Partially Observed Exchangeable Modeling

Yang Li [1] **Junier B. Oliva** [1]

Abstract

Modeling dependencies among features is fundamental for many machine learning tasks. Although there are often multiple related instances that may be leveraged to inform conditional dependencies, typical approaches only model conditional dependencies over individual instances. In this work, we propose a novel framework, partially observed exchangeable modeling (POEx) that takes in a set of related partially observed instances and infers the conditional distribution for the unobserved dimensions over multiple elements. Our approach jointly models the intra-instance (among features in a point) and inter-instance (among multiple points in a set) dependencies in data. POEx is a general framework that encompasses many existing tasks such as point cloud expansion and few-shot generation, as well as new tasks like few-shot imputation. Despite its generality, extensive empirical evaluations show that our model achieves state-of-the-art performance across a range of applications.

1. Introduction

Modeling dependencies among features is at the core of many unsupervised learning tasks. Typical approaches consider modeling dependencies in a vacuum. For example, one typically imputes the unobserved features of a single instance based only on that instance's observed features. However, there are often multiple related instances that may be leveraged to inform conditional dependencies. For instance, a patient may have multiple visits to a clinic with different sets of measurements, which may be used together to infer the missing ones. In this work, we propose to jointly model the intra-instance (among features in a point) and inter-instance (among multiple points in a set) dependencies by modeling sets of partially observed instances. To our knowledge, this is the first work that generalizes the concept of partially observed data to exchangeable sets. We consider modeling an exchangeable (permutation invariant) likelihood over a set $\mathbf{x} = \{x_i\}_{i=1}^N, x_i \in \mathbb{R}^d$. However, unlike previous approaches (Korshunova et al., 2018; Edwards & Storkey, 2016; Li et al., 2020b; Bender et al., 2020), we model a partially observed set, where "unobserved" features of points $\mathbf{x}_u = \{x_i^{(u_i)}\}_{i=1}^N$ are conditioned on "observed" features of points $\mathbf{x}_o = \{x_i^{(o_i)}\}_{i=1}^N$ and $o_i, u_i \subseteq \{1, \dots, d\}$ and $o_i \cap u_i = \emptyset$. Since each feature in $\mathbf{x}_u$ depends not only on features from the corresponding element but also on features from other set elements, the conditional likelihood $p(\mathbf{x}_u \mid \mathbf{x}_o)$ captures the dependencies across both features and set elements.

Probabilistic modeling of sets where each instance itself contains a collection of elements is challenging, since set elements are exchangeable and the cardinality may vary. Our partially observed setting brings another level of challenge due to the arbitrariness of the observed subset for each element. First, the subsets have arbitrary dimensionality, which poses challenges for modern neural network based models. Second, the combinatorial nature of the subsets renders the conditional distributions highly multi-modal, which makes it difficult to model accurately. To resolve these difficulties, we propose a variational weight-sharing scheme that is able to model the combinatorial cases in a single model.

Partially observed exchangeable modeling (POEx) is a general framework that encompasses many impactful applications, which we describe below. Despite its generality, we find that our single POEx approach provides competitive or better results than specialized approaches for these tasks.

Few-shot Imputation A direct application of the conditional distribution $p(\mathbf{x}_u \mid \mathbf{x}_o)$ enables a task we coin *few-shot imputation*, where one models a subset of covariates based on multiple related observations of an instance $\{x_i^{(o_i)}\}_{i=1}^N$. Our set imputation formulation leverages the dependencies across set elements to infer the missing values. For example, when modeling an occluded region in an image $x_i^{(u_i)}$, it would be beneficial to also condition on observed pixels from other angles $x_j^{(o_j)}$. This task is akin to multi-task imputation and is related to group mean imputation (Sim et al., 2015), which imputes missing features in an instance according to the mean value of the features in

[1]Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA. Correspondence to: Yang Li <yangli95@cs.unc.edu>, Junier B. Oliva <joliva@cs.unc.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

a related group. However, our approach models an entire distribution (rather than providing a single imputation) and captures richer dependencies beyond the mean of the features. Given diverse sets during training, our POEx model generalizes to unseen sets.

Set Expansion When some set elements have fully observed features, $o_i = \{1, \dots, d\}$, and others have fully unobserved features, $o_j = \emptyset$, POEx can generate novel elements based on the given set of illustrations. Representative examples of this application include point cloud completion and upsampling, where new points are generated from the underlying geometry to either complete an occluded point cloud or improve the resolution of a sparse point cloud.

Few-shot Generation The set expansion formulation can be viewed as a few-shot generative model, where novel instances are generated based on a few exemplars. Given diverse training examples, the model is expected to generate novel instances even on unseen sets.

Set Compression Instead of expanding a set, we may proceed in the opposite direction and compress a given set. For example, we can represent a large point set with its coreset to reduce storage and computing requirements. The likelihood from our POEx model can guide the selection of an optimal subset, which retains the most information.

Neural Processes If we introduce an index variable t_i for each set element and extend the original set $\{x_i\}_{i=1}^N$ to a set of index-value pairs $\{(t_i, x_i)\}_{i=1}^N$, our POEx model encapsulates the neural processes (Garnelo et al., 2018a;b; Kim et al., 2019) as a special case. New elements corresponding to the given indexes can be generated from a conditional version of POEx: $p(\mathbf{x}_u \mid \mathbf{x}_o, \mathbf{t})$, where $\mathbf{t} = \{t_i\}_{i=1}^N$. In this work, we focus on modeling processes in high-dimensional spaces, such as processes of images, which are challenging due to the multi-modality of the underlying distributions.

Set of Functions Instead of modeling a set of finite dimensional vectors, we may be interested in sets of functions, such as a set of correlated processes. By leveraging the dependencies across functions, we can fit each function better while utilizing fewer observations. Our formulation essentially generalizes the multi-task Gaussian processes (Bonilla et al., 2008) into multi-task neural processes.

The contributions of this work are as follows: 1) We extend the concept of partially observed data to exchangeable sets so that the dependencies among both features and set elements are captured in a single model. 2) We develop a deep latent variable based model to learn the conditional distributions for sets and propose a collapsed inference technique to optimize the ELBO. The collapsed inference simplifies the hierarchical inference framework to a single level. 3) We leverage the captured dependencies to perform various applications, which are difficult or even impossible for alternative approaches. 4) Our model handles neural processes as special cases and generalizes the original neural processes to high-dimensional distributions. 5) We propose a novel extension of neural process, dubbed multi-task neural process, where sets of infinite-dimensional functions are modeled together. 6) We conduct extensive experiments to verify the effectiveness of our proposed model and demonstrate state-of-the-art performance across a range of applications.

2. Background

Set Modeling The main challenge of modeling set structured data is to respect the permutation invariant property of sets. A straight-forward approach is to augment the training data with randomly permuted orders and treat them as sequences. Given infinite training data and model capacity, an autoregressive model can produce permutation invariant likelihoods. However, for real-world limited data and models, permutation invariance is not guaranteed. As pointed out in (Vinyals et al., 2015), the order actually matters for autoregressive models.

BRUNO (Korshunova et al., 2018) proposes using invertible transformations to project each set element to a latent space where dimensions are factorized independently. Then they build independent exchangeable processes for each dimension in the latent space to obtain the permutation invariant likelihoods. FlowScan (Bender et al., 2020) instead recommends using a scan sorting operation to convert the set likelihood to a familiar sequence likelihood and normalizing the likelihood accordingly. ExNODE (Li et al., 2020b) utilizes neural ODE based permutation equivariant transformations and permutation invariant base likelihoods to construct a continuous normalizing flow model for exchangeable data.

De Finetti's theorem provides a principled way of modeling exchangeable data, where each element is modeled independently conditioned on a latent variable θ:

$$p(\{x_i\}_{i=1}^N) = \int \prod_{i=1}^N p(x_i \mid \theta) p(\theta) d\theta. \qquad (1)$$

Latent Dirichlet allocation (LDA) (Blei et al., 2003) and its variants (Teh et al., 2006b; Blei et al., 2007) are classic models of this form, where the likelihood and prior are expressed as simple known distributions. Recently, deep neural network based models have been proposed (Yang et al., 2019; Edwards & Storkey, 2016; Yang et al., 2020), in which a VAE is trained to optimize a lower bound of (1).

Arbitrary Conditional Models Instead of modeling the joint distribution $p(x)$, where $x \in \mathbb{R}^d$, arbitrary conditional models learn the conditional distributions for an arbitrary subset of features x_u conditioned on another non-overlapping arbitrary subset x_o, where $u, o \subseteq \{1, \dots, d\}$. Graphical models are a natural choice for such tasks (Koster et al., 2002), where conditioning usually has a closed-form solution. However, the graph structure is usually unknown for general data, and learning the graph structure from observational data has its own challenges (Heinze-Deml et al., 2018; Scutari et al., 2019). Sum-Product Network (SPN) (Poon & Domingos, 2011) and its variants (Jaini et al., 2018; Butz et al., 2019; Tan & Peharz, 2019) borrow the idea from graphical models and build deep neural networks by stacking sum and product operations alternately so that the arbitrary conditionals are tractable.

Deep generative models have also been used for this task. Universal Marginalizer (Douglas et al., 2017) builds a feed-forward network to approximate the conditional marginal distributions of each dimension conditioned on x_o. VAEAC (Ivanov et al., 2018) utilizes a conditional VAE to learn the conditional distribution $p(x_u \mid x_o)$. ACFlow (Li et al., 2020a) uses a normalizing flow based model for learning the arbitrary conditionals, where invertible transformations are specially designed to deal with arbitrary dimensionalities. GAN based approaches (Belghazi et al., 2019) have also been proposed to model arbitrary conditionals.

Stochastic Processes Stochastic processes are usually defined as the marginal distribution over a collection of indexed random variables $\{x_t; t \in \mathcal{T}\}$. For example, Gaussian process (Rasmussen, 2003) specifies that the marginal distribution $p(x_{t_1:t_n} \mid \{t_i\}_{i=1}^n)$ follows a multivariate Gaussian distribution, where the covariance is defined by some kernel function $K(t,t')$. The Kolmogrov extension theorem (Øksendal, 2003) provides the sufficient condition for designing a valid stochastic process:

- **Exchangeability:** The marginal distribution is invariant to any permutation π, i.e.,

$$p(x_{t_1:t_n} \mid \{t_i\}_{i=1}^n) = p(x_{t_{\pi_1}:t_{\pi_n}} \mid \pi(\{t_i\}_{i=1}^n)).$$

- **Consistency:** Marginalizing out part of the variables is the same as the one obtained from the original process, i.e., for any $1 \leq m \leq n$

$$p(x_{t_1:t_m} \mid \{t_i\}_{i=1}^m) = \int p(x_{t_1:t_n} \mid \{t_i\}_{i=1}^n) dx_{t_{m+1}:t_n}.$$

Stochastic processes can be viewed as a distribution over the space of functions and can be used for modeling exchangeable data. However, classic Gaussian processes (Rasmussen, 2003) and Student-t processes (Shah et al., 2014) assume the marginals follow a simple known distribution for tractability and have an $O(n^3)$ complexity, which render them impractical for large-scale complex dataset.

Neural Processes (Garnelo et al., 2018a;b; Kim et al., 2019) overcome the above limitations by learning a latent variable based model conditioned on a set of context points $X^{(C)} = \{(t_i^{(C)}, x_i^{(C)})\}_{i=1}^{N_C}$. The latent variable θ implicitly parametrizes a distribution over the underlying functions so that values on target points $\{t_j^{(T)}\}_{j=1}^{N_T}$ can be evaluated over random draws of the latent variable, i.e.,

$$p(x_{t_1:t_{N_T}}^{(T)} \mid \{t_j^{(T)}\}_{j=1}^{N_T}) = \int \prod_{j=1}^{N_T} p(x_j^{(T)} \mid t_j^{(T)}, \theta) p(\theta \mid X^{(C)}) d\theta.$$

Neural processes generalize the kernel based stochastic processes with deep neural networks and scale with $O(n)$ due to the amortized inference. The exchangeability requirement is met by using exchangeable neural networks for inference, and the consistency requirement is roughly satisfied with the variational approximation.

3. Method

In this section, we develop our approach for modeling sets of partially observed elements. We describe the variants of POEx and their corresponding applications. We also introduce our inference techniques used to train the model.

3.1. Partially Observed Exchangeable Modeling

3.1.1. Arbitrary Conditionals

Consider a set of vectors $\{x_i\}_{i=1}^N$, where $x_i \in \mathbb{R}^d$ and N is the cardinality of the set. For each set element x_i, only a subset of features $x_i^{(o_i)}$ are observed and we would like to predict the values for another subset of features $x_i^{(u_i)}$. Here, $u_i, o_i \subseteq \{1, \dots, d\}$ and $u_i \cap o_i = \emptyset$. We denote the set of observed features as $\mathbf{x}_o = \{x_i^{(o_i)}\}_{i=1}^N$ and the set of unobserved features as $\mathbf{x}_u = \{x_i^{(u_i)}\}_{i=1}^N$. Our goal is to model the distribution $p(\mathbf{x}_u \mid \mathbf{x}_o)$ for arbitrary u_i and o_i. Throughout the experiment, we assume features are missing completely at random (MCAR) for each element.

In order to model the arbitrary conditional distributions for sets, we introduce a latent variable θ. The following theorem states that there exists a latent variable θ such that conditioning on θ renders the set elements of $\mathbf{x}_u$ i.i.d.. Please see appendix for the proof.

Theorem 1. *Given a set of observations* $\mathbf{x} = \{x_i\}_{i=1}^N$ *from an infinitely exchangeable process, denote the observed and unobserved part as* $\mathbf{x}_o = \{x_i^{(o_i)}\}_{i=1}^N$ *and* $\mathbf{x}_u = \{x_i^{(u_i)}\}_{i=1}^N$ *respectively. Then the arbitrary conditional distribution* $p(\mathbf{x}_u \mid \mathbf{x}_o)$ *can be decomposed as follows:*

$$p(\mathbf{x}_u \mid \mathbf{x}_o) = \int \prod_{i=1}^N p(x_i^{(u_i)} \mid x_i^{(o_i)}, \theta) p(\theta \mid \mathbf{x}_o) d\theta. \quad (2)$$

Optimizing (2), however, is intractable due to the high-dimensional integration over θ. Therefore, we resort to variational approximation and optimize a lower bound:

$$\log p(\mathbf{x}_u \mid \mathbf{x}_o) \geq \sum_{i=1}^N \mathbb{E}_{q(\theta \mid \mathbf{x}_u, \mathbf{x}_o)} \log p(x_i^{(u_i)} \mid x_i^{(o_i)}, \theta) - D_{KL}(q(\theta \mid \mathbf{x}_u, \mathbf{x}_o) \parallel p(\theta \mid \mathbf{x}_o)), \quad (3)$$

where $q(\theta \mid \mathbf{x}_u, \mathbf{x}_o)$ and $p(\theta \mid \mathbf{x}_o)$ are variational posterior and prior that are permutation invariant w.r.t. the conditioning set. The arbitrary conditional likelihoods $p(x_i^{(u_i)} \mid x_i^{(o_i)}, \theta)$ are over a $\mathbb{R}^d$ feature space and can be implemented as in previous works (Ivanov et al., 2018; Li et al., 2020a; Belghazi et al., 2019).

Note that $\mathbf{x}_o$ and $\mathbf{x}_u$ are sets of vectors with arbitrary dimensionality. To represent vectors with arbitrary dimensionality so that a neural network can handle them easily, we impute missing features with zeros and introduce a binary mask to indicate whether the corresponding dimensions are missing or not. We denote the zero imputation operation as $\mathcal{I}(\cdot)$ that takes in a set of features with arbitrary dimensionality and outputs a set of d-dimensional features and the corresponding set of binary masks.

3.1.2. Set Compression

Given a pretrained POEx model, we can use the arbitrary conditional likelihoods $p(\mathbf{x}_u \mid \mathbf{x}_o)$ to guide the selection of a subset for compression. The principle is to select a subset that preserves the most information. Set compression is a type of combinatorial optimization problem, which is NP-hard. Here, we propose a sequential approach that selects one element at a time. We start from $o_j = \emptyset, u_j = \{1, \dots, d\}$ for each element x_j, that is, all elements are fully unobserved. The next element i to select should be the one that maximizes the conditional entropy $H(x_i \mid \mathbf{x}_o)$, which represents the most uncertain element across the remaining unobserved ones given the current selected elements. Since the original set $\mathbf{x}$ is given, we can estimate the entropy with

$$H(x_i \mid \mathbf{x}_o) = \mathbb{E}_{p(x_i|\mathbf{x}_o)} - \log p(x_i \mid \mathbf{x}_o) \approx - \log p(x_i \mid \mathbf{x}_o).$$

Therefore, the next element to chose is simply the one with minimum likelihood $p(x_i \mid \mathbf{x}_o)$ based on the current selection $\mathbf{x}_o$. Afterwards, we update $o_i = \{1, \dots, d\}, u_i = \emptyset$ and proceed to the next selection step.

3.1.3. Neural Process

Some applications may introduce index variables for each set element. For example, a collection of frames from a video are naturally indexed by their timestamps. Here, we consider a set of index-value pairs $\{(t_i, x_i)\}_{i=1}^N$, where t_i can be either discrete or continuous. Similarly, x_i are partially observed, and we define $\mathbf{x}_u$ and $\mathbf{x}_o$ accordingly. We also define $\mathbf{t} = \{t_i\}_{i=1}^N$ for notation simplicity, which are typically given. By conditioning on the index variables $\mathbf{t}$, we modify the lower bound (3) to

$$\log p(\mathbf{x}_u \mid \mathbf{x}_o, \mathbf{t}) \geq \sum_{i=1}^{N} \mathbb{E}_{q(\theta|\mathbf{x}_u, \mathbf{x}_o, \mathbf{t})} \log p(x_i^{(u_i)} \mid x_i^{(o_i)}, t_i, \theta) - D_{KL}(q(\theta \mid \mathbf{x}_u, \mathbf{x}_o, \mathbf{t}) \| p(\theta \mid \mathbf{x}_o, \mathbf{t})). \tag{4}$$

If we further generalize the cardinality N to be infinite and specify a context set $\mathbf{x}_c$ and a target set $\mathbf{x}_t$ to be arbitrary subsets of all set elements, i.e., $\mathbf{x}_c, \mathbf{x}_t \subseteq \{(t_i, x_i)\}_{i=1}^N$, we recover the exact setting for neural process. This is a special case of our POEx model in that features are fully observed ($o_i = \{1, \dots, d\}, u_i = \emptyset$) for elements of $\mathbf{x}_c$ and fully unobserved ($o_i = \emptyset, u_i = \{1, \dots, d\}$) for elements of $\mathbf{x}_t$. That is, $\mathbf{x}_o = \mathbf{x}_c$ and $\mathbf{x}_u = \mathbf{x}_t$. The ELBO objective is exactly the same as (4). Similar to neural processes, we use a finite set of data points to optimize the ELBO (4) and sample a subset at random as the context.

Neural processes usually use simple feed-forward networks and Gaussian distributions for the conditional likelihood $p(x_i^{(u_i)} \mid x_i^{(o_i)}, t_i, \theta)$, which makes it unsuitable for multi-modal distributions. Furthermore, they typically deal with low-dimensional data. Our model, however, utilizes arbitrary conditional likelihoods, which can deal with high-dimensional and multi-modal distributions.

3.1.4. Multi-task Neural Process

Neural processes model the distributions over functions, where one input variable is mapped to one target variable. In a multi-task learning scenario, there exists multiple target variables. Therefore, we propose a multi-task neural process extension to capture the correlations among target variables. For notation simplicity, we assume the target variables are exchangeable here. Non-exchangeable targets can be easily transformed to exchangeable ones by concatenating with their indexes. Consider a set of functions $\{\mathcal{F}_k\}_{k=1}^K$ for K target variables. Inspired by neural process, we represent each function $\mathcal{F}_k$ by a set of input-output pairs $\{(t_{ki}, x_{ki})\}_{i=1}^{N_k}$. The goal of multi-task neural process is to learn an arbitrary conditional model given arbitrarily observed subsets from each function. We similarly define $\mathbf{x}_u = \{\mathcal{F}_k^{(u_k)}\}_{k=1}^{N_k} = \{\{(t_{ki}, x_{ki}^{(u_{ki})})\}_{i=1}^{N_k}\}_{k=1}^K$ and $\mathbf{x}_o = \{\mathcal{F}_k^{(o_k)}\}_{k=1}^{N_k} = \{\{(t_{ki}, x_{ki}^{(o_{ki})})\}_{i=1}^{N_k}\}_{k=1}^K$.

The multi-task neural process described above models a set of sets. A straight-forward approach is to use a hierarchical model

$$p(\mathbf{x}_u \mid \mathbf{x}_o) = \int \prod_{k=1}^{K} p(\mathcal{F}_k^{(u_k)} \mid \mathcal{F}_k^{(o_k)}, \theta) p(\theta \mid \mathbf{x}_o) d\theta = \int \prod_{k=1}^{K} \left[\int \prod_{i=1}^{N_k} p(x_{ki}^{(u_{ki})} \mid x_{ki}^{(o_{ki})}, t_{ki}, \phi) p(\phi \mid \mathcal{F}_k^{(o_k)}, \theta) d\phi \right] p(\theta \mid \mathbf{x}_o) d\theta, \tag{5}$$

which utilizes the Theorem 1 twice. However, inference with such a model is challenging since complex inter-dependencies need to be captured across two set levels. Moreover, the latent variables are not of direct interest. Therefore, we propose an inference technique that collapses the two latent variables into one. Specifically, we assume the uncertainties across θ and ϕ are both absorbed into θ and define $p(\phi \mid \mathcal{F}_k^{(o_k)}, \theta) = \delta(G(\mathcal{F}_k^{(o_k)}, \theta))$, where G represents a deterministic mapping. Therefore, (5) can be simplified as

$$p(\mathbf{x}_u \mid \mathbf{x}_o) = \int \prod_{k=1}^{K} \prod_{i=1}^{N_k} p(x_{ki}^{(u_{ki})} \mid x_{ki}^{(o_{ki})}, t_{ki}, \phi) \delta(G(\mathcal{F}_k^{(o_k)}, \theta)) p(\theta \mid \mathbf{x}_o) d\phi d\theta. \tag{6}$$

Further collapsing ϕ and θ into one latent variable ψ gives

$$p(\mathbf{x}_u \mid \mathbf{x}_o) = \int \prod_{k=1}^{K} \prod_{i=1}^{N_k} p(x_{ki}^{(u_{ki})} \mid x_{ki}^{(o_{ki})}, t_{ki}, \psi) p(\psi \mid \mathbf{x}_o) d\psi, \tag{7}$$

where ψ is permutation invariant for both set levels. The collapsed model may seem restricted at first sight, but we show empirically that it remains powerful when we use a flexible likelihood model for the arbitrary conditionals. More importantly, it significantly simplifies the implementation.

A similar collapsed inference technique has been used in (Griffiths & Steyvers, 2004; Teh et al., 2006a; Porteous et al., 2008) to reduce computational cost and accelerate inference for LDA models. Recently, Yang et al. (2020) propose to use collapsed inference in the neural process framework to marginalize out the index variables. Here, we utilize collapsed inference to reduce a hierarchical generative model to a single level.

Given the generative process (7), it is straightforward to optimize using the ELBO

$$\log p(\mathbf{x}_u \mid \mathbf{x}_o) \geq \sum_{k=1}^{K} \sum_{i=1}^{N_k} \mathbb{E}_{q(\psi \mid \mathbf{x}_u, \mathbf{x}_o, \mathbf{t})} \log p(x_{ki}^{(u_{ki})} \mid x_{ki}^{(o_{ki})}, t_{ki}, \psi) - D_{KL}(q(\psi \mid \mathbf{x}_u, \mathbf{x}_o, \mathbf{t}) \| p(\psi \mid \mathbf{x}_o, \mathbf{t})). \tag{8}$$

3.2. Implementation

In this section, we describe some implementation details of POEx that are important for good empirical performance. Please refer to Sec. B in the appendix for more details. Our code is publicly available at `https://github.com/lupalab/POEx`.

Given the ELBO objectives defined in (3), (4) and (8), it is straightforward to implement them as conditional VAEs. Please see Fig. 1 for an illustration. The posterior and prior are implemented with permutation invariant networks. For sets of vectors (such as point clouds), we first use Set Transformer (Lee et al., 2019) to extract a permutation equivariant embedding, then average over the sets. For sets of images, we use a convolutional neural network to process each image independently and take the mean embedding over the sets. For sets of sets/functions, Set Transformer (with a global pooling) is used to extract the embedding for each function respectively, then the average embedding is taken as the final permutation invariant embedding. Index variables are tiled as the same sized tensor as the corresponding inputs and concatenated together. The posterior is then defined as a Gaussian distribution, where the mean and variance are derived from the set representation. The prior is defined as a normalizing flow model Q with base distribution defined as a Gaussian conditioned on the set representation. The KL-divergence terms are calculated by Monte-Carlo estimation: $D_{KL}(q\|p) = -H(q) - \mathbb{E}_q \log p$, where both $H(q)$ and $\log p$ are tractable.

In addition to the permutation invariant latent code, we also use a permutation equivariant embedding of x_o to assist the learning of the arbitrary conditional likelihood. For a set of vectors, we use Set Transformer to capture the inter-dependencies. For images, Set Transformer is computationally too expensive. Therefore, we propose to decompose the computation across spatial dimensions and set dimension. Specifically, for a set of images $\{x_i\}_{i=1}^N$, shared convolutional layers are applied to each set element, and self-attention layers are applied to each spatial position. Such layers and pooling layers can be stacked alternately to extract a permutation equivariant embedding for a set of images. For sets of sets $\{\{x_{ki}\}_{i=1}^{N_k}\}_{k=1}^K$, the permutation equivariant embedding contains two parts. One part is the self attention embedding that attends only to the features in the same set (i.e., same k) $\{\textbf{SelfAttention}(\{x_{ki}\}_{i=1}^{N_k})\}_{k=1}^K$. Note that **SelfAttention** outputs a feature vector for each element, which is a weighted sum of a certain embedding from each element. Another part is the attention embedding across different sets $\{\frac{1}{K-1}\sum_{k'=1}^K \textbf{CrossAttention}(\{x_{ki}\}_{i=1}^{N_k}, \{x_{k'j}\}_{j=1}^{N_{k'}}) \mathbb{I}(k \neq k')\}_{k=1}^K$. For each element in the query set, the cross attention outputs an attentive embedding over the key set.

Given the permutation equivariant embedding of $\mathbf{x}_o$ (denoted as ζ), the arbitrary conditionals $p(x_i^{(u_i)} \mid x_i^{(o_i)}, \theta)$, $p(x_i^{(u_i)} \mid x_i^{(o_i)}, t_i, \theta)$ and $p(x_{ki}^{(u_{ki})} \mid x_{ki}^{(o_{ki})}, t_{ki}, \psi)$ in (3), (4) and (8) are rewritten as $p(x_i^{(u_i)} \mid x_i^{(o_i)}, \theta, \zeta)$, $p(x_i^{(u_i)} \mid x_i^{(o_i)}, t_i, \theta, \zeta)$ and $p(x_{ki}^{(u_{ki})} \mid x_{ki}^{(o_{ki})}, t_{ki}, \psi, \zeta)$ respectively, which can be implemented by any arbitrary conditional models (Li et al., 2020a; Ivanov et al., 2018; Douglas et al., 2017). Here, we choose ACFlow for most of the experiments and modify it to a conditional version, where both the transformations and the base likelihood are conditioned on the corresponding tensors. For low-dimensional data, such as 1D function approximation, a simple feed-forward network that maps the conditioning tensor to a Gaussian distribution also works well.

4. Experiments

In this section, we conduct extensive experiments with POEx for the aforementioned applications. In order to verify the effectiveness of the set level dependencies, we compare

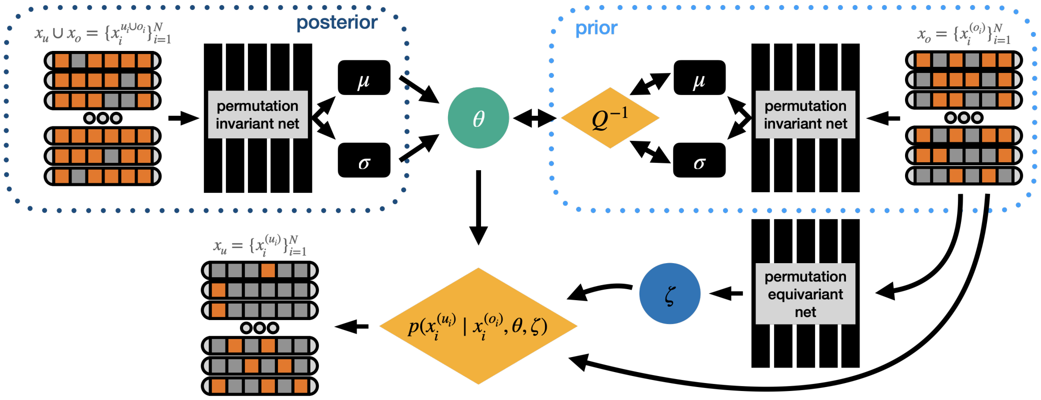
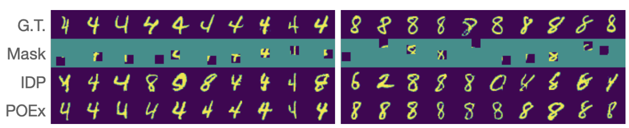
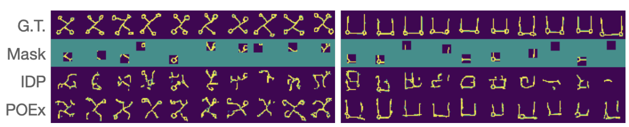


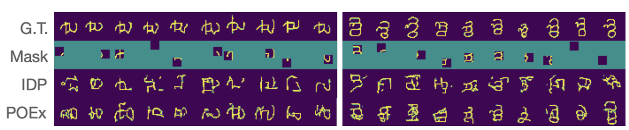
Figure 1. VAE model for partially observed exchangeable modeling.



(a) MNIST



(b) Omniglot



(c) Omniglot from unseen classes

Figure 2. Inpaint the missing values for a set of images.

to a model that treats each set element as independent input (denoted as IDP). IDP uses the same architecture as the decoder of POEx. We also compare to some specially designed approaches for each application. Due to space limitations, we put the experimental details and additional results in the appendix. In this work, we emphasize the versatility of POEx, and note that certain domain specific techniques may further improve performance, which we leave for future works.

We first utilize our POEx model to impute missing values for a set of images from MNIST and Omniglot datasets, where several images from the same class are considered a set. We consider a setting where only a small portion of pixels are observed for each image. Figure 2 and Table 1 compare the results for POEx, IDP, and a tensor completion based approach TRC (Wang et al., 2017). The results demonstrate clearly that the dependencies among set elements can significantly improve the imputation performance. Even when the given information is limited for each image, our POEx model can still accurately recover the missing parts. TRC fails to recover any meaningful structures for both MNIST and Omniglot, see Fig. C.1 for several examples. Our POEx model can also perform few-shot imputation on unseen classes, see Fig. 2(c) for several examples.

Table 1. PSNR of inpainting sets of images.

	MNIST	Omniglot
TRC	7.80	8.87
IDP	11.38	11.49
POEx	**13.02**	**12.09**

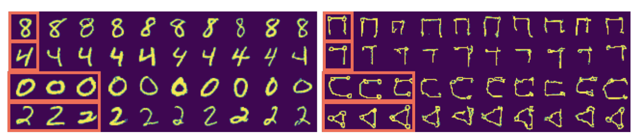


Figure 3. Expand a set by generating similar elements. Red boxes indicate the given elements. Left: MNIST. Right: Omniglot.

If we change the distribution of the masks so that some elements are fully observed, our POEx model can perform set expansion by generating similar elements to the given ones. Figure 3 shows several examples for MNIST and Omniglot datasets. Our POEx model can generate realistic and novel images even if only one element is given.

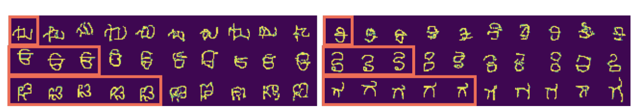
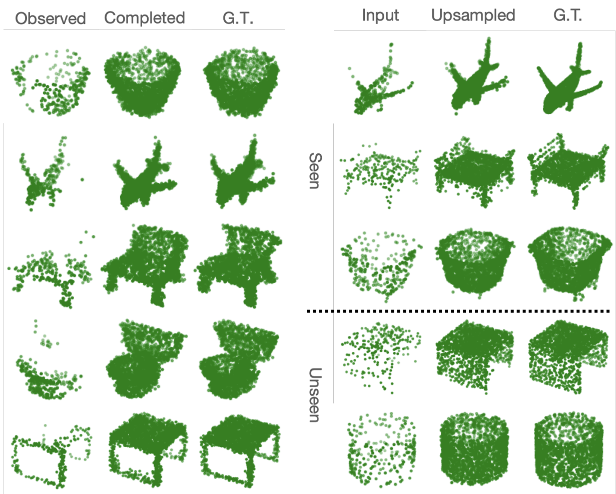


Figure 4. Few-shot generation with unseen Omniglot characters.

To further test the generalization ability of POEx, we provide the model with several unseen characters and utilize the POEx model to generate new elements. Figure 4 demonstrates the few-shot generation results given

Table 2. 5-way-1-shot classification with MAML.

Algorithm	Acc.
MAML	89.7
MAML(aug=5)	93.8
MAML(aug=10)	94.7
MAML(aug=20)	95.1



(a) completion (b) upsampling

Figure 5. Point cloud completion and upsampling.

several unseen Omniglot images. We can see the generated images appear similar to the given ones. To quantitatively evaluate the quality of generated images, we perform few-shot classification by augmenting the few-shot support sets with our POEx model. We evaluate the 5-way-1-shot accuracy of a fully connected network using MAML (Finn et al., 2017). Table 2 reports the accuracy of MAML with and without augmentation. We can see the few-shot accuracy improves as we provide more synthetic data.

In addition to sets of images, our POEx model can deal with point clouds. Figure 5 presents several examples for point cloud completion and upsampling. Point cloud completion predicts the occluded parts based on a partial point cloud. Partial point clouds are common in practice due to limited sensor resolution and occlusion. We use the dataset created by Wang et al. (2020), where the point cloud is self occluded due to a single camera view point. We sample 256 points uniformly from the observed partial point cloud to generate 2048 points from the complete one using our POEx model. For comparison, we train a PCN (Yuan et al., 2018) using the same dataset. PCN is specially designed for the completion task and uses a multi-scale generation process. For quantitative comparison, we report the Chamfer Distance (CD) and Earth Mover's Distance (EMD) in Table 3. Despite the general purpose of our POEx model, we achieve comparable performance compared to PCN.

Table 3. Point cloud completion.

	CD	EMD
PCN	0.0033	0.1393
POEx	0.0044	0.0994

For point cloud upsampling, we use the ModelNet40 dataset. We uniformly sample 2048 points as the target and create a low resolution point cloud by uniformly sampling a subset. Note we use arbitrary sized subset during training. For evaluation, we upsample a point cloud with 256 points. We use PUNet (Yu et al., 2018) as the baseline, which is trained to upsample 256 points to 2048 points. Table 4 reports the CD and EMD between the upsampled point clouds and the ground truth. We can see our POEx model produces slightly higher distances, but we stress that our model is not specifically designed for this task, nor was it trained w.r.t. these metrics. We believe some task specific tricks, such as multi-scale generation and folding (Yang et al., 2018), can help improve the performance further, which we leave as future work. Similar to the image case, we can also generalize a pretrained POEx model to upsample point clouds in unseen categories.

Table 4. Point cloud upsampling.

		PUNet	POEx
Seen	CD	0.0025	0.0035
	EMD	0.0733	0.0880
Unseen	CD	0.0031	0.0048
	EMD	0.0793	0.1018

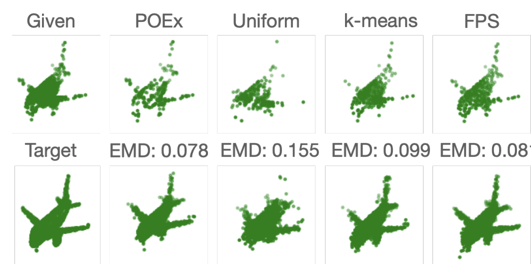


Figure 6. Point cloud compression. The EMD scores are calculated over the entire testset.

In contrast to upsampling, we propose using our POEx model to compress a given point cloud. Here we use a POEx model trained for airplane to summarize 2048 points into 256 points. To showcase the significance of leveraging set dependencies, we simulate a non-uniformly captured point cloud, where points close to the center have higher probability of being captured by the sensor. We expect the compressed point cloud to preserve the underlying geometry, thus we evaluate the distance between the recovered point cloud and a uniformly sampled one. Figure 6 compares the compression performance with several sampling approaches, where FPS represents the farthest point sampling (Qi et al., 2017). We can see the baselines tend to select center points more frequently, while POEx distributes the points more evenly. Quantitative results (Fig.6) verify the superiority of POEx for compression.

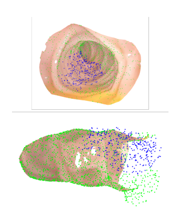


Figure 7. Impute missing points for colonoscopy data. Green: observed. Blue: imputed.

In addition to these synthetic point cloud data, we also evaluate on a real-world colonoscopy dataset. We uniformly sample 2048 points from the meshes and manually drop some points to simulate the blind spots. Our POEx model is then used to predict those missing points.

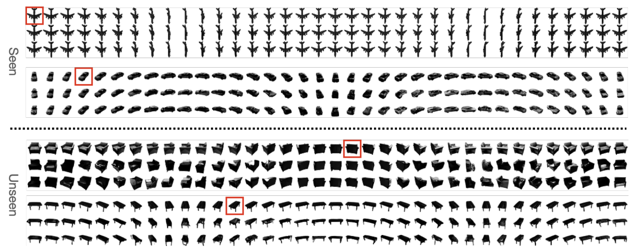
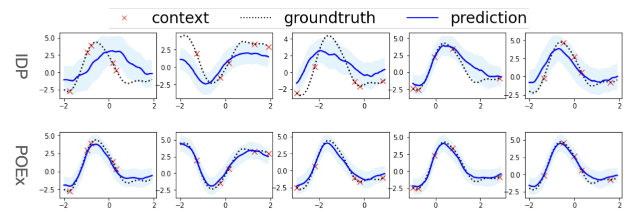


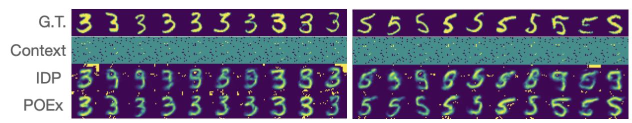
Figure 8. Neural processes on ShapeNet. First row: ground truth, red boxes indicate the context. Second row: predicted views given the context. Third row: predicted views from unseen angles.

(a) Occlusion removal (b) Youtube

Figure 9. Video inpainting. Better viewed with zoom-in.



(a) Multi-task Gaussian processes



(b) MNIST with 50 context points

Figure 10. Modeling a set of functions.

To provide guidance about where to fill in those missing points, we divide the space into small cubes and pair each point with its cube coordinates. Missing points are then predicted conditioned on their cube coordinates. Figure 7 presents several imputed point clouds mapped onto their corresponding meshes. We can see the imputed points align well with the meshes.

The conditional version of POEx can be viewed as a neural process which learns a distribution over functions. Instead of modeling low-dimensional functions, we model a process over images here. A subtle but important difference between NP and POEx is that POEx model the high-dimensional processes. Although NP models have been applied to images, they treat images as low-dimensional functions, where the input is the 2D pixel positions and the output is the corresponding pixel values. Instead, we consider domains over the $H \times W \times C$ dimensions. We evaluate on ShapeNet dataset (Chang et al., 2015), which is constructed by viewing the objects from different angles. Our POEx model takes several images from random angles as context and predicts the views for arbitrary angles. Figure 8 presents several examples for both seen and unseen categories from ShapeNet. We can see our POEx model generates sharp images and smoothly transits between different viewpoints given just one context view. Our model can also generalize to unseen categories. Conditional BRUNO (Korshunova et al., 2020) trained with the same dataset sometimes generates images not in the same class as the specified context, while POEx generation always matches with the context classes. Please see Fig. C.6 for additional examples. In Table 5, we report the bits per dimension (bpd) for generating a sequence of views given one context. Our model achieves lower bpd on both seen and unseen categories.

Table 5. Bpd for generating 10 views given one random view.

	Seen	Unseen
cBRUNO	1.43	1.62
POEx	**1.34**	**1.41**

With a conditional version of POEx, we can consider a collection of video frames conditioned on their timestamps as a set. Figure 9 shows the inpainting results on two video datasets from Liao et al. (2020) and Xu et al. (2018), and Table 6 reports the quantitative results. In addition to IDP, we compare to group mean imputation (GMI) (Sim et al., 2015) and TCC (Huang et al., 2016), which utilizes optical flow to infer the correspondence between adjacent frames. We can see POEx outperforms IDP and GMI. There is still room for improvement with the help of optical flow, but we leave it for future works. GMI works well only if the content in the video does not move much. TCC does not work when the missing rate is high due to the difficulty of estimating the optical flow. Please see Fig. C.7 and C.8 for additional examples.

Table 6. Video inpainting.

	Occlusion		Youtube	
	PSNR	SSIM	PSNR	SSIM
IDP	15.01	0.77	15.10	0.95
GMI	19.85	0.82	16.49	0.96
TCC	**31.35**	0.84	**30.18**	0.98
POEx	21.69	**0.92**	21.62	**0.99**

Further generalizing to the infinite dimensional set elements, we propose to model a set of functions using our POEx model. Similar to Neural Processes, we evaluate on Gaussian processes and simulated functions from images. Here, we use multi-task Gaussian processes (Bonilla et al., 2008). For functions based on images, a set of MNIST images from the same class is used so that the set of functions are correlated. Figure 10 present examples

Table 7. NLL for modeling a set of functions.

	MTGP	MNIST
IDP	2.04	-1.08
POEx	**1.79**	**-1.10**

of modeling a set of correlated functions. We can see our POEx model manages to recover the processes with low uncertainty using just a few context points, while the IDP model that treat each element independently fails. Table 7 reports the negative log likelihood (NLL), and POEx model obtains lower NLL on both datasets.

5. Conclusion

In this work, we develop the first model to work with sets of partially observed elements. Our POEx model captures the intra-instance and inter-instance dependencies in a holistic framework by modeling the conditional distributions of the unobserved part conditioned on the observed part. We further reinterpret various applications as partially observed set modeling tasks and apply POEx to solve them. POEx is versatile and performs well for many challenging tasks even compared with domain specific approaches. For future works, we will explore domain specific architectures and techniques to further improve the performance.

Acknowledgements

This work was supported in part by NIH 1R01AA026879-01A1. We would like to thank Professor Stephen M. Pizer for providing the colonoscopy images and guidance for preprocessing them.

References

Belghazi, M., Oquab, M., and Lopez-Paz, D. Learning about an exponential amount of conditional distributions. In *Advances in Neural Information Processing Systems*, pp. 13703–13714, 2019.

Bender, C., O'Connor, K., Li, Y., Garcia, J., Oliva, J., and Zaheer, M. Exchangeable generative models with flow scans. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 10053–10060, 2020.

Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan): 993–1022, 2003.

Blei, D. M., Lafferty, J. D., et al. A correlated topic model of science. *The Annals of Applied Statistics*, 1(1):17–35, 2007.

Bonilla, E. V., Chai, K. M., and Williams, C. Multi-task gaussian process prediction. In *Advances in neural information processing systems*, pp. 153–160, 2008.

Butz, C. J., Oliveira, J. S., dos Santos, A. E., and Teixeira, A. L. Deep convolutional sum-product networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3248–3255, 2019.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Douglas, L., Zarov, I., Gourgoulias, K., Lucas, C., Hart, C., Baker, A., Sahani, M., Perov, Y., and Johri, S. A universal marginalizer for amortized inference in generative models. *arXiv preprint arXiv:1711.00695*, 2017.

Edwards, H. and Storkey, A. Towards a neural statistician. *arXiv preprint arXiv:1606.02185*, 2016.

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

Garnelo, M., Rosenbaum, D., Maddison, C. J., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D. J., and Eslami, S. Conditional neural processes. *arXiv preprint arXiv:1807.01613*, 2018a.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

Gordon, J., Bruinsma, W. P., Foong, A. Y. K., Requeima, J., Dubois, Y., and Turner, R. E. Convolutional conditional neural processes. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Skey4eBYPS.

Griffiths, T. L. and Steyvers, M. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101 (suppl 1):5228–5235, 2004.

Heinze-Deml, C., Maathuis, M. H., and Meinshausen, N. Causal structure learning. *Annual Review of Statistics and Its Application*, 5:371–391, 2018.

Huang, J.-B., Kang, S. B., Ahuja, N., and Kopf, J. Temporally coherent completion of dynamic video. *ACM Transactions on Graphics (TOG)*, 35(6):1–11, 2016.

Ivanov, O., Figurnov, M., and Vetrov, D. Variational autoencoder with arbitrary conditioning. In *International Conference on Learning Representations*, 2018.

Jaini, P., Poupart, P., and Yu, Y. Deep homogeneous mixture models: representation, separation, and approximation. In *Advances in Neural Information Processing Systems*, pp. 7136–7145, 2018.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. Attentive neural processes. *arXiv preprint arXiv:1901.05761*, 2019.

Korshunova, I., Degrave, J., Huszár, F., Gal, Y., Gretton, A., and Dambre, J. Bruno: A deep recurrent model for exchangeable data. *Advances in Neural Information Processing Systems*, 31:7190–7198, 2018.

Korshunova, I., Gal, Y., Gretton, A., and Dambre, J. Conditional bruno: A neural process for exchangeable labelled data. *Neurocomputing*, 2020.

Koster, J. T. et al. Marginalizing and conditioning in graphical models. *Bernoulli*, 8(6):817–840, 2002.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pp. 3744–3753. PMLR, 2019.

Li, Y., Akbar, S., and Oliva, J. ACFlow: Flow models for arbitrary conditional likelihoods. In *Proceedings of the 37th International Conference on Machine Learning*, 2020a.

Li, Y., Yi, H., Bender, C., Shan, S., and Oliva, J. B. Exchangeable neural ode for set modeling. *Advances in Neural Information Processing Systems*, 33, 2020b.

Liao, J., Duan, H., Li, X., Xu, H., Yang, Y., Cai, W., Chen, Y., and Chen, L. Occlusion detection for automatic video editing. In *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 2255–2263, 2020.

Øksendal, B. Stochastic differential equations. In *Stochastic differential equations*, pp. 65–84. Springer, 2003.

Poon, H. and Domingos, P. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 689–690. IEEE, 2011.

Porteous, I., Newman, D., Ihler, A., Asuncion, A., Smyth, P., and Welling, M. Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 569–577, 2008.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. 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.

Rasmussen, C. E. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pp. 63–71. Springer, 2003.

Scutari, M., Graafland, C. E., and Gutiérrez, J. M. Who learns better bayesian network structures: Accuracy and speed of structure learning algorithms. *International Journal of Approximate Reasoning*, 115:235–253, 2019.

Shah, A., Wilson, A., and Ghahramani, Z. Student-t processes as alternatives to gaussian processes. In *Artificial intelligence and statistics*, pp. 877–885, 2014.

Sim, J., Lee, J. S., and Kwon, O. Missing values and optimal selection of an imputation method and classification algorithm to improve the accuracy of ubiquitous computing applications. *Mathematical problems in engineering*, 2015.

Tan, P. L. and Peharz, R. Hierarchical decompositional mixtures of variational autoencoders. In *International Conference on Machine Learning*, pp. 6115–6124, 2019.

Teh, Y., Newman, D., and Welling, M. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. *Advances in neural information processing systems*, 19:1353–1360, 2006a.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. Hierarchical dirichlet processes. *Journal of the american statistical association*, 101(476):1566–1581, 2006b.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Vinyals, O., Bengio, S., and Kudlur, M. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

Wang, H., Liu, Q., Yue, X., Lasenby, J., and Kusner, M. J. Pre-training by completing point clouds. *arXiv preprint arXiv:2010.01089*, 2020.

Wang, W., Aggarwal, V., and Aeron, S. Efficient low rank tensor ring completion. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5697–5705, 2017.

Xu, N., Yang, L., Fan, Y., Yue, D., Liang, Y., Yang, J., and Huang, T. Youtube-vos: A large-scale video object segmentation benchmark. *arXiv preprint arXiv:1809.03327*, 2018.

Yang, G., Huang, X., Hao, Z., Liu, M.-Y., Belongie, S., and Hariharan, B. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4541–4550, 2019.

Yang, M., Dai, B., Dai, H., and Schuurmans, D. Energy-based processes for exchangeable data. *arXiv preprint arXiv:2003.07521*, 2020.

Yang, Y., Feng, C., Shen, Y., and Tian, D. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 206–215, 2018.

Yu, L., Li, X., Fu, C.-W., Cohen-Or, D., and Heng, P.-A. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2790–2799, 2018.

Yuan, W., Khot, T., Held, D., Mertz, C., and Hebert, M. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pp. 728–737. IEEE, 2018.