Cluster-wise Hierarchical Generative Model for Deep Amortized Clustering

Huafeng Liu, Jiaqi Wang, Liping Jing*
School of Computer and Information Technology
Beijing Key Lab of Traffic Data Analysis and Mining
Beijing Jiaotong University, Beijing, China
{huafeng, jiaqi.wang, lpjing}@bjtu.edu.cn

Abstract

*In this paper, we propose **C**luster-wise **Hi**erarchical **G**enerative Model for deep **a**mortized **c**lustering (CHiGac). It provides an efficient neural clustering architecture by grouping data points in a cluster-wise view rather than point-wise view. CHiGac simultaneously learns what makes a cluster, how to group data points into clusters, and how to adaptively control the number of clusters. The dedicated cluster generative process is able to sufficiently exploit pairwise or higher-order interactions between data points in both inter- and intra-cluster, which is useful to sufficiently mine the hidden structure among data. To efficiently minimize the generalized lower bound of CHiGac, we design an Ergodic Amortized Inference (EAI) strategy by considering the average behavior over sequence on an inner variational parameter trajectory, which is theoretically proven to reduce the amortization gap. A series of experiments have been conducted on both synthetic and real-world data. The experimental results demonstrated that CHiGac can efficiently and accurately cluster datasets in terms of both internal and external evaluation metrics (DBI and ACC).*

1. Introduction

Clustering is a fundamental task in unsupervised machine learning to group similar data points into multiple clusters. Aside from its usefulness in many downstream tasks, clustering is an important tool for visualizing and understanding the underlying structures of datasets, as well as a model for categorization in cognitive science. A plethora of clustering methods have been developed and successfully employed in various fields, including computer vision [13, 6], natural language processing [18, 26], social network analysis [12], and medical informatics [35]. Among various clustering algorithms [47, 48], probabilistic clustering model has been widely concerned because of its flexibility and interpretability.

Probabilistic generative clustering models (or equivalently, mixture models) [8] are a staple of statistical modeling in which a discrete latent variable is introduced for each observation, indicating its cluster identity. These generative clustering models can be roughly divided into two categories: finite mixture model [8] and infinite mixture model [44]. In recent years, finite mixture models have been increasingly applied in unsupervised learning problems with the aid of deep neural networks. The most relevant research is that of deep generative clustering models [21, 31, 49], where neural networks are trained to predict the states of latent variables given observations in a deep generative model or probabilistic program [23, 42]. Instead of using an arbitrary prior for the latent variable, these methods adopted finite mixture prior, such as Gaussian Mixture Model (GMM) [21, 49]. A finite mixture model with a fixed number of clusters may fit the given data set well, however, it may be sub-optimal to use the same number of clusters if more data comes under a slightly changed distribution. It would be ideal if the clustering models can figure out the unknown number of clusters automatically.

Alternatively, infinite mixture model is the application of nonparametric Bayesian techniques to mixture modeling, which allows for the automatic determination of an appropriate number of mixture components. The prior distribution can be specified in terms of a data point sequential process called Dirichlet process (e.g., Chinese restaurant process (CRP)), where the number of clusters can arbitrarily grow to better accommodate data as needed. To approximate the corresponding infinite mixture posterior, recently, deep amortized clustering [37, 38, 29, 39, 50] methods are proposed in a black-box fashion [45]. Specifically, they make use of neural networks for amortized inferring the cluster assignments and parameters, which is flexible to define the clusters. To adaptively determine the number of clusters, however, they have to construct the non-parametric prior, which largely depends on the sequential data point modeling. For large-scale dataset, this process will be time-

*Corresponding author

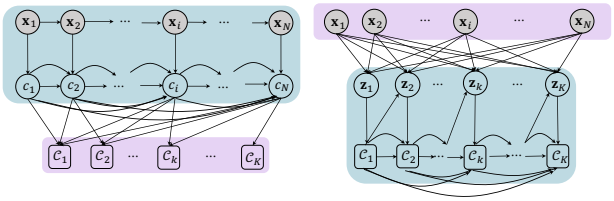
consuming and difficult to converge.

In this paper, we build on these prior works and propose a Cluster-wise Hierarchical Generative model for deep amortized clustering (CHiGac), which aims to learn non-parametric deep Bayesian posterior in a cluster-wise view. In other words, CHiGac targets on generating clusters rather than generating data points (all existing generative clustering methods adopted the later). This cluster generation process has two good facets. One is that the whole generation process is efficient, because it depends on the number of clusters (K) rather than the number of data points (N), where $K \ll N$. The other is that inter-cluster and intra-cluster structure can be sufficiently exploited during the learning process, because each cluster is generated according to the previous clusters and the current left data points. To efficiently approximate the non-parametric Bayesian posterior for CHiGac, we propose an Ergodic Amortized Inference (EAI) algorithm by considering the average behavior over sequence on an inner variational parameter trajectory. EAI is theoretically proven to reduce the *amortization gap* and provide flexible parameterization for the neural amortized inference model. To illustrate the superior of the proposed method, we perform experiments on both synthetic data and real-world data in terms of clustering performance and inference optimization performance.

2. Related Work

Deep clustering There is a growing interest in developing clustering methods using deep networks for complex data [21, 38, 31, 39, 49, 29, 46, 6]. The main focus of these methods is to learn a representation of input data amenable to clustering via deep neural networks. Learning representations and assigning data points to the clusters are usually trained alternatively. However, like the traditional clustering algorithms, these methods aim to cluster particular datasets. Since such methods typically learn a data representation using deep neural networks, the representation is prone to overfitting when applied to small datasets. Among the aforementioned work, variational autoencoder (VAE)-based clustering [21, 49] and amortized clustering [38, 39] are most relevant to our work. These two types of clustering methods utilize the Variational Bayes framework, where a probabilistic generative process is constructed to model the data points, and connect it to clustering process. Yet, our work directly models the cluster generative process, which is much more efficient than the previous work and has ability to exploit the structure among the previous generated clusters during learning process.

Amortized inference The term amortized inference (AI) refers to utilizing inferences from past computations to support future computations [14]. For Variational inference (VI), amortized inference usually refers to inference over local variables. Instead of approximating separate variables



(a) Point-wise sequence (b) Cluster-wise sequence

Figure 1. The clustering process for (a) point-wise sequential modeling and (b) cluster-wise sequential modeling. Purple panel contains unordered information and Blue panel contains ordered information. The Arrows indicate the corresponding order.

for each data point, amortized VI assumes that these latent variables can be predicted by a parameterized function of the data. Thus, once this function is estimated, the latent variables can be acquired by passing new data points through the function. Deep neural networks used in this context are also called inference networks [23]. The amortized inference is definitely fast, but the variational parameters are approximated by a parametric function of the input data, which may be too strict and cause the *amortization gap*. In order to reduce amortization gap, researchers focus on blending amortized inference via a local Stochastic Variational Inference (SVI) procedure [9, 34, 41, 22, 25]. Among them, Krishnan et al. [25] decoupled the training of inference and generative models by introducing an inner sub-optimization for variational parameters. To enable end-to-end training of the inference and generative models, Kim et al. [22] proposed to back-propagate through the SVI steps via a finite-difference estimation of the necessary Hessian-vector products. Alternatively, Marino et al. [34] adopts a learning-to-learn framework where an inference model iteratively outputs variational parameters. Although above methods reduce amortization gap to some extent, they can not take advantage of the sufficient information among inner sub-optimization which is helpful to find the final optimal point. To achieve an efficient and generalized model inference, we propose a new optimization strategy (Ergodic Amortized Inference, EAI) by using the average behavior over sequence on an inner variational parameter trajectory.

3. The proposed Method

In this section, we present the cluster-wise hierarchical deep generative clustering model and a new amortized inference method for effective posterior approximation.

3.1. Notations and Problem Formulation

Let calligraphic letter (e.g., $\mathcal{A}$) indicate set, capital letter (e.g., A) for scalar, lower-case bold letter (e.g., $\mathbf{a}$) for vector, and capital bold letter (e.g., $\mathbf{A}$) for matrix. Suppose there are N data points, $\mathbf{X} = (\mathbf{x}_i, \cdots, \mathbf{x}_N)^\top \in \mathbb{R}^{N \times D}$

indicates feature information over the whole dataset. Let $\{c_1, \cdots, c_N\}$ indicate the cluster assignment of data, where $c_i \in \{1, \cdots, K\}$ denotes the cluster index to which the data point $\mathbf{x}_i$ is assigned, and K is the number of clusters. Let $\{\mathcal{C}_1, \cdots, \mathcal{C}_K\}$ indicate the final cluster partitions, $\mathcal{C}_k$ represents the set of data points belonging to the k-th cluster.

Probabilistic model for clustering is usually presented by sequential sampling procedure to generate clusters. One of the prototypical tools for nonparametric clustering modeling is the Dirichlet Process. It allows for a discrete distribution of observations drawn from an arbitrary base measure over the domain in such a way as that the marginals match draws from, while simultaneously obtaining a countable set of distinct data points. A useful view of the Dirichlet Process Mixture Model [11] is the Chinese Restaurant metaphor [15], which sequentially computes the conditional probability of assigning the current data point to one of already constructed clusters or a new one. The cluster assignment for each data point can be sampled by

$$\begin{aligned} c_i &\sim p(c_i|c_{1:i-1}, \mathbf{x}_{1:i-1}) \\ &\propto p(c_i|c_{1:i-1}, \alpha) p(\mathbf{x}_{1:i-1}|c_{1:i-1}) \quad i = 1, \cdots, n \end{aligned} \tag{1}$$

Here $c_i \in \{1, \cdots, K\}$ encodes the cluster assignment of the data point $\mathbf{x}_i$ and $c_{1:i-1} = \{c_1, \cdots, c_{i-1}\}$ indicates the existing clusters formed by already assigned data points $\{\mathbf{x}_1, \cdots, \mathbf{x}_{i-1}\}$. α is hyperparameter controlling the ability to create a new cluster.

Given N data points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$, it is natural to draw independent samples to form clusters using decomposition

$$p(c_{1:N}|\mathbf{X}) = p(c_1) \prod_{i=2}^{N} p(c_i|c_{1:i-1}, \mathbf{x}_{1:i-1}) \tag{2}$$

After all data points are assigned, the final cluster is formed as $\{\mathcal{C}_k\}_{k=1}^K$. Inference proceeds by traversing data points and re-sampling their cluster assignments in sequence. As we can see, above process sequentially computes the conditional probability of assigning the current data point to one of already constructed clusters or a new one, and does not posit any particular prior on partitions, as shown in Figure 1(a). The sequential sampling procedure makes it impossible to parallelize handling data points with modern GPUs, which limits its scalability on large datasets (the computational cost for full i.i.d. sampling $\{c_1, \cdots, c_N\}$ is $O(NK)$) [19]. Furthermore, the clustering result is sensitive to the sequential processing order. To obtain stable result, it needs a sufficient number of random samples, which is time-consuming.

3.2. Hierarchical Generative Clustering: CHiGac

For the seek of effective cluster generative process and preventing the effect of sequential processing order, we focus on an alternative of clustering generative process from the view of cluster rather tan the view of data point. Let $\mathcal{C} = \{\mathcal{C}_1, \cdots, \mathcal{C}_K\}$ indicate K clusters, we define $\mathcal{C}_k$ as

$$\mathcal{C}_k = (\mathcal{C}_k^{(1)}, \cdots, \mathcal{C}_k^{(i)}, \cdots, \mathcal{C}_k^{(N_k)}) \quad k = 1, \cdots, K \tag{3}$$

where $\mathcal{C}_k^{(i)}$ is the i-th data point of the k-th cluster. N_k is the number of data points belonging to k-th cluster. Based on above definition, we are interested in sampling each cluster assignment by

$$\mathcal{C}_k \sim p_\theta(\mathcal{C}_k|\mathcal{C}_{1:k-1}, \mathbf{S}^k) \quad k = 1, \cdots, K \tag{4}$$

where $\mathbf{S}^k \in \mathbb{R}^{M_k \times D}$ indicates the available data for constructing the k-th cluster. M_k is the number of left data points after generating the previous $k-1$ clusters, defined by $M_k = N - \sum_{i=j}^{k-1} N_j$ (N_j is the number of data points belonging to j-th cluster). We generate cluster assignment in the form of *cluster-wise sequence* ($\mathcal{C}_1 \rightarrow \mathcal{C}_2 \rightarrow \cdots \rightarrow \mathcal{C}_K$) (as shown in Figure 1 (b)) rather than *data-wise sequence* ($c_1 \rightarrow c_2 \rightarrow \cdots \rightarrow c_N$) (as shown in Figure 1 (a)).

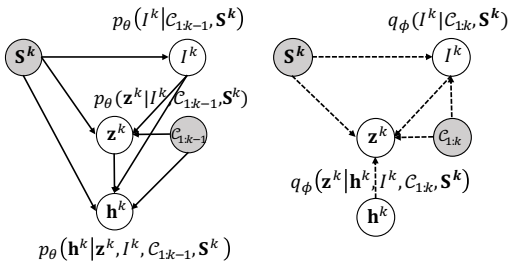
Given N data points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$, we are interested in sampling $\mathcal{C}_k$ via a decomposition

$$p_\theta(\mathcal{C}_{1:K}|\mathbf{X}) = p_\theta(\mathcal{C}_1|\mathbf{X}) \prod_{k=2}^{K} p_\theta(\mathcal{C}_k|\mathcal{C}_{1:k-1}, \mathbf{S}^k) \tag{5}$$

Following (5), it is easy to model cluster sequentially, and the arbitrary number of clusters can be adaptively determined until there is no remaining point in $\mathbf{S}^k$.

Hierarchical generative processes Consequently, the key task becomes modeling $p_\theta(\mathcal{C}_k|\mathcal{C}_{1:k-1}, \mathbf{S}^k)$ which reflects the generation of k-th cluster according to the existing clusters $\mathcal{C}_{1:k-1}$ and unassigned data points $\mathbf{S}^k$. To model it, we define a hierarchical generative process which can be broken into three parts: (1) selecting the pioneer data point I^k; (2) generating the cluster representation $\mathbf{z}^k$ and (3) making the cluster assignments $\mathbf{h}^k$. Specifically, the generative process can be formularized as follows

$$\begin{aligned} &I^k \sim p_\theta(I^k|\mathcal{C}_{1:k-1}, \mathbf{S}^k) \qquad \mathbf{z}^k \sim p_\theta(\mathbf{z}^k|I^k, \mathcal{C}_{1:k-1}, \mathbf{S}^k) \\ &h_i^k \sim p_\theta(h_i^k|\mathbf{z}^k, I^k, \mathcal{C}_{1:k-1}, \mathbf{S}^k) \quad i = 1, \cdots, M_k - 1. \end{aligned} \tag{6}$$



(a) Generative model (b) Inference model

Figure 2. The graphical architecture of CHiGac for (a) generative model and (b) inference model.

Here I^k indicates the index of the first data point (called as pioneer data point) for the k-th cluster selected from $\mathbf{S}^k$. Denote by $\mathbf{z}^k \in \mathbb{R}^{D_1}$ the cluster representation of the k-th cluster, we are interested in capturing the cluster representation to indicate a group of data points related to k-th cluster. Conditioned on $\{\mathbf{z}^k, I^k, \mathcal{C}_{1:k-1}, \mathbf{S}^k\}$, a binary vector $\mathbf{h}^k = (h_1^k, \cdots, h_{M_k-1}^k) \in \mathbb{R}^{M_k-1}$ can be sampled to indicate whether each left data point is affiliated with the k-th cluster. After obtaining $\mathbf{h}^k$, the cluster result $\mathcal{C}_k$ will be explicit. The whole generative process is shown in Figure 2(a). We will describe later the implementation details of $p_\theta(I^k|\mathcal{C}_{1:k-1}, \mathbf{S}^k)$, $p_\theta(\mathbf{z}^k|I^k, \mathcal{C}_{1:k-1}, \mathbf{S}^k)$ and $p_\theta(h_i^k|\mathbf{z}^k, I^k, \mathcal{C}_{1:k-1}, \mathbf{S}^k)$. Note that $p_\theta(I^k|\mathcal{C}_{1:k-1}, \mathbf{S}^k)$ requires careful design to prevent mode collapse, i.e., the degenerate case where almost all data points have the same potential to be the pioneer data point. Considering the equivalence of $\mathcal{C}_k$ and $\mathbf{h}^k$, we have $p_\theta(\mathcal{C}_k|\mathcal{C}_{1:k-1}, \mathbf{S}^k) \triangleq p_\theta(\mathbf{h}^k|\mathcal{C}_{1:k-1}, \mathbf{S}^k)$.

Joint modeling The generative processes are iterated until there is no available data point left. For convenience, we denote $\{I^k, \mathcal{C}_{1:k-1}, \mathbf{S}^k\}$ by $\mathbf{S}_{I^k}$. Consequently, the joint distribution over all latent variables is given by

$$\begin{aligned} &p_\theta(\mathbf{h}^k, \mathbf{z}^k, I^k|\mathcal{C}_{1:k-1}, \mathbf{S}^k) \\ &= \prod_{i=1}^{M_k-1} p_\theta(h_i^k|\mathbf{z}^k, \mathbf{S}_{I^k}) p_\theta(\mathbf{z}^k|\mathbf{S}_{I^k}) p_\theta(I^k|\mathcal{C}_{1:k-1}, \mathbf{S}^k) \end{aligned} \tag{7}$$

Crucially, the distribution of $\{h_i^k\}_{i=1}^{M_k-1}$ are conditionally independent. Therefore, once obtaining $p_\theta(\mathbf{z}^k|\mathbf{S}_{I^k})$, h_i^k can be sampled in parallel along all left data points, which is extremely efficient than sequential process in equation (2).

Amortized Variational Inference Following the conditional variational auto-encoder (CVAE) paradigm [42], θ and ϕ can be optimized by maximizing a low bound of $\mathbb{E}_{p(\mathbf{X}, \mathcal{C}_{1:K})} \log p_\theta(\mathcal{C}_{1:K}|\mathbf{X})$, i.e.,

$$\mathcal{L}_{\theta,\phi} = \mathbb{E}_{p(\mathbf{X}, \mathcal{C}_{1:K})} \sum_{k=1}^{K} \mathbb{E}_{q_\phi(\mathbf{z}^k, I^k|\mathcal{C}_{1:k}, \mathbf{S}^k)} \log \left[\frac{p_\theta(\mathbf{h}^k, \mathbf{z}^k, I^k|\mathcal{C}_{1:k-1}, \mathbf{S}^k)}{q_\phi(\mathbf{z}^k, I^k|\mathcal{C}_{1:k}, \mathbf{S}^k)} \right] \tag{8}$$

Here $q_\phi(\mathbf{z}^k, I^k|\mathcal{C}_{1:k}, \mathbf{S}^k)$ is amortized inference model which can be factorized by introducing a factorized variational posterior distribution, as shown in Figure 2(b)

$$q_\phi(\mathbf{z}^k, I^k|\mathcal{C}_{1:k}, \mathbf{S}^k) = q_\phi(\mathbf{z}^k|\mathbf{h}^k, \mathbf{S}_{I^k}) q_\phi(I^k|\mathcal{C}_{1:k}, \mathbf{S}^k) \tag{9}$$

This operation will encourage cluster inference process from I^k to $\mathbf{z}^k$, which will be presented in implementation. The two expectations, i.e., $\mathbb{E}_{p(\mathbf{X}, \mathcal{C}_{1:K})}[\cdot]$ and $\mathbb{E}_{q_\phi(\mathbf{z}^k, I^k|\mathcal{C}_{1:k}, \mathbf{S}^k)}[\cdot]$, are intractable, and can be estimated by the Gumbel-Softmax trick [20, 33] and the Gaussian reparameterization trick [23], respectively. Once the training procedure is finished, the cluster assignment can be estimated by

$$p(\mathbf{h}^k|\mathbf{S}^k) \approx \frac{1}{L} \sum_{l=1}^{L} p_\theta(\mathbf{h}^k|(\mathbf{z}^k)_l, \mathbf{S}_{I^k}) \quad (\mathbf{z}^k)_l \sim p_\theta(\mathbf{z}^k|\mathbf{S}_{I^k}) \tag{10}$$

Here L is the number of Monte Carlo samples.

3.3. Implementation

In this section, we describe the implementation of $p_\theta(I^k|\mathcal{C}_{1:k-1}, \mathbf{S}^k)$, $p_\theta(\mathbf{z}^k|I^k, \mathcal{C}_{1:k-1}, \mathbf{S}^k)$ and $p_\theta(h_i^k|\mathbf{z}^k, I^k, \mathcal{C}_{1:k-1}, \mathbf{S}^k)$, which are related to the decoder, and $q_\phi(I^k|\mathcal{C}_{1:k-1}, \mathbf{S}^k)$ and $q_\phi(\mathbf{z}^k|\mathbf{h}^k, \mathbf{S}_{I^k})$ related to the encoder. Furthermore, an efficient strategy is proposed to combat mode collapse. The parameters $\{\theta, \phi\}$ include: N data point embedding $\{\mathbf{e}_i\}_{i=1}^N \in \mathbb{R}^{N \times D}$, K pioneer data point indices $\{I^k\}_{k=1}^K \in \{1, \cdots, N\}^K$ related to each cluster, K cluster prototypes $\{\mathbf{m}^k\}_{k=1}^K \in \mathbb{R}^{K \times D_1}$ and cluster representations $\{\mathbf{z}^k\}_{k=1}^K \in \mathbb{R}^{K \times D_1}$, K cluster assignments $\{\mathbf{h}^k\}_{k=1}^K$, and the parameters of neural networks. We optimize $\{\theta, \phi\}$ to maximize the training objective (8).

Prototype-based pioneer point selection A straightforward approach would be to assume $p_\theta(I^k|\mathcal{C}_{1:k-1}, \mathbf{S}^k)$ follow a uniform distribution related to random sampling or categorical distribution with its own set of $M_k - 1$ parameters. As we all known, uniform distribution is too simple to contain any prior, and complex categorical distribution would result in over-parameterization and low sample efficiency. We instead propose a prototype-based implementation. To be specific, we introduce K cluster prototypes $\{\mathbf{m}^k\}_{k=1}^K$ and make use of the data point representations to draw the one-hot vector $\mathbf{o}^k$ following *categorical* distribution:

$$\begin{aligned} \mathbf{o}^k &\sim \textsc{Categorical}\left(\textsc{Softmax}([s_1^{(k)}, \cdots, s_j^{(k)}, \cdots, s_{M_k}^{(k)}])\right) \\ s_j^{(k)} &= \frac{\textsc{Cosine}(\mathbf{e}_j^{(k)}, \mathbf{m}^k)}{\tau} - \frac{1}{k-1} \sum_{l=1}^{k-1} \frac{\textsc{Cosine}(\mathbf{m}^l, \mathbf{m}^k)}{\tau} \end{aligned} \tag{11}$$

Then, the index of pioneer data point can be obtained by $I^k = \text{index}(\max(\mathbf{o}^k))$. $\mathbf{e}_j^{(k)}$ is the representation of the j-th data points in $\mathbf{S}^k$. Here the cosine similarity, $\text{Cosine}(\mathbf{a}, \mathbf{b}) = (\mathbf{a}^\top \mathbf{b})/(||\mathbf{a}||_2 ||\mathbf{b}||_2)$, instead of the inner product similarity adopted by most existing deep learning methods [32], is used to evaluate the correlation between data point and cluster and prevent mode collapse. In fact, with inner product, the majority of the available data points are highly like to be selected as the first data point for the k-th cluster, which will result in improper data assignment to cluster. Moreover, cosine similarity can be taken as Euclidean distance on the unit hypersphere, which is more suitable for inferring the cluster structure than inner product [36]. The hyper-parameter τ scales the similarity from

$[-1, 1]$ to $[-1/\tau, 1/\tau]$, which is set as $\tau = 0.1$ to obtain a more skewed distribution. The first term in $s_j^{(k)}$ aims to capture the similarity between data point and the current cluster, which can be taken as intra-cluster correlation. The second term tries to evaluate the difference between the current cluster and the existing clusters, which can be taken as inter-cluster separateness. Thus, the proposed learning process is expected to effectively mine the hidden cluster structure.

Prior and Decoder The prior $p_\theta(\mathbf{z}^k|I^k, \mathcal{C}_{1:k-1}, \mathbf{S}^k) \triangleq p_\theta(\mathbf{z}^k|\mathbf{S}_{I^k})$ targets to capture the relation between existing clusters, when given the first selected data point $\mathbf{x}_{I^k}$, existing clusters $\mathcal{C}_{1:K-1}$ and the unsigned data points $\mathbf{S}^k$. For convenience, $\mathbf{z}^k$ is assumed following a multivariate *Gaussian* distribution with diagonal covariance matrix and sampled via $p_\theta(\mathbf{z}^k|\mathbf{S}_{I^k}) = \mathcal{N}\left(\mathbf{z}^k|\boldsymbol{\mu}_\theta^{(k)}, [\mathrm{diag}(\boldsymbol{\sigma}_\theta^{(k)})]^2\right)$ where mean $\mu_\theta(\cdot)$ and variance $\sigma_\theta(\cdot)$ are parameterized by neural networks $f_\theta(\cdot)$:

$$\begin{aligned}\mathbf{w}_\theta^{(k)} &= \mathbf{e}_{I^k}^{(k)} + f\left(\sum_{i=1}^{k-1} \textsc{Mha}(\mathbf{e}_{I^k}^{(k)}, \mathbf{V}_i), \textsc{Mha}(\mathbf{e}_{I^k}^{(k)}, \mathbf{U}_k)\right)\\ (\boldsymbol{\mu}_\theta^{(k)}, \boldsymbol{\sigma}_\theta^{(k)}) &= f_\theta(\mathbf{w}_\theta^{(k)} + f(\mathbf{w}_\theta^{(k)}))\end{aligned} \tag{12}$$

here $\mathbf{e}_{I^k}^{(k)} \in \mathbb{R}^D$ is the representation of $\mathbf{x}_{I^k}$, $\mathbf{V}_i \in \mathbb{R}^{N_i \times D}$ indicates data belonging to the i-th cluster and $\mathbf{U}_k \in \mathbb{R}^{(M_k-1)\times D}$ represents the unassigned data points. $\textsc{Mha}(\mathbf{A}, \mathbf{B})$ indicates multi-head attention for capturing the relation between $\mathbf{A}$ and $\mathbf{B}$, which is able to exploit pairwise or higher-order interactions between data points in both inter- and intra-cluster [28, 29]. $f(\cdot)$ and f_θ are feed-forward layers with layer normalization [3].

The decoder predicts which data points out of M_k ones are mostly to be selected to form the k-th cluster, i.e., $p_\theta(h_i^k|\mathbf{z}^k, \mathbf{S}_{I^k}) = \mathrm{Sigmoid}(g_{\theta,i}(\mathbf{S}_{I^k}))$, where we introduce neural network parameterized by $g_{\theta,i}(\cdot)$ defined in terms of

$$\begin{aligned}s_i^{(k)} &= \textsc{Cosine}(\mathbf{e}_i^{(k)}, \mathbf{z}^k)/\tau\\ \mathbf{w}_i^{(k)} &= \mathbf{e}_i^{(k)} + f\left(\sum_{i=1}^{k-1} \textsc{Mha}(\mathbf{e}_i^{(k)}, \mathbf{V}_i), \textsc{Mha}(\mathbf{e}_i^{(k)}, \mathbf{U}_k)\right)\\ g_{\theta,i}(\mathbf{S}_{I_k}) &= g_{\theta,i}(s_i^{(k)}, \mathbf{w}_i^{(k)} + f(\mathbf{w}_i^{(k)}))\end{aligned} \tag{13}$$

here $\mathbf{e}_i^{(k)}$ is the representation of i-th data point in $\mathbf{S}^k$. The neural network $g_{\theta,i}(\cdot)$ captures the nonlinear structure among $M_k - 1$ data points. In our cluster-wise generative process, the cluster affiliation of these remaining data points for the k-th cluster (i.e., h^k) are conditionally independent, thus, they be sampled in parallel, i.e., $p_\theta(\mathbf{h}^k|\mathbf{z}^k, \mathbf{S}_{I^k}) = \prod_{i=1}^{M_k-1} p_\theta(h_i^k|\mathbf{z}^k, \mathbf{S}_{I^k})$.

Encoder The encoder contains two parts: $q_\phi(I^k|\mathcal{C}_{1:k}, \mathbf{S}^k)$ and $q_\phi(\mathbf{z}^k|\mathbf{h}^k, \mathbf{S}_{I^k})$. For I^k, the variational distribution has the same architecture as generative distribution. Difference is that generative model focuses on selecting I^k from $\mathbf{S}^k$ and inference model aims to select I^k from $\mathcal{C}_k$.

Similarly, $\mathbf{z}^k$ is sampled via a multivariable *Gaussian* distribution, i.e., $q_\phi(\mathbf{z}^k|\mathbf{h}^k, \mathbf{S}_{I^k}) = \mathcal{N}(\mathbf{z}^k|\boldsymbol{\mu}_\phi^{(k)}, [\mathrm{diag}(\boldsymbol{\sigma}_\phi^{(k)})]^2)$, here mean and standard deviation are parameterized by a neural network $f_\phi(\cdot)$:

$$\begin{aligned}(\mathbf{a}_\phi^{(k)}, \mathbf{b}_\phi^{(k)}) &= f_\phi\left(\frac{\sum_{i=1}^{N_k} h_i^k \cdot \mathbf{e}_i^{(k)}}{\sqrt{\sum_{i=1}^{N_k}(h_i^k)^2}}, \sum_{i=1}^{k} \textsc{Mha}(\mathbf{e}_{I^k}^{(k)}, \mathbf{V}_i)\right)\\ \boldsymbol{\mu}_\phi^{(k)} &= \frac{\mathbf{a}_\phi^{(k)}}{||\mathbf{a}_\phi^{(k)}||_2} \qquad \boldsymbol{\sigma}_\phi^{(k)} \leftarrow \sigma_0 \cdot \exp\left(-\frac{1}{2}\mathbf{b}_\phi^{(k)}\right)\end{aligned} \tag{14}$$

where $h_{I^k}^k = 1$. We normalize the mean to be consistent with the use of cosine similarity which projects the representations onto a unit hypersphere. Note that σ_0 should be set to a small value, e.g., around 0.1, since the learned representations are well normalized.

3.4. Ergodic Amortized Inference

A straightforward Inference is to optimize $\{\theta, \phi\}$ by maximizing the training objective $\mathcal{L}_{\theta,\phi}$ (see Eq.(8)). Actually, the lower bound that we want to maximize is given by an amortized model $q_\phi(\mathbf{z}^k, I^k|\mathcal{C}_{1:k}, \mathbf{S}^k)$, which learns an efficient mapping from samples to proposal distributions and reduces the cost of variational inference. The task focuses on optimizing variational parameters $\boldsymbol{\psi}^{(k)} = \{\boldsymbol{\mu}_\phi^{(k)}, \boldsymbol{\sigma}_\phi^{(k)}\}$ for the k-th cluster, where $\boldsymbol{\psi}_k$ is the output of amortized inference model. However, this procedure introduces an *amortization gap* [9] in which the less flexible parameterization of the amortized inference model replace the original instance-specific variational distribution, e.g., Stochastic Variational Inference [17].

The amortized inference has fast inference, but having the variational parameters be a parametric function of the input may be too strict of a restriction. In order to reduce amortization gap, researchers focus on blending amortized inference with a local SVI procedure [9, 34, 41, 22, 25]. SVI allows fine-tuning the initial proposal distribution $\boldsymbol{\psi}_0^{(k)}$ via an amortized inference model. However, the fine-tuned SVI procedure requires several iterative optimization for $\boldsymbol{\psi}^{(k)}$. Suppose the optimal variational parameter $\boldsymbol{\psi}_M^{(k)}$ is obtained with M iterations. The training of the inference and generative models can be decoupled with the initial one $\boldsymbol{\psi}_0^{(k)}$ and the optimal one $\boldsymbol{\psi}_M^{(k)}$.

Actually, to get the optimal $\boldsymbol{\psi}_M^{(k)}$, SVI procedure generates a sequence of variational parameters $\Psi^{(k)} = \{\boldsymbol{\psi}_0^{(k)}, \cdots, \boldsymbol{\psi}_M^{(k)}\}$, where each one can be taken as one reasonable approximations of the posterior. It is potentially wasteful to discard their corresponding contributions. Inspired by Importance weighted Autoencoder [5] and

ergodic theorem in stochastic programming models[24], we define a new lower bound for parameter θ in the generative model. For convenience, let $w_m^{(k)}$ denote $\frac{p_\theta(\mathbf{h}^k,\mathbf{z}_m^k,I^k|\mathcal{C}_{1:k-1},\mathbf{S}^k)}{q_\phi(\mathbf{z}_m^k,I^k|\mathcal{C}_{1:k},\mathbf{S}^k)}$, the objectives related to θ and ϕ via different strategies can be formalized as follow.

$$\mathcal{L}_\phi = \mathcal{L}_\phi^\triangle = \mathcal{L}_\phi^* = \mathbb{E}_{p(\mathbf{X},\mathcal{C}_{1:K})} \sum_{k=1}^{K} \mathbb{E}_{q_\phi(\mathbf{z}_0^k,I^k|\mathcal{C}_{1:k},\mathbf{S}^k)} \log w_0^{(k)}$$

$$\mathcal{L}_\theta = \mathbb{E}_{p(\mathbf{X},\mathcal{C}_{1:K})} \sum_{k=1}^{K} \mathbb{E}_{q_\phi(\mathbf{z}_0^k,I^k|\mathcal{C}_{1:k},\mathbf{S}^k)} \log w_0^{(k)}$$

$$\mathcal{L}_\theta^\triangle = \mathbb{E}_{p(\mathbf{X},\mathcal{C}_{1:K})} \sum_{k=1}^{K} \mathbb{E}_{q_\phi(\mathbf{z}_M^k,I^k|\mathcal{C}_{1:k},\mathbf{S}^k)} \log w_M^{(k)}$$

$$\mathcal{L}_\theta^* = \mathbb{E}_{p(\mathbf{X},\mathcal{C}_{1:K})} \sum_{k=1}^{K} \mathbb{E}_{q_\phi(\mathbf{z}_{(0:M)}^{(k)},I^k|\mathcal{C}_{1:k},\mathbf{S}^k)} \log \sum_{m=0}^{M} \pi_m w_m^{(k)}$$

Among them, $\mathcal{L}_\phi$ and $\mathcal{L}_\theta$ are original objectives, $\mathcal{L}_\phi^\triangle$ and $\mathcal{L}_\theta^\triangle$ are SVI-based objectives, and $\mathcal{L}_\phi^*$ and $\mathcal{L}_\theta^*$ are proposed amortized inference objectives. To keep preceding variational parameters along the entire local procedure, we use the average of $\{w_m^{(k)}\}_{m=0}^M$ to define the objective for optimizing θ. For simplicity, π_m is set to a uniform weight (e.g., $\pi_m = 1/(M+1)$). $\mathbf{z}_{(0:M)}^{(k)}$ indicates $\{\mathbf{z}_0^{(k)}, \cdots, \mathbf{z}_M^{(k)}\}$. This strategy considers average behavior over sequence on an inner variational parameter trajectory, thus we call this procedure as *Ergodic Amortized Inference* (*EAI*). As shown in line $4-10$ of Algorithm 1, the local procedure generates a series of local variational parameters $\Psi^{(k)} = \{\boldsymbol{\psi}_0^{(k)}, \cdots, \boldsymbol{\psi}_M^{(k)}\}$. Then, ϕ and θ can be updated to optimize the amortized ELBO objective (as shown in line $14-15$ of Algorithm 1). Among it, θ is trained to approximately optimize the weighted form with the aid of variational parameters trajectory. Note that M in the inner optimization process is predefined as a small value, e.g., $M = 10$.

We theoretically demonstrate that the proposed EAI objective $\mathcal{L}_{\theta,\phi}^*$ is a valid lower bound to the log-likelihood $\mathbb{E}_{p(\mathbf{X},\mathcal{C}_{1:K})} \log p_\theta(\mathcal{C}_{1:K}|\mathbf{X})$, and it is tighter than the original amortized objective $\mathcal{L}_{\theta,\phi}$ and SVI-based amortized inference objective $\mathcal{L}_{\theta,\phi}^\triangle$ [9, 34, 41, 22, 25]. See supplementary material for more details.

Algorithm 1 Learning with *Ergodic Amortized Inference* (EAI) for CHiGac

Input: $\mathbf{X} = (\mathbf{x}_i, \cdots, \mathbf{x}_N)^\top$ with N data points. Randomly initialize θ, ϕ. $\Psi^{(k)} = \varnothing$.

1: **while** not new cluster is generated **do**
2: Sample a batch of data points $\mathcal{D}$
3: **for** all $\mathbf{x}_i \in \mathcal{D}$ **do**
4: $\boldsymbol{\psi}_0^{(k)} = \{\boldsymbol{\mu}_\phi^{(k)}, \boldsymbol{\sigma}_\phi^{(k)}\} \leftarrow f_\phi(\mathbf{h}^k, \mathbf{S}_{I^k})$
5: $\Psi^{(k)} \leftarrow \{\boldsymbol{\psi}_0^{(k)}\} \cup \Psi^{(k)}$
6: Compute $\mathbf{z}_k$ via reparametrization trick
7: **for** $\tau = 1, \cdots, M$ **do**
8: $\boldsymbol{\psi}_{(\tau)}^{(k)} \leftarrow \boldsymbol{\psi}_{(\tau-1)}^{(k)} + \eta_\psi \nabla_\psi \mathcal{L}_\phi^*$
9: $\Psi^{(k)} \leftarrow \{\boldsymbol{\psi}_{(\tau)}^{(k)}\} \cup \Psi^{(k)}$
10: **end for**
11: Compute noisy gradients $\nabla_\phi \mathcal{L}_\phi^*$ and $\nabla_\theta \mathcal{L}_\theta^*$
12: **end for**
13: Average noisy gradients from batch
14: Update ϕ: $\phi^{t+1} \leftarrow \phi^t + \eta_\phi \nabla_\phi \mathcal{L}_\phi^*$
15: Update θ: $\theta^{t+1} \leftarrow \theta^t + \eta_\theta \nabla_\theta \mathcal{L}_\theta^*$
16: **end while**

4. Experiments

4.1. Experimental Setup

Datasets: A series of experiments have been conducted on four real-world datasets, *MNIST*, *STL-10*, *Reuters*, and *HHAR*. Specifically, *MNIST* [27] and *STL-10* [7] are image datasets. *Reuters* [30] is a textual dataset where each document is represented as a word vector with their TF-IDF values. *HHAR* [2] is a sensor signal dataset. We flatten each image in *MNIST* to a 784-dimensional vector and subtract features for *STL-10* by a pretrained ResNet-50 [16]. After preprocessing, the *MNIST* contains 10 classes of 784-dimensional training samples and each class has 7000 samples. *STL-10* contains 10 classes of 2048-dimensional training samples, where each class has 1300 samples. *Reuters* has 4 classes containing 10000 samples with 2000 dimensionality, and the *HHAR* contains 10200 samples belonging to 6 classes, and each sample is represented as a 561-dimensional vector.

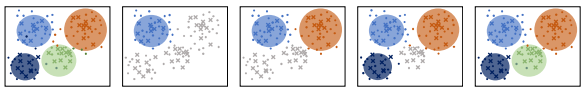
Baselines: Three kinds of methods are adopted as baselines, including traditional methods: MCMC [1] and VI [4]; the state-of-art VAE based deep clustering methods: VaDE [21], LTVAE [31] and DGG [49] (where DGG considers local structure by pre-constructing the nearest neighbor graph for input data points); nonparametric deep amortized clustering methods: DAC [29], NCP [39].

Evaluation Metrics: Two evaluation metrics are adopted. One is internal evaluation, *Davies-Bouldin Index (DBI)* [10]: $DBI = \frac{1}{K}\sum_{k=1}^{K} \max_{j\neq k}((a_k + a_j)/d(\mathbf{c}_k, \mathbf{c}_j))$ where $\mathbf{c}_k$ is the centroid of the k-th cluster, a_k is the average distance of all elements in cluster k to centroid $\mathbf{c}_k$. $d(\cdot)$ is cosine similarity. Smaller DBI value indicates better performance. The other is external evaluation, clustering accuracy (ACC): $ACC = \max_{m\in\mathcal{M}} \frac{1}{N}\mathbb{1}\{y_i = m(\hat{y}(\mathbf{x}_i))\}$ where N is the total number of data samples, y_i is the ground-truth label that corresponds to that $\mathbf{x}_i$ sample, $\hat{y}(\mathbf{x}_i)$ is the cluster assignment obtained by the model, and m ranges over the set $\mathcal{M}$ of all possible one-to-one mappings between cluster assignments and labels. Larger ACC value indicates better performance.

Parameters: The encoder-decoder structure is used for the data generation-based VAE methods (VaDE, LTVAE, DGG).Each encoder network uses five dense layers with

Table 1. Clustering performance on synthetic 2D MoG

	Metrics	MCMC	VI	DAC	NCP	CHiGa
s1	*DBI*	0.0313	0.0426	0.0367	0.0551	**0.0218**
	ACC	0.9566	0.9432	0.9623	0.9531	**0.9698**
	Time[s]	0.053	0.033	0.012	0.034	**0.006**
s2	*DBI*	0.0554	0.0827	0.0761	0.0759	**0.0512**
	ACC	0.9331	0.9244	0.9421	0.9233	**0.9426**
	Time[s]	0.095	0.036	0.015	0.036	**0.006**
Training time[s]		5.223	**2.116**	2.423	4.644	2.214



(a) Ground-truth (b) Epoch 5 (c) Epoch 13 (d) Epoch 16 (e) Epoch 21

Figure 3. The cluster generative process of CHiGac on 2D MoG.

size $D-500-500-2000-K$ and each decoder uses five dense layers with sizes $K-2000-500-500-D$. For DAC and NCP, the structure of network is the same as the original paper. For CHiGac, we use a 1-D ConvNet as feature extractor for datasets *MNIST* and *STL-10* and multilayer perceptron for *Reuters* and *HHAR*. The ConvNet uses a ResNet architecture [16] with 4 residual blocks. The size of cluster representation ($\mathbf{z}^k$ and $\mathbf{m}^k$ is set as $D_1 = 100$. The neural networks $f(\cdot)$, $f_\theta(\cdot)$ and $g_{\theta,i}(\cdot)$ are multilayer perceptrons.

4.2. Clustering Performance on Synthetic Data

The first experiment was conducted on synthetic data with arbitrary number of clusters. Thus, four existing methods handling variable number of clusters, MCMC, VI, DAC, and NCP, are used as baselines. The synthetic data is generated via a 2D mixture of Gaussian (MoG). Among them, the training set contains 200 samples belonging to 4 clusters. Two testing scenarios (*s1* and *s2*) are constructed to evaluate the effectiveness of the proposed method. The testing set in *s1* has the same configuration (200 samples and 4 clusters) as training set, while *s2* contains different numbers of samples and clusters (400 samples and 6 clusters) in order to verify whether the clustering method can generalize to the unseen clusters.

Table 1 summarizes the results.As expected, the proposed CHiGac consistently outperforms baselines on both *s1* and *s2*. Although *s2* is more challenging, CHiGac can capture cluster uncertainty and obtain the best results. For training stage, optimization-based (VI, DAC, CHiGa) methods are obviously faster than the sampling based methods (MCMC, NCP). In testing stage, the amortized clustering methods (DAC, NCP, CHiGac) are superior to the traditional methods (MCMC, VI). Further, we demonstrate the cluster-wise generation process by plotting the clustering results in the corresponding epochs. As shown in Figure 3, CHiGac has ability to automatically determine the true number of clusters without any prior.

4.3. Clustering Performance on Real-world Data

In Table 2, we depict the quantitative clustering results over four real-world benchmarks, compared to five popular deep clustering methods with state-of-the-art performance. The best and second results are marked in bold and underlined. The average clustering results are recorded over 10 training sessions with different random parameter initialization. It can be seen that our CHiGac outperforms all baselines in terms of both internal and external evaluation metrics. The main reason, we believe, is that CHiGac not only exploits amortized properties between data points in both inter- and intra-clusters, but also has flexible generative process, which simultaneously emphasizes the superiority of variational generative framework (VaDE, LTVAE, DGG) and amortized inference (DAC, NCP). The baseline, DGG, achieves a competitive performance since it exploits the additional local graph information and is trained in a self-supervised manner, which is time-consuming because local graph among all points has to be pre-computed. For the propsed CHiGac, it has ability to mine both local and global interactions with the aid of dedicated cluster generative process, thus it outperforms DGG even without extral information. DAC and CHiGac outperform other methods on computational time by a large margin, because DAC identifies each cluster after one forward pass and CHiGac generates clusters rather points. However, DAC is pretty worse than the proposed CHiGac on clustering performance.

Among amortized clustering methods, DAC, NCP and CHiGac are able to process arbitrary number of clusters. A digit subset from *MNIST* testing set is randomly sampled. To illustrate how the clustering models capture the shape ambiguity of some of the digits, we plot the ground-truth digit clusters and clustering results (obtained by DAC, NCP and CHiGac respectively) in Figure 4. Comparing with DAC and NCP, our method can get the most accurate results. For example, DAC assigns the digit 7 (with similar appearance to 9) to cluster 9, and NCP generates a new cluster for it. Fortunately, CHiGac correctly assigns it to cluster 7. Furthermore, we demonstrate the selected pioneer data points of each cluster for *MNIST* and *STL-10* datasets in Figure 5. It can be seen that these selected points have unique characteristics of their corresponding clusters, which will further leverage the cluster generation process.

4.4. Inference Optimization Analysis

In order to investigate the convergence property of the proposed ergodic amortized inference (EAI) method, we compare it with several popular amortized inference methods on constructing the CHiGac model, including amortized inference (AI) [23], semi-amortized inference (Semi-AI) [22], iterative amortized inference (IAI) [34], stochastic variational inference-amortized inference (SVI-AI) [25]

Table 2. Comparing clustering performance (DBI and ACC) on four real-world datasets.

	MNIST			*STL-10*			*Reuters*			*HHAR*		
Method	*DBI*	*ACC*	Time[s]	*DBI*	*ACC*	Time[s]	*DBI*	*ACC*	Time[s]	*DBI*	*ACC*	Time[s]
VaDE	0.4354	0.9446	43.62	1.2306	0.8545	36.33	1.1630	0.7984	26.44	1.6921	0.8446	17.33
LTVAE	1.5946	0.8630	56.33	0.4412	0.9006	52.34	1.1046	0.8096	35.67	1.5522	0.8500	23.46
DGG	0.2394	0.9758	63.46	0.4728	0.9059	57.34	1.0613	0.8230	39.56	0.3602	0.8904	24.56
DAC	0.2578	0.9596	24.33	0.8474	0.8822	23.56	1.1063	0.8033	**16.37**	1.7163	0.8452	**13.44**
NCP	0.2525	0.9633	73.34	0.8160	0.8959	77.35	1.2740	0.8069	45.81	1.6806	0.8441	36.79
CHiGac	**0.2327**	**0.9796**	**22.56**	**0.4364**	**0.9123**	**21.66**	**1.0582**	**0.8285**	17.44	**0.3487**	**0.9076**	15.33

The standard deviations (stds) of *DBI* are around 0.01 or 0.02 for all methods and stds of *ACC* are around 0.1.

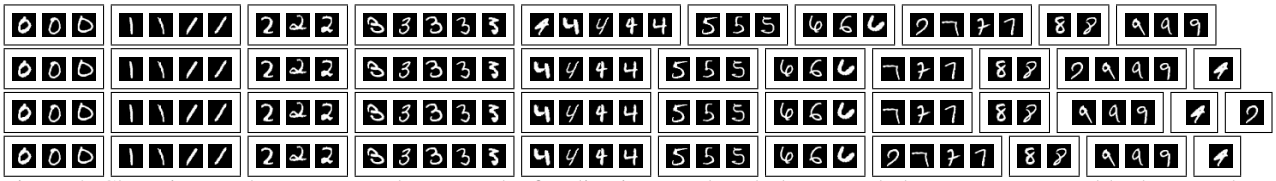


Figure 4. Clustering results on *MNIST* datasets. The first line is ground-truth dataset and clusters are separated by boxes. The results obtained by DAC, NCP and CHiGac are listed from the second line to forth line.

(a) *MNIST* (b) *STL-10*

Figure 5. Illustration of the selected pioneer points for each cluster of *MNIST* and *STL-10*.

Table 3. Clustering performance comparisons among amortized inference methods on CHiGac for *MNIST* and *STL-10* datasets.

	MNIST		*STL-10*	
Method	*DBI*	*ACC*	*DBI*	*ACC*
AI	0.2357	0.9769	0.4406	0.9102
Semi-AI	0.2389	0.9773	0.4412	0.9114
IAI	0.2347	0.9773	0.4388	0.9116
SVI-AI	0.2377	0.9767	0.4374	0.9109
EAI	**0.2327**	**0.9796**	**0.4364**	**0.9123**

and our ergodic amortized inference (EAI). Table 3 lists the clustering results on two image datasets. Obviously, improved amortized inference methods work better than the traditional amortized inference (AI). As expected, the proposed EAI obtains the best performance. This is because it considers the whole variational parameter trajectory which contains more optimization information.

Table 4 shows the optimal *Log-likelihood*, evidence lower-bound (*ELBO*), and *Reconstruction* error of CHiGac with different strategies (AI, Semi-AI, IAI, SVI-AI) on *MNIST* dataset. Obviously, EAI performs best because it iteratively refines the proposal distribution during amortized inference process. Meanwhile, Figure 6 demonstrates the *Log-likelihood* versus iterations, which further confirms that EAI method has ability to obtain a tighter bound for log-likelihood than other amortized inference methods.

Table 4. Comparing inference optimization performance for CHiGac model on *MNIST* dataset.

Method	*Log-likelihood*	*ELBO*	*Reconstruction*
AI	-2253.54	-2267.45	2266.76
Semi-AI	-2051.21	-2065.34	2063.45
IAI	-1998.44	-2005.47	2004.44
SVI-AI	-1985.66	-1996.33	1992.33
EAI	**-1933.27**	**-1945.92**	**1923.55**

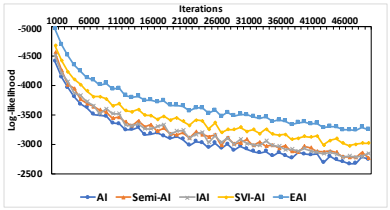


Figure 6. Comparing *Log-likelihood* of CHiGac versus iterations on *MNIST* dataset.

5. Conclusions

In this paper, we proposed Cluster-wise Hierarchical Generative Model for deep amortized clustering (CHiGac). It targets on learning to cluster from data without specifying the number of clusters. The proposed ergodic amortized inference method brings a tighter bound for CHiGac generative model and achieves more accurate clustering results.

Acknowledgment: This work was partly supported by the Beijing Natural Science Foundation under Grant Z180006; The National Natural Science Foundation of China under Grant 61822601, 61773050, and 61632004; The National Key Research and Development Program (2020AAA0106800); The Fundamental Research Funds for the Central Universities (2019JBZ110); Science and Technology Innovation Planning Foundation of Universities from Ministry of Education.

References

[1] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43, 2003. 6

[2] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, page 3, 2013. 6

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5

[4] David M Blei, Michael I Jordan, et al. Variational inference for dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143, 2006. 6

[5] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015. 5

[6] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the ECCV*, pages 132–149, 2018. 1, 2

[7] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the AISTATS*, pages 215–223, 2011. 6

[8] Review By: A. Clifford Cohen. Finite mixture distributions. *Technometrics*, 24(4):339, 1982. 1

[9] Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *Proceedings of ICML*, pages 1078–1086, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. 2, 5, 6

[10] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979. 6

[11] Thomas S Ferguson. A bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230, 1973. 3

[12] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010. 1

[13] Hichem Frigui and Raghu Krishnapuram. A robust competitive clustering algorithm with applications in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):450–465, 1999. 1

[14] Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the CogSci*, volume 36, 2014. 2

[15] Thomas L Griffiths, Michael I Jordan, Joshua B Tenenbaum, and David M Blei. Hierarchical topic models and the nested chinese restaurant process. In *Proceedings of the NeurIPS*, pages 17–24, 2004. 3

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the CVPR*, pages 770–778, 2016. 6, 7

[17] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013. 5

[18] Anna Huang. Similarity measures for text document clustering. In *Proceedings of the NZCSRSC*, volume 4, pages 9–56, 2008. 1

[19] Hemant Ishwaran and Lancelot F James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173, 2001. 3

[20] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 4

[21] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *Proceedings of the IJCAI*, IJCAI'17, page 1965–1972. AAAI Press, 2017. 1, 2, 6

[22] Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-amortized variational autoencoders. In Jennifer Dy and Andreas Krause, editors, *Proceedings of ICML*, pages 2678–2687. PMLR, 10–15 Jul 2018. 2, 5, 6, 7

[23] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1, 2, 4, 7

[24] Lisa A Korf and Roger JB Wets. An ergodic theorem for stochastic programming problems. In *Optimization*, pages 203–217. Springer, 2000. 6

[25] Rahul Krishnan, Dawen Liang, and Matthew Hoffman. On the challenges of learning with inference networks on sparse, high-dimensional data. In *Proceedings of AISTATS*, pages 143–151, 2018. 2, 5, 6, 7

[26] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the ACM SIGKDD*, pages 16–22, 1999. 1

[27] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 6

[28] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the ICML*, pages 3744–3753. PMLR, 2019. 5

[29] Juho Lee, Yoonho Lee, and Yee Whye Teh. Deep amortized clustering. *arXiv preprint arXiv:1909.13433*, 2019. 1, 2, 5, 6

[30] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: a new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, Dec. 2004. 6

[31] Xiaopeng Li, Zhourong Chen, Leonard K. M. Poon, and Nevin L. Zhang. Learning latent superstructures in variational autoencoders for deep multidimensional clustering. In *Proceedings of the ICLR*, 2019. 1, 2, 6

[32] Wenjie Luo, Alexander G. Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the CVPR*, June 2016. 4

[33] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete

random variables. *arXiv preprint arXiv:1611.00712*, 2016. 4

[34] Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *Proceedings of ICML*, pages 3403–3412, 10–15 Jul 2018. 2, 5, 6, 7

[35] Francesco Masulli and Andrea Schenone. A fuzzy clustering based segmentation system as support to diagnosis in medical imaging. *Artificial intelligence in medicine*, 16(2):129–147, 1999. 1

[36] Pascal Mettes, Elise van der Pol, and Cees Snoek. Hyperspherical prototype networks. In *Proceedings of the NeurIPS*, pages 1487–1497, 2019. 4

[37] Eric Nalisnick and Padhraic Smyth. Stick-breaking variational autoencoders. In *Proceedings of the SenSys*, pages 127–140, 2016. 1

[38] Ari Pakman and Liam Paninski. Amortized bayesian inference for clustering models. *arXiv preprint arXiv:1811.09747*, 2018. 1, 2

[39] Ari Pakman, Yueqi Wang, Catalin Mitelut, JinHyung Lee, and Liam Paninski. Neural clustering processes. In *Proceedings of the ICML*, 2020. 1, 2, 6

[40] Uri Shaham, Xiuyuan Cheng, Omer Dror, Ariel Jaffe, Boaz Nadler, Joseph Chang, and Yuval Kluger. A deep learning approach to unsupervised ensemble learning. In *Proceedings of the ICML*, pages 30–39, 2016.

[41] Rui Shu, Hung H Bui, Shengjia Zhao, Mykel J Kochenderfer, and Stefano Ermon. Amortized inference regularization. In *Proceedings of NeurIPS*, pages 4393–4402, 2018. 2, 5, 6

[42] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Proceedings of the NeurIPS*, pages 3483–3491, 2015. 1, 4

[43] Gabor J Szekely, Maria L Rizzo, et al. Hierarchical clustering via joint between-within distances: Extending ward's minimum variance method. *Journal of classification*, 22(2):151–184, 2005.

[44] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the american statistical association*, 101(476):1566–1581, 2006. 1

[45] Cinzia Viroli and Geoffrey J McLachlan. Deep gaussian mixture models. *Statistics and Computing*, 29(1):43–51, 2019. 1

[46] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Proceedings of the ICML*, pages 478–487, 2016. 2

[47] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015. 1

[48] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005. 1

[49] Linxiao Yang, Ngai-Man Cheung, Jiaying Li, and Jun Fang. Deep clustering by gaussian mixture variational autoencoders with graph embedding. In *Proceedings of the ICCV*, pages 6440–6449, 2019. 1, 2, 6

[50] Qingyu Zhao, Nicolas Honnorat, Ehsan Adeli, and Kilian M Pohl. Truncated gaussian-mixture variational autoencoder. *arXiv preprint arXiv:1902.03717*, 2019. 1