LEARNING PROTOTYPE-ORIENTED SET REPRESENTATIONS FOR META-LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning from set-structured data is a fundamental problem that has recently attracted increasing attention, where a series of summary networks are introduced to deal with the set input. In fact, many meta-learning problems can be treated as set-input tasks. Most existing summary networks aim to design different architectures for the input set in order to enforce permutation invariance. However, scant attention has been paid to the common cases where different sets in a meta-distribution are closely related and share certain statistical properties. Viewing each set as a distribution over a set of global prototypes, this paper provides a novel optimal transport (OT) based way to improve existing summary networks. To learn the distribution over the global prototypes, we minimize its OT distance to the set empirical distribution over data points, providing a natural unsupervised way to improve the summary network. Since our plug-and-play framework can be applied to many meta-learning problems, we further instantiate it to the cases of few-shot classification and implicit meta generative modeling. Extensive experiments demonstrate that our framework significantly improves the existing summary networks on learning more powerful summary statistics from sets and can be successfully integrated into metric-based few-shot classification and generative modeling applications, providing a promising tool for addressing set-input and meta-learning problems.

1 INTRODUCTION

Machine learning models, such as convolutional neural networks for images (He et al. 2016) and recurrent neural networks for sequential data (Sutskever et al. 2014), have achieved great success in taking advantage of the structure in the input space (Maron et al. 2020). However, extending them to handle unstructured input in the form of sets, where a set can be defined as an unordered collections of elements, is not trivial and has recently attracted increasing attention (Jurewicz & Strømberg-Derczynski, 2021). Set-input is relevant to a range of problems, such as understanding a scene formed of a set of objects (Eslami et al. 2016), classifying an object composed of a set of 3D points (Qi et al. 2017), and estimating summary statistics from a set of data points for implicit generative models (Chen et al. 2021). Moreover, many meta-learning problems, which process different but related tasks, may also be viewed as set-input tasks (Lee et al., 2019), where an input set corresponds to the training dataset of a single task. Therefore, we broaden the scope of set related applications by including traditional set-structured input problems and most meta-learning problems. Both of them aim to improve the quick adaptation ability for unseen sets, even though the latter is more difficult because of limited samples or the occurrence of new categories for classification problems.

For a set-input, the output of the model must not change if the elements of the input set are reordered, which entails permutation invariance of the model. To enforce this property, multiple researchers have recently focused on designing different network architectures, which can be referred to as a *summary network* for compressing the set-structured data into a fixed-size output. For example, the prominent works of Zaheer et al. (2017) and Edwards & Storkey (2017) combined the standard feed-forward neural networks with a set-pooling layer, which have been proven to be universal approximators of continuous permutation invariant functions (Zaheer et al. 2017). Lee et al. (2019) further introduced Set Transformer to encode and aggregate the features within the set using the multi-head attention mechanism. Maron et al. (2020) designed deep models and presented a principled approach to learn sets of symmetric elements. Despite the effectiveness and recent popularity of

these works in set-input problems, there are several shortcomings for existing summary networks, which could hinder their applicability and further extensions: 1) The parameters of the summary network are typically optimized by a task-specific loss function, which could limit the models' flexibility. 2) A desideratum of a summary network is to extract set features, which have enough ability to represent the summary statistics of the input set and thus benefit the corresponding set-specific task; but for many existing summary networks, there is no clear evidence or constraint that the outputs of the summary network could describe the set's summary statistics well. These limits still remain even using the recent more carefully designed summary networks, while sets with limited samples further exacerbate the problem.

To address the above shortcomings, we present a novel and generic approach to improve the summary networks for set-structured data and adapt them to meta-learning problems. Motivated by meta-learning that aims to extract transferable patterns useful for all related tasks, we assume that there are K global prototypes (i.e., centers) among the collection of related sets, and each prototype or center is encouraged to capture the statistical information shared by those sets, similar to the "topic" in topic modeling or "dictionary atom" in dictionary learning. Specifically, for the jth set, we consider it as a discrete distribution P_j over all the samples within the set (in data or feature space). At the same time, we also represent this set with another distribution Q_j (in the same space with P_j), supported on K global prototypes with a K-dimensional set representation $\boldsymbol{h}_j$. Since $\boldsymbol{h}_j$ measures the importance of global prototypes for set j, it can be treated as the prototype proportion for summarizing the salient characteristics of set j. Moreover, the existing summary networks can be adopted to encode set j as $\boldsymbol{h}_j$ for their desired property of permutation invariance. In this way, we can formulate the learning of summary networks as the process of learning a P_j to be as close to Q_j as possible, a process facilitated by leveraging the optimal transport (OT) distance (Peyré & Cuturi 2019). Therefore, the global prototypes and summary network can be learned by jointly optimizing the task-specific loss and OT distance between P_j and Q_j in an *end-to-end* manner. We can refer to this method as prototype-oriented framework for meta-learning, which is applicable to a range of unsupervised and supervised tasks, such as set-input problems solved by summary networks, meta generation (Hong et al., 2020c; Antoniou et al., 2017), metric-based few-shot classification (Snell et al., 2017), and learning statistics for approximate Bayesian computation (Chen et al., 2021).

Since our plug-and-play framework can be applied to many meta-learning problems, this paper further instantiates it to the cases of metric-based few-shot classification and implicit meta generative modeling. We summarize our contributions as follows: (1) We formulate the learning of summary network as the distribution approximation problem by minimizing the distance between distribution over data points and another distribution over global prototypes. (2) We leverage optimal transport distance to measure the distance between the distributions for use in a joint learning algorithm. (3) We apply our method to metric-based few-shot classification and construct implicit meta generative model, where summary network is used to extract the summary statistics from set and optimized by the OT loss. Experiments on several meta-learning tasks demonstrate that introducing the OT loss into existing summary networks can extract more effective set representations for the corresponding tasks, which can be also integrated into existing few-shot classification and GAN frameworks, producing a new way to learn the set' summary statistics applicable to many applications.

2 Background

2.1 Summary networks for set-structured input

To deal with the set-structured input $D_j = \{\boldsymbol{x}_{j,1:N_j}\}$ and satisfy the permutation invariance in set, a remarkably simple but effective summary network is to perform pooling over embedding vectors extracted from the elements of a set. More formally,

$$S_\phi\left(D_j\right) = g_{\phi_2}\left(\text{pool}\left(\left\{f_{\phi_1}\left(\boldsymbol{x}_{j1}\right), \ldots, f_{\phi_1}\left(\boldsymbol{x}_{jN_j}\right)\right\}\right)\right), \tag{1}$$

where $f_{\phi_1}(\cdot)$ acts on each element of a set and $g_{\phi_2}(\text{pool}(\cdot))$ aggregates these encoded features and produces desired output, and $\phi = \{\phi_1, \phi_2\}$ denotes the parameters of the summary network. Most network architectures for set-structured data follow this structure; see more details from the previous works (Lee et al., 2019; Zaheer et al., 2017; Edwards & Storkey, 2017; Maron et al., 2020).

2.2 Optimal Transport

Although OT has a rich theory, we limit our discussion to OT for discrete distributions and refer the reader to Peyré & Cuturi (2019) for more details. Let us consider p and q as two discrete probability

distributions on the arbitrary space $X \subseteq \mathbb{R}^d$, which can be formulated as $p = \sum_{i=1}^{n} a_i \delta_{x_i}$ and $q = \sum_{j=1}^{m} b_j \delta_{y_j}$. In this case, $\boldsymbol{a} \in \Sigma^n$ and $\boldsymbol{b} \in \Sigma^m$, where Σ^m denotes the probability simplex of $\mathbb{R}^m$. The OT distance between $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined as

$$\text{OT}(\boldsymbol{a}, \boldsymbol{b}) = \min_{\mathbf{T} \in U(\boldsymbol{a}, \boldsymbol{b})} \langle \mathbf{T}, \mathbf{C} \rangle, \tag{2}$$

where $\langle \cdot, \cdot \rangle$ means the Frobenius dot-product; $\mathbf{C} \in \mathbb{R}_{\geq 0}^{n \times m}$ is the transport cost function with element $C_{ij} = C(x_i, y_j)$; $\mathbf{T} \in \mathbb{R}_{>0}^{n \times m}$ denotes the doubly stochastic transport probability matrix such that $U(\boldsymbol{a}, \boldsymbol{b}) := \{\mathbf{T} \mid \sum_i^n T_{ij} = b_j, \sum_j^m T_{ij} = a_i\}$. To relax the time-consuming problem when optimising the OT distance, Cuturi (2013) introduced the entropic regularization, $H = -\sum_{ij} T_{ij} \ln T_{ij}$, leading to the widely-used Sinkhorn algorithm for discrete OT problems.

3 Proposed framework

In meta-learning, given a meta-distribution $p_{\mathcal{M}}$ of tasks, the marginal distribution p_j of task j is sampled from $p_{\mathcal{M}}$ for $j \in \mathcal{J}$, where $\mathcal{J}$ denotes a finite set of indices. For example, we can sample p_j from $p_{\mathcal{M}}$ with probability $\frac{1}{\mathcal{J}}$ when $p_{\mathcal{M}}$ is uniform over a finite number of marginals. During meta-training, direct access to the distribution of interest p_j is usually not available. Instead, we will observe a set of data points $D_j = \{\boldsymbol{x}_{ji}\}_{i=1}^{N_j}$, which consists of N_j i.i.d samples from p_j over $\mathbb{R}^d$. We can roughly treat the meta-learning problems as the set-input tasks, where dataset D_j from p_j corresponds to an input set. To learn more representative features from related but unseen sets in meta-learning problems, we adopt the summary network as the encoder to extract set representations and improve it by introducing the OT loss and global prototypes, providing many applications. Besides, we also provide the applications to metric-based few-shot classification and implicit generative framework by assimilating the summary statistics. Below we describe our model in detail.

3.1 Learning global prototypes and set representation via optimal transport

Given J sets from meta-distribution $p_{\mathcal{M}}$, we can represent each set D_j from meta-distribution $p_{\mathcal{M}}$ as an empirical distribution over N_j samples on the original data space, formulated as

$$P_j = \sum\nolimits_{i=1}^{N_j} \frac{1}{N_j} \delta_{\boldsymbol{x}_{ji}}, \boldsymbol{x}_{ji} \in \mathbb{R}^d. \tag{3}$$

Since all sets (distributions) draw from meta-distribution $p_{\mathcal{M}}$ are closely related, it is reasonable to assume that these sets share some statistical information. Motivated by dictionary learning and topic modeling, we define the shared information as the learnable global prototype matrix $\mathbf{B} = \{\boldsymbol{\beta}_k\} \in \mathbb{R}^{d \times K}$, where K means the number of global prototypes and $\boldsymbol{\beta}_k$ denotes the distributed representation of the k-th prototype in the same space of the observed data points (i.e., "topic" in topic modeling). Given the global prototype matrix $\mathbf{B}$, each set j can be represented with a K-dimensional weight vector $\boldsymbol{h}_j \in \Sigma_k$ ($e.g.$, "topic proportion" in topic modeling), where h_{jk} indicates the weight of the prototype $\boldsymbol{\beta}_k$ for set j. Therefore, we can represent set D_j with another distribution Q_j on global prototypes $\boldsymbol{\beta}_{1:K}$, defined as

$$Q_j = \sum\nolimits_{k=1}^{K} h_{jk} \delta_{\boldsymbol{\beta}_k}, \boldsymbol{\beta}_k \in \mathbb{R}^d, \tag{4}$$

where $\boldsymbol{h}_j$ can be viewed as set representation for describing set j. Since set j can be represented as Q_j and P_j, we propose to learn set-specific representation $\boldsymbol{h}_j$ and K global prototypes $\mathbf{B}$ by pushing Q_j towards P_j:

$$\text{OT}(P_j, Q_j) = \min_{\mathbf{B}, \boldsymbol{h}_j} \langle \mathbf{T}, \mathbf{C} \rangle \overset{\text{def.}}{=} \sum_i^{N_j} \sum_k^{K} C_{ik} T_{ik}, \tag{5}$$

where $\mathbf{C} \in \mathbb{R}_{\geq 0}^{N_j \times K}$ is the transport cost matrix. In this paper, to measure the distance between data point $\boldsymbol{x}_{ji}$ in set j and prototype $\boldsymbol{\beta}_k$, unless specified otherwise, we specify the construction of $\mathbf{C}$ as $C_{ik} = 1 - \cos(\boldsymbol{x}_{ji}, \boldsymbol{\beta}_k)$, for providing an upper-bounded positive similarity metric. Besides, the transport probability matrix $\mathbf{T} \in \mathbb{R}_{>0}^{N_j \times K}$ should satisfy $\Pi(\boldsymbol{a}, \boldsymbol{b}) := \left\{\mathbf{T} \mid \mathbf{T}\mathbf{1}_K = \boldsymbol{a}, \mathbf{T}^\top \mathbf{1}_{N_j} = \boldsymbol{b}\right\}$ with $T_{ik} = T(\boldsymbol{x}_{ji}, \boldsymbol{\beta}_k)$, where $\boldsymbol{a} = [\frac{1}{N_j}] \in \Sigma^{N_j}$ and $\boldsymbol{b} = [h_{jk}] \in \Sigma^K$ denote the respective probability vectors for distribution P_j in Equation 3 and Q_j Equation 4.

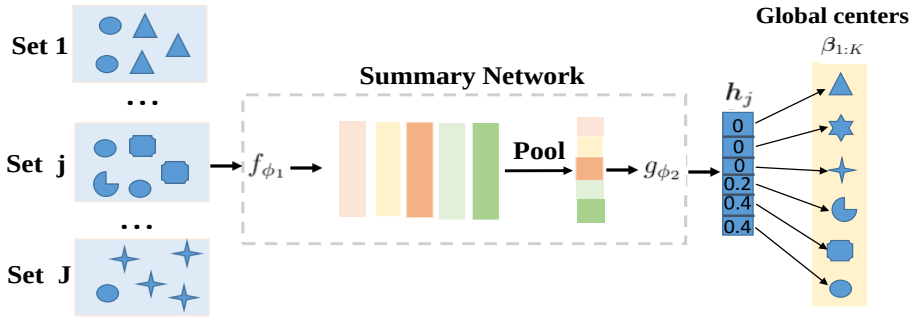


Figure 1: An overview of our proposed framework, where "pool" operation including mean, sum, max or similar. We compute the representation $\boldsymbol{h}_j$ for set j using the summary network in Equation 1, which is the weight vector of the global prototypes (i.e., centers) $\boldsymbol{\beta}_{1:K}$ in the corresponding set.

Since $\boldsymbol{h}_j$ should be invariant to the permutations of the samples in set j, we adopt summary network to encode the set of N_j points. For unsupervised tasks, taking the summary network in Equation 1 as the example, we can directly add a Softmax activation function into S_ϕ to enforce the simplex constraint in set representation $\boldsymbol{h}_j$, denoted as $\boldsymbol{h}_j = \mathrm{Softmax}(S_\phi(D_j))$. As shown in Fig. 1, given J sets, to learn the global prototype matrix $\mathbf{B}$ and the summary network parameterized by ϕ, we adopt the entropic constraint (Cuturi, 2013) and define the average OT loss for all training sets as

$$L_{\mathrm{OT}} = \min_{\mathbf{B},\phi} \frac{1}{\mathcal{J}} \sum_{j=1}^{\mathcal{J}} \left(\sum_{i}^{N_j} \sum_{k}^{K} C_{ik} T_{ik} - \epsilon \sum_{i}^{N_j} \sum_{k}^{K} -T_{ik} \ln T_{ik} \right) = \min_{\mathbf{B},\phi} \frac{1}{\mathcal{J}} \sum_{j=1}^{\mathcal{J}} \left(\mathrm{OT}_\epsilon(P_j, Q_j) \right), \quad (6)$$

where ϵ is a hyper-parameter for entropic constraint. Algorithm 1 describes the workflow of OT distance for improving summary network under unsupervised tasks. For supervised tasks, set j is denoted as $D_j = \{\boldsymbol{x}_{j,1:N_j}, \boldsymbol{y}_j\}$, where $\boldsymbol{y}_j$ is the ground-truth output determined by specific task. As $\boldsymbol{h}_j$ is a normalized weight vector, directly using it to realize the corresponding task may be undesired. Denoting $\boldsymbol{z}_j = \mathrm{pool}\left(\left\{f_{\phi_1}\left(\boldsymbol{x}_{j1}\right), \ldots, f_{\phi_1}\left(\boldsymbol{x}_{jN_j}\right)\right\}\right)$, we project it to the following vectors:

$$\boldsymbol{h}_j = f_e(\boldsymbol{z}_j), \quad \hat{\boldsymbol{y}}_j = f_\lambda(\boldsymbol{z}_j), \quad (7)$$

where $\boldsymbol{h}_j$ and $\hat{\boldsymbol{y}}_j$ are responsible for the OT and task-specific losses, respectively. Now the summary network parameters $\tilde{\phi} = \{e, \lambda, \phi_1\}$ and global prototypes $\mathbf{B}$ can be learned by jointly optimizing the task-specific loss (computed by $\hat{\boldsymbol{y}}_j$ and $\boldsymbol{y}_j$) and OT loss in Equation F. In summary, minimizing the OT distance between the prototype distribution Q_j and empirical distribution P_j provides a principled and unsupervised way to encourage the summary network to capture the set's summary statistics, where a suite of summary networks can be integrated into this plug-and-play framework. Therefore, it realizes efficient learning from new sets for both unsupervised and supervised tasks.

3.2 Application to Metric-based Few-shot Classification

As a challenging meta-learning problem, few-shot classification has recently attracted increasing attention, where one representative method is metric-based few-shot classification algorithms. Taking the ProtoNet (Snell et al., 2017) as the example, we provide a simple but effective method to improve its classification performance with the help of OT distance and summary network, where we refer the reader to ProtoNet for more details. In a specific, ProtoNet represents each class by computing an M-dimensional representation $\mathbf{c}_j \in \mathbb{R}^M$, with an embedding function $f_{\phi_1} : \mathbb{R}^d \to \mathbb{R}^M$, where we adopt the same ϕ_1 as the learnable parameters, following the feature extractor in summary network in Equation 1 for simplicity. Formally, ProtoNet adopts the average pooling to aggregate the embedded features of support points belonging to its class into vector $\mathbf{c}_j = \frac{1}{|S_j|} \sum_{(\mathbf{x}_{ji}, y_{ji}) \in S_j} f_{\phi_1}(\mathbf{x}_{ji})$. ProtoNet then compares the distance between a query point $f_{\phi_1}(\boldsymbol{x})$ to the $\mathbf{c}_j$ in the same embedding space. Motivated by the summary network, we further introduce a feed-forward network g_{ϕ_2} to map the $\mathbf{c}_j$ into the $\boldsymbol{h}_j$ used to define Q_j distribution over $\mathbf{B}$. Therefore, the functions g_{ϕ_2} and f_{ϕ_1} can be jointly optimized by minimizing the OT distance between P_j and Q_j and the original classification loss in ProtoNet, where $\mathbf{c}_j$ is still used to perform classification since $\boldsymbol{h}_j$ is a normalized weight vector. Only introducing matrix $\mathbf{B}$ and g_{ϕ_2}, whose parameters are usually negligible compared to the original feature extractor f_{ϕ_1}, our proposed method provides a simple but effective way to enforce

Algorithm 1 Workflow about minimizing the OT distance between P_j and Q_j.

Require: Datasets $\mathcal{D}_{1:J}$, batch size m, learning rate α, initial summary network parameters ϕ, initial global prototype matrix $\mathbf{B}$, cost function C and hyper-parameter ϵ.
while $\mathbf{B}, \phi$ has not converged **do**
  Randomly choose j from $1, 2, .., \mathcal{J}$
  Sample the real data $\{\boldsymbol{x}_{ji}\}_{i=1}^m$ from set j, which is denoted as empirical distribution P_j in Equation 3;
  Compute the set representation $\boldsymbol{h}_j = \mathrm{Softmax}(S_\phi(\{\boldsymbol{x}_{ji}\}_{i=1}^m))$ with summary network in Equation 1;
  Represent the Q_j with global prototype matrix $\mathbf{B}$ and statistics $\boldsymbol{h}_j$ in Equation 4;
  Compute the loss $\mathrm{OT}_\epsilon(P_j, Q_j)$ between P_j and Q_j with Sinkhorn algorithm in Equation F;
  $\mathbf{B} \leftarrow \mathbf{B} + \alpha g_{\mathbf{B}}$, where $g_{\mathbf{B}} \leftarrow \nabla_{\mathbf{B}} [\mathrm{OT}_\epsilon(P_j, Q_j)]$;
  $\phi \leftarrow \phi + \alpha g_\phi$, where $g_\phi \leftarrow \nabla_\phi [\mathrm{OT}_\epsilon(P_j, Q_j)]$;
end while

the f_{ϕ_1} to learn more powerful representation $\mathbf{c}_j$ of each class and feature $f_{\phi_1}(\boldsymbol{x})$ of query sample, benefiting the metric-based few-shot classification.

3.3 Application to Implicit Meta Generative models

Considering implicit meta generative modeling is still a challenging but important task in meta-learning, we further present how to construct the model by introducing set representation $\boldsymbol{h}_j$ as summary statistics, where we consider GAN-based implicit models. Specifically, given $p_j \sim p_{\mathcal{M}}$, we aim to construct a parametrized pushforward ($i.e.$, generator) of reference Gaussian distribution ρ, denoted as $T_\theta(\cdot, p_j)\sharp\rho$, to approximate the marginal distribution p_j, where θ summarizes the parameters of pushforward. Since it is unaccessible to the distribution of interest p_j, we replace p_j with P_j and use the summary network to encode set D_j into $\boldsymbol{h}_j$ as discussed above, which is further fed into the generator serving as the conditional information, denoted as $T_\theta(\boldsymbol{z}; \boldsymbol{h}_j), \boldsymbol{z} \sim \rho$. To enforce the pushforward $T_\theta(\boldsymbol{z}; \boldsymbol{h}_j)$ to fit the real distribution P_j as well as possible, we introduce a discriminator f_w following the standard GAN (Goodfellow et al., 2014). Generally, our GAN-based model consists of three components. Summary network $S_\phi(\cdot)$ focuses on learning the summary statistics $\boldsymbol{h}_j$ by minimizing the OT distance between P_j and Q_j. The pushforward aims to push the combination of a random noise vector $\boldsymbol{z}$ and statistics $\boldsymbol{h}_j$ to generate samples that resemble the ones from P_j, where we simply adopt a concatenation for $\boldsymbol{h}_j$ and $\boldsymbol{z}$ although other choices are also available. Besides, f_w tries to distinguish the "fake" samples from the "real" samples in set D_j. Therefore, we optimize the implicit meta generative model by defining the objective function as:

$$\min_{\mathbf{B},\phi,\theta} \max_{w} V(\mathbf{B}, \phi, \theta, w) = L_{\mathrm{OT}} + \frac{1}{\mathcal{J}} \sum_{j=1}^{\mathcal{J}} \mathbb{E}_{\boldsymbol{x}\sim P_j}[\log f_w(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim\rho}[\log(1 - f_w(T_\theta(\boldsymbol{z}; \boldsymbol{h}_j))] \quad (8)$$

In addition to the standard GAN loss, we can also adopt the Wasserstein GAN (WGAN) of Arjovsky et al. (2017) to approximate P_j. It is also flexibly to decide the input fed into f_w. For example, following the conditional GAN (CGAN) of Mirza & Osindero (2014), we can combine $\boldsymbol{h}_j$ and data points (generated or real), where the critic can be denoted as $f_w(\boldsymbol{x}, \boldsymbol{h}_j)$ and $f_w(T_\theta(\boldsymbol{z}; \boldsymbol{h}_j), \boldsymbol{h}_j)$, respectively. Since we focus on fitting the meta-distribution with the help of the summary network and OT loss, we leave the problem-specific design of generator, critic, and summary network as future work for considerable flexibility in architectures. We defer to Appendix A the illustration of our proposed model in Fig. 3 and detailed algorithm in Algorithm 2.

4 Related Work

Learning Summary Representation of Set-input. There are two lines for learning the set representation. The first line aims to design more powerful summary networks, which are reviewed in Introduction and Section 2.1 and omitted here due to the limited space. The another line assumes a/some to-be-learned reference set(s), and optimizes the distance between original sets (or features of the observed data points) and the reference set(s) with OT theory or other distance measures, to learn set representation. For example, RepSet (Skianis et al., 2020) computes some comparison costs between the input sets and some to-be-learned reference sets with a network flow algorithm, such as bipartite matching. These costs are then used as set representation in a subsequent neural network. However, unlike our framework, RepSet does not allow unsupervised learning and mainly focuses on classification tasks. The optimal Transport Kernel Embedding (OTKE) (Mialon et al.,

2021) marries ideas from OT theory and kernel methods (Schölkopf et al., 2002), and aligns features of a given set to a trainable reference distribution. Wasserstein Embedding for Graph Learning (WEGL) (Kolouri et al., 2021) also uses a similar idea to the linear Wasserstein embedding as a pooling operator for learning from sets of features. To our best knowledge, both of them view the reference distribution as the barycenter and compute the set-specific representation by aggregating the features (embedded with kernel methods) in a given set with adaptive weight, defined by the transport plan between the given set and the reference. Different from them, we assume J probability distributions (rather than one reference distribution with an uniform measure) over these shared prototypes by taking set representations $\boldsymbol{h}_{1:J}$ as the measures, to approximate the corresponding J empirical distributions, respectively. Then we naturally use the summary network as the encoder to compute $\boldsymbol{h}_{1:J}$, which can be jointly optimized with the shared prototypes by minimizing the general OT loss in an unsupervised way. For a given set, we can directly compute its representation with summary network, avoiding iteratively optimizing the transport plan between given set and learned reference like OTKE and WEGL. These differences between them and ours lead to different views of set representation learning and different frameworks as well. To learn compact representations for sequential data, Cherian & Aeron (2020) blend contrastive learning, adversarial learning, OT, and Riemannian geometry into one framework. However, our work directly minimises the OT distance between empirical distribution and the to-be-learned distribution, without contrastive learning and explicit generative processes from original set to the adversarial noise set.

Metric-based few-shot classification methods. Our method has a close connection with metric-based few-shot classification algorithms. For example, MatchingNet (Vinyals et al., 2016) and ProtoNet (Snell et al., 2017) learned to classify samples by computing distances to representatives of each class. Using an attention mechanism over a learned embedding of the support set to predict classes for the query set, MatchingNet (Vinyals et al., 2016) can be viewed as a weighted nearest-neighbor classifier applied within an embedding space. ProtoNet (Snell et al., 2017) takes a class's prototype to be the mean of its support set in the learned embedding space, which further performs classification for an embedded query point by finding the nearest class prototype. Importantly, the global prototypes in our paper are shared among all sets, which is different from the specific prototype for each class in ProtoNet but suitable for our case. Due to the flexibility of our method, we can project the average aggregated feature vector derived from the encoder in ProtoNet or MatchingNet, into $\boldsymbol{h}_j$ by introducing a simple neural network, which can be jointly optimized with the encoder by minimizing the classification loss and OT loss. Our novelty is that OT loss can be naturally used to improve the learning of encoder without changing the existing model architectures or algorithms a lot. Another recent work for learning multiple centres is infinite mixture prototypes (IMP) (Allen et al., 2019), which represents each class by a set of clusters and infers the number of clusters with Bayesian nonparametrics. However, in our work, the centers are shared for all classes and set-specific feature extracted from the summary network serves as the proportion of centers, where the centers and summary network can be jointly learned with OT loss.

Meta GAN-based Models. As discussed by Hong et al. (2020a), meta GAN-based models can be roughly divided into optimization-based, fusion-based, and transformation-based methods. Clouâtre & Demers (2019) and Liang et al. (2020) integrated GANs with meta-learning algorithms to realize the optimization-based methods, including model-agnostic meta-learning (MAML) (Finn et al., 2017) and Reptile (Nichol et al., 2018). Hong et al. (2020b;c) fused multiple conditional images by combining matching procedure with GANs, providing fusion-based methods. For transformation based methods, Antoniou et al. (2017) and Hong et al. (2020a) combine only one image and the random noise into the generator to produce a slightly different image from the same category, without using the multiple images from same category. Besides, a recent work that connects existing summary network with GAN is MetaGAN (Zhang et al. 2018), which feeds the output of the summary network into the generator and focuses on few-shot classification using MAML. The key difference of these models from ours is that we adopt the OT distance to optimize the summary network for capturing each sets' summary statistics, which are assimilated into the existing GAN-based models as conditional information. In this general framework, the summary network, generator, and discriminator can be flexibly chosen for specific tasks.

5 Experiments

We conduct extensive experiments to evaluate the performance of our proposed prototype-oriented framework for meta-learning. We show that our method outperforms state-of-art baselines, including

improving summary networks, few-shot generation, and few-shot classification. Unless specified otherwise, we set the hyper-parameter of entropic constraint as $\epsilon = 0.1$, the maximum iteration number in Sinkhorn algorithm as Itermax $= 200$ for all experiments and adopt the Adam optimizer (Kingma & Ba, 2015) with learning rate 0.001. We repeat all experiments 5 times and report the mean and standard deviation on corresponding test datasets.

5.1 Experiments about OT distance in Summary Network

To evaluate the effectiveness of OT loss in improving the summary network designed for set-structured input, we conduct three tasks on two classical architectures: DeepSets (Zaheer et al. 2017) and Set Transformer (Lee et al. 2019), where the former uses standard feed-forward neural networks and the latter adopts the attention-based network architecture. For Set Transformer and DeepSets, the summary network is defined in Equation 1 and optimized by the task-specific loss; for Set Transformer(+OT) and DeepSets(+OT), the summary network, defined as in Equations 1 and 7, is optimized with matrix $\mathbf{B}$ by both the OT loss and task-specific loss. We provide more details about toy datasets, network architectures, and training schemes in Appendix B.

Amortized Clustering with Mixture of Gaussians (MoGs): We consider the task of maximum likelihood of MoGs with C components, denoted as $P(x;\boldsymbol{\theta}) = \sum_{c=1}^{C} \pi_c N\left(x \mid \mu_c, \operatorname{diag}\left(\sigma_c^2\right)\right)$. Given the dataset $X = \{x_{1:n}\}$ generated from the MoG, the goal is to train a neural network, which takes X as input set and outputs parameters $\boldsymbol{\theta}$. Each dataset contains $n \in [100, 500]$ points on a 2D plane, each of which is sampled from one of C Gaussians. Table 1 reports the test average likelihood of different models with varying $C \in \{4, 8\}$, where we set $K = 50$ for all C. We observe that Set Transformer outperforms DeepSets largely, validating the effectiveness of attention mechanisms in this task. Besides, both Set Transformer(+OT) and DeepSets(+OT) improve their baselines, showing that the OT loss can encourage the summary networks to learn more efficient summary statistics.

Point Cloud Classification: Here, we evaluate our method on the task of point cloud classification using the ModelNet40 (Chang et al., 2015) dataset [1], which consists of 3D objects from 40 different categories. By treating each object as a point cloud, we represent it as a set of N vectors in $\mathbb{R}^3$ (x; y; z-coordinates). Table 1 reports the classification accuracy, where we perform the experiments with varying $N \in \{64, 1024\}$, set $K = 40$. Clearly, both of the DeepSets and Set Transformer can be improved by adding the OT loss. Notably, fewer points would lead to lower performance, where the OT loss plays a more important role. Taking this task as the example, we further study our model's sensitivity to the hyper-parameter ϵ in Fig. 4 of the Appendix B.5.

Sum of Digits: Following Zaheer et al. (2017), we aim to compute the sum of a given set of digits, where we consider MNIST8m (Loosli et al., 2007) containing 8 million instances of 28×28 grey-scale stamps of digits in $\{0, ..., 9\}$. By randomly sampling a subset of maximum $M = 10$ images from MNIST8m, we build $N = 100k$ "sets" of training, where we denote the sum of digits in that set as the set-label. We construct $100k$ sets of test MNIST digits, where we vary the M starting from 10 all the way up to 100. The output of the summary network is a scalar, predicting the sum of M digits. In this case, we adopt L1 as the task-specific loss and set $K = 10$. We show the accuracy of digit summation for different algorithms in Fig. 2 and find that the OT loss can enhance the summary networks to achieve better generalization. In this task, we also explore the convergence rate and the learned transport plan matrix of Sinkhorn algorithm in Fig. 5 of the Appendix B.6.

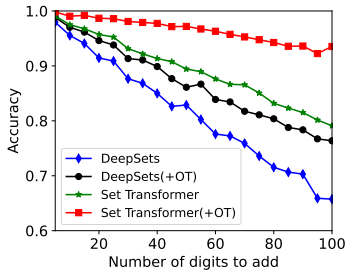


Figure 2: Accuracy of digit summation with image inputs, where all models are trained on tasks of length 10 at most and tested on examples of length up to 100.

5.2 Experiments on Few-shot Classification

To explore whether our proposed method can improve the metric-based few-shot classification, we consider two commonly-used algorithms as the baselines, including ProtoNet (Snell et al., 2017) and MatchNet (Vinyals et al., 2016). Denoting the feature extractor in each algorithm as f_{ϕ_1}, we

[1] We adopt the point-cloud dataset directly from the authors of Zaheer et al. 2017

Table 1: Test performance of different methods, where left table denotes the likelihood for MoG with varying C (number of components), oracle is the likelihood of true parameters for the test data, and right denotes the test accuracy for point cloud classification task with varying N (number of points).

Task	**Test likelihood for MoG**					**Test accuracy for the point cloud classification**				
Algorithm	**C=4**	**C=5**	**C=6**	**C=7**	**C=8**	**N=64**	**N=128**	**N=256**	**N=512**	**N=1024**
Oracle	-1.473	-1.660	-1.820	-1.946	-2.058	-	-	-	-	-
DeepSets	-1.809 ±0.015	-1.812 ±0.016	-1.897 ±0.017	-2.115 ±0.016	-2.261 ±0.014	79.14 ±0.035	82.51 ±0.028	84.62 ±0.037	85.74 ±0.045	86.83 ±0.042
DeepSets(+OT)	**-1.723** ±0.015	**-1.743** ±0.017	**-1.861** ±0.012	**-2.078** ±0.018	**-2.214** ±0.015	**79.91** ±0.050	**83.65** ±0.060	**85.22** ±0.055	**86.25** ±0.067	**86.93** ±0.075
Set Transformer	-1.501 ±0.006	-1.721 ±0.006	-1.859 ±0.007	-2.003 ±0.007	-2.106 ±0.007	79.01 ±0.103	82.31 ±0.117	84.46 ±0.125	85.82 ±0.114	86.34 ±0.122
Set Transformer(+OT)	**-1.486** ± 0.007	**-1.676** ± 0.007	**-1.828** ± 0.006	**-1.967** ±0.007	**-2.084** ±0.007	**80.00** ±0.111	**83.32** ±0.130	**85.64** ±0.121	**86.51** ±0.124	**86.84** ±0.115

consider several popular backbones, including ResNet10 and ResNet34 (He et al., 2016). Recalling the discussions in section 3.2, to enforce f_{ϕ_1} to learn more powerful image features, we additionally introduce matrix $\mathbf{B}$ and net g_{ϕ_2} and learn the model by minimizing the OT distance and classification errors. We perform the experiments on the CUB (Welinder et al., 2010) and miniImageNet (Ravi & Larochelle, 2016). As a fine-grained few-shot classification benchmark, CUB contains 200 different classes of birds with a total of $11,788$ images of size $84 \times 84 \times 3$, where we split the dataset into 100 base classes, 50 validation classes, and 50 novel classes following Chen et al. (2019). miniImageNet is derived from ILSVRC-12 dataset (Russakovsky et al., 2015), consisting of $84 \times 84 \times 3$ images from 100 classes with 600 random samples in each class. We follow the splits used in previous work (Ravi & Larochelle, 2016), which splits the dataset into 64 base classes, 16 validation classes, and 20 novel classes. Table 2 reports the 5way5shot and 5way10shot classification results of different methods on miniImageNet and CUB. We see that introducing the OT distance and summary network can consistently improve over baseline classifiers, and the performance gain gradually increases with the development of number of network layers. This suggests that our proposed plug-and-play framework can be flexibly used to enhance the metric-based few-shot classification, without the requirement of designing complicated models on purpose.

Table 2: 5way5shot and 5way10shot classification accuracy (%) on CUB and miniImageNet, respectively. Here, p-value indicates $p < 0.05$, computed with two-sample T-test, and p-value with blue (red) color means the increase (reduce) of performance when introducing OT loss.

Datasets	**CUB**		**miniImageNet**	
ProtoNet(resnet10)	84.32 ± 0.51	87.41 ± 0.49	72.74±0.63	78.14 ± 0.56
ProtoNet(+OT) (resnet10)	**84.44 ±0.51** (p-value)	**87.69 ±0.53** (p-value)	**72.94 ±0.66** (p-value)	**78.76 ± 0.49** (p-value)
ProtoNet(resnet34)	87.33 ± 0.48	91.75 ± 0.47	73.99 ± 0.64	78.64 ± 0.56
ProtoNet(+OT) (resnet34)	**88.34± 0.46** (p-value)	**92.17 ± 0.48** (p-value)	**75.15± 0.63**(p-value)	**79.05 ± 0.52** (p-value)
MatchNet (resnet10)	82.98± 0.56	85.97± 0.53	68.82 ± 0.65	**72.06 ± 0.54**
MatchNet(+OT) (resnet10)	**83.64 ± 0.58** (p-value)	**86.02± 0.56** (p-value)	**68.95 ± 0.62**(p-value)	71.94 ± 0.56 (p-value)
MatchNet (resnet34)	84.66± 0.55	86.32 ± 0.56	68.32 ± 0.66	**72.41 ± 0.63**
MatchNet(+OT) (resnet34)	**85.50 ± 0.66** (p-value)	**86.75 ± 0.61** (p-value)	**68.51 ± 0.64** (p-value)	71.98 ± 0.59 (p-value)

5.3 Experiments on Few-shot Generation

Here, we consider few-shot generation task to investigate the effectiveness of our proposed implicit meta generative framework, where we consider CGAN (Mirza & Osindero, 2014) and DAGAN (Antoniou et al., 2017) as baselines for their ability to generate conditional samples. For CGAN-based models, we adopt summary network as the encoder to extract the feature vector $\boldsymbol{h}_j$ from a given set D_j, which is further fed into the generator and discriminator as the conditional information. Since the original DAGAN only assimilates one image into the generator, in our framework, we replace the encoder in DAGAN with summary network to learn the set representation. For DAGAN-based models, we adopt the same way with the original DAGAN to construct the real/samples for the critic and explain here for clarity: we sample two sets (D_{1j}, D_{2j}) from the same distribution or category; then we represent the real samples as the combination of D_{1j} and D_{2j} and the fake ones as the combination of D_{1j} and $\hat{D}_{1j}$ from the generator (conditioned on D_{1j}). Different from our framework that separately optimizes the summary network using the OT loss, all other models for comparison in this paper jointly optimize the encoder (e.g., summary network) with the generator by the generator loss. Besides, for a fair comparison, we also consider introducing the additional reconstruction loss (mean square error, MSE) to optimize the generator and encoder in baselines. We consider the DeepSets as the summary network for its simple architecture. For 3D natural images,

we adopt the pretrained densenet (Iandola et al., 2014) to extract features from each data pints, and take the features as the input to summary network. To evaluate the quality of generated samples, we adopt commonly used metric Fréchet Inception Distance (FID) (Heusel et al., 2017), where we only report the FID score (Heusel et al., 2017) considering the notable performance gap between our model and the compared ones. We provide several examples on toy datasets to show the efficiency of our proposed model in Appendix D.

Table 3: FID ↓ of images generated by different methods with varying unseen angles A on MNIST.

Algorithms	**A=-160**	**A=-120**	**A=-80**	**A=-40**	**A=0**	**A=40**	**A=80**	**A=120**	**A=160**
CGAN	227.94 ± 2.56	203.59 ±2.21	211.74 ±1.88	231.63 ± 2.11	228.35 ±1.94	222.87 ± 2.02	195.30 ±1.58	202.69 ±1.75	202.35 ±1.16
CGAN+MSE	225.56 ± 2.05	201.09 ± 1.62	209.11 ±0.91	222.54 ± 2.13	223.61 ±1.55	220.18 ±1.47	193.75 ±1.39	200.12 ±1.08	201.13 ± 2.33
CGAN+OT	**213.30** ± 1.35	**193.42** ± 1.25	**196.56** ± 1.61	**217.28** ± 1.58	**210.27** ± 2.04	**206.44** ±1.78	**181.05** ± 0.99	**190.19** ±1.22	**191.22** ±1.37
DAGAN	169.85 ± 1.56	201.58 ±1.98	157.95 ±1.45	204.23 ±2.31	218.42 ±1.96	196.70 ±1.75	160.35 ±1.57	198.24 ±2.14	164.64 ±1.85
DAGAN+MSE	169.77 ±1.51	200.12 ±2.10	155.87 ±1.41	202.16 ±1.89	215.12 ±1.97	194.61 ±2.10	159.08 ±1.74	197.48 ±2.05	163.23 ±1.84
DAGAN+ OT	**159.51** ± 1.57	**174.07** ±1.63	**135.89** ±1.17	**177.64** ±1.88	**197.09** ±1.46	**186.17** ±1.90	**142.40** ±1.33	**174.75** ±1.59	**146.69** ±1.47

Rotated MNIST: Following Wu et al. (2020), we artificially transform each image in MNIST dataset (LeCun, 1998) with 18 rotations (-180 to 180 by 20 degrees), leading to 18 distributions characterized by angle A. We choose 9 interleaved distributions for training and the rest as unseen distributions for testing. We consider the CGAN-based and DAGAN-based models, respectively. During the test stage, for each unseen distribution, we randomly sample 1000 real images and generate 20 fake images based on every 20 real images and repeat this process 50 times (i.e., $1000/20$), resulting in 1000 generated samples for each method. We summarize the test performance in Table 3 with varying A. We can find that our proposed framework allows for better generalization to related but unseen distributions at test time, indicating the OT loss can enforce the summary network to capture more salient characteristics.

Natural Images: We further consider few-shot image generation on Flowers (Nilsback & Zisserman, 2008) and Animal Faces (Deng et al., 2009), where we follow seen/unseen split provided in Liu et al. (2019). Flowers dataset contains 8189 images of 102 categories, which are divided into 85 training seen and 17 testing unseen categories; Animal Faces dataset contains $117,574$ animal faces collected from 149 carnivorous animal categories, which are split into 119 training seen and 30 testing unseen categories. We present the example images generated by DAGAN and DAGAN(+OT) and network architectures in Appendix E for the limited space, where we also compute the FID scores with the similar way in Rotated MNIST experiment. We find that our method achieves the lowest FID and has the ability to generate more realistic natural images compared with baselines. This indicates the summary network in our proposed framework can successfully capture the important summary statistics within the set, beneficial for the few-shot image generation.

6 Conclusion

In this paper, we present a novel method to improve existing summary networks designed for set-structured input based on optimal transport, where a set is endowed with two distributions: one is the empirical distribution over the data points, and another is the distribution over the learnable global prototypes. Moreover, we use the summary network to encode input set as the prototype proportion (i.e., set representation) for global centers in corresponding set. To learn the distribution over global prototypes and summary network, we minimize the OT distance between two distributions in terms of the defined cost function. Only additionally introducing the acceptable parameters, our proposed model provides a natural and unsupervised way to improve the summary network. In addition to the set-input problems, our plug and play framework has shown appealing properties that can be applied to many meta-learning tasks, where we consider the cases of metric-based few-shot classification and implicit meta generative modeling. Extensive experiments have been conducted, showing that our proposed framework achieves state-of-the-art performance on both improving existing summary networks and meta-learning models for set-input problems. Due to the flexibility and simplicity of our proposed framework, there are still some exciting extensions. For example, an interesting future work would be to apply our method into approximate Bayesian computation for posterior inference.

REFERENCES

Kelsey Allen, Evan Shelhamer, Hanul Shin, and Joshua Tenenbaum. Infinite mixture prototypes for few-shot learning. In *International Conference on Machine Learning*, pp. 232–241. PMLR, 2019.

Jason M. Altschuler and Enric Boix-Adserà. Wasserstein barycenters can be computed in polynomial time in fixed dimension. *J. Mach. Learn. Res.*, 22:44:1–44:19, 2021.

Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Yanzhi Chen, Dinghuai Zhang, Michael U. Gutmann, Aaron C. Courville, and Zhanxing Zhu. Neural approximate sufficient statistics for implicit models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Anoop Cherian and Shuchin Aeron. Representation learning via adversarially-contrastive optimal transport. In *International Conference on Machine Learning*, volume 119, pp. 1820–1830, 2020.

Louis Clouâtre and Marc Demers. Figr: Few-shot image generation with reptile. *arXiv preprint arXiv:1901.02199*, 2019.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300, 2013.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pp. 248–255. IEEE Computer Society, 2009.

Harrison Edwards and Amos J. Storkey. Towards a neural statistician. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

SM Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Geoffrey E Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. *Advances in Neural Information Processing Systems*, 29:3225–3233, 2016.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. volume 27, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Yan Hong, Li Niu, Jianfu Zhang, Jing Liang, and Liqing Zhang. Deltagan: Towards diverse few-shot image generation with sample-specific delta. *arXiv preprint arXiv:2009.08753*, 2020a.

Yan Hong, Li Niu, Jianfu Zhang, and Liqing Zhang. Matchinggan: Matching-based few-shot image generation. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6. IEEE, 2020b.

Yan Hong, Li Niu, Jianfu Zhang, Weijie Zhao, Chen Fu, and Liqing Zhang. F2gan: Fusing-and-filling gan for few-shot image generation. In *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 2535–2543, 2020c.

Gao Huang, Shichen Liu, Laurens Van der Maaten, and Kilian Q Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2752–2761, 2018.

Forrest Iandola, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014.

Mateusz Jurewicz and Leon Strømberg-Derczynski. Set-to-sequence methods in machine learning: a review. *arXiv preprint arXiv:2103.09656*, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015.

Soheil Kolouri, Navid Naderializadeh, Gustavo K. Rohde, and Heiko Hoffmann. Wasserstein embedding for graph learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

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 *International Conference on Machine Learning*, pp. 3744–3753. PMLR, 2019.

Weixin Liang, Zixuan Liu, and Can Liu. Dawson: A domain adaptive few shot generation framework. *arXiv preprint arXiv:2001.00576*, 2020.

Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 10550–10559. IEEE, 2019.

Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training invariant support vector machines using selective sampling. *Large scale kernel machines*, 2, 2007.

Haggai Maron, Or Litany, Gal Chechik, and Ethan Fetaya. On learning sets of symmetric elements. In *International Conference on Machine Learning*, pp. 6734–6744. PMLR, 2020.

Grégoire Mialon, Dexiong Chen, Alexandre d'Aspremont, and Julien Mairal. A trainable optimal transport embedding for feature aggregation and its relationship to attention. In *ICLR 2021-The Ninth International Conference on Learning Representations*, 2021.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Sixth Indian Conference on Computer Vision, Graphics & Image Processing, ICVGIP 2008, Bhubaneswar, India, 16-19 December 2008*, pp. 722–729. IEEE Computer Society, 2008.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Found. Trends Mach. Learn.*, 11 (5-6):355–607, 2019.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015.

Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

Konstantinos Skianis, Giannis Nikolentzos, Stratis Limnios, and Michalis Vazirgiannis. Rep the set: Neural networks for learning set representations. In *International conference on artificial intelligence and statistics*, pp. 1410–1420. PMLR, 2020.

Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4077–4087, 2017.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3630–3638, 2016.

Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 2010.

Mike Wu, Kristy Choi, Noah Goodman, and Stefano Ermon. Meta-amortized variational inference and learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6404–6412, 2020.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. In *NeurIPS*, pp. 3391–3401, 2017.

Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. Metagan: An adversarial approach to few-shot learning. *NeurIPS*, 2:8, 2018.

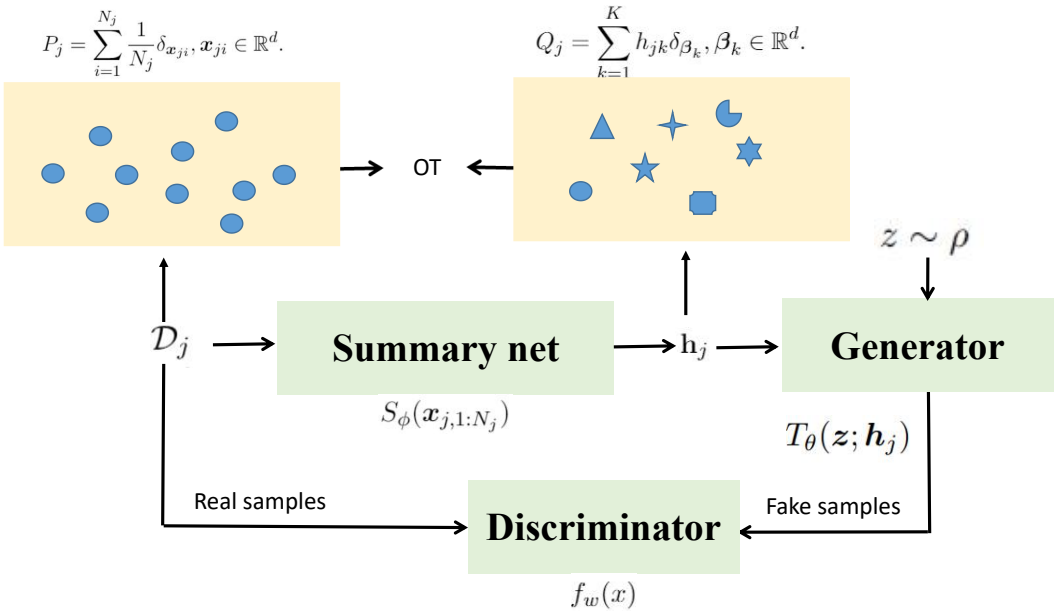


Figure 3: The overview of our proposed implicit meta generative framework, where we sample the j-th distribution from training sets, feed the data points into the summary network, generate the fake samples with the random noise and summary output as the input, discriminate the real/fake samples.

Algorithm 2 The workflow of our proposed implicit meta generative framework.

Require: Datasets $\mathcal{D}_{1:J}$, initial discriminator parameters w, initial generator parameters θ, initial summary network parameters ϕ, initial matrix $\mathbf{B}$, the cost function C, the number of critic iterations per generator iteration η_{critic}, the batch size m, learning rate α and the hyper-parameter ϵ.

while w, θ,$\mathbf{B}$, ϕ has not converged **do**
  Randomly choose j from $1, 2, .., \mathcal{J}$
  for $t = 1, \cdots, \eta_{critic}$ **do**
    Sample the real data set $D_j = \{\boldsymbol{x}_{ji}\}_{i=1}^m$ from j-th empirical distribution P_j;
    Embed the observed batch data into the statistics $\boldsymbol{h}_j = \mathrm{Softmax}(S_\phi(D_j))$;
    Sample a batch of prior samples $\{z_i\}_{i=1}^m$ from $p(z)$;
    $g_w \leftarrow -\nabla_w \frac{1}{m} \sum_{i=1}^m [\log f_w(x_j^i) + \log(1 - f_w(T_\theta(z_i, \boldsymbol{h}_j)))]$;
    $w \leftarrow w + \alpha g_w$
  end for;
  Represent the Q_j with global prototype matrix $\mathbf{B}$ and set representation/statistics $\boldsymbol{h}_j$
  Compute the loss $\mathrm{OT}_\epsilon(P_j, Q_j)$ between P_j and Q_j with Sinkhorn algorithm in Equation F
  $g_{\mathbf{B}} \leftarrow \nabla_{\mathbf{B}} [\mathrm{OT}_\epsilon(P_j, Q_j)]$;
  $\mathbf{B} \leftarrow \mathbf{B} + \alpha g_{\mathbf{B}}$;
  $g_\phi \leftarrow \nabla_\phi [\mathrm{OT}_\epsilon(P_j, Q_j)]$;
  $\phi \leftarrow \phi + \alpha g_\phi$; see Algorithm 1 for more details;
  Sample a batch of latent variables $\left\{z^{(i)}\right\}_{i=1}^m \sim p(z)$;
  $g_\theta \leftarrow \nabla_\theta \left[\frac{1}{m} \sum_{i=1}^m \log\left(1 - f_w\left(T_\theta(z_i, \boldsymbol{h}_j)\right)\right)\right]$;
  $\theta \leftarrow \theta + \alpha g_\theta$;
end while

A ALGORITHMS AND ILLUSTRATION OF OUR PROPOSED MODEL

The pseudo code for the implicit meta generative modeling is provided in Algorithm 2.

B EXPERIMENTAL SETTINGS ABOUT INTRODUCING OT DISTANCE INTO THE SUMMARY NETWORKS

B.1 DETAILS FOR AMORTIZED CLUSTERING WITH MIXTURES OF GAUSSIANS

We generate the 2D toy datasets following the Lee et al. (2019), where we additionally vary the C (the number of components) from 4 to 8. Below, we present the detailed generation process about the toy datasets:

1. Specify the number of components C for 2D toy dataset.
2. Generate the number of data points, $n \sim \mathrm{Uniform}(100, 500)$.
3. Sample the mean vector for C components.

$$\mu_{c,d} \sim \mathrm{Uniform}(-4, 4), \quad c = 1, \ldots, C, \quad d = 1, 2$$

4. Sample the cluster labels.

$$\pi \sim \mathrm{Dir}\left(\mathbf{1}^{\top}\right), \quad z_i \sim \mathrm{Categorical}(\pi), i = 1, \ldots, n, z_i = 1, \ldots, C$$

5. Generate data from spherical Gaussian.

$$x_i \sim \mathcal{N}\left(\mu_{z_i}, (0.3)^2 I\right), i = 1, \ldots, n$$

B.2 DETAILS ABOUT SET-TRANSFORMER-BASED AND DEEPSETS-BASED ARCHITECTURES USED IN MOGS EXPERIMENTS

DeepSets In terms of the DeepSets, the f_{ϕ_1} in summary network contains 3 permutation-equivariant layers with 256 channels followed by mean-pooling over the set structure. Then the resulting vector representation $\boldsymbol{z}_j$ of the set is then fed to a fully connected layer with 512 units followed by a linear layer $512 \times C(1 + 2 * 2)$, where C denotes the number of components. We use ELU activation at all layers. To introduce the OT loss into the DeepSets, we further feed the $\boldsymbol{z}_j$ into a fully connected layer with 512 units followed by a 50-way softmax unit and also introduce the global matrix $\mathbf{B} \in \mathbb{R}^{2\times 50}$.
Set Transformer To perform the MoGs experiments, we adopt the same architecture for Set Transformer following Lee et al. (2019), whose parameters are reported in Table 4. To introduce the OT loss, we also add the two fully connected layers with 256 units on the resulting vector $\boldsymbol{z}_j$ followed by a 50-way softmax unit, and a global prototype matrix $\mathbf{B} \in \mathbb{R}^{2\times 50}$.

B.3 DETAILS ABOUT SET-TRANSFORMER-BASED AND DEEPSETS-BASED ARCHITECTURES USED IN SUM OF DIGITS

DeepSets Following the official code in Zaheer et al. (2017), we adopt the default architecture to implement the DeepSets, where we first project the image into a 128-dimensional vector with three convolutional layers and apply summary network on the 128-dimensional vectors. To build DeepSets(+OT), we take the set representation $\boldsymbol{z}_j$ after sum-pooling in summary network as the input and introduce fully connected layer with 128 units followed by a 10-way softmax unit, and a global prototype matrix $\mathbf{B} \in \mathbb{R}^{128\times 10}$.
Set Transformer For Set Transformer, we follow the similar structure used in MoGs experiments, where we also project the image with three convolutional layers and output a scalar. To build Set Transformer(+OT), we take the representation $\boldsymbol{z}_j$ after sum-pooling in summary network as the input and introduce fully connected layer with 128 units followed by a 10-way softmax unit, and a center matrix $\mathbf{B} \in \mathbb{R}^{128\times 10}$.

B.4 DETAILS ABOUT SET-TRANSFORMER-BASED AND DEEPSETS-BASED ARCHITECTURES USED IN POINT CLOUD CLASSIFICATION

DeepSets For original DeepSets, we adopt the same architecture with Zaheer et al. (2017). In a specific, the f_{ϕ_1} in summary network contains 3 permutation-equivariant layers with 256 channels followed by max-pooling over the set structure. Then the resulting vector representation $\boldsymbol{z}_j$ of the set is then fed to a fully connected layer with 256 units followed by a 40-way softmax unit. We

Table 4: Detailed architectures of Set Transformer used in the MoGs experiments, cited from Lee et al. (2019), where C denotes the number of components.

Encoder			Decoder	
rFF	**SAB**	**ISAB**	**Pooling**	**PMA**
FC(128, ReLU)	SAB(128, 4)	ISABm(128, 4)	mean	PMA4(128, 4)
FC(128, ReLU)	SAB(128, 4)	ISABm(128, 4)	FC(128, ReLU)	SAB(128, 4)
FC(128, ReLU)	-	-	FC(128, ReLU)	FC(C · (1 + 2 · 2),)
FC(128, ReLU)	-	-	FC(128, ReLU)	FC(C · (1 + 2 · 2),)
-	-	-	FC(C · (1 + 2 · 2),)	-

Table 5: Detailed architectures of Set Transformer used in the point cloud classification experiments, cited from Lee et al. (2019).

Encoder		Decoder	
rFF	**ISAB**	**Pooling**	**PMA**
FC(256, ReLU)	ISAB(256, 4)	max	Dropout(0.5)
FC(256, ReLU)	ISAB(256, 4)	Dropout(0.5)	PMA1(256, 4)
FC(256, ReLU)	-	FC(256, ReLU)	Dropout(0.5)
FC(256,)	-	Dropout(0.5)	FC(40,)
-	-	FC(40,)	-

use Tanh activation at all layers and dropout on the layers after set-max-pooling (i.e. two dropout operations) with 50% dropout rate. To introduce the OT loss into the DeepSets, we further feed the $\boldsymbol{z}_j$ into a fully connected layer with 256 units followed by a 40-way softmax unit, with 70% dropout rate. Besides, we additionally introduce the center matrix $\mathbf{B} \in \mathbb{R}^{3\times 40}$.

Set Transformer We also adopt the same architecture to implement the Set Transformer, where we summarize the parameters in Table 5, following Lee et al. (2019). To improve the Set Transformer with OT distance, we also introduce a fully connected layer with 256 units followed by a 40-way softmax unit, with 90% dropout rate, and a center matrix $\mathbf{B} \in \mathbb{R}^{3\times 40}$.

B.5 Parameter sensitivity

In the previous experiments, we fix the value of ϵ as 0.1, controlling the weight of the entropic regularisation in the Sinkhorn algorithm. Notably, unless specified otherwise, we specify the construction of $\mathbf{C}$ as $C_{ik} = 1 - \cos\left(\boldsymbol{x}_{ji}, \boldsymbol{\beta}_k\right)$. Therefore, the cost function provides an upper-bounded positive similarity metric, making the ϵ has the corresponding reasonable range as a prior knowledge. Here, we study our DeepSets(+OT)'s sensitivity to ϵ. We consider the point cloud classification task and each object is represented as a set of $N = 20$ vectors. As shown in Fig. 4, we report the performance of DeepSets(+OT) on point cloud classification task with varying ϵ, where DeepSets serves as the baseline. It can be seen that our model is robust to the ϵ. Besides, all the results of DeeepSets(+OT) with different ϵ are superior than that of DeeepSets, indicating the effectiveness of our method. By fine-tuning ϵ for each dataset in each task, we might obtain better results than those reported in our experiments. However, we aim to validate our method instead of exhaustively tuning this hyper-parameter and thus we set $\epsilon = 0.1$, which can achieve the acceptable result.

B.6 Convergence rate of Sinkhorn algorithm

In this paper, we set the maximum iteration number as $\text{Itermax} = 200$ in Sinkhorn algorithm for all experiments. As shown in Fig. 5, we visualize the convergence rate of Sinkhorn algorithm, where we consider the task about "sum of digits" (DeepSets+OT). The upper figure shows the convergence rate of Sinkhorn algorithm. The bottom figure visualizes the transport plan matrix with varying iterations. We find that the 200 iterations are typically enough for Sinkhorn algorithm and we can learn a sparse transport plan matrix $\mathbf{T}$ when the algorithm converge. Notably, the transport plan matrix needs to satisfy two marginal constraints, defined by the probability measures of two distributions, respectively. Recall that the empirical distribution has an unchanged uniform probability measure, so the learned transport plan matrix is dense for the N_j observed samples. In terms of another dis-

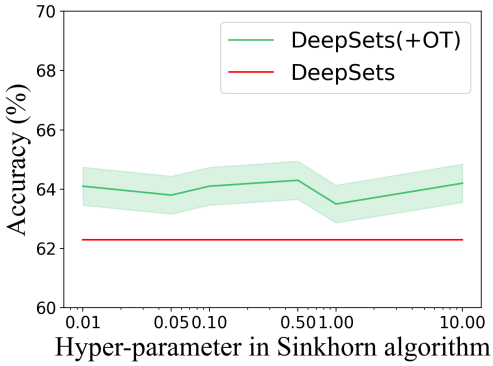


Figure 4: Parameter sensitivity of DeepSets(+OT) on point cloud classification task, with varying ϵ, where each object is represented as a set of $N = 20$ vectors.

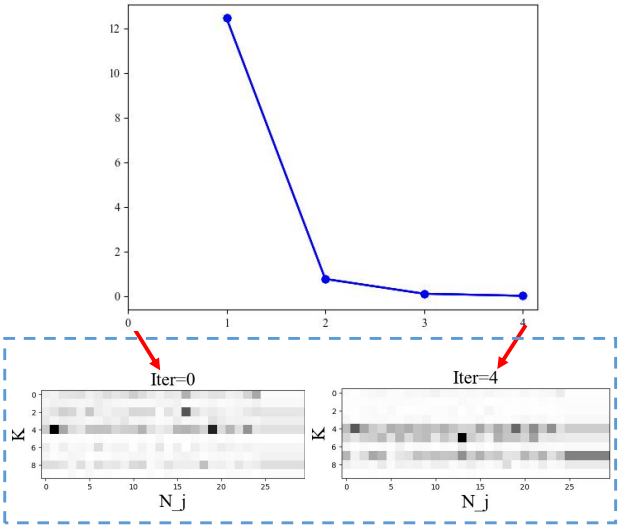


Figure 5: Top: the convergence rate of Sinkhorn for $c(x, y) = 1 - cosine(x, y)$, and $\epsilon = 0.1$, as measured in term of marginal constraint violation $\frac{1}{J}\sum_j^J \sum_i^{N_j} |u_{ij}^{l+1} - u_{ij}^l|$, where l is the iteration index and $\boldsymbol{u}$ is the scaling variable; please see page 67 in Peyré & Cuturi (2019) for more details. Bottom: evolution of the transport plan matrix $\mathbf{T} = \text{diag}\left(\boldsymbol{u}^{(\ell)}\right) \mathbf{K}\, \text{diag}\left(\boldsymbol{v}^{(\ell)}\right)$ computed at iteration of Sinkhorn's iterations.

tribution, its probability measure is the set-specific representation, weighting the importance of K shared centers for corresponding set. Therefore, it is reasonable that transport plan matrix is sparse for K centers.

C EXPERIMENTAL SETTINGS ABOUT FEW-SHOT CLASSIFICATION

Denote the prototype for set j (computed by f_{ϕ_1}) in few-shot classification as $\boldsymbol{c}_j$. We consider two backbones for f_{ϕ_1}, including ResNet10 and ResNet34, which produce the 512-dimensional $\boldsymbol{c}_j$. To improve the metric-based few-shot classification with our framework, taking the $\boldsymbol{c}_j$ as input, we further construct the g_{ϕ_2}. Specifically, we introduce a fully connected network with architecture as $512 \rightarrow 256$ units with ReLU function followed by a **X**-way (**X**=64 for CUB, and **X**=128 for miniImageNet) Softmax function and a center matrix $\mathbf{B} \in \mathbb{R}^{512\times\mathbf{X}}$. We conduct 10000 tasks of the training set $\mathcal{D}_{tr}$ to train the model while 1000 tasks of the test set $\mathcal{D}_{te}$ to evaluate the learned model. And $\mathcal{D}_{tr} \cap \mathcal{D}_{te} = \emptyset$. We run 60 epochs to train the model on CUB and miniImageNet. The model is trained using Adam optimizer with default settings (learning rate $1e-3$, $\beta = (0.9, 0.999)$, and $\epsilon = 1e-8$) on one Nvidia Geforce RTX3090 GPU.

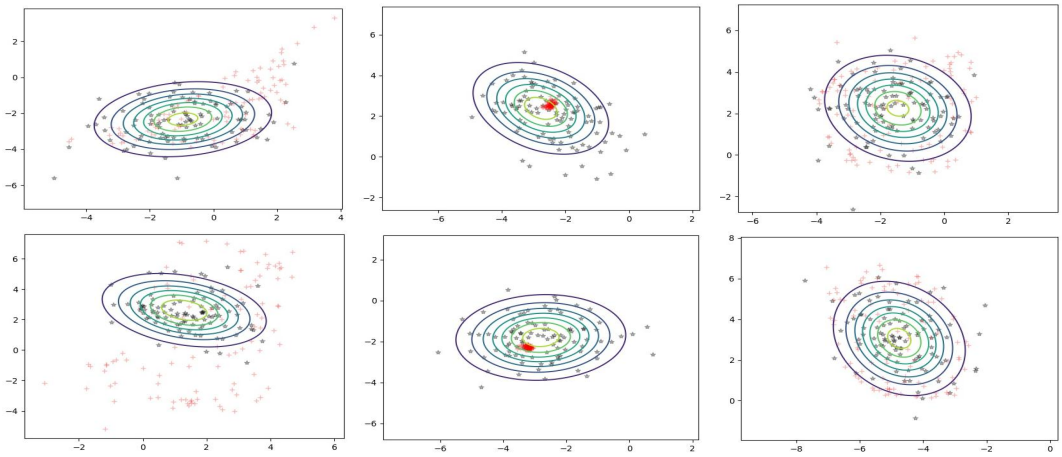


Figure 6: Visualization of real samples from the true distributions (gray points) and generated samples (red points) by different models (from first to third column: CGAN, CGAN+MSE and our proposed CGAN+OT), where we also plot the contour of each Gaussian distribution.

D ADDITIONAL EXPERIMENTAL RESULTS ON FEW-SHOT GENERATION ABOUT TOY DATASETS

We test our algorithm through a series of synthetic data sets and realistic data sets. For synthetic datasets, we set T_θ , f_w and S_ϕ as fully connected neural networks, where T_θ , f_w have 4 hidden layers and f_{ϕ_1} and g_{ϕ_2} (we adopt DeepSets) have 3 hidden layers. Each layer has 200 nodes, and the activation function is chosen as RELU, where we adopt the softmax in the final layer.

Normal distribution on 2D toy data: We first consider the 2D normal case, where the training data contains $10K$ sets and each set contains 100 data points from $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. We sample the mean, variance, and covariance from $U[-5, 5]$, $U[1, 2]$, and $U[-0.5, 0.5]$, respectively. Fig. 6 shows the real (gray points) and generated samples (red points) by different models given unseen test sets, where we only consider CGAN-based methods for the simple toy data. We find that our model (third column) can improve the resistance to mode collapse compared with CGAN+MSE (second column) and better fit the unseen test distributions than CGAN (first column). This result indicates the OT loss can spur the summary network to capture more desired statistics for unseen distributions.

One-dimensional Gaussian distributions: In this case, we generate another collection of synthetic $1-D$ datasets based on Gaussian parametric family, where the means and variances are sampled from $\mathrm{U}[-1, 1]$ and $\mathrm{U}[0.5, 2]$ respectively. The training data contains $10K$ sets each containing 50 samples. We also visualize the pdfs of gaussian distributions with randomly means and variance in Fig 7, which are used to sample test data, and show the 500 data points generated by the push-forward. For this experiment, we set the dimension of z and summary vector s as 2.

Multi-family distribution on 1D toy data: To validate if our proposed model can capture many types of distributional families simultaneously, we construct a collection of synthetic 1-D datasets each containing 100 samples from either an Exponential, Gaussian or Laplacian distribution with equal probability. For Gaussian and Laplacian distributions, means and variances are sampled from $U[-1, 1]$ and $U[0.5, 2]$ respectively; for Exponential distributions, rates are sampled from $U[0.5, 2]$. Fig. 8 visualizes the pdfs of six one-dimensional test distributions with different means and variances and the generated data points. It is interesting to observe that the generated data points can fit the corresponding pdf well, indicating our model can generalize to different distributions with varying parameters. Besides, our model performs slightly worse on the Exponential distributions, perhaps attributing to the fact that it is the only non-symmetric distribution.

E DETAILS ABOUT NATURAL IMAGE GENERATIONS AND THE RESULTS

We use denseNet proposed by Huang et al. (2018) as the backbone of summary network, then a pooling operation is conducted as Zaheer et al. (2017) does. And a 2-layer fully connected network with ReLU activation function is finally employed to embed the 4096-D visual features

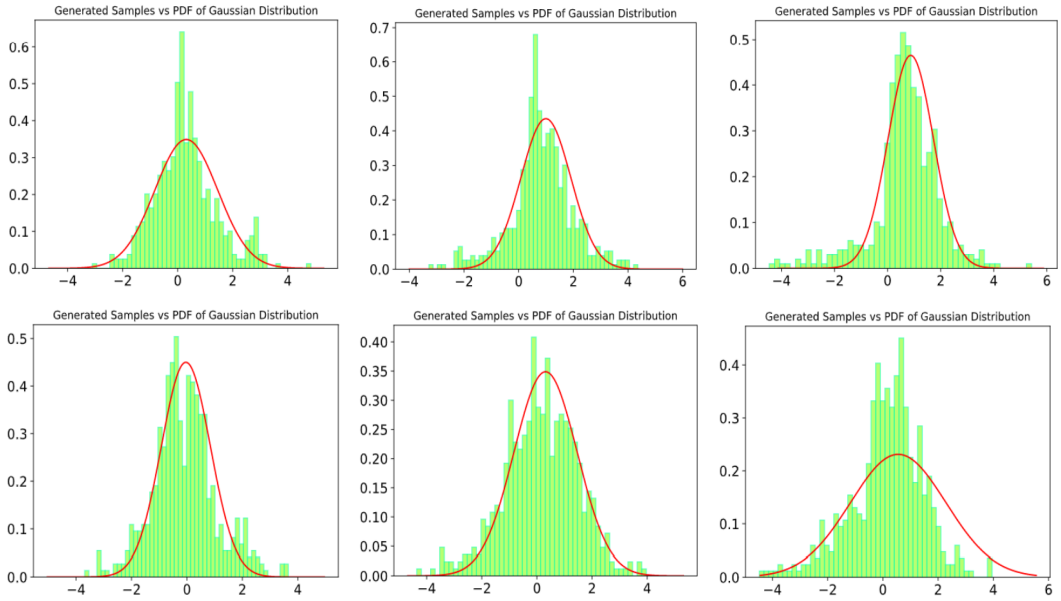


Figure 7: The generated samples by the generator vs the pdf of the true distribution conditioned on the one dimensional Gaussian distributions.

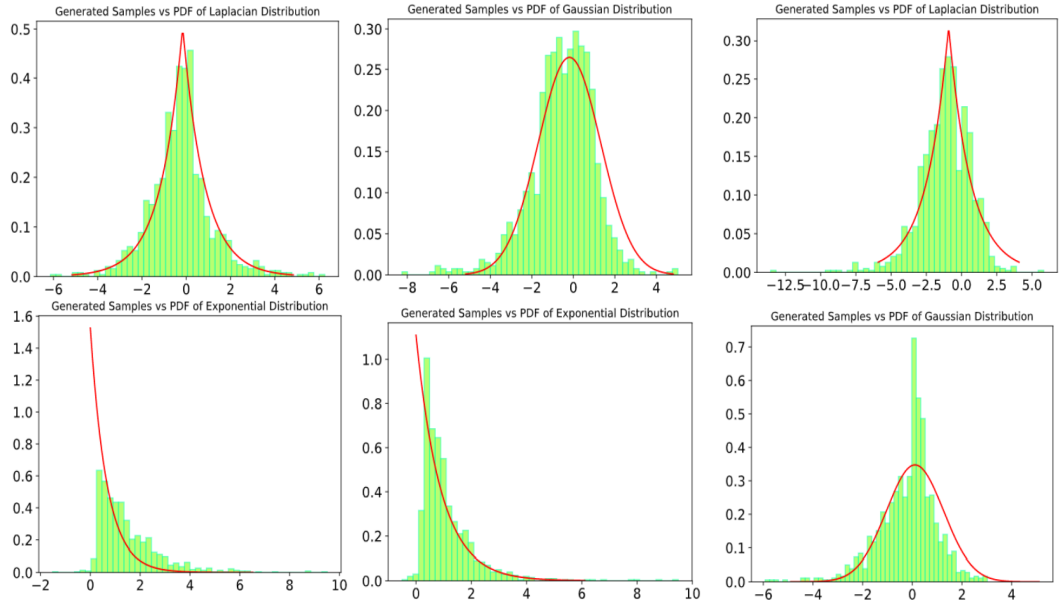
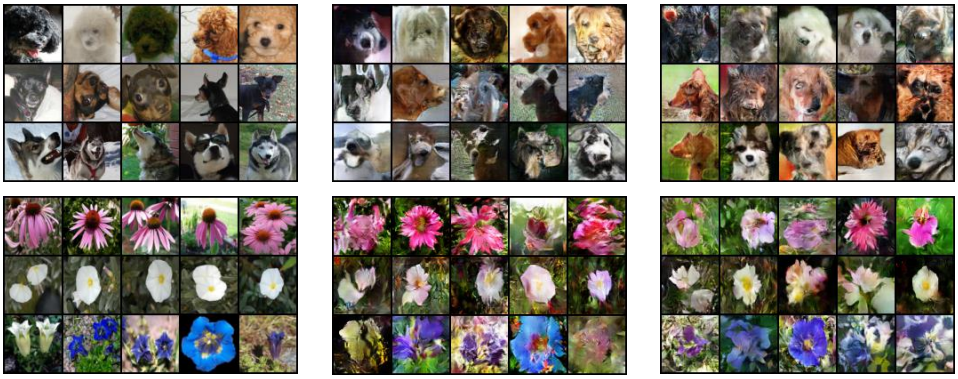


Figure 8: The generated samples by the generator vs the pdf of the true distribution conditioned on the multi distributions.

into the corresponding 512-D set representations $\boldsymbol{h}_j$. As for conditional generator (conditioned on set representations as well as Gaussian noise), we introduce a 2-layer embedding network [100→600→100] with LeakyReLU activation function to embed the input noise $\mathbf{n}$. Besides, we use a 5-layer deconvolution network [ConvTranspose2d($\mathbf{X}$ + 100, 512, 4, 1, 0)→ ConvTranspose2d(512, 256, 4, 2, 1)→ConvTranspose2d(512, 128, 4, 2, 1)→ConvTranspose2d(128, 64, 4,

2, 1)→ConvTranspose2d(64, 3, 4, 2, 1)], where $\mathbf{X} = 64, 128$ for oxford and animal face datasets respectively with BatchNorm along channels and ReLU activation function to deconvolute the concatenated noise embeddings and set representations $cat([\mathbf{n}, \boldsymbol{h}_j])$ as fake output images $\boldsymbol{x}_{fake}$: $\mathbb{R}^{N\times3\times64\times64}$. Finally, a 5-layer discriminator network [Conv2d(2*3, 64, 4, 2, 1)→Conv2d(64, 128, 4, 2, 1))→Conv2d(128, 256, 4, 2, 1)→Conv2d(256, 512, 4, 2, 1)→Conv2d(512, 1, 4, 1, 0)] with BatchNorm as well as LeakyReLU activation function at the first fourth deconvolutional layers and the last layer without BatchNorm while with sigmoid activation funtion to distinguish the true or fake generated images. We present the generated results in Figure 9.



(a) Real images
Above: Animal Face
Below: Oxford flower

(b) Fake images generated by DAGAN
Above: Animal Face
Below: Oxford flower

(c) Fake images generated by DAGAN+OT
Above: Animal Face
Below: Oxford flower

Figure 9: The images generated by DAGAN and DAGAN+OT conditioned on 3 different categories on Oxford(flower) and animal face datasets. The FID ↓ scores of DAGAN and DAGAN+OT on Oxford are 97.25 and **91.78**, respectively. The FIDs of DAGAN and DAGAN+OT on animal face are 139.14 and **131.51**.

F THE DIFFERENCE BETWEEN OUR MODEL AND BARYCENTER PROBLEM

In this section, we clarify the difference between Wasserstein barycenter and our method. Specifically, for j-th distribution, we denoted P_j as its empirical distribution consisting of N_j samples, expressed as $P_j = \sum_{i=1}^{N_j} \frac{1}{N_j}\delta_{\boldsymbol{x}_{ji}}, \boldsymbol{x}_{ji} \in \mathbb{R}^d$. Notably, $\boldsymbol{a}_j = [\frac{1}{N_j}] \in \Sigma^{N_j}$ represents the probability measure for distribution P_j.
For another thing, we can represent P_j with another to-be-learned distribution Q_j, defined as $Q_j = \sum_{k=1}^{K} h_{jk}\delta_{\boldsymbol{\beta}_k}, \boldsymbol{\beta}_k \in \mathbb{R}^d$. Here $\boldsymbol{b}_j = [h_{jk}] \in \Sigma^K$ is the probability measure for distribution Q_j, which can be computed using summary network $S_\phi(\boldsymbol{x}_{i,1:N_j})$ and serves as the representation for set j. And $\boldsymbol{\beta}_k$ is the k-th prototype in the same space of the observed data points, which is the k-th column of $\mathbf{B} \in \mathbb{R}^{K\times d}$, a learnable global prototype matrix. To optimize the $\boldsymbol{\beta}_{1:K}$ and the summary network S_ϕ for computing $\boldsymbol{h}_j$, we minimize the average OT loss (between Q_j and P_j) for all training sets. We rewrite the Equation (6) here for convenience:

$$L_{\text{OT}} = \min_{\mathbf{B},\phi} \frac{1}{\mathcal{J}} \sum_{j=1}^{\mathcal{J}} \left(\sum_{i}^{N_j} \sum_{k}^{K} C_{ik}T_{ik} - \epsilon \sum_{i}^{N_j} \sum_{k}^{K} -T_{ik}\ln T_{ik} \right) = \min_{\mathbf{B},\phi} \frac{1}{\mathcal{J}} \sum_{j=1}^{\mathcal{J}} \left(\text{OT}_\epsilon(P_j, Q_j)\right).$$

Usually, this equation can also be represented:

$$\min_{\mathbf{B},\phi} \frac{1}{\mathcal{J}} \sum_{j=1}^{\mathcal{J}} \left(\text{OT}_\epsilon(\boldsymbol{a}_j, \boldsymbol{b}_j)\right). \tag{9}$$

In terms of Wasserstein barycenter, we adopt the same notations for consistency. Following (Altschuler & Boix-Adserà, 2021), given J empirical distributions $P_{1:J}$ and their respective probability measures $[\boldsymbol{a}_1, \ldots, \boldsymbol{a}_J]$ supported on $\mathbb{R}^d$ and a vector $\boldsymbol{\lambda} \in \Sigma^J$, their corresponding Wasserstein

barycenter can be viewed as another distribution Q, i.e., $Q = \sum_{k=1}^{K} m_k \delta_{\boldsymbol{\beta}_k}, \boldsymbol{\beta}_k \in \mathbb{R}^d$, where $\boldsymbol{\beta}_k$ is the k-th column of $\mathbf{B}$, $\boldsymbol{m} = [m_k] \in \Sigma^K$ is the probability measure for distribution Q. Then we can learn the barycenter (i.e., $\mathbf{B}$ and $\boldsymbol{m}$) by minimizing

$$\min_{\mathbf{B},\boldsymbol{m}} \frac{1}{\mathcal{J}} \sum_{j=1}^{\mathcal{J}} \lambda_j \mathcal{W}\left(P_j, Q\right) = \min_{\mathbf{B},\boldsymbol{m}} \frac{1}{\mathcal{J}} \sum_{j=1}^{\mathcal{J}} \lambda_j \mathcal{W}\left(\boldsymbol{a}_j, \boldsymbol{m}\right) \tag{10}$$

where above $\mathcal{W}(\cdot,\cdot)$ denotes the squared 2-Wasserstein distance. By comparing the equation 10 and equation 9, we can find that our model learns a Q_j to approximate P_j for each distribution j but "barycenter" problem learns a shared Q as the barycenter for all $P_{1:J}$. Therefore, after minimizing the average loss for all training sets with Equation (6), we can use the probability measure $\boldsymbol{b}_j = [h_{jk}]$ (i.e., $\boldsymbol{h}_j$) to represent the empirical distribution P_j. Especially, we can directly map the test set to its representation by using the summary network. However, since the probability measure $\boldsymbol{m}$ in "barycenter" is shared by all distributions, it can not represent a specific set. Therefore, to achieve the set representation, it might need to first compute the transport plan between test set and the barycenter Q and then aggregate the data points (or features) within the test set by taking the transport plan as the weight. Therefore, our model produces a more intuitive solution to learn the set representation, which can take full advantage of the existing summary networks and provides a promising tool for addressing set-input and meta-learning problems. We will clarify the difference between our model and "barycenter" problem in the appendix of the revised version if necessary.