymbol{v})p(\boldsymbol{v})p(\boldsymbol{\phi})}{q_\psi(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{y})q(\boldsymbol{v})q(\boldsymbol{\phi}))}\right] \\ &= \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] + \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log p(\boldsymbol{z}|\boldsymbol{y},\boldsymbol{\phi})\right] \\ &- \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log q_\psi(\boldsymbol{z}|\boldsymbol{x})\right] \\ &+ \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log p(\boldsymbol{y}|\boldsymbol{v})\right] + \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log p(\boldsymbol{v})\right] \\ &- \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log q(\boldsymbol{y})\right] - \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log q(\boldsymbol{v})\right] \\ &- \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log q(\boldsymbol{\phi})\right] + \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log p(\boldsymbol{\phi})\right]\end{aligned} \tag{5}$$

We assume our variational distribution takes the form of

$$\begin{aligned}&q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x}) = q_\psi(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{\phi})q(\boldsymbol{v})q(\boldsymbol{y}) \\ &= \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x};\psi), \mathrm{diag}(\boldsymbol{\sigma}^2(\boldsymbol{x};\psi))) \prod_{t=1}^{T-1} q_{\eta_t}(v_t) \prod_{t=1}^{T} q_{\zeta_t}(\boldsymbol{\phi}_t) \prod_{n=1}^{N} q_{\rho_n}(\boldsymbol{y}_n) \\ &= \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x};\psi), \mathrm{diag}(\boldsymbol{\sigma}^2(\boldsymbol{x};\psi))) \prod_{t=1}^{T-1} \mathrm{Beta}(\eta_{t_1}, \eta_{t_2}) \\ &\prod_{t=1}^{T} \mathcal{N}(\boldsymbol{\mu}_t|\boldsymbol{m}_t, (\boldsymbol{\beta}_t\boldsymbol{\Lambda}_t)^{-1})\mathcal{W}(\boldsymbol{\Lambda}_t|W_t,\nu_t) \prod_{n=1}^{N} \mathrm{Mult}(T,\rho_n),\end{aligned} \tag{6}$$

where we denote $f_\psi(\boldsymbol{x}) = (\boldsymbol{\mu}(\boldsymbol{x};\psi), \boldsymbol{\sigma}^2(\boldsymbol{x};\psi))$, which is a neural network, T is the number of mixture components in the DP of the variational distribution, $\boldsymbol{z}_n \sim \mathcal{N}(\boldsymbol{z}_n|\boldsymbol{\mu}_t, \boldsymbol{\Lambda}_t^{-1})$, $\boldsymbol{\phi}_t = (\boldsymbol{\mu}_t, \boldsymbol{\Lambda}_t)$, and $\mathrm{Mult}(T,\rho_n)$ is a Multinomial distribution. The notation definitions in equation (A.1), (A.2) and (A.6) are provided in

Table 1. Our inference strategy starts with only one mixture component and uses CERR technique described in Section 5.6 to either increase or merge the number of clusters.

5.2. General Parameter Learning Strategy

In equation (A.1), there are mainly two types of parameters we need to optimize the ELBO. The first type includes the neural network's parameters θ and ϕ when learning the latent representation $\boldsymbol{z}$ for the original data $\boldsymbol{x}$. The other type involves the latent cluster membership $\boldsymbol{y}$ and the parameters for the DP Gaussian mixture model.

In order to perform joint inference for both types of parameters, we adopt the alternating optimization strategy. First we update the neural network parameters (θ and ψ) to learn the latent representation $\boldsymbol{z}$ given the DP Gaussian mixture parameters. This is achieved by optimizing $\mathcal{L}_{\text{ELBO-VAE}}$, which only involves the first three terms of equation (A.1) that make a contribution to optimize θ, ψ and $\boldsymbol{z}$. Under our variational distribution assumptions in (A.2), by taking advantage of the reparameterization trick [23] and the Monte Carlo estimate of expectations, we obtain

$$\begin{aligned}
&\mathcal{L}_{\text{ELBO}-\text{VAE}}(\boldsymbol{x})\\
&= -\frac{1}{2}\sum_{k=1}^{T} N_k\nu_k \left\{\text{Trace}(\boldsymbol{U}_k W_k)\right\} - \frac{1}{2}\sum_{k=1}^{T} N_k\nu_k \left\{(\bar{z}_k - \boldsymbol{m}_k)^T W_k (\bar{z}_k - \boldsymbol{m}_k)\right\}\\
&\quad - \frac{1}{2}\frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{N}\sum_{j=1}^{D}\left(\log(\boldsymbol{\sigma}(\boldsymbol{z},\theta)^2)_j^{(l)} + \frac{\left(\boldsymbol{x}_{ij} - \boldsymbol{\mu}(\boldsymbol{z};\theta)_j^{(l)}\right)^2}{(\boldsymbol{\sigma}(\boldsymbol{z};\theta)^2)_j^{(l)}}\right) + \frac{1}{2}\log(\text{Det}(2\pi e\Sigma)).
\end{aligned} \tag{7}$$

We define the notations in Table 1. The derivation details are provided in Appendix A. Then, we update the DP Gaussian mixture parameters and the cluster membership given the current neural network parameters θ, ψ and the latent representation $\boldsymbol{z}$. This allows us to use improved latent representation to infer latent cluster memberships and the updated clustering will in turn facilitate learning latent knowledge representation. The update equations for

Table 1: Notations in $\mathcal{L}_{\text{ELBO}-\text{VAE}}(\boldsymbol{x})$.

Notations in the ELBO
N: the total number of observations.
D: the dimension of the latent representation $\boldsymbol{z}$.
Σ : $\text{diag}(\boldsymbol{\sigma}^2(\boldsymbol{x};\psi))$.
$\boldsymbol{x}_{ij}$: the jth dimension of the nth observation.
$\boldsymbol{y}_n$: cluster membership for the nth observation.
$p(\boldsymbol{y}_n = k) = \gamma_{ik}, N_k = \sum_{n=1}^{N} \gamma_{nk}$.
L: the number of Monte Carlo samples in Stochastic Gradient Variational Bayes (SGVB).
$\hat{\boldsymbol{z}}_n = \frac{1}{L}\sum_{l=1}^{L} \boldsymbol{z}_n^{(l)}$. $\bar{\boldsymbol{z}}_k = \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{nk}\hat{\boldsymbol{z}}_n$.
$\boldsymbol{U}_k = \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{nk}(\hat{\boldsymbol{z}}_n - \bar{\boldsymbol{z}}_k)(\hat{\boldsymbol{z}}_n - \bar{\boldsymbol{z}}_k)^T$.
$\beta_k = \beta_0 + N_k$: the posterior scalar precision in NW distribution.
$\boldsymbol{m}_k = \frac{1}{\beta_k}(\beta_0\boldsymbol{m}_0 + N_k\bar{z}_k)$: the posterior mean of cluster k.
$W_k^{-1} = W_0^{-1} + N_k S_k + \frac{\beta_0 N_k}{\beta_0 + N_k}(\bar{\boldsymbol{z}}_k - \boldsymbol{m}_0)(\bar{\boldsymbol{z}}_k - \boldsymbol{m}_0)^T$.
$\nu_k = \nu_0 + N_k$: the kth posterior degrees of freedom of NW.

DP mixture model parameters can be found in [10]. We describe the core idea of automatic cluster expansion and redundancy removal strategy in our inference in Section 5.6 to explain how we start with only one cluster and achieve dynamic model expansion by creating new mixture components (clusters) given new data in LL.

Our general parameter learning strategy via variational inference may seem straightforward for a batch setting at first glance. However, both the derivation and the implementation is nontrivial especially when incorporating our new components in the end-to-end inference procedure to address the additional challenges in a LL setting. For illustration purposes, we choose to describe the high level core idea of our inference procedure. The main difficulty lies in how to adapt our inference algorithm from a batch setting to a LL setting, which requires us to overcome catastrophic forgetting, maintain past knowledge and develop a dynamic model that can expand with automatic cluster discovery and redundancy removal capacity. Next, we describe our novel solutions.

5.3. Our Ingredients for Alleviating Catastrophic Forgetting

Catastrophic forgetting or *catastrophic interference* is a dramatic issue for DNNs as witnessed in SLL [14, 19]. In our ULL setting, the issue is even more challenging since we have more sources than SLL that may lead to abrupt model performance decrease due to the interference of training with new data. The first source is the same as in SLL when the DNNs forget previously learned information upon learning new information. Additionally, in an unsupervised setting, the model is not able to recover the learned cluster membership and clustering related parameters in the DP mixture model when the previous data is no longer available, or when the previous learned information of DNNs has been wiped out upon learning new information, since the clustering structure learned depends on the latent representation of the raw data, which is determined by the DNNs' parameters and the data distributions.

To resolve these issues, we develop our own novel solution via a combination of two ingredients: (1) generating and replaying a fixed small number of samples based on our generative process in Section 3.2 given the current DNNs and DP Gaussian mixture parameter estimates, which is a computationally effective byproduct of our algorithm; and, (2) developing a novel hierarchical sufficient statistics knowledge preservation strategy to remember the clustering information in an unsupervised setting.

We choose to replay a number of generative samples to preserve the previous data distribution instead of using a subset of past real data, since storing past data may require large memory and such data storage and replay may not be feasible in real big data applications. More details of replaying deep generative samples over real data in LL have been discussed in [19]. Moreover, our proposal to use sufficient statistics is novel and has the advantage of allowing incremental updates of the clustering information as new data arrive without the need of access to previous data because of the additive property of sufficient statistics. We introduce this novel strategy in the next section.

5.4. Sufficient Statistics for Knowledge Preservation

As LL is an emerging field, and there is no well-accepted knowledge definition or an appropriate representation scheme to efficiently maintain past knowledge from seen data. Researchers have adopted prior distributions [18] or model parameters [33] to represent past knowledge in most SLL problems, where achieving high prediction accuracy incrementally is the main objective. However, there is no guidance on preserving past knowledge in an unsupervised learning setup.

We propose a novel knowledge preservation strategy in DBULL. In our problem, there are two types of knowledge to maintain. The first contains previously learned DNNs' parameters needed to encode the latent knowledge representation $\boldsymbol{z}$ of the raw data and the reconstruction of the real data from $\boldsymbol{z}$. The other involves the DP Gaussian mixture parameters to represent different cluster characteristics and different cluster mixing proportions. Our novel knowledge representation scheme uses *hierarchical sufficient statistics* to preserve the information related to the DP Gaussian mixture. We develop a sequential updating rule to update our knowledge.

Assume that we have encountered N datasets $\{\mathcal{D}_j\}_{j=1}^N$ and each time only one dataset $\mathcal{D}_j$ can be in memory. While in memory, each dataset can be divided into M mini-batches $\{\mathcal{B}_i\}_{i=1}^M$. To define the sufficient statistics, we first define the *global* parameters of the DP Gaussian mixture as probabilities of each mixture component (cluster) $\pi_k s$ and the mixture parameters $(\boldsymbol{\mu}_k^*, \boldsymbol{\sigma}_k^*)$ for each cluster k. We define the *local* parameters as the cluster membership for each observation in memory. To remember the characteristics of all encountered data and the local information of the current dataset, we memorize three levels of sufficient statistics. The ith *mini-batch sufficient statistics* $S_k^{j,i} = (N_k(\mathcal{B}_i), s_k(\mathcal{B}_i))$ of the current dataset D_j, where $s_k(\mathcal{B}_i) = \sum_{n \in \mathcal{B}_i} \hat{\gamma}_{nk} t(z_n)$ and $t(\boldsymbol{z}_n)$ is the sufficient statistics to represent a distribution within the exponential family (Gaussian distribution is within the exponential family and $t(\boldsymbol{z}_n) = (\boldsymbol{z}_n, \boldsymbol{z}_n^T \boldsymbol{z}_n)$ in our case) and $\hat{\gamma}_{nk}$ represents the estimated probability of the nth observations in mini-batch $\mathcal{B}_i$ belonging to cluster k. We also define the *stream suffi-*

cient statistics $S_k^j = \sum_{i=1}^{M} S_k^{j,i}$ of dataset $\mathcal{D}_j$ and the *overall sufficient statistics* $S_k^0 = (N_k, s_k(\boldsymbol{z}))$ of all encountered datasets $\{\mathcal{D}_j\}_{j=1}^N$.

To efficiently maintain and update our knowledge, we develop our updating algorithm as: (1) substract the old summary of each mini-batch and update the local parameters; (2) compute a new summary for each mini-batch; and, (3) update the stream sufficient statistics for each cluster learned in the current dataset.

$$S_k^j \leftarrow S_k^j - S_k^{j,i}, \tag{8}$$

$$S_k^{j,i} \leftarrow \left(\sum_{n \in \mathcal{B}_i} \hat{\gamma}_{nk}, \sum_{n \in \mathcal{B}_i} \hat{\gamma}_{nk} t(\boldsymbol{z}_n) \right), \tag{9}$$

$$S_k^j \leftarrow S_k^j + S_k^{j,i}. \tag{10}$$

For the dataset $\mathcal{D}_j$ in the learning phase, we repeat the updating process multiple iterations to refine our training while learning the DP Gaussian mixture parameters and the cluster membership. Finally, we update the overall sufficient statistics by $S_k^0 \leftarrow S_k^0 + S_k^j$. The correctness of the algorithm is guaranteed by the additive property of the sufficient statistics.

5.5. Contribution of Sufficient Statistics to Alleviate Forgetting

The sufficient statistics alleviate forgetting by preserving data characteristics and allow sequential updates in LL without the need of saving real data. To be precise, the sufficient statistics allow us to update the log-likelihood and the ELBO sequentially since both terms are linear functions of the expectation of the sufficient statistics. Given the expected sufficient statistics, we are able to evaluate the first two terms of $\mathcal{L}_{\text{ELBO}-\text{VAE}}(\boldsymbol{x})$ in equation (A.6) and $p(\boldsymbol{z}|\boldsymbol{y})p(\boldsymbol{y}|\boldsymbol{v})$ in the joint probability density function of our model in equation (3). Next, we provide mathematical derivations to illustrate this.

Define sufficient statistics of all data $S^0 = (S_1^0, S_2^0, ., S_k^0)$, where k is the number of clusters. Define $S_k^0 = (N_k, \sum_{n=1}^N r_{nk} t(\boldsymbol{z}_n))$, where $t(\boldsymbol{z}_n) = (\boldsymbol{z}_n, \boldsymbol{z}_n^T \boldsymbol{z}_n)$ and r_{nk} denotes the probability of the nth observation belonging to cluster k, $N_k = \sum_{n=1}^N r_{nk}$, and N is the total number of observations. Given the sufficient statistics and current mixture parameters $(\boldsymbol{\mu}_k^*, \boldsymbol{\sigma}_k^*)$ and $\pi(v)$, we can evaluate $\sum_{k=1}^{\infty} N(\boldsymbol{z}_n|\boldsymbol{\mu}_k^*, \boldsymbol{\sigma}_k^{*2} I)P(\boldsymbol{y}_n = k|\pi(v))$ in the joint probability density function in equation (3) without storing each latent representation $\boldsymbol{z}_n$ for all data. Similarly, we can also evaluate the first two terms of $\mathcal{L}_{\text{ELBO}-\text{VAE}}(\boldsymbol{x})$ in equation (A.6) with notations defined in Table 1.

5.6. Cluster Expansion and Redundancy Removal Strategy

Our model starts with one cluster and, as we have new data with different characteristics, we expect the model to either dynamically grow the number of mixture components (clusters) or merge clusters if they have similar characteristics. To achieve this, we perform birth and merge moves in a similar fashion as the Nonparametric Bayesian literature [34] to allow automatic CERR. However, we would like to emphasize that our work is different from [34] since our merge moves have extra constraints. To avoid losing information about clusters learned earlier, we only allow merge moves to happen between two novel clusters from the birth move or one existing cluster with a newborn cluster. Two previously existing clusters cannot be merged. Reference [34] is designed for a batch learning, thus, it does not require this constraint (but in LL this constraint is important for avoiding information loss).

It is challenging to give birth to new clusters with streaming data since the number of observations may not be sufficient to inform good proposals for new clusters. To resolve this issue, we follow the procedure in [34] by collecting a subsample of data for each learned cluster k. Then, we cache the samples in the subsample if the probability of the nth observation to be assigned to cluster k is bigger than a threshold of value 0.1. We fit the DP Gaussian mixture to the cached samples with $k = 1$ cluster and expand the current model with 10 novel clusters. However, only adopting the birth moves may overcluster the observations into different clusters. After the birth move, we merge the clusters by (1) selecting candidate clusters to merge and by (2) merging two selected clusters if ELBO improves. The candidate clusters are selected if the marginal likelihood of the two merged clusters is bigger than the marginal likelihood when

Table 2: Summary statistics for benchmark datasets.

Dataset	# Samples	Dimension	# Classes
MNIST	70000	784	10
Reuters10k	10000	2000	4
STL-10	13000	2048	10

keeping the two clusters separate.

6. Experiments

Datasets. We adopt the most common text and image benchmark datasets in LL to evaluate the performance of our method. The *MNIST* database of 70,000 handwritten digit images [35] is widely used to evaluate deep generative models [23, 25, 24, 27, 26] for representation learning and LL models in both supervised [14, 18, 19] and unsupervised learning contexts [22]. To provide a fair comparison with state-of-the-art competing methods and easy interpretation, we mainly use *MNIST* to evaluate the performance of our method and interpret our results with intuitive visualization patterns. To examine our method on more complex datasets, we use text *Reuters10k* [36] and image *STL-10* [37] databases. STL-10 is at least as hard as a well-known image database CIFAR-10 [38] since STL-10 has fewer labeled training examples within each class. The summary statistics for the datasets are provided in Table 2.

We adopt the same neural network architecture as in [26]. All experiments and implementation details will be publicly available once accepted.

Competing Methods in Unsupervised Lifelong Learning. CURL is the only lifelong unsupervised learning method currently with both representation learning and new cluster discovery capacity, which makes CURL the latest, most related, and comparable method to ours. We use **CURL-D** to represent CURL without the true number of clusters provided but to detect it **D**ynamically given unlabelled streaming data.

Competing Methods in a Classic Batch Setting. Although designed for LL, to show the generality of DBULL in a batch training mode, we compare it against recent deep (generative) methods with representation learning and clustering capacity designed for batch settings, including DEC [25], VaDE [26], CURL-F [22] and VAE+DP. CURL-F represents CURL with the true number of clusters provided as a **F**ixed value. VAE+DP fits a Variational Autoencoders (VAE) to learn latent representations first and then uses a DP to learn clustering in two separate steps. We list the capabilities of different methods in Table 3.

Table 3: DBULL and competing methods capacity comparison.

	Lifelong Learning		**Batch Setting**			
	DBULL	CURL-D	DEC	VaDE	VAE+DP	CURL-F
Representation Learning	**yes**	**yes**	**yes**	**yes**	**yes**	**yes**
Learns # of Clusters	**yes**	**yes**	no	no	**yes**	no
Dynamic Expansion	**yes**	**yes**	no	no	**yes**	no
Overcome Forgetting	**yes**	**yes**	no	no	no	**yes**

Evaluation Metrics. One of the main objectives of our method is to perform new cluster discovery with streaming non-stationary data. Thus, it is desired if our method can achieve superior clustering quality in both ULL and batch settings. We adopt the clustering quality metrics including Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), Homogeneity Score (HS), Completeness Score (CS) and V-measure Score (VM). These are all normalized metrics ranging from zero to one, and larger values indicate better clustering quality. NMI, ARI and VM of value one represent perfect clustering as the ground truth. CS is a symmetrical metric to HS. Detailed definitions for these metrics can be found in [39].

6.1. Lifelong Learning Performance Comparison

Experiment Objective. It is desired if LL methods can adapt a learned model to new data while retaining the information learned earlier. The objective of this experiment is to demonstrate DBULL has such desired capacity and effectiveness compared with state-of-the-art LL methods such that there is no dramatic performance decrease on past data even if the model has been updated with new information.

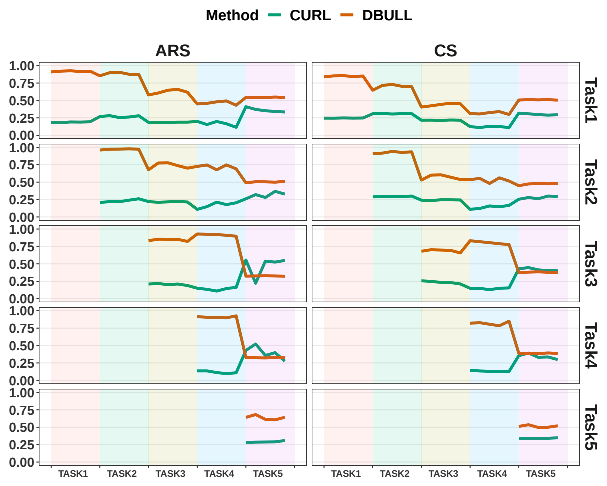

Figure 2: Clustering quality performances in terms of ARS and CS measured on each task after sequential training from TASK$_1$ to TASK$_5$ across every 500 iterations for each task.

Experiment Setup. To evaluate the performance of DBULL, we adopt the most common experiment setup called *Split MNIST* in LL, which used images from MNIST [15, 18]. We divide MNIST into 5 disjoint subsets with each subset containing 10,000 random samples of two digit classes in the order of digits 0-1, 2-3, 4-5, 6-7 and 8-9, denoted as DS$_1$, DS$_2$, DS$_3$, DS$_4$, and DS$_5$. Each dataset is divided into 20 subsets that arrive in a sequential order to mimic a LL setting. We denote DS$_{i:j}$ as all data from DS$_i$ to DS$_j$, where $i, j = 1, 2, \ldots, 5$.

Discussion on Performance. To check if our method has dramatic performance loss due to catastrophic forgetting, we sequentially train our method DBULL and its LL competitor CURL-D on $\text{DS}_1, \text{DS}_2, \ldots, \text{DS}_5$. We define $TASK_i$ as training on DS_i, where $i = 1, 2, \ldots, 5$. We measure the performance of TASK$_1$ after training TASK$_1$, TASK$_2$, TASK$_3$, TASK$_4$, TASK$_5$ with datasets $\text{DS}_1, \text{DS}_{1:2}, \text{DS}_{1:3}, \text{DS}_{1:4}$, and DS$_{1:5}$, the performance of TASK$_2$ after training $\text{TASK}_2, \text{TASK}_3, \text{TASK}_4, \text{TASK}_5$ with datasets $\text{DS}_2, \text{DS}_{2:3}, \text{DS}_{2:4}, \text{DS}_{2:5}$, etc. We report the LL clustering quality performance for each task after sequential training five tasks in Fig. 2 and Fig. 3.

Fig. 2 and Fig. 3 reflect that DBULL has better performance in handling

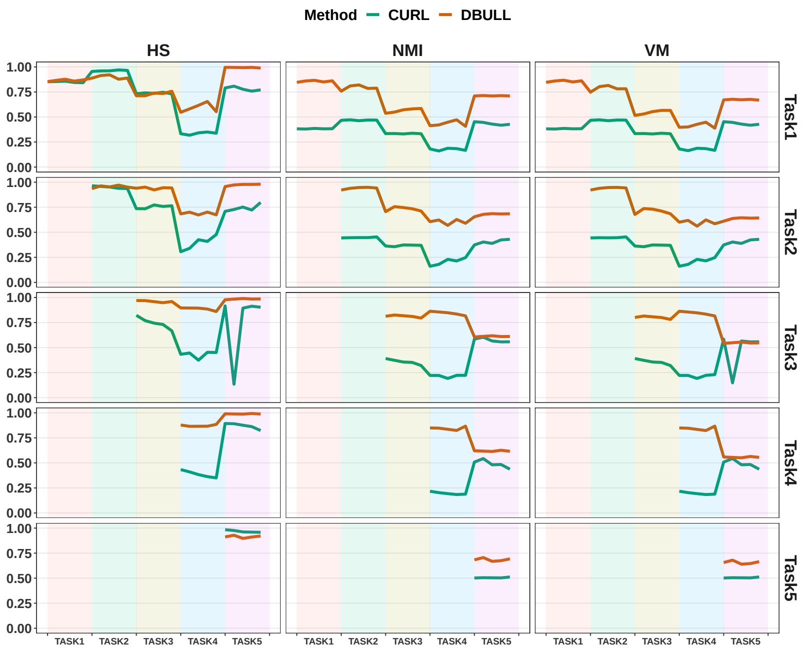

Figure 3: Clustering quality performances in terms of HS, NMI and VM measured on each task after sequential training from TASK_1 to TASK_5 across every 500 iterations for each task.

catastrophic forgetting than CURL-D since DBULL has slightly less performance drop than CURL-D for previous tasks in almost all scenarios in terms of nearly all clustering metrics.

Fig. 4 reflects that DBULL has advantages over CURL-D in handling over-clustering issues. Since each task has two digits, the true number of clusters seen after training each task sequentially is 2, 4, 6, 8, 10. The number of clusters automatically detected by DBULL after training $\text{TASK}_1, \ldots, \text{TASK}_5$ is 4, 6, 8, 10, 12. DBULL clusters digit 1 into three clusters of different handwritten patterns in TASK_1. For other digits, DBULL discovers each new digit into one exact cluster as the ground truth. In contrast, CURL-D clustered digits 0-1 into 14-16 clusters of TASK_1 and obtained 23-25 clusters for 10 digits after training five tasks sequentially. We provide visualization of the reconstructed cluster mean from the DP mixture model via our trained decoder of DBULL in Fig. 5.

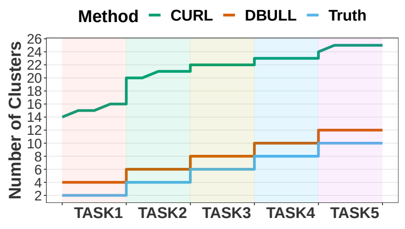

Figure 4: Number of clusters detected by CURL-D, DBULL after sequentially training from TASK_1 to TASK_5 across every 500 iterations for each task compared with the ground truth.

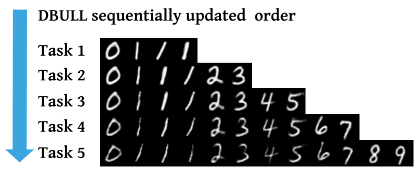

Figure 5: Decoded images using the DP Gaussian mixture posterior mean after sequentially trained from TASK_1 to TASK_5 using DBULL.

Besides the overall clustering quality reported, we also provide the precision and recall of DBULL to view the performance for each digit after sequentially training all tasks. CURL-D overclusters the 10 digits into 25 clusters, making it hard to report the precision and recall of each digit. To visualize the results, the three sub-clusters of digit one by DBULL have been merged into one cluster. Overall, there is no significant performance loss of previous tasks after sequentially training multiple tasks for digits 0, 1, 3, 4, 6, 8, 9. Digit 2 has experienced precision decrease after training TASK_4 of digits 6 and 7 since DBULL has trouble in differentiating some samples from digits 2 and 7.

6.2. Batch Setting Clustering Performance Comparison

Experiment Objective. The goal of this experiment is to demonstrate the generality our LL method DBULL, which can achieve comparable performance as competing methods in an unsupervised batch setting.

Discussion on Performance. The true number of clusters is provided to competing methods: DEC, VaDE and CURL-F in advance since the total number of clusters is required. DBULL, CURL-D, VAE+DP have less information than DEC, VaDE and CURL-F since they have no knowledge about the true number of clusters. DBULL, CURL-D and VAE+DP all start with one cluster and detect the number of clusters on the fly. Thus, if DBULL can achieve similar performance to DEC, VaDE and CURL-F and outperforms its LL counterpart CURL-D, it demonstrates DBULLs effectiveness. Table 4 shows DBULL performs the best in NMI, VM for MNIST and NMI, ARI and VM for STL10 and outperforms CURL-D in MNIST and STL10. Moreover, DBULL and DEC are more stable in terms of all evaluation metrics because of smaller standard error than other methods. We also report the number of clusters found by DBULL, CURL-D for MNIST, Reuters10k and STL-10 in Table 5 out of five replications. Table 5 shows that DBULL handles overclustering issues better in comparison with CURL-D. In summary, Table 4 and 5 demonstrate DBULL's effectiveness in a batch setting.

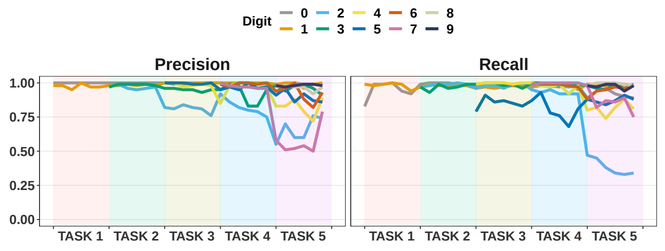

Figure 6: Precision and recall for each digit of DBULL evaluated after sequentially training from TASK$_1$ to TASK$_5$ across every 500 iterations for each task.

Table 4: Clustering quality (%) comparison averaged over five replications with both the average value and the standard error (in the parenthesis) provided.

Dataset	Method	NMI	ARI
MNIST	DEC	84.67 (2.25)	**83.67** (4.53)
	VaDE	80.35 (4.68)	74.06 (9.11)
	VAE+DP	81.70 (**0.825**)	70.49 (1.654)
	CURL-F	69.76 (2.51)	56.47 (4.11)
	CURL-D	63.51 (1.32)	36.84 (**1.98**)
	DBULL	**85.72** (1.02)	83.53 (2.35)
Reuters10k	DEC	46.56 (5.36)	46.86 (7.98)
	VaDE	41.64 (4.73)	38.49 (5.44)
	VAE + DP	41.62 (2.99)	37.93 (4.57)
	CURL-F	**51.92** (3.22)	**47.72** (4.00)
	CURL-D	46.31 (1.83)	22.00 (**3.60**)
	DBULL	45.32 (**1.79**)	42.66 (5.73)
STL10	DEC	71.92 (2.66)	58.73 (5.09)
	VaDE	68.35 (3.85)	59.42 (6.84)
	VAE+DP	43.18 (1.41)	26.58 (1.32)
	CURL-F	66.98 (3.38)	51.24 (4.06)
	CURL-D	65.71 (1.33)	37.96 (4.69)
	DBULL	**75.26** (**0.53**)	**70.72** (**0.81**)

Dataset	Method	HS	VM
MNIST	DEC	84.67 (2.25)	84.67 (2.25)
	VaDE	79.86 (4.93)	80.36 (4.69)
	VAE+DP	**91.27** (**0.215**)	81.19 (0.904)
	CURL-F	68.60 (2.56)	69.75 (2.51)
	CURL-D	76.35 (1.53)	62.45 (1.32)
	DBULL	89.34 (0.25)	**85.65** (**0.51**)
Reuters10k	DEC	48.44 (5.44)	46.52(5.36)
	VaDE	43.64 (4.88)	41.60 (4.73)
	VAE + DP	46.64 (3.85)	41.34 (2.94)
	CURL-D	**66.90** (2.09)	43.34 (**2.00**)
	CURL-F	54.38 (3.49)	**51.86** (3.21)
	DBULL	48.88 (**1.86**)	45.40 (2.04)
STL10	DEC	68.47 (3.48)	71.83 (2.72)
	VaDE	67.24 (4.23)	68.37 (3.92)
	VAE+DP	42.28 (**1.03**)	43.16 (1.39)
	CURL-F	65.46 (3.27)	66.96 (3.37)
	CURL-D	**80.86** (2.94)	64.31 (1.24)
	DBULL	77.61 (1.29)	**75.22** (**0.52**)

Table 5: Number of clusters found by DBULL and CURL-D out of five replications, where the upperbounds for the number of clusters for Reuters10k and STL-10 are set at 40 and 50.

Datasets	True # of Clusters	DBULL	CURL-D
MNIST	10	11-15	34
Reuters10k	4	5-10	40
STL-10	10	12-15	50

7. Conclusion

In this work, we introduce our approach DBULL for unsupervised LL problems. DBULL is a novel end-to-end approximate Bayesian inference algorithm, which is able to perform automatic new task discovery via our proposed dynamic model expansion strategy, adapt to changes in the evolving data distributions, and overcome forgetting using our proposed information extraction mechanism via summary sufficient statistics while learning the underlying representation simultaneously. Experiments on MNIST, Reuters10k and STL-10 demonstrate that DBULL has competitive performance compared with state-of-the-art methods in both a batch setting and an unsupervised LL setting.

Acknowledgment

The work described was supported in part by Award Numbers U01 HL089856 from the National Heart, Lung, and Blood Institute and NIH/NCI R01 CA199673.

References

[1] M. McCloskey, N. J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, in: Psychology of learning and motivation, Vol. 24, Elsevier, 1989, pp. 109–165.

[2] S. Thrun, T. M. Mitchell, Lifelong robot learning, Robotics and autonomous systems 15 (1-2) (1995) 25–46.

[3] S. Thrun, Is learning the n-th thing any easier than learning the first?, in: Advances in neural information processing systems, 1996, pp. 640–646.

[4] P. Ruvolo, E. Eaton, Ella: An efficient lifelong learning algorithm, in: International Conference on Machine Learning, 2013, pp. 507–515.

[5] Z. Chen, N. Ma, B. Liu, Lifelong learning for sentiment classification, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), 2015, pp. 750–756.

[6] S. S. Sarwar, A. Ankit, K. Roy, Incremental learning in deep convolutional neural networks using partial network sharing, IEEE Access.

[7] S. Hou, X. Pan, C. Change Loy, Z. Wang, D. Lin, Lifelong learning via progressive distillation and retrospection, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 437–452.

[8] P. Ruvolo, E. Eaton, Active task selection for lifelong machine learning, in: Twenty-seventh AAAI conference on artificial intelligence, 2013.

[9] D. Isele, M. Rostami, E. Eaton, Using task features for zero-shot knowledge transfer in lifelong learning., in: IJCAI, 2016, pp. 1620–1626.

[10] D. M. Blei, M. I. Jordan, et al., Variational inference for dirichlet process mixtures, Bayesian analysis 1 (1) (2006) 121–143.

[11] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, S. Wermter, Continual lifelong learning with neural networks: A review, Neural Networks.

[12] J. L. McClelland, B. L. McNaughton, R. C. O'Reilly, Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory., Psychological review 102 (3) (1995) 419.

[13] Z. Chen, B. Liu, Lifelong Machine Learning, Morgan & Claypool Publishers, 2018.

[14] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., Overcoming catastrophic forgetting in neural networks, Proceedings of the national academy of sciences 114 (13) (2017) 3521–3526.

[15] F. Zenke, B. Poole, S. Ganguli, Continual learning through synaptic intelligence, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 3987–3995.

[16] A. Robins, Catastrophic forgetting, rehearsal and pseudorehearsal, Connection Science 7 (2) (1995) 123–146.

[17] H. Xu, B. Liu, L. Shu, P. S. Yu, Lifelong domain word embedding via meta-learning, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 4510–4516.

[18] C. V. Nguyen, Y. Li, T. D. Bui, R. E. Turner, Variational continual learning, in: International Conference on Learning Representations (ICLR), 2018.

[19] H. Shin, J. K. Lee, J. Kim, J. Kim, Continual learning with deep generative replay, in: Advances in Neural Information Processing Systems, 2017, pp. 2990–2999.

[20] L. Shu, B. Liu, H. Xu, A. Kim, Lifelong-rl: Lifelong relaxation labeling for separating entities and aspects in opinion targets, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing, Vol. 2016, NIH Public Access, 2016, p. 225.

[21] L. Shu, H. Xu, B. Liu, Lifelong learning crf for supervised aspect extraction, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2017.

[22] D. Rao, F. Visin, A. Rusu, R. Pascanu, Y. W. Teh, R. Hadsell, Continual unsupervised representation learning, in: Advances in Neural Information Processing Systems, 2019, pp. 7645–7655.

[23] D. P. Kingma, M. Welling, Auto-encoding variational bayes, in: International Conference on Learning Representations (ICLR), 2014.

[24] M. Johnson, D. K. Duvenaud, A. Wiltschko, R. P. Adams, S. R. Datta, Composing graphical models with neural networks for structured representations and fast inference, in: Advances in neural information processing systems, 2016, pp. 2946–2954.

[25] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: International conference on machine learning, 2016, pp. 478–487.

[26] Z. Jiang, Y. Zheng, H. Tan, B. Tang, H. Zhou, Variational deep embedding: an unsupervised and generative approach to clustering, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, AAAI Press, 2017, pp. 1965–1972.

[27] P. Goyal, Z. Hu, X. Liang, C. Wang, E. P. Xing, Nonparametric variational auto-encoders for hierarchical representation learning, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5094–5102.

[28] J. Sethuraman, R. C. Tiwari, Convergence of dirichlet measures and the interpretation of their parameter, in: Statistical decision theory and related topics III, Elsevier, 1982, pp. 305–315.

[29] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural networks 4 (2) (1991) 251–257.

[30] D. P. Kingma, S. Mohamed, D. J. Rezende, M. Welling, Semi-supervised learning with deep generative models, in: Advances in neural information processing systems, 2014, pp. 3581–3589.

[31] E. Nalisnick, L. Hertel, P. Smyth, Approximate inference for deep latent gaussian mixtures, in: NIPS Workshop on Bayesian Deep Learning, Vol. 2, 2016.

[32] H. Ishwaran, L. F. James, Gibbs sampling methods for stick-breaking priors, Journal of the American Statistical Association 96 (453) (2001) 161–173.

[33] S. Lee, J. Stokes, E. Eatonr, Learning shared knowledge for deep lifelong learning using deconvolutional networks, in: 6th Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI), 2019, pp. 2837–2844.

[34] M. C. Hughes, E. Sudderth, Memoized online variational inference for dirichlet process mixture models, in: Advances in Neural Information Processing Systems, 2013, pp. 1133–1141.

[35] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al., Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[36] D. D. Lewis, Y. Yang, T. G. Rose, F. Li, Rcv1: A new benchmark collection for text categorization research, Journal of machine learning research 5 (Apr) (2004) 361–397.

[37] A. Coates, A. Ng, H. Lee, An analysis of single-layer networks in unsupervised feature learning, in: Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, pp. 215–223.

[38] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images.

[39] A. Rosenberg, J. Hirschberg, V-measure: A conditional entropy-based external cluster evaluation measure, in: Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL), 2007.

Appendix A.

Here we provide the derivation of $\mathcal{L}_{\text{ELBO-VAE}}$ described in Section 5.1. The notations are defined in Table A.6.

Recall that in Section 5.1, the ELBO is

$$\begin{aligned}
&\mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}|\boldsymbol{y},\boldsymbol{\phi})p(\boldsymbol{y}|\boldsymbol{v})p(\boldsymbol{v})p(\boldsymbol{\phi})}{q_\psi(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{y})q(\boldsymbol{v})q(\boldsymbol{\phi}))}\right] \\
&= \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] + \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log p(\boldsymbol{z}|\boldsymbol{y},\boldsymbol{\phi})\right] \\
&- \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log q_\psi(\boldsymbol{z}|\boldsymbol{x})\right] \\
&+ \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log p(\boldsymbol{y}|\boldsymbol{v})\right] + \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log p(\boldsymbol{v})\right] \\
&- \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log q(\boldsymbol{y})\right] - \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log q(\boldsymbol{v})\right] \\
&- \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log q(\boldsymbol{\phi})\right] + \mathbb{E}_{q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x})}\left[\log p(\boldsymbol{\phi})\right].
\end{aligned} \tag{A.1}$$

660 $\mathcal{L}_{\text{ELBO-VAE}}$ only involves the first three terms of equation (A.1) that make a contribution to optimize θ, ψ and $\boldsymbol{z}$ since we adopt the alternating optimization technique.

We assume our variational distribution takes the form of

$$\begin{aligned}
&q(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\phi},\boldsymbol{v}|\boldsymbol{x}) = q_\psi(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{\phi})q(\boldsymbol{v})q(\boldsymbol{y}) \\
&= \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x};\psi), \operatorname{diag}(\boldsymbol{\sigma}^2(\boldsymbol{x};\psi))) \prod_{t=1}^{T-1} q_{\eta_t}(v_t) \prod_{t=1}^{T} q_{\zeta_t}(\boldsymbol{\phi}_t) \prod_{n=1}^{N} q_{\rho_n}(\boldsymbol{y}_n) \\
&= \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x};\psi), \operatorname{diag}(\boldsymbol{\sigma}^2(\boldsymbol{x};\psi))) \prod_{t=1}^{T-1} \operatorname{Beta}(\eta_{t_1}, \eta_{t_2}) \\
&\prod_{t=1}^{T} \mathcal{N}(\boldsymbol{\mu}_t|\boldsymbol{m}_t, (\boldsymbol{\beta}_t\boldsymbol{\Lambda}_t)^{-1})\mathcal{W}(\boldsymbol{\Lambda}_t|W_t,\nu_t) \prod_{n=1}^{N} \operatorname{Mult}(T,\rho_n),
\end{aligned} \tag{A.2}$$

where we denote $f_\psi(\boldsymbol{x}) = (\boldsymbol{\mu}(\boldsymbol{x};\psi), \boldsymbol{\sigma}^2(\boldsymbol{x};\psi))$, which is a neural network, T is the number of mixture components in the DP of the variational distribution, $\boldsymbol{z}_n \sim$ 665 $\mathcal{N}(\boldsymbol{z}_n|\boldsymbol{\mu}_t, \boldsymbol{\Lambda}_t^{-1})$, $\boldsymbol{\phi}_t = (\boldsymbol{\mu}_t, \boldsymbol{\Lambda}_t)$, and $\operatorname{Mult}(T,\rho_n)$ is a Multinomial distribution. Under the assumptions in (A.2), we derive each of the first three terms in Equation A.1 to obtain $\mathcal{L}_{\text{ELBO-VAE}}$.

(1) $\mathbb{E}_{q_\psi(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{y})q(\boldsymbol{v})q(\boldsymbol{\phi})}\left[\log P_\theta(\boldsymbol{x}|\boldsymbol{z})\right]$:

We assume in the generative model that $\boldsymbol{x}|\boldsymbol{z} = \boldsymbol{z} \sim \mathcal{N}\left(\boldsymbol{\mu}(\boldsymbol{z};\theta), \operatorname{diag}(\boldsymbol{\sigma}^2(\boldsymbol{z};\theta))\right)$

Table A.6: Notations in the ELBO.

Notations in the ELBO
N: the total number of observations.
L: the number of Monte Carlo samples in Stochastic Gradient Variational Bayes (SGVB).
$\Sigma : \mathrm{diag}(\boldsymbol{\sigma}^2(\boldsymbol{x};\psi))$.
$\boldsymbol{x}_n$: the nth observation.
$\boldsymbol{y}_n$: cluster membership for the nth observation.
$p(\boldsymbol{y}_n = k) = \gamma_{ik}, N_k = \sum_{n=1}^{N} \gamma_{nk}$.
$\hat{\boldsymbol{z}}_n = \frac{1}{L}\sum_{l=1}^{L} \boldsymbol{z}_n^{(l)}$.
$\bar{\boldsymbol{z}}_k = \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{nk}\hat{\boldsymbol{z}}_n$.
$\boldsymbol{S}_k = \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{nk}(\hat{\boldsymbol{z}}_n - \bar{\boldsymbol{z}}_k)(\hat{\boldsymbol{z}}_n - \bar{\boldsymbol{z}}_k)^T$.
$\beta_k = \beta_0 + N_k$: the scalar precision in NW distribution.
$\boldsymbol{m}_k = \frac{1}{\beta_k}(\beta_0 \boldsymbol{m}_0 + N_k \bar{z}_k)$: the posterior mean of cluster k.
$W_k^{-1} = W_0^{-1} + N_k S_k + \frac{\beta_0 N_k}{\beta_0 + N_k}(\bar{\boldsymbol{z}}_k - \boldsymbol{m}_0)(\bar{\boldsymbol{z}}_k - \boldsymbol{m}_0)^T$.
$\nu_k = \nu_0 + N_k$: the kth posterior degrees of freedom of NW.
$\boldsymbol{\phi}$: variational parameters of the kth NW components.
η_t: variational parameters of a Beta distribution for the tth component in Equation A.2.
ζ_t:variational parameters of the NW distribution for $\boldsymbol{\phi}_t$.
ρ_n: the variational parameters of a categorical distribution for the cluster membership for each observation.

and $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is parameterized by a neural network $g_\theta : Z \to X$ and $g_\theta(\boldsymbol{z}) = \big(\boldsymbol{\mu}(\boldsymbol{z};\theta), \boldsymbol{\sigma}^2(\boldsymbol{z};\theta)\big)$. Using the reparameterization trick [23] and the Monte Carlo estimate of expectations, we have

$$\begin{aligned}
&\mathbb{E}_{q_\psi(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{y})q(\boldsymbol{v})q(\boldsymbol{\phi})}[\log P_\theta(\boldsymbol{x}|\boldsymbol{z})] \\
&\quad= \mathbb{E}_{q_\psi(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{y})q(\boldsymbol{v})q(\boldsymbol{\phi})}\bigg(-\frac{1}{2}\log(\boldsymbol{\sigma}^2(\boldsymbol{z};\theta)_j) + \frac{(\boldsymbol{x}_j - \boldsymbol{\mu}(\boldsymbol{z};\theta)_j)^2}{\boldsymbol{\sigma}^2(\boldsymbol{z};\theta)_j}\bigg) \\
&\quad= -\frac{1}{2}\frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{N}\sum_{j=1}^{D}\Bigg(\log(\boldsymbol{\sigma}^2(\boldsymbol{z};\theta)_j^{(l)} + \frac{\Big(\boldsymbol{x}_{ij} - \boldsymbol{\mu}(\boldsymbol{z};\theta)_j^{(l)}\Big)^2}{\boldsymbol{\sigma}^2(\boldsymbol{z};\theta)_j^{(l)}}\Bigg). \qquad (A.3)
\end{aligned}$$

(2) $\mathbb{E}_{q_\psi(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{y})q(\boldsymbol{v})q(\boldsymbol{\phi})}[\log p(\boldsymbol{z}|\boldsymbol{y},\boldsymbol{\phi})]$:

Recall that ψ, where $q_\psi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x};\psi), \boldsymbol{\sigma}^2(\boldsymbol{x};\psi))$ and $(\boldsymbol{\mu}(\boldsymbol{x};\psi), \boldsymbol{\sigma}^2(\boldsymbol{x};\psi)) = f(\boldsymbol{x};\psi)$, where f is a neural network. Following [23], we use the reparameterization and sampling trick to allow backpropagation, for $l = 1, 2, \ldots, L$, where L is the number of Monte Carlo samples, we have

$$\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \quad \text{and} \quad \boldsymbol{z}^{(l)} = \boldsymbol{\mu}(\boldsymbol{x};\psi) + \boldsymbol{\epsilon}^{(l)}\boldsymbol{\sigma}(\boldsymbol{x};\psi).$$

Define $\hat{\boldsymbol{z}}_n = \frac{1}{L}\sum_{l=1}^{L} \boldsymbol{z}_n^{(l)}$. We have

$$\begin{aligned}
&\mathbb{E}_{q_\psi(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{y})q(\boldsymbol{v})q(\boldsymbol{\phi})}[\log p(\boldsymbol{z}|\boldsymbol{y},\boldsymbol{\phi})] \\
&= \frac{1}{2}\sum_{k=1}^{T} N_k \left\{ \log \tilde{\Lambda}_k - D\beta_k^{-1} - \nu_k \mathrm{Tr}(S_k W_k) \right\} \\
&- \frac{1}{2}\sum_{k=1}^{T} N_k \left\{ \nu_k (\bar{z}_k - \boldsymbol{m}_k)^T W_k (\bar{z}_k - \boldsymbol{m}_k) - D\log(2\pi) \right\},
\end{aligned}$$

where

$$\log \tilde{\Lambda}_k = \mathbb{E}[\log \Lambda_k] = \sum_{j=1}^{D} \psi\left(\frac{\nu_k + 1 - i}{2}\right) + D\log 2 + \log |W_k|. \tag{A.4}$$

(3) $\mathbb{E}q_\psi(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{y})q(\boldsymbol{v})q(\boldsymbol{\phi})(\log q_\psi(\boldsymbol{z}|\boldsymbol{x}))$:

Since $q_\psi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x};\psi), \mathrm{diag}(\boldsymbol{\sigma}^2(\boldsymbol{x};\psi)))$, $\mathbb{E}q_\psi(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{y})q(\boldsymbol{v})q(\boldsymbol{\phi})(\log q_\psi(\boldsymbol{z}|\boldsymbol{x}))$ is equal to the negative entropy of a multivariate Gaussian distribution:

$$\mathbb{E}q_\psi(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{y})q(\boldsymbol{v})q(\boldsymbol{\phi})(\log q_\psi(\boldsymbol{z}|\boldsymbol{x})) = \frac{1}{2}\log(\mathrm{Det}(2\pi e\Sigma)), \tag{A.5}$$

where $\Sigma = \mathrm{diag}(\boldsymbol{\sigma}^2(\boldsymbol{x};\psi))$.

When we update the neural network parameters θ and ψ and the latent representation $\boldsymbol{z}$, the DPMM parameters will be fixed. Thus, the terms that do not involve $\boldsymbol{z}$, θ, ψ will not contribute to the $L_{\text{ELBO-VAE}}$. Hence,

$$\begin{aligned}\mathcal{L}_{\text{ELBO-VAE}}(\boldsymbol{x}) &= -\frac{1}{2}\sum_{k=1}^{T} N_k\nu_k\left\{\text{Tr}(S_kW_k)\right\}\\ &-\frac{1}{2}\sum_{k=1}^{T} N_k\nu_k\left\{(\bar{z}_k-\boldsymbol{m}_k)^TW_k(\bar{z}_k-\boldsymbol{m}_k)\right\}\\ &-\frac{1}{2}\frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{N}\sum_{j=1}^{D}\left(\log(\boldsymbol{\sigma}(\boldsymbol{z},\theta)^2)_j^{(l)}+\frac{\left(\boldsymbol{x}_{ij}-\boldsymbol{\mu}(\boldsymbol{z};\theta)_j^{(l)}\right)^2}{(\boldsymbol{\sigma}(\boldsymbol{z};\theta)^2)_j^{(l)}}\right)\\ &+\frac{1}{2}\log(\text{Det}(2\pi e\Sigma)). \qquad \text{(A.6)}\end{aligned}$$

670 Details of the standard variational inference for parameters in the DP Gaussian mixture can be found in [10]. We only list the updating equations for the variational parameters below.

- $q(\boldsymbol{y}_n = i) = \gamma_{n,i}$.
- $q(\boldsymbol{y}_n > i) = \sum_{j=i+1}^{T}\gamma_{n,j}$.
- 675 $\mathbb{E}_q[\log V_i] = \Psi(\gamma_{i,1}) - \Psi(\gamma_{i,1}+\gamma_{i,2})$.
- $\mathbb{E}_q[\log(1-V_i)] = \Psi(\gamma_{i,2}) - \Psi(\gamma_{i,1}+\gamma_{i,2})$.
- $S_t = \mathbb{E}[\log V_i]+\sum_{i=1}^{t-1}\mathbb{E}_q[\log(1-V_i)]+\frac{1}{2}\log\tilde{\Lambda}_k-\frac{D}{2\beta_k}-\frac{\nu_k}{2}(\hat{z}_n-\boldsymbol{m}_k)^TW_k(\hat{z}_n-\boldsymbol{m}_k)$.
- $\gamma_{n,t}\propto\exp(S_t)$, $\gamma_{n,t}=\frac{\exp(S_t)}{\sum_{t=1}^{T}\exp(S_t)}$.
- Under the Normal-Wishart variational distribution assumption,

$$\begin{aligned}&\mathbb{E}q_\psi(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{y})q(\boldsymbol{v})q(\boldsymbol{\phi})(\log p(\boldsymbol{\phi}))\\ &=\frac{1}{2}\sum_{k=1}^{T}\left\{D\log(\beta_0/2\pi)+\log\tilde{\Lambda}_k\right\}\\ &-\frac{1}{2}\sum_{k=1}^{T}\left\{\frac{D\beta_0}{\beta_k}+\beta_0\nu_k(\boldsymbol{m}_k-\boldsymbol{m}_0)^TW_k(\boldsymbol{m}_k-\boldsymbol{m}_0)\right\}\\ &+T\log\text{B}(W_0,\nu_0)\frac{\nu_0-D-1}{2}\sum_{k=1}^{T}\log\tilde{\Lambda}_k-\frac{1}{2}\sum_{i=1}^{T}\nu_k\text{Tr}(W_0^{-1}W_k),\end{aligned}$$

where

$$\mathrm{B}(W,\nu) = |W|^{-\nu/2}\left(2^{\nu D/2}\pi^{D(D-1)/4}\prod_{i=1}^{D}\Gamma\left(\frac{\nu+1-i}{2}\right)\right)^{-1}.$$

- Similarly, we have

$$\begin{aligned}&\mathbb{E}q_{\psi}(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{y})q(\boldsymbol{v})q(\boldsymbol{\phi})[\log q(\boldsymbol{\phi})]\\&=\sum_{k=1}^{T}\left(\frac{1}{2}\log\tilde{\Lambda}_k+\frac{D}{2}\log\left(\frac{\beta_k}{2\pi}\right)-\frac{D}{2}-H[q(\Lambda_k)]\right),\end{aligned}$$

$$H[\Lambda] = -\log B(W,\nu) - \frac{\nu-D-1}{2}\mathbb{E}[\log|\Lambda|] + \frac{\nu D}{2}.$$