Amortized Mixing Coupling Processes for Clustering

Huafeng Liu[1,2], **Liping Jing**[1]
[1]Beijing Key Lab of Traffic Data Analysis and Mining, Beijing Jiaotong University, Beijing, China
[2]The Department of Mathematics, The University of HongKong, Hong Kong SAR, China
huafeng.liu@outlook.com, lpjing@bjtu.edu.cn

Abstract

Considering the ever-increasing scale of data, which may contain tens of thousands of data points or complicated latent structures, the issue of scalability and algorithmic efficiency becomes of vital importance for clustering. In this paper, we propose cluster-wise amortized mixing coupling processes (AMCP), which is able to achieve efficient amortized clustering in a well-defined non-parametric Bayesian posterior. Specifically, AMCP learns clusters sequentially with the aid of the proposed intra-cluster mixing (IntraCM) and inter-cluster coupling (InterCC) strategies, which investigate the relationship between data points and reference distribution in a linear optimal transport mixing view, and coupling the unassigned set and assigned set to generate new cluster. IntraCM and InterCC avoid pairwise calculation of distances between clusters and reduce the computational complexity from quadratic to linear in the current number of clusters. Furthermore, cluster-wise sequential process is able to improve the quick adaptation ability for the next cluster generation. In this case, AMCP simultaneously learns what makes a cluster, how to group data points into clusters, and how to adaptively control the number of clusters. To illustrate the superiority of the proposed method, we perform experiments on both synthetic data and real-world data in terms of clustering performance and computational efficiency.

1 Introduction

Clustering aims to group samples into a certain number of clusters, such that similar samples belong to the same cluster, while dissimilar ones to different clusters. Aside from its usefulness in many downstream tasks, clustering is an important but challenging topic which has seen applications in various settings including social-media recommendation [35], customer partitioning [7], discovering social networks [9] and partitioning protein-protein interaction networks [5]. Among various clustering algorithms [40], probabilistic generative clustering models [42] have been widely concerned because of its flexibility and interpretability.

Probabilistic generative clustering models (as known as, mixture models) are a staple of statistical modeling in which a discrete latent variable is introduced for each observation, indicating its mixture component identity. These generative clustering models can be roughly divided into two categories: finite mixture model [8] and infinite mixture model [34]. In recent years, finite mixture models have been increasingly applied in unsupervised learning problems with the aid of deep neural networks. Instead of using an arbitrary prior for the latent variable, these methods adopted finite mixture prior, such as Gaussian mixture model (GMM) [17, 41]. A finite mixture model with a fixed number of clusters may fit the given dataset well, while it may be sub-optimal to use the same number of clusters if more data comes under a slightly changed distribution [34]. 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

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

distribution can be specified in terms of a data point sequential process called Dirichlet process (e.g., Chinese restaurant process (CRP)[32]), where the number of clusters can arbitrarily grow to better accommodate data as needed [15]. However, the inference of nonparametric posterior largely rely on the sequential data point modeling (e.g., Markov Chain Monte Carlo (MCMC)), which is time-consuming, with convergence that is difficult to assess.

While above probabilistic clustering is generally approached as an unsupervised learning problem, the amortization power of non-linear transformation (e.g., neural networks) has recently spurred progress on supervised formulations, based on novel tractable objective functions for generic data and an arbitrary number of clusters [23, 24, 30]. In exchange for the additional cost of procuring labeled training data and training the model, amortized clustering offers several benefits on top of time efficiency at test time. Traditional unsupervised clustering, especially probabilistic clustering, defines specific likelihood and prior on the data geometry, even without any prior knowledge of the data, which are not needed with amortized clustering, since the model learns based on the statistics of the labeled training data [24]. Moreover, to achieve the best result (e.g., agreement with human labeling or with a well-defined posterior), unsupervised methods usually need parameter tuning and post-processing (e.g., cluster merging) [18], which are obviated with amortized inference by implicitly learning the definition of clusters underlying the training datasets. In a sense this shares a similar philosophy as Neural Processes [12, 11, 19], which learns from multiple datasets to learn a prior over functions in a meta learning manner.

In this paper, motivated by these prior work, we propose cluster-wise Amortized Mixing Coupling Processes (AMCP), which is able to provides an efficient clustering architecture by grouping data points in an optimal transport cluster-wise view. In each cluster-wise step, AMCP generates cluster sequentially and finds one cluster a time. In this case, two key strategies, Intra-Cluster Mixing (*IntraCM*) and Inter-Cluster Coupling (*InterCC*), are proposed to measure the relations between different data points and generate cluster. Specifically, *IntraCM* focus on investigating the relationship between data points and reference distribution in a linear optimal transport view [36], which avoid pairwise calculation of distances between clusters and reduce the computational complexity from quadratic to linear in the current number of clusters. Then, *InterCC* leverages new insights on defining dissimilarity between assigned data points and the unassigned data points based on the couplings obtained in *IntraCM*, which is able to select the dissimilar data points to form the next cluster and update existing clusters. The key contributions of our work are as follows: (1) *Efficiency*. The proposed AMCP not only inherent the efficient cluster-wise learning manner, but also reduce the quadratic cost of traditional similarity weight computation with the aid of optimal transport mixing. (2) *Novelty*. To the best of our knowledge, AMCP is the first to introduce optimal transport for amortized clustering and effectively correlate the relationship between optimal transport and mixture models. (3) *Adaptivity*. Non-parametric learning manners exist not only in model parameter learning but also in cluster generation, which allows unsupervised parameter learning and non-handcrafted intervention. (4) *Flexibility*. Different from existing amortized clustering methods without changing the already-generated clusters, AMCP is able to dynamically adjust the previous generated clusters to avoid the accumulated errors with the aid of mixing and coupling manners. (5) *Effectiveness*. Extensive experiments conducted on both synthetic and real-world datasets demonstrate that AMCP can cluster data effectively and efficiently.

2 Related Work

In this section, we briefly introduce some recent developments in several related topics, namely probabilistic clustering and amortized clustering.

Probabilistic Clustering Probabilistic clustering models (or equivalently, mixture models) are a staple of statistical modeling in which a discrete latent variable is introduced for each observation, indicating its mixture component identity. There is a growing interest in developing probabilistic clustering methods using parametric or nonparametric prior for complex data modeling [8, 34, 17, 41], and finite and infinite mixture models were developed. The main focus of finite model introduces finite mixture prior with a fixed number of cluster to clustering. In this case, Gaussian distribution or non Gaussian distributions [27, 26] are widely used to model data distribution [8]. Alternately, infinite mixture models [4] adopt non-parametric prior to control the number of clusters dynamically, such as Dirichlet process and its variants [32]. Recently, researchers focus on deep generative clustering by combining probabilistic clustering modeling and neural networks [17, 25, 30, 41, 24, 39, 6]. The

main focus of these methods is to learn a representation of input data amenable to clustering via deep neural networks.

Amortized Clustering Clustering is generally formulated as an unsupervised learning manner and while amortized clustering [30, 23, 24] investigates the amortized power steam from a predefined supervised formulation to achieve an arbitrary number of clusters in new task. Lee et al. [23, 24] adopt attention networks to output instead the cluster labels of each data point, for general mixture model, with an arbitrary number K of clusters and a cost of $O(K)$ forward passes. Later, Pakman et al. [30] presented Neural Clustering Process (NCP), which sequentially computes the conditional probability of assigning the current data point to one of already constructed clusters or a new one. The most similar work to our methods is NCP [30] and CHiGac [28]. Both of them explain clustering from the perspective of generative models, which similar in spirit to the popular Gibbs sampling algorithm for Dirichlet process mixture models, but without positing particular priors on partitions. Besides, NCP and CHiGac rely heavily on the selecting of anchor points and the processing ordering and often exhibits unstable properties. Different from previous work, AMCP connects the relationship of mixture model and optimal transport to describe the cluster's summary statistics well, and seamlessly combines mixing and coupling to achieve efficient amortized clustering.

3 Method

3.1 Notations and Problem Formulation

Given N data points $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ indicates feature representation for the i-th data point represented in d-dim latent space. Probabilistic models for clustering is usually presented by sequential sampling procedure to generate clusters. Usually, the generative process is $c_i \sim p(c_i|\alpha_1)$, $\boldsymbol{\mu}_k \sim p(\boldsymbol{\mu}_k|\alpha_2)$, $\mathbf{x}_i \sim p(\mathbf{x}_i|\boldsymbol{\mu}_{c_i})$, here $c_i \in \{1, 2, \cdots, K\}$ encodes the cluster assignment of the data point $\mathbf{x}_i$. α_1 and α_2 are hyperparameters. The integer random variable K indicates the number of distinct values among the sampled c_i, and $\boldsymbol{\mu}_k$ denotes a parameter controlling the distribution of the k-th cluster. Specifically, these models include Gaussian mixture model [17, 41], Dirichlet process mixture model [10] and etc.

Point-wise Clustering Posterior 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 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 $c_i \sim p(c_i|c_{1:i-1}, \mathcal{D})$. Here $c_{1:i-1} = \{c_1, c_2, \cdots, c_{i-1}\}$ indicates the existing clusters formed by already assigned data points $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{i-1}\}$. Based on existing clusters $c_{1:i-1}$ and observations $\mathcal{D}$, the cluster assignment for the i-th data point is sampled.

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

$$p(c_{1:N}|\mathcal{D}) = p(c_1|\mathcal{D})p(c_2|c_1, \mathcal{D}) \cdots p(c_N|c_{1:N-1}, \mathcal{D}) \tag{1}$$

After all data points are assigned, the final clusters are formed. Inference proceeds by traversing data points and re-sampling their cluster assignments in sequence.

Challenges 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. The sequential sampling procedure makes it process data points one by one, which limits its scalability on large datasets (the computational cost for full i.i.d. sampling $\{c_1, c_2, \cdots, c_N\}$ is high) [16]. Furthermore, the clustering result is sensitive to the sequential processing order. It needs a sufficient number of random samples to obtain stable result, which is time-consuming.

3.2 Amortized Mixing Coupling Processes

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 than the view of data point [30, 28]. Let $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_K\}$ indicate K clusters, we define $\mathcal{C}_k$ as $\mathcal{C}_k = (\mathcal{C}_k^{(1)}, \cdots, \mathcal{C}_k^{(i)}, \cdots, \mathcal{C}_k^{(N_k)})$, 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 the k-th cluster. Based on above definition, we are interested in sampling each cluster assignment by $\mathcal{C}_k \sim p(\mathcal{C}_k|\mathcal{C}_{1:k-1}, \mathcal{S}_k), k = 1, \cdots, K$, where $\mathcal{S}_k$ indicates the conditional information containing M_k available unassigned data points for constructing the k-th cluster. M_k is the number of left data points after generating the previous $k-1$ clusters, defined by $M_k = N - \sum_{j=1}^{k-1} N_j$ (N_j is the number of data points belonging to the j-th cluster).

We formulate cluster assignment process in the form of *cluster-wise* ($\mathcal{C}_1 \to \mathcal{C}_2 \to \cdots \to \mathcal{C}_K$) rather than *point-wise* ($c_1 \to c_2 \to \cdots \to c_N$), which enables an efficient clustering process. We are interested in sampling $\{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_K\}$ via a decomposition as follows:

$$p(\mathcal{C}_{1:K}|\mathcal{D}) = p(\mathcal{C}_1|\mathcal{S}_1)p(\mathcal{C}_2|\mathcal{C}_1, \mathcal{S}_2) \cdots p(\mathcal{C}_K|\mathcal{C}_{1:K-1}, \mathcal{S}_K), \tag{2}$$

which is easy to model cluster sequentially, and the arbitrary number of clusters can be adaptively determined until there is no remaining point in $\mathcal{S}_k$. The sampling process defined in Eq.(2) indicates a clustering posterior using K sets of indices.

To model above cluster-wise sequential posterior, we propose a cluster-wise amortized mixing coupling processes (AMCP), which is able to provides an efficient amortized clustering architecture by grouping data points in a cluster-wise view rather than point-wise view. As shown in Figure 1(a), AMCP generates clusters sequentially and two key strategies, *Intra-Cluster Mixing* (*IntraCM*) and *Inter-Cluster Coupling* (*InterCC*), are proposed. Specifically, as shown in Figure 1, *IntraCM* focus on investigating the relationship between data points and reference distribution $\boldsymbol{\phi}^{(k,T)} = [\boldsymbol{\phi}_1^{(k,T)}, \boldsymbol{\phi}_2^{(k,T)}, \cdots, \boldsymbol{\phi}_k^{(k,T)}]$ with $\boldsymbol{\phi}_j^{(k,T)} \in \mathbb{R}^d$ in a linear optimal transport view [36] with T inner update steps. Then, *InterCC* leverages new insights on defining dissimilarity between unassigned data points in $\mathcal{S}_k$ and assigned data points in $\{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_{k-1}\}$ based on the learned optimal transport plan $\mathbf{Q}^{(k,T)}$ obtained in *IntraCM* in a coupling manner, which is able to select the dissimilar data points to form the next cluster $\mathcal{C}_k$ and update existing clusters $\{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_{k-1}\}$.

3.2.1 Intra-Cluster Mixing

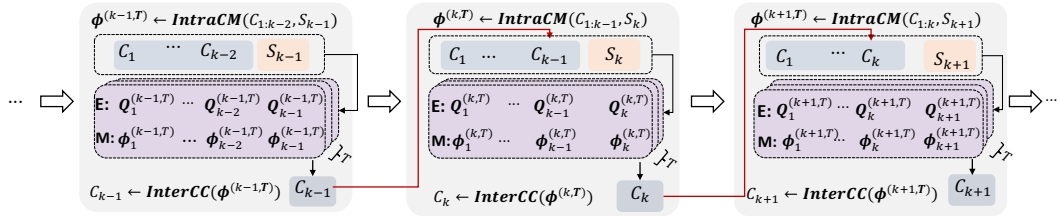
We present intra-cluster mixing (*IntraCM*) from generative mixing view, a novel and fast framework for embedding each cluster in a vector space and defining the relations of each data point from the reference distribution, which give a clear evidence or constraint that the learned distributions could describe the cluster's summary statistics well, as shown in Figure 1(b). Specifically, in the k-th cluster generation step, data points are partitioned into several subsets, i.e., $\mathcal{D}^{(k)}$ has structures with $\{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_{k-1}, \mathcal{S}_k\}$. We model each data point $\mathbf{x}_i \in \mathcal{D}^{(k)}$ as a Bayesian mixture model with k component parametrized by reference vectors $\boldsymbol{\phi}^{(k)} = [\boldsymbol{\phi}_1^{(k)}, \boldsymbol{\phi}_2^{(k)}, \cdots, \boldsymbol{\phi}_k^{(k)}] \in \mathbb{R}^{k \times d}$ and non-fixed mixing proportions over $k-1$ clusters $\{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_{k-1}\}$ and unassigned data points $\mathcal{S}_k$, as shown in Figure 1(a). A differentiable non-linear function $f_\theta(\cdot)$ (a neural network) is used to transform these representations $\boldsymbol{\phi}_j^{(k)}$ into parameters $f_\theta(\boldsymbol{\phi}_j^{(k)})$. By introducing latent variable $b_{i,j}^{(k)}$ indicating the data point $\mathbf{x}_i$ comes from the j-th component, we can get the full mixing likelihood related to the k-th cluster generation as follows:

$$p(\mathcal{D}^{(k)}|\boldsymbol{\phi}^{(k)}) = \prod_{\mathcal{D}^{(k)}} \sum_{\mathbf{b}_i^{(k)}} p\left(\mathbf{x}_i, \mathbf{b}_i^{(k)}|f_\theta(\boldsymbol{\phi}^{(k)})\right) = \prod_{\mathcal{D}^{(k)}} \sum_{j=1}^{k} \underbrace{p(b_{i,j}^{(k)})}_{\pi_j^{(k)}} p\left(\mathbf{x}_i|f_\theta(\boldsymbol{\phi}_j^{(k)}), b_{i,j}^{(k)}\right), \tag{3}$$

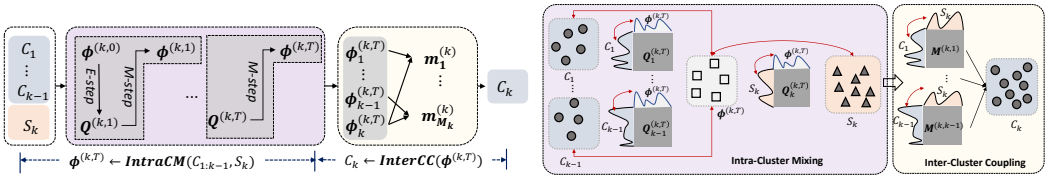
where $\mathbf{b}_i^{(k)} = [b_{i,1}^{(k)}, b_{i,2}^{(k)}, \cdots, b_{i,k}^{(k)}]$ encodes the data point assignments. $\boldsymbol{\pi}^{(k)} = [\pi_1^{(k)}, \pi_2^{(k)}, \cdots, \pi_k^{(k)}]$ are the mixing coefficients.

However, directly optimizing $\log \prod_{\mathcal{D}^{(k)}} p(\mathbf{x}_i|f_\theta(\boldsymbol{\phi}^{(k)}))$ with respect to $\boldsymbol{\phi}^{(k)}$ is difficult due to marginalization over $\mathbf{b}_i^{(k)}$, while for many distributions optimizing $\log \prod_{\mathcal{D}^{(k)}} p(\mathbf{x}_i, \mathbf{b}_i^{(k)}|f_\theta(\boldsymbol{\phi}_j^{(k)}))$ is much easier. In order to reflect the relationship of data in different sets, we introduce a variational distribution $q_{i,j}^{(k)} = q(b_{i,j}^{(k)}|\mathbf{x}_i)$ indicating a variational probability that the data point $\mathbf{x}_i$ belongs to the j-th component. In this case, we introduce a constrained variational lower bound to satisfy the optimal transport measure among different components and focus on performing the expected log likelihood by optimizing a lower bound:

$$\begin{aligned} \sum_{\mathcal{D}^{(k)}} \log \sum_{j=1}^{k} p\left(\mathbf{x}_i, b_{i,j}^{(k)}|f_\theta(\boldsymbol{\phi}^{(k)})\right) &\geq \sum_{\mathcal{D}^{(k)}} \sum_{j=1}^{k} q(b_{i,j}^{(k)}|\mathbf{x}_i) \log \frac{p\left(\mathbf{x}_i, b_{i,j}^{(k)}|f_\theta(\boldsymbol{\phi}^{(k)})\right)}{q(b_{i,j}^{(k)}|\mathbf{x}_i)} \\ &=: J(\boldsymbol{\phi}^{(k)}, \mathbf{Q}^{(k)}) \end{aligned} \tag{4}$$



(a) The architecture of the proposed cluster-wise AMCP including two key strategies: Intra-cluster mixing (IntraCM) and inter-cluster coupling (InterCC). IntraCM and InterCC performs to generate cluster one by one.



(b) Intra-cluster mixing and inter-cluster coupling. Among them, IntraCM cooperates optimal transport from inter-cluster coupling, which focus on embedding each mixture view and describe the cluster's summary statis- tics with the aid of multiple EM steps. Then, InterCC regarding the output of IntraCM as coupling to generate the next cluster.

(c) Optimal transport view of intra-cluster mixing and each data points from the reference distribution. cluster in a vector space and defining the relations of

Figure 1: The architecture of the proposed AMCP.

Optimal Transport View of *IntraCM* Here we give a detailed analysis to connect the proposed mixing strategy with optimal transport measure [36, 14, 20], which defines the transport plan between data points as weighted points clouds or discrete measures. We aim to formulate an optimization problem for $q(b_{i,j}^{(k)}|\mathbf{x}_i)$ given in Eq.(4). By replacing the variational distribution to a joint distribution, i.e., $q_{i,j}^{(k)} = q(b_{i,j}^{(k)}|\mathbf{x}_i)q(\mathbf{x}_i)$, the optimization problem of maximizing the lower bound in Eq.(4) with respect to the transport plan matrix $\mathbf{Q}^{(k)} = (q_{i,j}^{(k)}) \in \mathbb{R}^{N\times k}$ is equivalent to solving following entropic regularized Kantorovich relaxation of optimal transport [31]:

$$J(\boldsymbol{\phi}^{(k)}, \mathbf{Q}^{(k)}) = \min_{\mathbf{Q}^{(k)}} \sum_{\mathcal{D}^{(k)}} \sum_{j=1}^{k} -\log p\left(\mathbf{x}_i, b_{i,j}^{(k)}|f_\theta(\boldsymbol{\phi}^{(k)})\right) q_{i,j}^{(k)} - \epsilon H(\mathbf{Q}^{(k)}), \tag{5}$$

where $-\log p(\mathbf{x}_i, b_{i,j}^{(k)}|f_\theta(\boldsymbol{\phi}^{(k)}))$ indicates the pairwise transport cost between data point and reference vector. If *Gaussian* distribution is used as data likelihood, the transport cost $-\log p(\mathbf{x}_i, b_{i,j}^{(k)}|f_\theta(\boldsymbol{\phi}^{(k)}))$ is equivalent to a negative kernel $-\mathcal{K}(\mathbf{x}_i, f_\theta(\boldsymbol{\phi}_j^{(k)}))$. The ϵ impacted entropic regularized term $H(\mathbf{Q}^{(k)}) = -\sum_{i,j} q_{i,j}^{(k)} \log q_{i,j}^{(k)}$ controls the sparsity of $\mathbf{Q}^{(k)}$.

In original entropic regularized Kantorovich relaxation of optimal transport [31], there are some constraints on transport plan $\mathbf{Q}^{(k)}$, i.e., $\sum_{i=1}^{N} q_{i,j}^{(k)} = 1/k$ and $\sum_{j=1}^{k} q_{i,j}^{(k)} = 1/N$. If we set the factorized variational distribution to $q(\mathbf{x}_i) = 1/N$, the constraint $\sum_j q_{i,j}^{(k)} = 1/N$ is automatically met. Optimal transport view induces various advantages of our proposed mixing strategy. First, it is is computationally efficient, scalable to vary large dataset. Unlike the quadratic cost of traditional similarity weight computation (e.g., the attention score in self-attention), our optimal transport mixing uses a fixed number (k) of references serving as queries and the number of references is set depends on data without manual setting. Secondly, it allows unsupervised learning where the model parameters $\{\boldsymbol{\phi}_j^{(k)}\}_{j=1}^{k}$ can be learned without target (class) labels. Third, it is able to sufficiently exploit pairwise interactions between data points in intra-cluster level and benefit for the next inter-cluster coupling, which is useful to sufficiently mine the hidden structure among data.

Expectation Maximization for *IntraCM* Considering the latent variable $\mathbf{b}^{(k)}$ is introduced, naturally the expectation maximization (EM) method can be utilized [14]. Iterative optimization of this bound alternates between two steps:

- **E-step**: The E-step focuses on maximizing above bound with respect to the variational probability $q(b_{i,j}^{(k,t)}|\mathbf{x}_i)$. We compute a new estimate of the posterior probability distribution over the latent

variables from the previous iteration, which yields a soft-assignment of the data points to the components $q(b_{i,j}^{(k,t)}|\mathbf{x}_i) = \frac{\pi_j^{(k,t-1)} p\left(\mathbf{x}_i|b_{i,j}^{(k,t-1)}, f_\theta(\boldsymbol{\phi}^{(k,t-1)})\right)}{\sum_{j=1}^{k} \pi_j^{(k,t-1)} p\left(\mathbf{x}_i|b_{i,j}^{(k,t-1)}, f_\theta(\boldsymbol{\phi}^{(k,t-1)})\right)}$, where $\pi_j^{(k,t-1)}$ and $\boldsymbol{\phi}^{(k,t-1)}$ indicates model parameters from previous iteration. This soft-assignment is a regular E-step without the balanced assignment constraints. Furthermore, the variational lower bound can be formulated as the optimal transport setting (as shown in Eq.(5)), yielding an Sinkhorn-based optimal transport solution [33] as follows:

$$q(b_{i,j}^{(k,t)}|\mathbf{x}_i) = u_i \exp\left(p\left(\mathbf{x}_i|b_{i,j}^{(k,t-1)}, f_\theta(\boldsymbol{\phi}^{(k,t-1)})\right)/\epsilon\right) v_j, \tag{6}$$

where $u_i = \frac{1}{N\sum_{j=1}^{k} \exp(p(\mathbf{x}_i|b_{i,j}^{(k,t-1)}, f_\theta(\boldsymbol{\phi}^{(k,t-1)}))/\epsilon) v_j}$ and $v_j = \frac{1}{k\sum_{i=1}^{N} \exp(p(\mathbf{x}_i|b_{i,j}^{(k,t-1)}, f_\theta(\boldsymbol{\phi}^{(k,t-1)}))/\epsilon) u_i}$. Note that above solution usually converges quickly after a few iterations. Hence, the E-step coincides with the optimal transport based similarity weight computation, which leads to the various advantages of our proposed mixing strategy and forms the transport plan $\mathbf{Q}^{(k,t)}$ for all data points.

- **M-step**: Once the E-step is done (i.e., $\mathbf{Q}^{(k,t)}$ is found), we perform the M-step, that is, maximizing the lower bound in Eq.(4) with respect to $\boldsymbol{\phi}^{(k,t)}$ when $\mathbf{Q}^{(k,t)}$ is fixed. Due to the non-linearity of $f_\theta(\cdot)$ there exists no analytical solution to $\arg\max_{\boldsymbol{\phi}^{(k)}} J(\boldsymbol{\phi}^{(k)}, \mathbf{Q}^{(k,t)})$. With the benefit of differentiable $f_\theta(\cdot)$, we can solve it with the aid of gradient-based strategy. Assume that likelihood $p(\mathbf{x}_i|f_\theta(\boldsymbol{\phi}_j^{(k)}), b_{i,j}^{(k)})$ follows a *Gaussian* distribution with fixed covariance, we can obtain following update rule:

$$\boldsymbol{\phi}_j^{(k,t)} = \boldsymbol{\phi}_j^{(k,t-1)} + \eta \sum_{\mathcal{D}^{(k)}} q(b_{i,j}^{(k,t)}|\mathbf{x}_i)\left(f_\theta(\boldsymbol{\phi}_j^{(k,t-1)}) - \mathbf{x}_i\right)\nabla f_\theta(\boldsymbol{\phi}_j^{(k,t-1)}). \tag{7}$$

For simplicity, if we assume the parameters $\boldsymbol{\phi}^{(k)}$ in likelihood with a linear transformation $f_\theta(\boldsymbol{\phi}^{(k)}) = \mathbf{W}\boldsymbol{\phi}^{(k)}$, it can yield an optimal transport kernel embedding solution [29] when we set it as *Gaussian* distribution with fixed variance, e.g., $\pi_j^{(k,t)} = \frac{\sum_i q(b_{i,j}^{(k,t-1)}|\mathbf{x}_i)}{N}$ and $\boldsymbol{\phi}_j^{(k,t)} = \sum_i \frac{\exp(\boldsymbol{\phi}_j^{(k,t-1)\top}(\mathbf{W}\mathbf{x}_i)/\sqrt{d})}{\sum_i \exp(\boldsymbol{\phi}_j^{(k,t-1)\top}(\mathbf{W}\mathbf{x}_i)/\sqrt{d})}\mathbf{W}\mathbf{x}_i$, which is equivalent to induced self-attention introduced in Lee et al. [23] providing an efficient solution for pairwise weights measuring.

Usually, we can take multiple EM steps for further optimization (e.g., $\boldsymbol{\phi}^{(k,0)} \to \boldsymbol{\phi}^{(k,1)} \to \cdots \to \boldsymbol{\phi}^{(k,T)}$). Here T is the number of EM steps. *IntraCM* belongs to the class of generalized EM algorithms and is guaranteed to converge to a local optimum of the data log likelihood [38].

3.2.2 Inter-Cluster Coupling

After T EM steps, the variational probability $q_{i,j}^{(k,T)} = q(b_{i,j}^{(k,T)}|\mathbf{x}_i)$ can be regarded as the optimal transport plan, which carries the information on how to distribute the mass of data points $\mathbf{x}_i$ to reference vectors $\boldsymbol{\phi}_j^{(k,T)}$. By regarding $\boldsymbol{\phi}_j^{(k,T)}$ as couplings between unassigned set and assigned set, we propose inter-cluster coupling (*InterCC*) to generate the next cluster, as shown in Figure 1. Considering the current data points have structures with k different sets, $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_{k-1}, \mathcal{S}_k$, we can get $k-1$ transport plan matrices $\{\mathbf{U}_1^{(k)}, \mathbf{U}_2^{(k)}, \cdots, \mathbf{U}_{k-1}^{(k)}\}$ related to $k-1$ existing clusters and $\mathbf{V}^{(k)}$ related to the unassigned set $\mathcal{S}_k$,

$$\mathbf{U}_l^{(k)} = [q(b_{i,j}^{(k,T)}|\mathbf{x}_i \in \mathcal{C}_l)]^{|\mathcal{C}_l|\times k} \quad l = \{1, 2, \cdots k\} \quad \text{and} \quad \mathbf{V}^{(k)} = [q(b_{i,j}^{(k,T)}|\mathbf{x}_i \in \mathcal{S}_k)]^{M_k \times k}. \tag{8}$$

We can measure the relations between unsigned data points in $\mathcal{S}_k$ and existing assigned data points in $k-1$ existing clusters $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_{k-1}$ with the aid of couplings $\boldsymbol{\phi}^{(k)}$ as follows

$$\mathbf{M}^{(k)} = \Big[\underbrace{\oplus_{l=1}^{k-1}(\mathbf{U}_l^{(k)}\boldsymbol{\phi}^{(k,T)}\mathbf{v}_1^{(k)\top})}_{\mathbf{m}_1^{(k)}}, \cdots, \underbrace{\oplus_{l=1}^{k-1}(\mathbf{U}_l^{(k)}\boldsymbol{\phi}^{(k,T)}\mathbf{v}_{M_k}^{(k)\top})}_{\mathbf{m}_{M_k}^{(k)}}\Big] \in \mathbb{R}^{(N-M_k)\times M_k}, \tag{9}$$

where $\oplus$ is a column-wise concatenation operation and $\mathbf{v}_i^{(k)}$ is the i-th column of $\mathbf{V}^{(k)}$. It is reasonable to select dissimilar data point in unassigned data points set $\mathcal{S}_k$ to form a new cluster $\mathcal{C}_k$, i.e.,

$$\mathcal{C}_k \sim \textsc{Mult}\left(\textsc{Softmax}\left(\left[1 - \mathbf{I}^\top\mathbf{m}_1^{(k)}, 1 - \mathbf{I}^\top\mathbf{m}_2^{(k)}, \cdots, 1 - \mathbf{I}^\top\mathbf{m}_{M_k}^{(k)}\right]\right)\right). \tag{10}$$

Here *Multinomial* distribution is used to model the generation of $\mathcal{C}_k$. $\mathbf{I} \in \mathbb{R}^{N-M_k}$ is a vector of whose elements are all 1s. $\mathbf{m}_j^{(k)}$ is the i-th column of $\mathbf{M}^{(k)}$. Thus, the data points related to dissimilar

indices in each matrix are selected to generate the new cluster $\mathcal{C}_k$. Furthermore, previous existing clusters $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_{k-1}$ can be updated based on dissimilarity matrix. For a given cluster $\mathcal{C}_l$, it can be updated by $\mathcal{C}_l := \mathcal{C}_l \cup \mathcal{C}_l^{\text{new}}$, where $\mathcal{C}_l^{\text{new}} := \textsc{Mult}(\textsc{Softmax}[1 - \mathbf{I}^{(l)\top}\mathbf{M}^{(k,l)}])$. $\mathbf{I}^{(l)} \in \mathbb{R}^{|\mathcal{C}_l|}$ is a vector of all 1s and $\mathbf{M}^{(k,l)} = \mathbf{U}_l^{(k)}\boldsymbol{\phi}^{(k,T)}\mathbf{V}^{(k)\top}$.

3.2.3 Iterative Unrolling Inference

IntraCM and *InterCC* are both differentiable procedures for optimal transport measuring and next cluster constructing, whose outcome relies on the current existing clusters and the statistical model parameters $\boldsymbol{\phi}^{(k)}$. As shown in Eq.(2), the whole cluster-wise clustering procedure can be regarded as a recurrent *IntraCM* with *InterCC* until there are no available data point left. In this case, the information about statistical regularities in our model that is required for generating the sequential clusters is encoded in a series of parameters $\{\boldsymbol{\phi}^{(1,T)}, \boldsymbol{\phi}^{(2,T)}, \cdots, \boldsymbol{\phi}^{(K,T)}\}$. By unrolling the iterations of the presented mixing process, we obtain an end-to-end differentiable clustering procedure based on the statistical model parameterized by $\{\boldsymbol{\phi}^{(1,T)}, \boldsymbol{\phi}^{(2,T)}, \cdots, \boldsymbol{\phi}^{(K,T)}\}$. We can therefore use (stochastic) gradient descent and fit the statistical model to capture the regularities corresponding to clusters for a given dataset.

$$\mathbb{E}_{p(\mathcal{C}_{1:K},\mathcal{D})}\log p_\theta(\mathcal{C}_{1:K}|\mathcal{D}) \geq \mathbb{E}_{p(\mathcal{C}_{1:K},\mathcal{D})}\sum_{k=1}^{K}\left[\underbrace{\sum_{j=1}^{k} q(b_{i,j}^{(k)}|\mathbf{x}_i)\log\frac{p\left(\mathbf{x}_i, b_{i,j}^{(k)}|f_\theta(\boldsymbol{\phi}^{(k)})\right)}{q(b_{i,j}^{(k)}|\mathbf{x}_i)}}_{\textit{IntraCM}} + \underbrace{p(\mathcal{C}_k|f_\theta(\boldsymbol{\phi}^{(k)}))}_{\textit{InterCC}}\right] \tag{11}$$

The likelihood is derived from approximating the $p_\theta(\mathcal{C}_{1:K}|\mathcal{D})$ to true posterior via KL divergence. Note that samples from $p(\mathcal{C}_{1:K}, \mathcal{D})$ are obtained from the explicit generative model, such as *Gaussian* mixture model, *Dirichlet* process mixture model. With infinite generated samples, we can train a model to approximate $p_\theta(\mathcal{C}_{1:K}|\mathcal{D})$ accurately. The whole inference procedure is given in Algorithm 1. By repeating the cluster generative process until there are no data points left, IntraCM and InterCC are processed alternatively, and the clusters are iteratively refined by deriving the explicit supervisory signal from the already formed clusters. Note that the complexity of traditional methods measuring weights between data points acquires $O(KN^2)$ (e.g., self-attention) which may be too expensive for large sets, and our AMCP only acquires $O(KNk)$ in the EM-step for the k-th iteration, which is more efficient.

Algorithm 1 Iterative Unrolling Inference for for AMCP

Input: $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ with N data points. Randomly initialized parameters $\{\boldsymbol{\phi}^{(k,0)}\}_{k=1}^{K}$.

Establish the first cluster $\mathcal{C}_1$ by selecting cluster with the highest probability from k-means or GMM
while not new cluster $\mathcal{C}_k$ is generated **do**
  // `Intra-Cluster Mixing`
  Establish expected lower bound $J(\boldsymbol{\phi}^{(k,0)}, \mathbf{Q}^{(k,0)})$ via Eq.(4)
  for $t = 1, 2, \cdots, T$ **do** ▷ `Multiple EM steps`
    $\mathbf{Q}^{(k,t)} \leftarrow J(\boldsymbol{\phi}^{(k,t-1)}, \mathbf{Q}^{(k,t-1)})$, $\boldsymbol{\phi}^{(k,t)} \leftarrow \arg\max_{\boldsymbol{\phi}^{(k)}} J(\boldsymbol{\phi}^{(k,t-1)}, \mathbf{Q}^{(k,t)})$ ▷ `EM-step`
  end for
  // `Inter-Cluster Coupling`
  Establish k transport plan matrices $\{\mathbf{U}_l^{(k)}\}_{l=1}^{k-1}$ and $\mathbf{V}^{(k)}$, and similarity matrix $\mathbf{M}^{(k)}$
  Generate $\mathcal{C}_k$
  Update existing clusters $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_{k-1}$ ▷ `Optional`
  if $\mathcal{S}_k \neq \emptyset$ **then**
    Establish unassigned set $\mathcal{S}_{k+1}$
  end if
end while

Output: updated parameters $\boldsymbol{\phi}^{(K,T)}$, generated clusters $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_K$

4 Experiments

In this section, we evaluate the proposed AMCP in terms of synthetic and real-world datasets by comparing with the state-of-the-art methods.

4.1 Experiments on Synthetic Datasets

The first experiment was conducted on synthetic data with arbitrary number of clusters. Thus, five existing methods handling variable number of clusters, MCMC[1], VI[4], DAC[24], NCP[30] and

Table 1: Clustering performance on synthetic 2D GMM. The numbers below Scenario 1 and 2 are oracle *LL* values computed by the true parameters. The average on 5 runs are reported.

Scenario	Metric	MCMC[1]	VI [4]	DAC [24]	NCP[30]	ST-ACT [23]	AMCP
Scenario 1 (-1.2452)	*LL*	-1.5111$_{\pm 0.021}$	-1.4230$_{\pm 0.081}$	-1.2671$_{\pm 0.041}$	-1.3251$_{\pm 0.042}$	-1.2928$_{\pm 0.051}$	**-1.2517**$_{\pm 0.035}$
	ACC	0.7066$_{\pm 0.003}$	0.6832$_{\pm 0.004}$	0.7923$_{\pm 0.001}$	0.8131$_{\pm 0.002}$	0.7680$_{\pm 0.003}$	**0.8532**$_{\pm 0.002}$
	NMI	0.8731$_{\pm 0.004}$	0.8531$_{\pm 0.002}$	0.8983$_{\pm 0.003}$	0.8923$_{\pm 0.004}$	0.9123$_{\pm 0.003}$	**0.9334**$_{\pm 0.001}$
	Time[s]	0.0832$_{\pm 0.032}$	0.0332$_{\pm 0.045}$	0.0124$_{\pm 0.047}$	0.0343$_{\pm 0.026}$	0.0114$_{\pm 0.002}$	**0.0113**$_{\pm 0.003}$
Scenario 2 (-1.8242)	*LL*	-1.924$_{\pm 0.022}$	-2.9026$_{\pm 0.022}$	-1.847$_{\pm 0.034}$	-1.8551$_{\pm 0.042}$	-1.8918$_{\pm 0.051}$	**-1.8420**$_{\pm 0.025}$
	ACC	0.6531$_{\pm 0.002}$	0.5944$_{\pm 0.003}$	0.7021$_{\pm 0.004}$	0.7233$_{\pm 0.002}$	0.6926$_{\pm 0.002}$	**0.7574**$_{\pm 0.003}$
	NMI	0.7931$_{\pm 0.005}$	0.8123$_{\pm 0.003}$	0.8231$_{\pm 0.002}$	0.8089$_{\pm 0.004}$	0.8412$_{\pm 0.002}$	**0.8592**$_{\pm 0.002}$
	Time[s]	0.0953$_{\pm 0.053}$	0.0362$_{\pm 0.056}$	0.0151$_{\pm 0.044}$	0.0366$_{\pm 0.035}$	0.0166$_{\pm 0.001}$	**0.0124**$_{\pm 0.002}$

ST-ACT[23] are used as baselines. Among them, MCMC and VI are traditional approximation methods, and DAC, NCP and ST-ACT are deep amortized clustering methods. The synthetic data is generated via a 2D *Gaussian* mixture model (GMM) which is defined by the following process $\alpha \sim \text{EXP}(1),\ c_{1:N} \sim \text{CRP}(\alpha),\ \boldsymbol{\mu}_k \sim \mathcal{N}(0, \sigma_\mu^2 \mathbf{1}),\ \mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\mu}_{c_i}, \sigma^2 \mathbf{I})$. We set $\sigma_\mu = \sigma = 10$. At each training step, we generate 10 random datasets according to the above generative process. Each dataset contains 200 points on a 2D plane, and each sampled from one of 4 Gaussians. Two testing scenarios (*Scenario 1* and *Scenario 2*) are constructed to evaluate the effectiveness of the proposed method. The test set in *Scenario 1* has the same configuration (200 samples and 4 clusters) as training set, while *Scenario 2* 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 in terms of two testing scenarios. Here we report log-likelihood (*LL*), clustering accuracy (*ACC*), Normalized Mutual Information (*NMI*) and running time (seconds). The best and second results are marked in bold and underlined. As expected, the proposed AMCP consistently outperforms baselines on both scenarios. Although *Scenario 2* is more challenging, amortized clustering methods (DAC, NCP, ST-ACT and our AMCP) can capture cluster uncertainty and obtain competitive results. For testing time, amortized methods are obviously faster than traditional sampling based method (MCMC). Although VI is much fast than MCMC, it is still lower than amortized methods. In cluster-wise amortized clustering methods, DAC, ST-ACT and our ACMP achieve competitive running time since cluster-wise sequential modeling is more efficient than point-wise sequential modeling.

4.2 Experiments on Real-world Datasets

Real-world datasets, *MNIST* [22], *Tiny-ImageNet* [21] and *CIFAR-10* [37], are used to validate the performance. To formulate the amortized mechanism, we generate training and test data via following *Dirichlet* process mixture model: $\alpha \sim \text{EXP}(1),\ c_{1:N} \sim \text{CRP}(\alpha), K-1 \sim \text{BINOMIAL}(K_g - 1, 0.5),\ l_k \sim U(0, K-1),\ \mathbf{x}_i \sim U[\mathcal{D}, l_{c_i}]$. Here $\mathbf{x}_i \sim U[\mathcal{D}, l_{c_i}]$ indicates sampling $\mathbf{x}_i$ with label l_{c_i} uniformly from dataset $\mathcal{D}$, which can be training set or test set. K_g indicates the number of clusters we can sample. For all original dataset, data containing half of the classes is used to generate the training sets, and the remaining half is used to generate the test sets, with no overlap between training classes and test classes. We sample multiple clustered datasets to form training sets with $N \in \{1000, 10000\}$. For test set, we generate 1000 randomly clustered datasets with $N \in \{1000, 5000\}$. More information is given in Appendix.

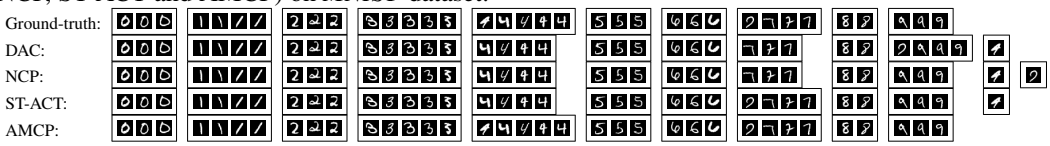
In Table 2, we depict the quantitative clustering results in real-world datasets. The best and second results are marked in bold and underlined. The average clustering results are recorded over 5 runs with different random parameter initialization. It can be seen that our AMCP outperforms all baselines. The main reason, we believe, is that AMCP not only exploits amortized properties between data points in both inter- and intra-clusters, but also has flexible clustering process, which simultaneously generate the new cluster and update the summary statistics for previous clusters. In model efficiency evaluation, the DAC, ST-ACT and AMCP consume a more competitive running time, which outperform other methods by a large margin. The reason is that DAC and ST-ACT identify each cluster after one forward pass, and AMCP generates clusters in cluster-wise manner and exploits data relations via the efficient optimal transport. However, DAC, ST-ACT are pretty worse than AMCP on clustering performance.

Among amortized clustering methods, DAC, NCP, ST-ACT and AMCP are able to process arbitrary number of clusters. For better visualization of cluster generative process, a digit subset from *MNIST* test set is randomly sampled. To illustrate how the clustering models capture the shape ambiguity of

Table 2: Clustering performance on real-world datasets. The average on 5 runs are reported.

Dataset	Metric	MCMC[1]	VI [4]	DAC [24]	NCP[30]	ST-ACT [23]	AMCP
MNIST	*ACC*	$0.8630_{\pm 0.004}$	$0.9758_{\pm 0.004}$	$0.9796_{\pm 0.001}$	$0.9633_{\pm 0.004}$	$0.9596_{\pm 0.003}$	$\mathbf{0.9845}_{\pm 0.001}$
	NMI	$0.9124_{\pm 0.005}$	$0.9482_{\pm 0.002}$	$0.9542_{\pm 0.001}$	$0.9321_{\pm 0.002}$	$0.9465_{\pm 0.002}$	$\mathbf{0.9624}_{\pm 0.001}$
	Time[s]	$56.33_{\pm 6.42}$	$63.46_{\pm 6.42}$	$24.33_{\pm 3.53}$	$73.34_{\pm 2.42}$	$22.56_{\pm 2.57}$	$\mathbf{18.22}_{\pm 3.74}$
Tiny-ImageNet	*ACC*	$0.0637_{\pm 0.005}$	$0.0785_{\pm 0.001}$	$0.0907_{\pm 0.004}$	$0.1028_{\pm 0.002}$	$0.0892_{\pm 0.003}$	$\mathbf{0.1324}_{\pm 0.002}$
	NMI	$0.3212_{\pm 0.002}$	$0.3231_{\pm 0.004}$	$0.2894_{\pm 0.005}$	$0.3254_{\pm 0.002}$	$0.2832_{\pm 0.003}$	$\mathbf{0.3421}_{\pm 0.003}$
	Time[s]	$3140.8_{\pm 18.26}$	$3450.5_{\pm 19.15}$	$1415.6_{\pm 16.57}$	$4732.8_{\pm 32.11}$	$1316.6_{\pm 21.15}$	$\mathbf{1186.4}_{\pm 19.37}$
CIFAR-10	*ACC*	$0.6322_{\pm 0.005}$	$0.8742_{\pm 0.002}$	$0.8835_{\pm 0.003}$	$0.7641_{\pm 0.005}$	$0.8632_{\pm 0.003}$	$\mathbf{0.9024}_{\pm 0.002}$
	NMI	$0.6782_{\pm 0.004}$	$0.7765_{\pm 0.004}$	$0.7873_{\pm 0.002}$	$0.7431_{\pm 0.004}$	$0.7673_{\pm 0.005}$	$\mathbf{0.8042}_{\pm 0.001}$
	Time[s]	$174.42_{\pm 6.12}$	$192.42_{\pm 5.11}$	$78.33_{\pm 4.51}$	$239.33_{\pm 9.17}$	$68.32_{\pm 6.51}$	$\mathbf{59.33}_{\pm 5.59}$

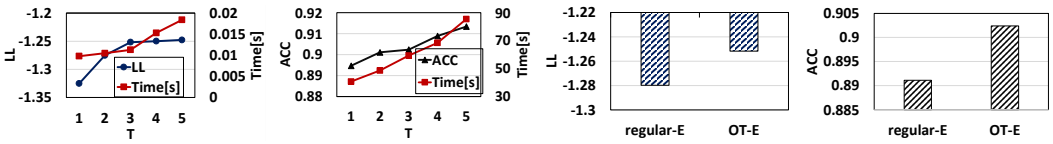
Table 3: A toy clustering results comparisons among different amortized clustering methods (DAC, NCP, ST-ACT and AMCP) on *MNIST* dataset.



some of the digits, we plot the ground-truth digit clusters and clustering results in Table 3. Comparing with DAC, NCP and ST-ACT, the proposed AMCP 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, AMCP correctly assigns it to cluster 7. All baselines are unable to assign the digit 4 written in a strange way to the right cluster, and our AMCP correctly assigns it to cluster 4. Results on *Tiny-ImageNet* and *CIFAR-10* can be obtained in Appendix.

4.3 Model Analysis

We empirically analyze the proposed model from the impact of various hyper-parameters. They are summarized as follows: (1)**The Number of EM Steps T:** AMCP can apply multiple EM steps to achieve more accurate $\phi^{(K,T)}$. Here we conduct experiments on varying T in $\{1, 2, 3, 4, 5\}$ to investigate the effect of T. Figure 2(a) and (b) show the effect of T on synthetic data (*Scenario 1*) and real-world data (*CIFAR-10*). We can see that, the performance (*LL* and *ACC*) become better and better with the increasing of T, while the running time is linearly scalable to the number of steps. (2) **OT E-step vs. Regular E-step** We introduce two different ways to conduct E-step in IntraCM: regular soft-assignment E-step and Sinkhorn-based optimal transport E-step. Figure 2(c) and (d) show the comparison between regular E-step and OT E-step on synthetic data (*Scenario 1*) and real-world data (*CIFAR-10*). The maximum number of Sinkhorn-Knopp iterations for solving the OT problem is set to 10. While the regular E-step is consistently faster than the OT E-step since it's closed-form solution, OT E-step can achieve more accurate likelihood and accuracy. Furthermore, we compare the difference between the strategy of updating the existing clusters and the strategy of not updating. More empirical results are given in Appendix.



(a) T-(*Scenario 1*) (b) T-(*CIFAR-10*) (c) E-step (*Scenario 1*) (d) E-step (*CIFAR-10*)

Figure 2: The impact of hyper-parameters.

5 Conclusion

In this paper, we studied the problem of amortized clustering, and presented our approach that perform cluster-wise amortized mixing coupling processes via intra-cluster mixing and inter-cluster coupling. An interesting direction for future research is to explore more complicated and realistic

hidden structure (e.g., hierarchical clusters) that is not possible with our approach. Furthermore, it is interested to explore novel applications that can be enabled by the interpretability and controllability brought by the amortized clustering.

6 Acknowledgement

This work was partly supported by the Beijing Natural Science Foundation under Grant (Z180006, L211016); the National Key Research and Development Program (2020AAA0106800); the National Natural Science Foundation of China under Grant 62176020; CAAI-Huawei MindSpore Open Fund; and Chinese Academy of Sciences(OEIP-O-202004).

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.

[2] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Proceedings of the NeurIPS*, volume 24, 2011.

[3] James Bergstra, Dan Yamins, David D Cox, et al. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in science conference*, pages 13–20, 2013.

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

[5] Sylvain Brohee and Jacques Van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC bioinformatics*, 7(1):1–19, 2006.

[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.

[7] Gianfranco Chicco, Roberto Napoli, and Federico Piglione. Comparisons among clustering techniques for electricity customer classification. *IEEE Transactions on power systems*, 21(2): 933–940, 2006.

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

[9] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84(6):066106, 2011.

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

[11] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In *Proceedings of the ICML*, pages 1704–1713, 2018.

[12] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

[13] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

[14] Klaus Greff, Sjoerd Van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. *Advances in Neural Information Processing Systems*, 30, 2017.

[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.

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

[17] 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*, page 1965–1972, 2017.

[18] Uzay Kaymak and Magne Setnes. Fuzzy clustering with volume prototypes and adaptive cluster merging. *IEEE Transactions on Fuzzy Systems*, 10(6):705–712, 2002.

[19] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *Proceedings of the ICLR*, 2019.

[20] Minyoung Kim. Differentiable expectation-maximization for set representation learning. In *International Conference on Learning Representations*, 2021.

[21] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7:7, 2015.

[22] 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.

[23] 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, 2019.

[24] Juho Lee, Yoonho Lee, and Yee Whye Teh. Deep amortized clustering. *arXiv preprint arXiv:1909.13433*, 2019.

[25] 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.

[26] Tsung I Lin, Jack C Lee, and Wan J Hsieh. Robust mixture modeling using the skew t distribution. *Statistics and computing*, 17(2):81–92, 2007.

[27] Tsung I Lin, Jack C Lee, and Shu Y Yen. Finite mixture modelling using the skew normal distribution. *Statistica Sinica*, pages 909–927, 2007.

[28] Huafeng Liu, Jiaqi Wang, and Liping Jing. Cluster-wise hierarchical generative model for deep amortized clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15109–15118, 2021.

[29] Grégoire Mialon, Dexiong Chen, Alexandre d'Aspremont, and Julien Mairal. A trainable optimal transport embedding for feature aggregation and its relationship to attention. *arXiv preprint arXiv:2006.12065*, 2021.

[30] Ari Pakman, Yueqi Wang, Catalin Mitelut, JinHyung Lee, and Liam Paninski. Neural clustering processes. In *Proceedings of the ICML*, pages 7455–7465, 2020.

[31] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[32] Jim Pitman et al. Combinatorial stochastic processes. Technical report, Technical Report 621, Dept. Statistics, UC Berkeley, 2002. Lecture notes, 2002.

[33] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

[34] 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.

[35] Chih-Fong Tsai and Chihli Hung. Cluster ensembles in collaborative filtering recommendation. *Applied Soft Computing*, 12(4):1417–1425, 2012.

[36] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

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

[38] CF Jeff Wu. On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103, 1983.

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

[40] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of data science*, 2(2):165–193, 2015.

[41] 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.

[42] Ron Zass and Amnon Shashua. A unifying approach to hard and probabilistic clustering. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 294–301. IEEE, 2005.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Abstract and Introduction.
   (b) Did you describe the limitations of your work? [Yes] See Conclusion
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] All assets used in this paper are open source
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] Does not contain personally identifiable information or offensive content
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Some Basic Concepts

Here, we revisit some concepts, which would help us understand our amortized mixing clustering process.

A.1 Optimal Transport

Our intra-cluster mixing is based on the transport plan between input data point $\mathbf{x}_i$ and reference vectors $\boldsymbol{\phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots, \boldsymbol{\phi}_k]$ with $\boldsymbol{\phi}_j \in \mathbb{R}^d$ seen as discrete measures, which is a by-product of the optimal transport problem. Here we adopt the Kantorovich relaxation of optimal transport with entropic regularization [31]. Let $\mathbf{C} \in \mathbb{R}^{N\times k}$ be a matrix representing the pairwise costs for aligning the elements of $\mathbf{x}_i$ and $\boldsymbol{\phi}_j$. The entropic regularized Kantorovich relaxation of optimal transport from $\mathbf{x}_i$ to $\boldsymbol{\phi}_j$ focuses on learning transport plan weight matrix $\mathbf{Q}$ in following manner:

$$J(\mathbf{Q}) = \min_{\mathbf{Q}} \sum_{\mathcal{D}} \sum_{j=1}^{k} c_{i,j} q_{i,j} - \epsilon H(\mathbf{Q}), \quad \text{s.t.} \quad \sum_{i=1}^{N} q_{i,j} = \frac{1}{k},\ \forall j \ \text{ and } \ \sum_{j=1}^{k} q_{i,j} = \frac{1}{N},\ \forall i \tag{12}$$

where the entropic regularized term $H(\mathbf{Q}) = -\sum_{i,j} q_{i,j} \log q_{i,j}$ with parameter ϵ controls the sparsity of $\mathbf{Q}$. Typically, the cost matrix $\mathbf{C}$ is defined as the negative kernel $c_{i,j} = -\mathcal{K}(\mathbf{x}_i, \boldsymbol{\phi}_j)$. Usually, above problem can be solved a matrix scaling procedure: Sinkhorn's algorithm [33].

A.2 Sinkhorn's Algorithm for Optimal Transport

Considering above entropic regularized Kantorovich relaxation of optimal transport (Eq. (12)), we can solve it efficiently by Sinkhorn's algorithm with a few fixed point matrix scaling iterations. Specifically, SK algorithm finds the optimal solution as

$$\mathbf{Q} = \text{diag}(\mathbf{u}^{(l)})\mathbf{A}\text{diag}(\mathbf{v}^{(l)}) \tag{13}$$

where the elements in $\mathbf{A} \in \mathbb{R}^{N\times k}$ are defined as $a_{i,j} = \exp(-{}^{c_{i,j}}/_{\epsilon})$ and

$$\mathbf{u}^{(l+1)} = \frac{{}^{1}/_{N}}{\mathbf{A}\mathbf{v}^{(l)}}, \quad \mathbf{v}^{(l+1)} = \frac{{}^{1}/_{k}}{\mathbf{A}\mathbf{u}^{(l)}} \tag{14}$$

Based on iterated updating, the matrix $\text{diag}(\mathbf{u}^{(l)})\mathbf{A}\text{diag}(\mathbf{v}^{(l)})$ converges to $\mathbf{Q}$ after a few iterations. Since the iterated SK algorithm constitutes the feed-forward pass for the whole inference procedure, IntraCC can be seamlessly composed to the whole inference procedure to achieve efficient computation. Furthermore, all the operations above at each step are differentiable, which enables $\boldsymbol{\phi}$ to be optimized through back-propagation. Consequently, this module can be easily injected into the whole AMCP.

A.3 Inductive Biases

First, we give a concept of Permutation Invariant Function which is the basic property of stochastic process, e.g., neural processes and gaussian process.

Definition 1. *(Permutation Invariant Function) A function $f(\cdot) : \times_i^N \mathbb{R}^M \to \mathbb{R}^D$ mapping a set of data points $\{x_i\}_{i=1}^N$ is Permutation Invariant Function if*

$$\mathbf{x} = [x_1, x_2, \cdots, x_N] \to f = [f_1(x_{\pi(1:N)}), f_2(x_{\pi(1:N)}), f_D(x_{\pi(1:N)})] \tag{15}$$

where $x_i \in \mathbb{R}^M$ and the function output is a D dimensional vector. Operation $\pi : [1, 2, \cdots, N] \to [\pi_1, \pi_2, \cdots, \pi_N]$ is a permutation set over the order of elements in the set.

Definition 2. *(Permutation Equivariant Function) A function $f(\cdot) : \times_i^N \mathbb{R}^M \to \mathbb{R}^N$ mapping a set of data points $\{x_i\}_{i=1}^N$ is Permutation Invariant Function if*

$$\mathbf{x}_\pi = [x_{\pi_1}, x_{\pi_2}, \cdots, x_{\pi_N}] \to f_\pi = \pi \circ f(x_{1:N}) \tag{16}$$

where the function output contains N elements keeping the order of inputs.

Permutation Equivariant Function keeps the order of elements in the output consistent with that in the input under any permutation operation π. Permutation invariant functions are candidate functions for learning embeddings of a set or other order uncorrelated data structure $\{x_i\}_{i=1}^N$, and the invariant property is easy to be verified. Here, we give a mean operation structure over the output

$$F(\mathbf{x}_{\pi(1:N)}) = \left(\frac{1}{N}\sum_{i=1}^{N}\phi_1(x_i), \frac{1}{N}\sum_{i=1}^{N}\phi_2(x_i), \cdots, \frac{1}{N}\sum_{i=1}^{N}\phi_M(x_i) \right) \tag{17}$$

A.4 ST-ACT

The ST-ACT [23] is a set transformer based amortized clustering with Adaptive Computation Time (ACT) [13], which is able to identify the clusters iteratively and make it to learn "when to stop". In Lee et al. [24]'s appendix, the authors give the detailed description of ST-ACT. By defining an adaptive-computation-time version of Point-inducing Multi-head Attention as follows:

$$aPMA(\mathbf{X}, k) = MAB([\mathbf{s}_1, ..., \mathbf{s}_k]^\top, \mathbf{X}), \mathbf{s}_j = PMA_1([\mathbf{s}_1, ..., \mathbf{s}_{j-1}]^\top) \text{ for } j = 2, ..., k, \tag{18}$$

which enables an RNN-like iterative computation by sequentially extending the parameters for PMA. The clustering network to output variable number of parameters is then defined as

$$\begin{aligned} &\mathbf{H}_X = ISAB_L(\mathbf{X}), \mathbf{H}_\theta^{(k)} = SAB_{L'}(aPMA(\mathbf{H}, k)), \\ &v_k = \text{SIGMOID}(\text{MEAN}(\text{ rFF}(\mathbf{H}_\theta^{(k)}[:, 1]))), \quad s_k = 1 - \prod_{j \le k} v_k \\ &(\text{logit}\pi_j^{(k)}, \theta + j^{(k)})_{j=1}^k = \text{rFF}(\mathbf{H}_\theta^{(k)}[:, 2:]), \end{aligned} \tag{19}$$

where $[:, 1]$ and $[:, 2:]$ are numpy-like notation indexing the columns. s_k is a "stop" variable where $s_k > 0.5$ means the iteration stops at k-th step and continues otherwise.

During training, ST-ACT utilize the true number of clusters as supervision for training c_t, yielding the overall loss function

$$\mathbb{E}_{p(\mathbf{X},k)} \left[-\sum_{i=1}^{N} \log \sum_{j=1}^{k} \pi_j^k p(\mathbf{x}_i, \theta_j^{(k)}) + \sum_{t=1}^{T} BCE(c_t, \mathbb{1}_{\{t<K\}}) \right] \tag{20}$$

where k is the true number of clusters, T is maximum number of steps to run.

B Model Analysis

B.1 Relations to Neural (Gaussian) Processes

It is illustrative to compare the clustering factor $p(\mathcal{C}_k|\mathcal{C}_{1:k-1}, \mathcal{S}_k)$ with Gaussian Processes (GPs) and their representation via Neural Processes (NPs) [11]. The GPs predictive distribution, given context pairs $(\mathbf{x}_G, \mathbf{y}_G) = (\mathbf{x}_i, \mathbf{y}_i)_{i \in G}$ and N targets $(\mathbf{x}_i, \mathbf{y}_i)_{i \in [1,N]}$, satisfies

$$\int p\big(\{\mathbf{y}_i\}_{i=1}^N | \{\mathbf{x}_i\}_{i=1}^N, \mathbf{x}_G, \mathbf{y}_G\big)\, d\mathbf{y}_1 = p\left(\{\mathbf{y}_i\}_{i=2}^N | \{\mathbf{x}_i\}_{i=2}^N, \mathbf{x}_G, \mathbf{y}_G\right) \tag{21}$$

i.e., the dependence on $\mathbf{x}_i$ disappears when we integrate it. Thus, unlike clustering models, there is no marginal persistency. The de Finetti representation of NPs for this distribution is

$$p\left(\{\mathbf{y}_i\}_{i=1}^N | \{\mathbf{x}_i\}_{i=1}^N, \mathbf{x}_G, \mathbf{y}_G\right) = \int \prod_{i=1}^{N} p\left(\mathbf{y}_i | \mathbf{x}_i, \mathbf{z}, \mathbf{x}_G, \mathbf{y}_G\right) p(\mathbf{z}|\mathbf{x}_G, \mathbf{y}_G) d\mathbf{z} \tag{22}$$

and thus Eq. (21) is satisfied because the prior $p(\mathbf{z}|\mathbf{x}_G, \mathbf{y}_G)$ is independent of the target $\mathbf{x}_i$.

In the analogy with our MCP, the target pairs $(\mathbf{x}_i, \mathbf{y}_i)$ correspond to $(\mathbf{x}_i, q(b_{i,j}^{(k)}|\mathbf{x}_i))$ where $\mathbf{x}_i \in \mathcal{S}_k$, the context $(\mathbf{x}_G, \mathbf{y}_G)$ corresponds to the already clustered points in $\mathcal{C}_{1:k-1}$, and the de Finetti representation corresponds to following conditional distribution:

$$p(q_{i,j}^{(k)}|\mathcal{S}_k) \simeq \int df_\theta(\boldsymbol{\phi}^{(k)}) \prod_{i=1}^{M_k} p\left(q_{i,j}^{(k)} | f_\theta(\boldsymbol{\phi}^{(k)}), \mathbf{x}_i\right) p(f_\theta(\boldsymbol{\phi}^{(k)})|\mathcal{S}_k) \tag{23}$$

which is a hierarchical modeling of clustering generation by introducing latent cluster representation $\boldsymbol{\phi}^{(k)}$. Here $q_{i,j}^{(k)}$ is $q(b_{i,j}^{(k)}|\mathbf{x}_i)$. However, the clustering prior $p(f_\theta(\boldsymbol{\phi}^{(k)})|\mathcal{S}_k)$ depends also on the target point $\mathbf{x}_i$, thus leading $p(f_\theta(\boldsymbol{\phi}^{(k)})|\mathcal{S}_k)$ to satisfy the following marginal persistence:

$$\sum_{q_{i,j}^{(k)}\in[0,1]} \sum_{j=1}^{k} p\left(q_{1,j}^{(k)}, \cdots, q_{M_k,j}^{(k)}|f_\theta(\boldsymbol{\phi}^{(k)}), \mathcal{S}_k\right) = \sum_{j=1}^{k} p\left(q_{2,j}^{(k)}, \cdots, q_{M_k,j}^{(k)}|f_\theta(\boldsymbol{\phi}^{(k)}), \mathcal{S}_k\right) \tag{24}$$

which results from marginalizing any binary variable $q_{i,j}^{(k)}$ in $p(q_{i,j}^{(k)}|f_\theta(\boldsymbol{\phi}^{(k)}), \mathcal{S}_k)$ still depends on the data point $\mathbf{x}_i$.

B.2 Connect Mixture Model to Optimal Transport

Here we analysis the proposed *IntraCC* is equivalent to solving the entropic regularized Kantorovich relaxation of optimal transport introduced in Section A.1.

We consider a mixture model with k components.

$$p(\mathbf{x}_i|\boldsymbol{\phi}) = \sum_{\mathbf{b}_i} p\left(\mathbf{x}_i, \mathbf{b}_i|\boldsymbol{\phi}\right) = \sum_{j=1}^{k} \underbrace{p(b_{i,j})}_{\pi_j} p\left(\mathbf{x}_i|\boldsymbol{\phi}_j, b_{i,j}\right); \tag{25}$$

where $\mathbf{b}_i = [b_{i,1}, b_{i,2}, \cdots, b_{i,k}]$ are latent variables encoding the data point assignments, such that $b_{i,j}$ indicates data point i was generated by component j. $\boldsymbol{\pi} = [\pi_1, \pi_2, \cdots, \pi_k]$ are the mixing coefficients.

For simplicity, we set equal mixing proportions $1/k$ and fixed shared spherical covariances of Gaussian likelihood as $p(\mathbf{x}_i|\boldsymbol{\phi}) = \sum_{j=1}^{k} \frac{1}{k}\mathcal{N}(\mathbf{x}_i; \boldsymbol{\phi}_j, \epsilon\mathbf{I})$. In this case, $\boldsymbol{\phi}_j \in \mathbb{R}^d$ is the mean vector. In this case, $p(b_{i,j}) = 1/k$. Suppose data points $\mathbf{x}_i \in \mathcal{D}$ are i.i.d sampled, the log-likelihood of $\mathcal{D}$ under the model is $\log p(\mathcal{D}|\boldsymbol{\phi}) = \sum_{\mathcal{D}} \log p(\mathbf{x}_i|\boldsymbol{\phi})$.

We focus on optimizing a lower bound derived by the following Jensen variational lower bound:

$$\sum_{\mathcal{D}} \log p(\mathbf{x}_i|\boldsymbol{\phi}) = \sum_{\mathcal{D}} \log \sum_{j=1}^{k} p(\mathbf{x}_i, b_{i,j}|\boldsymbol{\phi}) \geq \sum_{\mathcal{D}} \sum_{j=1}^{k} q(b_{i,j}|\mathbf{x}_i) \log \frac{p\left(\mathbf{x}_i, b_{i,j}|\boldsymbol{\phi}\right)}{q(b_{i,j}|\mathbf{x}_i)} \tag{26}$$

where $q(b_{i,j}|\mathbf{x}_i)$ is the variational distribution. Here we denote $q_{i,j}$ as $q(b_{i,j}|\mathbf{x}_i)q(\mathbf{x_i})$ with $q(\mathbf{x_i}) = 1/N$. Then maximizing Eq.(26) with respect to the $\mathbf{Q} = (q_{i,j}) \in \mathbb{R}^{N\times k}$ is equivalent to

$$J(\mathbf{Q}) = \min_{\mathbf{Q}} \sum_{\mathcal{D}} \sum_{j=1}^{k} c_{i,j} q_{i,j} - \epsilon H(\mathbf{Q}) \tag{27}$$

where $c_{i,j} = -\log p\left(\mathbf{x}_i, b_{i,j}|\boldsymbol{\phi}\right) = \frac{||\mathbf{x}_i - \boldsymbol{\phi}_j||^2}{2\epsilon} + \text{const}$. By having the quadratic kernel $\mathcal{K}(x, \phi) = -\frac{1}{2}||x - \phi||^2$, we can see Eq.(27) has strong connection to Eq.(12), i.e., Mixture-based OT loss Eq.(27) reduces to original OT loss Eq.(27) except for the constraints $\sum_{i=1}^{N} q_{i,j} = \frac{1}{k}, \forall j$ (Note that the constraint $\sum_{j=1}^{k} q_{i,j} = \frac{1}{N}, \forall i$ is automatically met).

B.3 Connect Induced Self-Attention Block (ISAB) to OT-based Mixture Model

First, we give the additional implementation details of self attention module. The whole training procedure related to each component is also given in this section. Before that, multi-head attention between two different sets is defined as [23]

$$MAB(\mathbf{X}, \mathbf{Y}) = \mathbf{H} + rFF(\mathbf{H}), \quad \mathbf{H} = \mathbf{X} + rFF(MhA(\mathbf{X}, \mathbf{Y})) \tag{28}$$

where $rFF()$ is a feed-forward layer applied row-wise (i.e., for each element). We use Multi-head attention Module $\text{MHA}(\cdot)$ to exploit pair-wise or higher-order interactions between data points in both inter-and intra-cluster. Considering we want to capture the elements-wise relationship between $\mathbf{X}$ and $\mathbf{Y}$, we set $\mathbf{X}$ as query, and set key and values are $\mathbf{Y}$. The Multi-head attention Module is defined as follow

$$\text{MHA}(\mathbf{X}, \mathbf{Y}) = \text{CONCAT}(\mathbf{O}_1, \cdots, \mathbf{O}_H)\mathbf{W}^h, \quad \mathbf{O}_i = \sigma\left(\mathbf{X}\mathbf{W}_i^Q \left(\mathbf{Y}\mathbf{O}_i^K\right)^\top\right)\mathbf{Y}\mathbf{W}_i^V \tag{29}$$

where $\sigma(\cdot)$ is activation function, $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V$ are head-specific transform matrices.

A Self-Attention Block (SAB) is simply MAB applied to the set itself: $SAB(\mathbf{X}) \triangleq MAB(\mathbf{X}, \mathbf{X})$. Note that the time-complexity of SAB is $O(N^2 d)$ because of pairwise computation. To reduce this, Lee et al. [23] proposed to use Induced Self-Attention Block (ISAB) defined as

$$ISAB(\mathbf{X}) = MAB(\mathbf{X}, MAB(\boldsymbol{\phi}, \mathbf{X})), \tag{30}$$

where $\boldsymbol{\phi} \in [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots, \boldsymbol{\phi}_k]^\top \in \mathbb{R}^{k \times d}$ are trainable inducing points. ISAB indirectly compares the elements of $\mathbf{X}$ through the inducing points, reducing the time-complexity to $O(Nk)$.

Actually, ISAB is equivalent our *IntraCM* [14, 20]. Here we give a detailed analysis. A matrix form of ISAB is given as follows:

$$\boldsymbol{\phi}_j \leftarrow \sum_i \frac{\exp((\mathbf{W}^Q \boldsymbol{\phi}_j)^\top (\mathbf{W}\mathbf{x}_i)/\sqrt{d})}{\sum_i \exp((\mathbf{W}^Q \boldsymbol{\phi}_j)^\top (\mathbf{W}\mathbf{x}_i)/\sqrt{d})} \mathbf{W}\mathbf{x}_i \tag{31}$$

where $\mathbf{W}^Q$ and $\mathbf{W}$ are the weight matrices for queries and keys/values, respectively. Considering $\mathbf{W}^Q$ and $\boldsymbol{\phi}$ are both learnable parameters, we can get a more general ISAB as follow:

$$\boldsymbol{\phi}_j \leftarrow \sum_i \frac{\exp(\boldsymbol{\phi}_j^\top (\mathbf{W}\mathbf{x}_i)/\sqrt{d})}{\sum_i \exp(\boldsymbol{\phi}_j^\top (\mathbf{W}\mathbf{x}_i)/\sqrt{d})} \mathbf{W}\mathbf{x}_i \tag{32}$$

We consider the optimal transport problem between $\boldsymbol{\phi}$ and $\mathbf{X}$ only with constraint $\sum_{i=1}^N q_{i,j} = \frac{1}{k}, \forall j$, i.e.,

$$J(\mathbf{Q}) = \min_{\mathbf{Q}} \sum_{\mathcal{D}} \sum_{j=1}^{k} c_{i,j} q_{i,j} - \epsilon H(\mathbf{Q}), \quad \text{s.t.} \quad \sum_{i=1}^{N} q_{i,j} = \frac{1}{k}, \forall j \tag{33}$$

where $c_{i,j} = -\mathcal{K}(\mathbf{x}_i, \boldsymbol{\phi}_j) = -\boldsymbol{\phi}_j^\top \mathbf{W}\mathbf{x}_i$ corresponds to a linear kernel. We can solve it via Lagrange multiplier and get

$$q_{i,j} = \exp(-c_{i,j}/\epsilon + \alpha_j) \tag{34}$$

where α_j is a constant. By combining the constraint $\sum_i q_{i,j} = 1/k$, we can get

$$\exp(\alpha_j) = \frac{1}{k \sum_i \exp(-c_{i,j}/\epsilon)}, \quad q_{i,j} = \frac{1}{k} \frac{\exp(\boldsymbol{\phi}_j^\top (\mathbf{W}\mathbf{x}_i)/\epsilon)}{\sum_i \exp(\boldsymbol{\phi}_j^\top (\mathbf{W}\mathbf{x}_i)/\epsilon)} \tag{35}$$

By setting $\epsilon = \sqrt{d}$, we can get the updated $\boldsymbol{\phi}_j$ as follows:

$$\boldsymbol{\phi}_j \leftarrow \sum_i \frac{\exp(\boldsymbol{\phi}_j^\top (\mathbf{W}\mathbf{x}_i)/\sqrt{d})}{\sum_i \exp(\boldsymbol{\phi}_j^\top (\mathbf{W}\mathbf{x}_i)/\sqrt{d})} \mathbf{W}\mathbf{x}_i \tag{36}$$

Note that here we set a linear kernel to analysis the relation between the proposed IntraCM and ISAB, we give a more general form (non-linear neural network-based kernel) in main manuscript which is able to achieve more flexible solution.

B.4 Connection to Existing Amortized Clustering

Different traditional clustering relies on point-wise learning manner, amortized clustering (e.g., ST-ACT, DAC, NCP, CHiGac and our AMCP) allows cluster-wise learning strategy (learning one cluster one time), which is more efficient. However, ST-ACT inherits the feature of MoG which needs to manually set the number of clusters. DAC generalizes ST-ACT and enables to produce a varying number of clusters. Both ST-ACT and DAC can be regarded as simply amortized inference of Gaussian mixture with the aid of neural networks. The attention mechanism used in ST-ACT and DAC are much complexed than the proposed optimal transport mixing reducing the quadratic cost of traditional similarity weight computation. The most similar work to our methods is NCP [30] and CHiGac [28]. Both of them explain clustering from the perspective of generative models, which similar in spirit to the popular Gibbs sampling algorithm for Dirichlet process mixture models, but without positing particular priors on partitions. Besides, NCP and CHiGac rely heavily on the selecting of anchor points and the processing ordering and often exhibits unstable properties. Different from previous work, AMCP connects the relationship of mixture model and optimal transport to describe the cluster's summary statistics well, and seamlessly combines mixing and coupling to achieve efficient amortized clustering.

B.5 Complexity Analysis

For our AMCP, *IntraCM* perform T EM steps, each E-step amounts to computing the Gaussian likelihoods, and additionally running of Sinkhorn's algorithm if we adopt OT-based E step. M-step mainly consists of the weighted sum of partial derivative. Thus the complexity of regular E-step takes $O(Nkd)$, and $O(Nk(L+d))$ for OT-based E step where L is the number of iterations of Sinkhorn's algorithm. The complexity of M-step takes $O(Nkd)$. Thence, the time complexity for EM-step is $O(Nkd)$ or $O(Nk(d+L))$. Usually, L is set to $5-10$, which is largely smaller than d.

We know that k is largely smaller than N, which indicates $O(Nkd)$ or $O(Nk(d+L))$ is largely lower than $O(N^2d)$ used in attention-based methods (e.g., DAC, ST-ACT). Although DAC and ST-ACT introduce induced self-attention block which is able to achieve complexity $O(Nkd)$ (for fair comparison, here we set the number of trainable inducing points to k), they cannot avoid computing the traditional attention module with complexity $O(N^2)$.

C Experimental Details and Additional Results

C.1 Infrastructure and Experimental Details

Infrastructure: We implement our model with Pytorch, and conduct our experiments with:

- CPU: Intel Xeon Silver 4116 @2.1GHz.
- GPU: 8x GeForce RTX 2080Ti.
- RAM: DDR4 256GB.
- ROM: 8x 1TB 7.2K 6Gb SATA and 1x 960G SATA 6Gb R SSD.
- Operating system: Ubuntu 18.04 LTS.
- Environments: Python 3.7; NumPy 1.18.1; SciPy 1.2.1; scikit-learn 0.23.2; seabornn 0.1; torch_geometric 1.6.1; matplotlib 3.1.3; dgl 0.4.2; pytorch 1.6.

Hyper-parameter search: We trained with the following hyperparameters: The neural network (e.g., $f_\theta(\cdot)$) in our model is a multilayer perceptron (MLP). We use the *tanh* activation function. We apply dropout before every layers, except the last layer. The model is trained using Adam. We then tune the other hyper-parameters of both our approach's and our baselines' automatically using the TPE [2] method implemented by Hyperopt [3]. We let Hyperopt conduct 200 trials to search for the optimal hyper-parameter configuration for each method on the validation of each dataset. The hyper-parameter search space is specified as follows:

- The number of hidden layers in a neural network: {0, 1, 2, 3 }.
- The number of neurons in a hidden layer: {100, 200, $\cdots$, 1000 }.
- Learning rate: $[10^{-8}, \cdots, 1]$.
- L2 regularization: $[10^{-12}, \cdots, 1]$.
- Dropout rate:$[0.05, \cdots, 1]$.
- Regularization coefficient λ: $[1, 10]$.
- The standard deviation of the prior for { θ}: $[0.01, 0.5]$.

Datasets: Three real-world datasets (*MNIST, Tiny-ImageNet, CIFAR-10*) are used to evaluate the proposed model.

- *MNIST*: The MNIST dataset consists of 70,000 hand-written digits. The images are centered and of size 28×28 pixels. We reshaped each image to a 784-dimensional vector.
- *Tiny-ImageNet*: Tiny-ImageNet contains 100,000 images of 200 classes (500 for each class) downsized to 64×64 colored images chosen from ImageNet ILSVRC-2012 challenge. Each class has 500 training images, 50 validation images and 50 test images.
- *CIFAR-10*: The CIFAR-10 dataset (Canadian Institute for Advanced Research, 10 classes) is a subset of the Tiny Images dataset and consists of 60,000 32$\times$32 color images. The images

Table 4: Summary of experimental real-world datasets.

Dataset	*MNIST*	*Tiny-ImageNet*	*CIFAR-10*
♯ Samples	70,000	110,000	60,000
♯ Input Dimensions	784	4,096	1,024
♯ Clusters	10	200	10

are labelled with one of 10 mutually exclusive classes: airplane, automobile (but not truck or pickup truck), bird, cat, deer, dog, frog, horse, ship, and truck (but not pickup truck). There are 6,000 images per class with 5000 training and 1000 testing images per class.

The overall statistical information about datasets is given in Table 4.

Note that both *MNIST* and *CIFAR-10* have 10 classes and Tiny-ImageNet have 200 classes. To formulate amortized mechanism, we process the dataset as follow:

- For *MNIST* and *CIFAR-10*, we define the data containing the first 5 classes as the training set, and the last 5 classes as the test set. Specifically, for *MNIST*, training set contains samples related to digits 0, 2, 4, 6, 8, and test set is related to 1, 3, 5, 7, 9. For *CIFAR-10*, training set contains samples related to airplane, bird, deer, frog, ship, and the rest is the test set.
- For *Tiny-ImageNet*, we define the data containing the first 100 classes as the training set, and the remaining 100 classes as the test set (the class order refers to the order in which it appeared in the original *Tiny-ImageNet*).

We sample multiple clustered datasets to form training sets with $N \in \{1000, 10000\}$. For test set, we generate 1000 randomly clustered datasets with $N \in \{1000, 5000\}$.

Evaluation Metrics: Two type of evaluation metrics are adopted. Clustering accuracy

$$ACC = \max_{m \in \mathcal{M}} \frac{1}{N} \mathbb{1}\{y_i = m(\hat{y}(\mathbf{x}_i))\}$$

and Normalized Mutual Information

$$NMI(\mathcal{C}_i, \mathcal{C}_j) = \frac{MI(\mathcal{C}_i, \mathcal{C}_j)}{\sqrt{H(\mathcal{C}_i)H(\mathcal{C}_j)}},$$

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. For *NMI*, $MI(\cdot,\cdot)$ denotes the mutual information and $H(\cdot)$ is the entropy. This measure can be applied to any assignment coming from the clusters or the true labels. If the two assignments $\mathcal{C}_k$ and $\mathcal{C}_j$ are independent, the *NMI* is equal to 0. If one of them is deterministically predictable from the other, the *NMI* is equal to 1.

Model Architecture: Here, we give the architectural details of the proposed AMCP models used for the synthetic and real-world experiments. We implement $f_\theta(\cdot)$ by means of a multilayer perceptron (MLP) with a sigmoid output. The representation $\boldsymbol{\phi}_j^{(k)}$ is a real-value d-dimensional vector squashed to the $(0, 1)$ range by a sigmoid function before being fed into the networks. For synthetic data, we set $d = 2$. For real-world datasets, we set $d = 512$.

For DAC, NCP and ST-ACT, we used the code released by the authors with default hyper-parameters. For ST-ACT and DAC, the network architecture is $ISAB(m, dh, 4) \rightarrow ISAB(m, dh, 4) \rightarrow PMA(4, dh, 4) \rightarrow SAB(dh, 4) - SAB(dh, 4) - FC(d_{out})$ where $SAB(d, h)$ means the set attention block with d/h units/heads, and $PMA(d, h, k)$ is the pooling multihead attention layer with k vectors and d/h units/heads. For NCF, there are composed by the deep networks h, g, u, f. For 2D Gaussians, h is $MLP[2 - 256 - 256 - 256 - 128]$ with ReLUs; u is $MLP[2 - 256 - 256 - 256 - 128]$ with ReLUs; g is $MLP[128 - 256 - 256 - 256 - 256]$ with ReLUs; f is $MLP[384 - 256 - 256 - 256 - 1]$ with ReLUs. For real-world datasets, h is 2 layers of $[convolutional + maxpool + ReLU] + MLP[320 - 256 - 128]$ with ReLUs; u is same as h; g is $MLP[256 - 128 - 128 - 128 - 128 - 256]$ with ReLUs; f is $MLP[384 - 256 - 256 - 256 - 1]$ with ReLUs.

For all baselines, we use ADAM as optimizer with learning rate 10^{-4} to train models..

Table 5: A toy clustering results comparisons among different amortized clustering methods (DAC, NCP, ST-ACT and AMCP) on *MNIST* dataset.

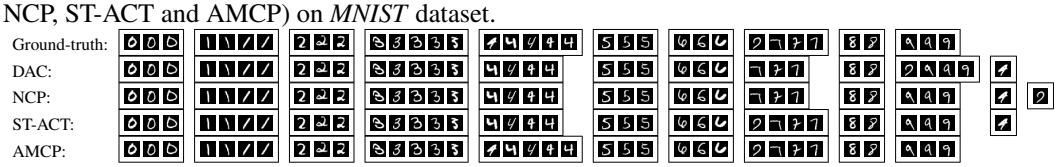


Table 6: A toy clustering results comparisons among different amortized clustering methods (DAC, NCP, ST-ACT and AMCP) on *Tiny-ImageNet* dataset.

C.2 Additional Experimental Results

C.2.1 Cluster Generative Process

To illustrate how the clustering models capture the shape ambiguity of some of the images, we plot the ground-truth image clusters and clustering results (obtained by DAC, NCP, ST-ACT and AMCP respectively) on *MNIST*, *Tiny-ImageNet* and *CIFAR-10*, as shown in Table 5, 6 and 7. Comparing with DAC, NCP, ST-ACT, the proposed AMCP can get the most accurate results. Taking *MNIST* dataset as an example, DAC assigns the digit 7 (with similar appearance to 9) to cluster 9, and NCP generates a new cluster for it. Fortunately, ST-ACT and DAC correctly assigns it to cluster 7. All baselines are unable to assign the digit 4 written in a strange way to the right cluster, and our AMCP correctly assigns it to cluster 4. Similar results can be found in other two datasets.

C.2.2 Convergence

We investigate the evolution of the model performance along training epochs and test epochs. Figure 3(a) and (b) shows the training and test likelihood vs. epoch for NCP and our AMCP on synthetic data. Figure 3(c) and (d) shows the training and test accuracy vs. epoch for NCP and our AMCP on *CIFAR-10*. We can see that AMCP can achieve better performance than NCP on both synthetic and real-world datasets, and AMCP converges faster than NCP.

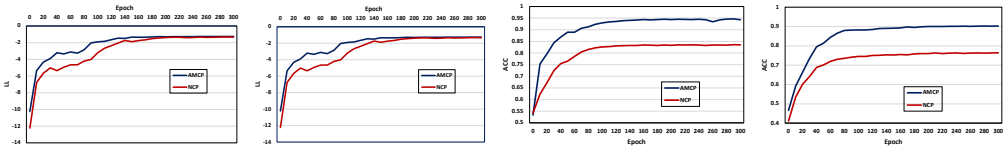
C.2.3 Empirical Study on Hyper-parameters Impacts

We empirically analyze the proposed model from the impact of various hyper-parameters. They are summarized as follows.

The Number of EM Steps T: AMCP can apply multiple EM steps to achieve more accurate $\phi^{(K,T)}$. Here we conduct experiments on varying T in $\{1, 2, 3, 4, 5\}$ to investigate the effect of T. Figure 4 and 5 show the effect of T on synthetic data (*Scenario 1* and *Scenario 2*) and real-world data (*MNIST*, *Tiny-ImageNet*, *CIFAR-10*). We can see that, the performance (*LL* and *ACC*) become better and better with the increasing of T, while the running time is linearly scalable to the number of steps.

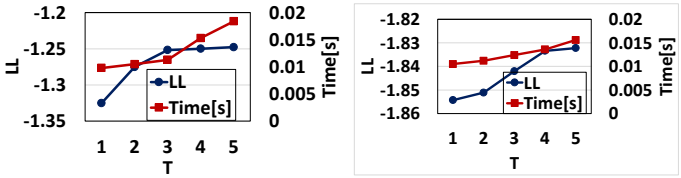
OT E-step vs. Regular E-step We introduce two different ways to conduct E-step in IntraCM: regular soft-assignment E-step and Sinkhorn-based optimal transport E-step. Figure 6 and 7 show the comparison between regular E-step and OT E-step on synthetic data (*Scenario 1* and *Scenario 2*)

Table 7: A toy clustering results comparisons among different amortized clustering methods (DAC, NCP, ST-ACT and AMCP) on *CIFAR-10* dataset.



(a) Training/synthetic (b) Test/Scenario 1 (c) Training/CIFAR-10 (d) Test/CIFAR-10

Figure 3: Training and test curves for the AMCP and NCP of synthetic data (Scenario 1) and *CIFAR-10*.



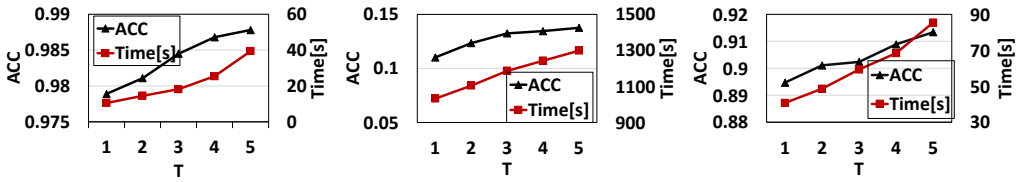
(a) T-(*Scenario 1*) (b) T-(*Scenario 2*)

Figure 4: The impact of T on synthetic datasets.

and real-world data (*MNIST*, *Tiny-ImageNet*, *CIFAR-10*). The maximum number of Sinkhorn-Knopp iterations for solving the OT problem is set to 10. While the regular E-step is consistently faster than the OT E-step since it's closed-form solution, OT E-step can achieve more accurate likelihood and accuracy.

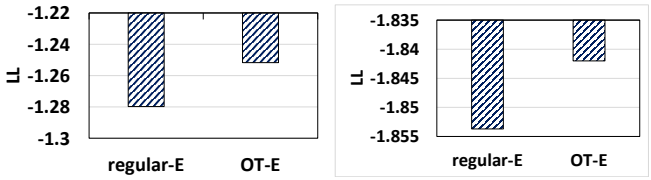
Update Previous Clusters In order to prevent the previous generated cluster from inaccurate, we can update previous clusters as stated in line 12 of Algorithm 1. Here we investigate the pros and cons of the two strategies by comparing whether previous clusters are updated or not, as shown in Table 8 and 9. We can see that AMCP* can achieve more better performance in more complicated scenario (*Scenario 2*, *Tiny-ImageNet* and *CIFAR-10*). The possible reason is that a single step to form cluster cannot exploiting data relations well, whole the update strategy is able to update previous clusters iteratively.

The Number of Classes in Training set Here we conducts experiments to investigate the model performance on different size of training set. Figure 8 shows the performance comparison on different training set with different number of classes. For *MNIST* and *CIFAR-10*, the number of classes in training set is turned from 1 to 5. For *Tiny-ImageNet*, the number of classes is turned in {20, 40, 60, 80, 100}. We can see the accuracy of all methods increases with the number of classes, because more classes in the training data help the test data to scale quickly. AMCP can achieve optimal results in all training scenarios, which proves the effectiveness of AMCP. Further, we report the trining time in Figure 9. We can see that AMCP still outperforms baselines.



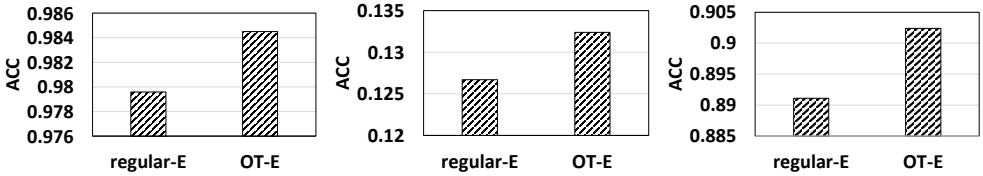
(a) T-(*MNIST*) (b) T-(*Tiny-ImageNet*) (c) T-(*CIFAR-10*)

Figure 5: The impact of T on real-world datasets.



(a) E-step (*Scenario 1*) (b) E-step (*Scenario 2*)

Figure 6: OT E-step vs regular E-step on synthetic datasets.



(a) E-step (*MNIST*) (b) E-step (*Tiny-ImageNet*) (c) E-step (*CIFAR-10*)

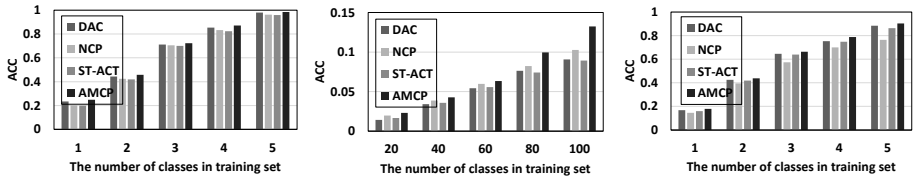
Figure 7: OT E-step vs regular E-step on real-world datasets.

Table 8: Comparing where previous clusters are updated or not in synthetic datasets. Here AMCP* indicates methods updating previous clusters.

Dataset	*Scenario 1*		*Scenario 2*	
Metric	*LL*	Time[s]	*LL*	Time[s]
AMCP	**-1.2517**$_{\pm 0.035}$	**0.0113**$_{\pm 0.003}$	-1.8420$_{\pm 0.025}$	**0.0124**$_{\pm 0.002}$
AMCP*	-1.2587$_{\pm 0.052}$	0.0143$_{\pm 0.003}$	**-1.8323**$_{\pm 0.056}$	0.0154$_{\pm 0.003}$

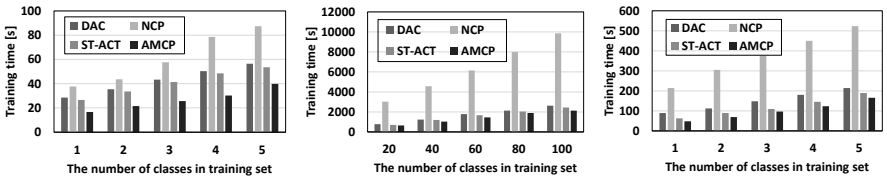
Table 9: Comparing where previous clusters are updated or not in real-world datasets. Here AMCP* indicates methods updating previous clusters.

Dataset	*MNIST*		*Tiny-ImageNet*		*CIFAR-10*	
Metric	*ACC*	Time[s]	*ACC*	Time[s]	*ACC*	Time[s]
AMCP	**0.9845**$_{\pm 0.001}$	**18.22**$_{\pm 3.74}$	0.1324$_{\pm 0.002}$	**1186.4**$_{\pm 19.37}$	0.9024$_{\pm 0.002}$	**59.33**$_{\pm 5.59}$
AMCP*	0.9764$_{\pm 0.002}$	21.44$_{\pm 2.85}$	**0.1378**$_{\pm 0.002}$	1273.4$_{\pm 26.25}$	**0.9085**$_{\pm 0.003}$	65.44$_{\pm 4.21}$



(a) *MNIST* (b) *Tiny-ImageNet* (c) *CIFAR-10*

Figure 8: Performance (ACC) comparison on different training set with different number of classes.



(a) *MNIST* (b) *Tiny-ImageNet* (c) *CIFAR-10*

Figure 9: Training time on different training sets with different number of classes.