Variational Flow Graphical Model

Anonymous Author(s)*

ABSTRACT

This paper introduces a novel approach to embed flow-based models with hierarchical structures. The proposed model learns the representation of high dimensional data via a message-passing scheme by integrating flow-based functions through variational inference. Meanwhile, our model produces a representation of the data using a lower dimension, thus overcoming the drawbacks of many flow-based models, usually requiring a high dimensional latent space involving many trivial variables. With the proposed aggregation nodes, the model provides a new approach for distribution modeling and numerical inference on datasets. Multiple experiments on synthetic and real datasets show the benefits of our proposed method.

ACM Reference Format:
Anonymous Author(s). 2022. Variational Flow Graphical Model. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (SIGKDD'22).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Learning tractable distribution or density functions from datasets has broad applications. Probabilistic graphical models (PGMs) provide a unifying framework for capturing complex dependencies among random variables [34]. There are two general approaches for probabilistic inference with PGMs and other models: exact inference and approximate inference. In most cases, exact inference is either computationally involved or simply intractable. Variational inference (VI) is computationally efficient and is applied to tackle large-scale inference problems [11, 13]. In variational inference, mean-field approximation [37] and variational message passing [3, 36] are two common approaches. These methods are limited by the choice of distributions that are inherently unable to recover the true posterior, often leading to a loose approximation.

To tackle the probabilistic inference problem, alternative models have been developed under the name of *tractable probabilistic models (TPMs)*. These models include probabilistic decision graphs [12], arithmetic circuits [6], and-or search spaces [21], multi-valued decision diagrams [7], cutset networks [25], sum-product nets [28], probabilistic sentential decision diagrams [17], and probabilistic circuits [5]. Probabilistic circuits (PCs) leverage the recursive mixture models and distributional factorization to establish tractable probabilistic inference. PCs also aim to attain a TPM with improved expressive power.

SIGKDD'22, August 14-18, 2022, Washington, DC

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

Apart from probabilistic inference, generative models have been developed to model high dimensional datasets and to learn meaningful hidden data representations by leveraging the approximation power of neural networks. These models also provide a possible approach to generate new samples from underlining distributions. Variational Auto-Encoders (VAEs) [16] and Generative Adversarial Networks (GAN) [10] are widely applied to different categories of datasets. Flow-based models [2, 8, 9, 26] leverage invertible neural networks and can estimate the density values of data samples as well. Unlike TPMs, it is usually difficult to directly use generative models to perform probabilistic inference on datasets.

In this paper, we introduce Variational Flow Graphical (VFG) models. By leveraging the expressive power of neural networks, VFGs can learn latent representations from data. VFGs also follow the stream of tractable neural networks that allow to perform inference on graphical structures. Sum-product networks [28] and probabilistic circuits [5] are falling into this type of models as well. Sum-product networks and probabilistic circuits depend on mixture models and probabilistic factorization in graphical structure for inference. VFGs rely on the consistency of aggregation nodes in graphical structures to achieve tractable inference. Our contributions are summarized as follows.

Contributions. Dealing with high dimensional data using graph structures exacerbates the systemic inability for effective distribution modeling and efficient inference. To overcome these limitations, we propose the VFG model to achieve the following goals:

- **Hierarchical and Flow-Based:** VFG is a novel graphical architecture uniting the hierarchical latent structures and flow-based models. Our model outputs a tractable posterior distribution used as an approximation of the true posterior of the hidden node states in the considered graph structure.
- **Distribution Modeling:** Our theoretical analysis shows that VFGs are universal approximators. In the experiments, VFGs can achieve improved evidence lower bound (ELBO) and likelihood values by leveraging the implicitly invertible flow-based model structure.
- **Numerical Inference:** Aggregation nodes are introduced in the model to integrate hierarchical information through a variational forward-backward message passing scheme. We highlight the benefits of our VFG model on applications: the missing entry imputation problem and the numerical inference on graphical data.

Moreover, experiments show that our model achieves to disentangle the factors of variation underlying high dimensional input data.

Roadmap: Section 2 presents important concepts used in the paper. Section 3 introduces the Variational Flow Graphical (VFG) model. The approximation property of VFGs is discussed in Section 4. Section 5 provides the algorithms used to train VFG models. Section 6 discusses how to perform inference with a VFG model. Section 7 showcases the advantages of VFG on various tasks. Section 8 and Section 9 provide a discussion and conclusion of the paper.

Notation: We use $[L]$ to denote the set $\{1, \cdots, L\}$, for all $L > 1$. $\mathbf{KL}(p||q) := \int_{\mathcal{Z}} p(z) \log(p(z)/q(z))\mathrm{d}z$ is the Kullback-Leibler divergence from q to p, two probability density functions defined on the set $\mathcal{Z} \subset \mathbb{R}^m$ for any dimension $m > 0$.

2 PRELIMINARIES

We introduce the general principles and notations of variational inference and flow-based models in this section.

Variational Inference: Following the setting discussed above, the functional mapping $\mathbf{f} : \mathcal{Z} \to \mathcal{X}$ can be viewed as a decoding process and the mapping $\mathbf{f}^{-1}: \mathcal{X} \to \mathcal{Z}$ as an encoding one between random variables $\mathbf{z} \in \mathcal{Z}$ and $\mathbf{x} \in \mathcal{X}$ with densities $\mathbf{z} \sim p(\mathbf{z}), \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$. To learn the parameters θ, VI employs a parameterized family of so-called variational distributions $q_\phi(\mathbf{z}|\mathbf{x})$ to approximate the true posterior $p(\mathbf{z}|\mathbf{x}) \propto p(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})$. The optimization problem of VI can be shown to be equivalent to maximizing the following evidence lower bound (ELBO) objective, noted $\mathcal{L}(\mathbf{x}; \theta, \phi)$:

$$\log p(\mathbf{x}) \geqslant \tag{1}$$
$$\mathcal{L}(\mathbf{x}; \theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] - \mathbf{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))\,.$$

In Variational Auto-Encoders (VAEs, [16, 27]), the calculation of the reconstruction term requires sampling from the posterior distribution along with using the reparameterization trick, i.e.,

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] \simeq \frac{1}{U}\sum_{u=1}^{U} \log p(\mathbf{x}|\mathbf{z}_u). \tag{2}$$

Here U is the number of latent variable samples drawn from the posterior $q_\phi(\mathbf{z}|\mathbf{x})$ regarding data $\mathbf{x}$.

Flow-based Models: Flow-based models [2, 8, 9, 26] correspond to a probability distribution transformation using a sequence of invertible and differentiable mappings, noted $\mathbf{f} : \mathcal{Z} \to \mathcal{X}$. By defining the aforementioned invertible maps $\{\mathbf{f}_\ell\}_{\ell=1}^{L}$, and by the chain rule and inverse function theorem, the variable $\mathbf{x} = \mathbf{f}(\mathbf{z})$ has a tractable probability density function (pdf) given as:

$$\log p_\theta(\mathbf{x}) = \log p(\mathbf{z}) + \sum_{i=1}^{L} \log \left|\det\left(\frac{\partial \mathbf{h}^i}{\partial \mathbf{h}^{i-1}}\right)\right|, \tag{3}$$

where we have $\mathbf{h}^0 = \mathbf{x}$ and $\mathbf{h}^L = \mathbf{z}$ for conciseness. The scalar value $\log|\det(\partial \mathbf{h}^i/\partial \mathbf{h}^{i-1})|$ is the logarithm of the absolute value of the determinant of the Jacobian matrix $\partial \mathbf{h}^i/\partial \mathbf{h}^{i-1}$, also called the log-determinant. Eq. (3) yields a simple mechanism to build families of distributions that, from an initial density and a succession of invertible transformations, returns tractable density functions that one can sample from. [26] propose an approach to construct flexible posteriors by transforming a simple base posterior with a sequence of flows. Firstly a stochastic latent variable is draw from base posterior $\mathcal{N}(\mathbf{z}_0|\mu(\mathbf{x}), \sigma(\mathbf{x}))$. With K flows, latent variable $\mathbf{z}_0$ is transformed to $\mathbf{z}_k$.The reformed EBLO is given by

$$\mathcal{L}(\mathbf{x}; \theta, \phi) = \mathbb{E}_{q_\phi}\big[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\big] =$$
$$\mathbb{E}_{q_0}\big[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_0(\mathbf{z}_0|\mathbf{x})\big] + \mathbb{E}_{q_0}\Big[\sum_{k=1}^{K} \log\Big|\det\Big(\frac{\partial \mathbf{f}_k(\mathbf{z}_k; \psi_k)}{\partial \mathbf{z}_k}\Big)\Big|\Big].$$

Here $\mathbf{f}_k$ is the k-th flow with parameter ψ_k, i.e., $\mathbf{z}_K = \mathbf{f}_K \circ \cdots \mathbf{f}_2 \circ \mathbf{f}_1(\mathbf{z}_0)$. The flows are considered as functions of data sample $\mathbf{x}$, and they determine the final distribution in amortized inference. Several recent models have been proposed by leveraging the invertible flow-based models. Graphical normalizing flow [35] learns a DAG structure from the input data under sparse penalty and maximum likelihood estimation. The bivariate causal discovery method proposed in [14] relies on autoregressive structure of flow-based models and the asymmetry of log-likelihood ratio for cause-effect pairs. In this paper, we propose a framework that generalizes flow-based models [2, 8, 9, 26] to graphical variable inference.

3 VARIATIONAL FLOW GRAPHICAL MODEL

Assume $\mathbf{k}$ sections in the data samples, i.e. $\mathbf{x} = [\mathbf{x}^{(1)}, ..., \mathbf{x}^{(k)}]$, and a relationship among these sections and the corresponding latent variable. Then, it is possible to define a graphical model using normalizing flows, as introduced Section 2, leading to exact latent variable inference and log-likelihood evaluation of data samples.

A VFG model $\mathbb{G} = \{\mathcal{V}, \mathbf{f}\}$ consists of a node set ($\mathcal{V}$) and an edge set ($\mathbf{f}$). An edge can be either a flow function or an identity function. There are two types of nodes in a VFG: *aggregation* nodes and *non-aggregation* nodes. A non-aggregation node connects with another node with a flow function or an identity function. An aggregation node has multiple children, and it connects each of them with an identity function. Figure 1-Left gives an illustration of an aggregation node and Figure 1-Right shows a tree VFG model. Unlike classical graphical models, a node in a VFG model may represent a single variable or multiple variables. Moreover, each latent variable belongs to only one node in a VFG. In the following sections of this paper, identity function is considered as a special case of flow functions.

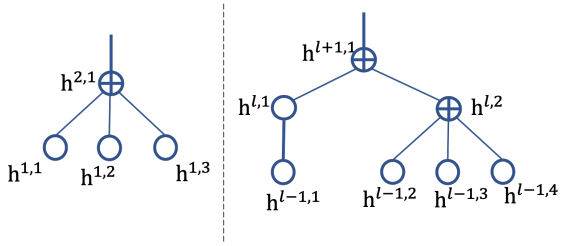

Figure 1: (Left) Node $\mathbf{h}^{2,1}$ connects its children with invertible functions. Messages from the children are aggregated at the parent node, $\mathbf{h}^{2,1}$. (Right) An illustration of the latent structure from layer $l-1$ to $l+1$. Thin lines are identity functions, and thick lines are flow functions. $\oplus$ is an aggregation node, and circles stand for non-aggregation nodes.

3.1 Evidence Lower Bound of VFGs

We apply variational inference to learn model parameters θ from data samples. Different from VAEs [16, 27], the recognition model (encoder) and the generative model (decoder) in a VFG share the same neural net structure and parameters. Moreover, the latent variables in a VFG lie in a hierarchy structure and are generated with deterministic flow functions.

We start with a tree VFG (Figure 2) to introduce the ELBO of the model. The hierarchical tree structure comprises L layers, $\mathbf{h}^l$ denotes the latent state in layer l of the tree. We use $\mathbf{h}^{(j)}$ to represent node j's latent state without specification of the layer number, and j is the node index in a tree or graph. The joint distribution for the

hierarchical model is then

$$p_\theta(\mathbf{x},\mathbf{h}) = p(\mathbf{h}^L)p(\mathbf{h}^{L-1}|\mathbf{h}^L)\cdots p(\mathbf{h}^1|\mathbf{h}^2)p(\mathbf{x}|\mathbf{h}^1)\,.$$

where $\mathbf{h} = \{\mathbf{h}^1, \cdots, \mathbf{h}^L\}$ denotes the set of latent states of the model. The hierarchical generative model is given by factorization $p(\mathbf{x}|\mathbf{h}^L) = p(\mathbf{x}|\mathbf{h}^1)\Pi_{l=1}^{L-1}p(\mathbf{h}^l|\mathbf{h}^{l+1})$, and the prior distribution is $p(\mathbf{h}^L)$. Note that only the *root nodes* have *prior distributions*. The probabilistic density function $p(\mathbf{h}^{l-1}|\mathbf{h}^l)$ in the generative model is parameterized with one or multiple invertible flow functions. By leveraging the invertible flow functions, we use variational inference to approximate the posterior distribution of latent states. The hierarchical posterior (recognition model) is factorized as

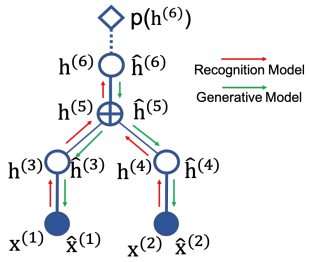

Figure 2: forward message from data to approximate posterior distributions; generative model is realized by backward message from the root node and generates the samples or reconstructions at each layer.

$$q_\theta(\mathbf{h}|\mathbf{x}) = q(\mathbf{h}^1|\mathbf{x})q(\mathbf{h}^2|\mathbf{h}^1)\cdots q(\mathbf{h}^L|\mathbf{h}^{L-1}). \tag{4}$$

Evaluation of the posterior (recognition model) (4) involves forward information flows from the bottom of the tree to the top, and similarly, sampling the generative model takes the reverse direction.

By leveraging the hierarchical conditional independence in both generative model and posterior, the ELBO regarding the model is

$$\log p_\theta(\mathbf{x}) \geqslant \mathcal{L}(\mathbf{x};\theta) = \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log p(\mathbf{x}|\mathbf{h}^{1:L})\big] - \sum_{l=1}^{L}\mathbf{KL}^l. \tag{5}$$

Here $\mathbf{KL}^l$ is the Kullback-Leibler divergence between the posterior and generative model in layer l. The first term in (5) evaluates data reconstruction. When $1 \leqslant l \leqslant L$,

$$\mathbf{KL}^l = \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log q(\mathbf{h}^l|\mathbf{h}^{l-1}) - \log p(\mathbf{h}^l|\mathbf{h}^{l+1})\big]. \tag{6}$$

When $l = L$, $\mathbf{KL}^L = \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log q(\mathbf{h}^L|\mathbf{h}^{L-1}) - \log p(\mathbf{h}^L)\big]$. It is easy to extend the computation of the ELBO (5) to DAGs with topology ordering of the nodes (and thus of the layers). Let $ch(i)$ and $pa(i)$ denote node i's child set and parent set, respectively. Then, the ELBO for a DAG structure reads:

$$\begin{aligned}\mathcal{L}(\mathbf{x};\theta) =& \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\big[\log p(\mathbf{x}|\mathbf{h})\big] - \sum_{i\in\mathcal{V}\setminus\mathcal{R}_\mathbb{G}}\mathbf{KL}^{(i)} \\ &- \sum_{i\in\mathcal{R}_\mathbb{G}}\mathbf{KL}\big(q(\mathbf{h}^{(i)}|\mathbf{h}^{ch(i)})||p(\mathbf{h}^{(i)})\big).\end{aligned} \tag{7}$$

Here $\mathbf{KL}^{(i)} = \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\big[\log q(\mathbf{h}^{(i)}|\mathbf{h}^{ch(i)}) - \log p(\mathbf{h}^{(i)}|\mathbf{h}^{pa(i)})\big]$. $\mathcal{R}_\mathbb{G}$ is the set of root nodes of DAG $\mathbb{G} = \{\mathcal{V}, \mathbf{f}\}$. Assuming there are k leaf nodes on a tree or a DAG model, corresponding to k sections of the input sample $\mathbf{x} = [\mathbf{x}^{(1)}, ..., \mathbf{x}^{(k)}]$.

Maximizing the ELBO (5) or (7) equals to optimizing the parameters of the flows, θ. Similar to VAEs [16, 27], we apply forward message passing (encoding) to approximate the posterior distribution of each layer's latent variables, and backward message passing (decoding) to generate the reconstructions as shown in Figure 2. For the following sections, we use $\mathbf{h}^i$ to represent node i's state in the forward message, and $\widehat{\mathbf{h}}^i$ for node i's state in the backward message. For all nodes, both $\mathbf{h}^i$ and $\widehat{\mathbf{h}}^i$ are sampled from the posterior. At the rood nodes, we have $\widehat{\mathbf{h}}^{\mathcal{R}} = \mathbf{h}^{\mathcal{R}}$.

3.2 Aggregation Nodes

There are two approaches to aggregate signals from different nodes: average-based and concatenation-based. We rather focus on average-based aggregation in this paper, and Figure 3 gives an example denoted by the operator $\oplus$. Let $\mathbf{f}_{(i,j)}$ be the direct edge (function) from node i to node j, and $\mathbf{f}^{-1}_{(i,j)}$ or $\mathbf{f}_{(j,i)}$ defined as its inverse function. Then, the aggregation operation at node i reads

$$\mathbf{h}^{(i)} = \frac{1}{|ch(i)|}\sum_{j\in ch(i)}\mathbf{f}_{(j,i)}(\mathbf{h}^{(j)}),\ \widehat{\mathbf{h}}^{(i)} = \frac{1}{|pa(i)|}\sum_{j\in pa(i)}\mathbf{f}_{(j,i)}(\widehat{\mathbf{h}}^{(j)})\,. \tag{8}$$

Notice that the above two equations hold even when node i has only one child or parent.

With the identity function between the parent and its children, there are *node consistency rules* regarding an average aggregation node: *(a)* a parent node's backward state equals the mean of its children's forward states, i.e., $\widehat{\mathbf{h}}^{(i)} = \frac{1}{|ch(i)|}\sum_{j\in ch(i)}\mathbf{h}^{(j)}$; *(b)* a child node's forward state equals to the average of its parents' backward states, i.e., $\mathbf{h}^{(i)} = \frac{1}{|pa(i)|}\sum_{j\in pa(i)}\widehat{\mathbf{h}}^{(j)}$. These rules empower VFGs with implicit invertibility.

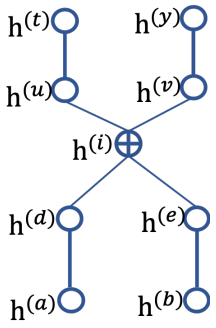

Figure 3: Aggregation node on a DAG VFG.

We use aggregation node i in the DAG presented in Figure 3 as an example to illustrate node consistency. Node i has two parents, u and v; and two children, d and e. Node i connects its parents and children with identity functions. According to (8), we have $\mathbf{h}^{(i)} = (\mathbf{h}^{(d)} + \mathbf{h}^{(e)})/2$ and $\widehat{\mathbf{h}}^{(i)} = (\widehat{\mathbf{h}}^{(u)} + \widehat{\mathbf{h}}^{(v)})/2$. Here aggregation *consistency* means, for i's children, their forward state should be consistent with i's backward state, i.e.,

$$\mathbf{h}^{(d)} = \mathbf{h}^{(e)} = \widehat{\mathbf{h}}^{(i)}. \tag{9}$$

For i's parents, their backward state should be consistent with i's forward state, i.e.,

$$\widehat{\mathbf{h}}^{(u)} = \widehat{\mathbf{h}}^{(v)} = \mathbf{h}^{(i)}. \tag{10}$$

We utilize the $\mathbf{KL}$ term in the ELBO (7) to ensure (9) and (10) can be satisfied during parameter updating. The $\mathbf{KL}$ term regarding node i is

$$\begin{aligned}\mathbf{KL}^{(i)} =& \mathbb{E}_{q(\mathbf{h},\widehat{\mathbf{h}}|\mathbf{x})}\big[\log q(\mathbf{h}^{(i)}|\mathbf{h}^{ch(i)}) - \log p(\mathbf{h}^{(i)}|\widehat{\mathbf{h}}^{pa(i)})\big] \\ \simeq& \log q(\mathbf{h}^{(i)}|\mathbf{h}^{ch(i)}) - \log p(\mathbf{h}^{(i)}|\widehat{\mathbf{h}}^{pa(i)}).\end{aligned} \tag{11}$$

As the term $\log q(\mathbf{h}^{(i)}|\mathbf{h}^{ch(i)})$ involves node states that are deterministic according to (8), it is omitted in the computation of (11). With Laplace as the latent state distribution, here

$$\begin{aligned}&\log p(\mathbf{h}^{(i)}|\widehat{\mathbf{h}}^{pa(i)})\\ =&\frac{1}{2}\big(\log p(\mathbf{h}^{(i)}|\widehat{\mathbf{h}}^{(u)}) + p(\mathbf{h}^{(i)}|\widehat{\mathbf{h}}^{(v)})\big)\\ =&\frac{1}{2}\big(-\|\mathbf{h}^{(i)}-\widehat{\mathbf{h}}^{(u)}\|_1 - \|\mathbf{h}^{(i)}-\widehat{\mathbf{h}}^{(v)}\|_1 - 2m\cdot\log 2\big).\end{aligned}$$

Hence minimizing $\mathbf{KL}^{(i)}$ is equal to minimizing $\{\|\mathbf{h}^{(i)}-\widehat{\mathbf{h}}^{(u)}\|_1 + \|\mathbf{h}^{(i)}-\widehat{\mathbf{h}}^{(v)}\|_1\}$ which achieves the consistent objective in (10).

Similarly, $\mathbf{KL}$s of i's children intend to realize consistency given in (9). We use node d as an example. The $\mathbf{KL}$ term regarding node d is

$$\begin{aligned}\mathbf{KL}^{(d)} =&\mathbb{E}_{q(\mathbf{h},\widehat{\mathbf{h}}|\mathbf{x})}\big[\log q(\mathbf{h}^{(d)}|\mathbf{h}^{ch(d)}) - \log p(\mathbf{h}^{(d)}|\widehat{\mathbf{h}}^{pa(d)})\big]\\ \simeq&\log q(\mathbf{h}^{(d)}|\mathbf{h}^{ch(d)}) - \log p(\mathbf{h}^{(d)}|\widehat{\mathbf{h}}^{pa(d)}).\end{aligned}$$

The first term $\log q(\mathbf{h}^{(d)}|\mathbf{h}^{ch(d)})$ is omitted in the calculation of $\mathbf{KL}^{(d)}$ due to the deterministic relation with (8). Knowing that

$$\log p(\mathbf{h}^{(d)}|\widehat{\mathbf{h}}^{pa(d)}) = \log p(\mathbf{h}^{(d)}|\widehat{\mathbf{h}}^{(i)}) = -\|\mathbf{h}^{(d)}-\widehat{\mathbf{h}}^{(i)}\|_1 - m\cdot\log 2,$$

we notice that minimizing $\mathbf{KL}^{(d)}$ boils down to minimizing $\|\mathbf{h}^{(d)}-\widehat{\mathbf{h}}^{(i)}\|_1$ that targets at (9). In summary, by maximizing the ELBO of a VFG, the aggregation consistency can be attained along with fitting the model to the data.

3.3 Implementation Details

The calculation of the data reconstruction term in (7) requires node states $\mathbf{h}^i$ and $\widehat{\mathbf{h}}^i$ ($\forall i \in \mathcal{V}$) from the posterior. It corresponds to the encoding and decoding procedures in VAE model [16, 27] as given by Eq. (2). At the root node, we have $\widehat{\mathbf{h}}^{\mathcal{R}} = \mathbf{h}^{\mathcal{R}}$. The reconstruction term in ELBO (7) can be computed with the backward message from the generative model $p(\mathbf{x}|\widehat{\mathbf{h}}^1)$, i.e.,

$$\begin{aligned}\mathbb{E}_{q(\mathbf{h},\widehat{\mathbf{h}}|\mathbf{x})}\big[\log p(\mathbf{x}|\mathbf{h},\widehat{\mathbf{h}})\big] \simeq& \frac{1}{U}\sum_{u=1}^{U}\log p(\mathbf{x}|\widehat{\mathbf{h}}_u^{1:L})\\ =&\frac{1}{U}\sum_{u=1}^{U}\log p(\mathbf{x}|\widehat{\mathbf{h}}_u^{pa(x)}).\end{aligned}$$

For a VFG model, we set $U=1$. In the last term, $p(\mathbf{x}|\widehat{\mathbf{h}}^{pa(x)})$ is either Gaussian or binary distribution parameterized with $\widehat{\mathbf{x}}$ generated via the flow function with $\widehat{\mathbf{h}}^{pa(x)}$ as the input.

4 UNIVERSAL APPROXIMATION PROPERTY

A universal approximation power of coupling-layer based flows has been highlighted in [31]. Following the analysis for flows [31], we can prove that coupling-layer based VFGs have universal approximation as well. We first give several additional definitions regarding universal approximation. For a measurable mapping $\mathbf{f}:\mathbb{R}^m\to\mathbb{R}^n$ and a subset $K\subset\mathbb{R}^m$, we define the following,

$$||\mathbf{f}||_{p,K} = \left(\int_K ||f(x)||^p dx\right)^{1/p}.$$

Here $||\cdot||$ is the Euclidean norm of $\mathbb{R}^n$. We also define $||\mathbf{f}||_{\sup,K} := \sup_{x\in K}||\mathbf{f}(x)||$.

Definition 4.1. (L^p-/sup-universality) Let $\mathcal{M}$ be a model which is a set of measurable mappings from $\mathbb{R}^m$ to $\mathbb{R}^n$. Let $p\in[1,\infty)$, and let $\mathcal{G}$ be a set of measurable mappings $\mathbf{g}: U_{\mathbf{g}}\to\mathcal{R}^n$, where $U_{\mathbf{g}}$ is a measurable subset of $\mathbb{R}^m$ which may depend on $\mathbf{g}$. We say that $\mathcal{M}$ has the L^p-universal approximation property for $\mathcal{G}$ if for any $\mathbf{g}\in\mathcal{G}$, any $\epsilon>0$, and any compact subset $K\in U_{\mathbf{g}}$, there exists $\mathbf{f}\in\mathcal{M}$ such that $||\mathbf{f}-\mathbf{g}||_{p,K}<\epsilon$. We define the sup-universality analogously by replacing $||\cdot||_{p,K}||$ with $||\cdot||_{p,K}||_{sup,K}$.

Definition 4.2. (Immersion and submanifold) $\mathbf{g}:\mathfrak{M}\to\mathfrak{N}$ is said to be an immersion if rank(g)= m =dim($\mathfrak{M}$) everywhere. If $\mathbf{g}$ is injective (one-to-one) immersion, then $\mathbf{g}$ establish an one-to-one correspondence of $\mathfrak{M}$ and the subset $\tilde{\mathfrak{M}}=\mathbf{g}(\mathfrak{M})$ of $\mathfrak{N}$. If we use this correspondence to endow $\tilde{\mathfrak{M}}$ with a topology and C^∞ structure, then $\tilde{\mathfrak{M}}$ will be called a submanifold (or immersed submanifold) and $\mathbf{g}:\mathfrak{M}\to\tilde{\mathfrak{M}}$ is a diffeomorphism.

Definition 4.3. (C^r-diffeomorphisms for submanifold: $\mathcal{Q}^r$). We define $\mathcal{Q}^r$ as the set of all C^r-diffeomorphisms $\mathbf{g}:U_{\mathbf{g}}\to\mathfrak{U}$, where $U_{\mathbf{g}}\subset\mathbb{R}^m$ is an open set C^r-diffeomorphic to $\mathfrak{U}$, which may depend on $\mathbf{g}$, and $\mathfrak{U}$ is a submanifold of $\mathbb{R}^n$.

We use m to represent the root dimension of a VFG, and n to denote the dimension of data samples. VFGs learn the data manifold embedded in $\mathbb{R}^n$. We define $C_c^\infty(\mathbb{R}^{m-1})$ as the set of all compactly-supported C^∞ mappings from $\mathbb{R}^{m-1}$ to $\mathbb{R}$. For a function set $\mathcal{T}$, we define $\mathcal{T}$-ACF as the set of affine coupling flows [31] that are assembled with functions in $\mathcal{T}$, and we use $\text{VFG}_{\mathcal{T}-ACF}$ to represent the set of VFGs constructed using flows in $\mathcal{T}$-ACF.

Theorem 4.4. *(L^p-universality) Let $p\in[0,\infty)$. Assume $\mathcal{H}$ is a sup-universal approximator for $C_c^\infty(\mathbb{R}^{m-1})$, and that it concists of C^1-functions. Then $\text{VFG}_{\mathcal{H}-ACF}$ is an L^p-universal approximator for $\mathcal{Q}_c^0$.*

Proof. We construct a VFG structure that forms a mapping from $\mathbb{R}^m$ to $\mathbb{R}^n$. Let $r = n \mod m$.

If $r=0$, it is easy to construct a one-layer tree VFG $\mathbf{f}$ with the root as an aggregation node. The children divide the n input entries into $\tau=n/m$ even sections, and each section connects the aggregation node with a flow function.

Given an injective immersion $\mathbf{g}:\mathfrak{M}\to\mathfrak{N}$, function $\mathbf{g}$ can be represented with the concatenation of a set of functions, i.e., $\mathbf{g}=[\mathbf{g}_1,...,\mathbf{g}_\tau]^\top$, each invertible $\mathbf{g}_i$ has dimension m. According to the function decomposition theory [18], its inverse can be represent as the summation of functions $\mathbf{g}_i^{-1}, 1\le i\le\tau$, i.e., $\mathbf{g}^{-1}=\frac{1}{\tau}\sum_{i=1}^{\tau}\mathbf{g}_i^{-1}$. For each $\mathbf{g}_i$, and $\tilde{\mathfrak{M}}_i=\mathbf{g}_i(\mathfrak{M})$ is a submanifold in $\mathfrak{N}$, and it is diffeomorphic to $\mathfrak{M}$. According to Theorem 2 in [31], $\mathcal{H}-ACF$ is an universal approximater for each $\mathbf{g}_i, 1\le i\le\tau$. Therefore, VFG $\mathbf{f}$ has universal approximation for immersion $\mathbf{g}:\mathfrak{M}\to\mathfrak{N}$.

If $r\neq 0$, let $\tau=\lfloor n/m\rfloor$. We divide the τ-th section and the remaining r entries into two equal small sections that are denoted with τ and $\tau+1$. Sections τ and $\tau+1$ have r overlapped entries. Similarly, we can construct an one-layer VFG $\mathbf{f}$ with $\tau+1$ children, and each child takes a section as the input.

The input coordinate index of $\mathbf{g}_\tau$ in $\mathbb{R}^m$ is $I_\tau=\big[1,2,...,\lceil(m+r)/2\rceil\big]$, and the out put index of $\mathbf{g}_\tau$ in $\mathbb{R}^n$ is $I_\tau+\gamma=\big[\gamma+1,\gamma+2,...,\gamma+\lceil(m+r)/2\rceil\big]$, and $\gamma=(\tau-1)m$. The input coordinate index of $\mathbf{g}_{\tau+1}$ in $\mathbb{R}^m$ is $I_{\tau+1}=\big[m-\lceil(m+r)/2\rceil+1,...,m-1,m\big]$, and the out put index of $\mathbf{g}_{\tau+1}$ in $\mathbb{R}^n$ is $I_{\tau+1}+\gamma$. We can see that

the m dimension is divided into two sets, the overlapped set $O = \lfloor m-\lceil (m+r)/2 \rceil+1, \lceil (m+r)/2 \rceil \rfloor$, and the remaining set R containing the rest dimensions.

The mapping $\mathbf{g} : \mathfrak{M} \to \mathfrak{R}$ can be decomposed into $\tau+1$ functions, i.e., $\mathbf{g} = [\mathbf{g}_1, ..., \mathbf{g}_\tau, \mathbf{g}_{\tau+1}]^\top$, and the inverse $\mathbf{g}^{-1}$ is adjusted here: $\mathbf{g}_j^{-1} = \frac{1}{\omega} \sum_{i=1}^{\omega} \mathbf{g}_{i(j)}^{-1}$. When $j \in O$, $\omega = \tau + 1$, and all $\mathbf{g}_i^{-1}$s will be involved; when $j \in R$, $\omega = \tau$, and either $\mathbf{g}_\tau^{-1}$ or $\mathbf{g}_{\tau+1}^{-1}$ is omitted due to the missing of entry j in the function output.

The mapping $\mathbf{g}_\tau$ is a diffeomorphism from manifold $\mathfrak{M}_\tau$ ($\mathfrak{M}_\tau \subset \mathfrak{M}$) to sub-manifold $\tilde{\mathfrak{M}}_\tau$ in $\mathfrak{R}$. Similarly $\mathbf{g}_{\tau+1}$ is a diffeomorphism from $\mathfrak{M}_{\tau+1}$ to manifold $\tilde{\mathfrak{M}}_{\tau+1}$. For each $\mathbf{g}_i$, $1 \le i \le \tau+1$, it can be universally approximated with a function in $\mathcal{H} - ACF$ [31]. Hence, we can construct a VFG with universal approximation for any $\mathbf{g}$ in $\mathcal{Q}_c^0$. □

With the conditions in Theorem 4.4, $\text{VFG}_{\mathcal{H}-ACF}$ is a distributional universal approximator as well [31].

5 ALGORITHM

In this section, we develop the training algorithm (Algorithm 1) to maximize the ELBO objective function (7). In Algorithm 1, the inference of the latent states is performed via forwarding message passing, cf. Line 6, and their reconstructions are computed in backward message passing, cf. Line 11. A VFG is a deterministic network passing latent variable values between nodes. Ignoring explicit neural network parameterized variances for all latent nodes enables us to use flow-based models as both the encoders and decoders. Hence, we obtain a deterministic ELBO objective (5)- (7) that can efficiently be optimized with standard stochastic optimizers.

Algorithm 1 Inference model parameters with forward and backward message propagation

1: **Input:** Data distribution $\mathcal{D}$, $\mathbb{G} = \{\mathcal{V}, \mathbf{f}\}$
2: **for** $s = 0, 1, ...$ **do**
3: Sample minibatch b samples $\{\mathbf{x}_1, ..., \mathbf{x}_b\}$ from $\mathcal{D}$;
4: **for** $i \in \mathcal{V}$ **do**
5: // forward message passing
6: $\mathbf{h}^{(i)} = \frac{1}{|ch(i)|} \sum_{j \in ch(i)} \mathbf{f}_{(j,i)}(\mathbf{h}^{(j)})$;
7: **end for**
8: $\widehat{\mathbf{h}}^{(i)} = \mathbf{h}^{(i)}$ if $i \in \mathcal{R}_\mathbb{G}$ or $i \in$ layer L;
9: **for** $i \in \mathcal{V}$ **do**
10: // backward message passing
11: $\widehat{\mathbf{h}}^{(i)} = \frac{1}{|pa(i)|} \sum_{j \in pa(i)} \mathbf{f}_{(i,j)}^{-1}(\widehat{\mathbf{h}}^{(j)})$;
12: **end for**
13: $\mathbf{h} = \{\mathbf{h}^{(t)} | t \in \mathcal{V}\}$, $\widehat{\mathbf{h}} = \{\widehat{\mathbf{h}}^{(t)} | t \in \mathcal{V}\}$;
14: Approximate the **KL** terms in ELBO for each layer with b samples;
15: Updating VFG model $\mathbb{G}$ with gradient ascending: $\theta_\mathbf{f}^{(s+1)} = \theta_\mathbf{f}^{(s)} + \nabla_{\theta_\mathbf{f}} \frac{1}{b} \sum_{i=1}^{b} \mathcal{L}(\mathbf{x}_b; \theta_\mathbf{f}^{(s)})$.
16: **end for**

In training algorithm 1, the backward variable state $\widehat{\mathbf{h}}^l$ in layer l is generated according to $p(\widehat{\mathbf{h}}^l | \widehat{\mathbf{h}}^{l+1})$, and at the root layer, node state $\widehat{\mathbf{h}}^\mathcal{R}$ is set equal to $\mathbf{h}^\mathcal{R}$ that is from the posterior $q(\mathbf{h}|\mathbf{x})$, not from the prior $p(\mathbf{h}^\mathcal{R})$. So we can see all the forward and backward latent variables are sampled from the posterior $q(\mathbf{h}|\mathbf{x})$.

From a practical perspective, layer-wise training strategy can improve the accuracy of a model especially when it is constructed of more than two layers. In such a case, the parameters of only one layer are updated with backpropagation of the gradient of the loss function while keeping the other layers fixed at each optimization step. By maximizing the ELBO (7) with the above algorithm, the node consistency rules in Section 3.2 are expected to be satisfied.

6 INFERENCE ON VFGS

With a VFG, we aim to infer node states given observed ones. The hidden state of a parent node j in $l = 1$ can be computed with the observed children as follows:

$$\mathbf{h}^{(j)} = \frac{1}{|ch(j) \cap O|} \sum_{i \in ch(j) \cap O} \mathbf{h}^{(i)}, \tag{12}$$

where O is the set of observed leaf nodes, see Figure 4-left for an illustration. Observe that for either a tree or a DAG, the state of any hidden node is updated via messages received from its children. After reaching the root node, we can update any nodes with backward message passing. Figure 4 illustrates this inference mechanism for trees in which the structure enables us to perform message passing among the nodes. We derive the following lemma establishing the relation between two leaf nodes.

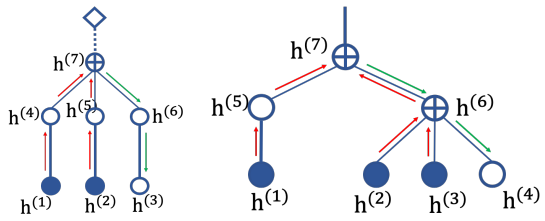

Figure 4: (Left) Inference on a VFG with single aggregation node. Node 7 aggregates information from node 1 and 2, and pass down the updated state to node 3 for prediction. (Right) Inference on a tree VFG. Observed node states are gathered at node 7 to predict the state of node 4. Red and green lines are forward and backward messages, respectively.

Lemma 6.1. *Let $\mathbb{G}$ be a tree VFG with L layers, and i and j are two leaf nodes with a as the closest common ancestor node. Given observed value at node i, the value of node j can be approximated by $\widehat{\mathbf{x}}^j = \mathbf{f}_{(a,j)}(\mathbf{f}_{(i,a)}(\mathbf{x}^{(i)}))$. Here $\mathbf{f}_{(i,a)}$ is the flow function path from node i to node a.*

Proof. According to the aggregation operation (8) discussed in Section 3.2, at an aggregation node a, the reconstruction state of a child node j is the mean reconstruction states of the parent nodes. The reconstruction of the child node j can be calculated with the average reconstruction state of its parent node. Apply it sequentially, we have $\widehat{\mathbf{x}}^{(j)} = \mathbf{f}_{(a,j)}(\widehat{\mathbf{h}}^a)$.

The forward state of node a can be computed by sequentially applying forward aggregating starting from its descendent i, i.e., $\mathbf{h}^{(a)} = \mathbf{f}_{(i,a)}(\mathbf{x}^{(i)})$. As there is no other observations, with forward and backward message passing to and from the root node, at node a, we have $\mathbf{h}^{(a)} = \widehat{\mathbf{h}}^{(a)}$. Therefore, we have $\widehat{\mathbf{x}}^{(j)} = \mathbf{f}_{(a,j)}(\mathbf{f}_{(i,a)}(\mathbf{x}^{(i)}))$.

□

Considering the flow-based model (3), we have the following identity for each node of the graph structure:

$$p(\mathbf{h}^{(i)}|\mathbf{h}^{pa(i)}) = p(\mathbf{h}^{pa(i)})\left|\det\left(\frac{\partial \mathbf{h}^{pa(i)}}{\partial \mathbf{h}^{(i)}}\right)\right|$$
$$= p(\mathbf{h}^{pa(i)})\left|\det(\mathbf{J}_{\mathbf{h}^{pa(i)}}(\mathbf{h}^{(i)}))\right|.$$

Lemma 6.1 provides an approach to conduct inference on a tree and impute missing values in the data. It is easy to extend the inference method to DAG VFGs.

7 NUMERICAL EXPERIMENTS

In this section, we provide several studies to validate the proposed VFG models. The first application we present is missing value imputation. We compare our method with different baseline models on several datasets. The second set of experiments is to evaluate VFG models on three different datasets, i.e., MNIST, Caltech101, and Omniglot, with ELBO and likelihoods as the score. The third application we present here is the task of learning posterior distribution of the latent variables corresponding to the hidden explanatory factors of variations in the data [1]. For that latter application, the model is trained and evaluated on the MNIST handwritten digits dataset.

In this paper, we would rather assume the VFG graph structures are given and fixed. In the following experiments, the VFG structures are given in the dataset or designed heuristically (as other neural networks) for the sake of numerical illustrations. Learning the structure of VFG is an interesting research problem and is left for future works. A simple approach for VFG structure learning is to regularize the graph with the DAG structure penalty [35, 38].

All the experiments are conducted on NVIDIA-TITAN X (Pascal) GPUs. In the experiments, we use the same coupling block [9] to construct different flow functions. The coupling block consists of three fully connected layers (of dimension 64) separated by two RELU layers along with the coupling trick. Each flow function has block number $\mathcal{B} > 3$.

7.1 Evaluation on Inference with Missing Entries Imputation

We now focus on the task of imputing missing entries in a graph structure. For all the following experiments, the models are trained on the training set and are used to infer the missing entries of samples in the testing set. We first study the proposed VFGs on two datasets without given graph structures, and we compare VFGs with several conventional methods that do not require the graph structures in the data. We then compare VFGs with graphical models that can perform inference on explicit graphs.

7.1.1 Synthetic dataset. In this set of experiments, we study different methods with synthetic datasets. The baselines for this set of experiments include mean value method (Means), iterative imputation (Iterative) [4], and multivariate imputation by chained equation (MICE) [33]. Mean Squared Error as the metric of reference in order to compare the different methods for the imputation task. We use the baseline implementations in [23] in the experiments.

We generate 10 synthetic datasets (using different seeds) of 1 300 data points, 1 000 for the training phase of the model, 300 for imputation testing. Each data sample has 8 dimensions with 2 latent variables. Let $z_1 \sim \mathcal{N}(0, 1.0^2)$ and $z_2 \sim \mathcal{N}(1.0, 2.0^2)$ be the latent variables. For a sample $\mathbf{x}$, we have $x_1 = x_2 = z_1, x_3 = x_4 = 2\sin(z_1), x_5 = x_6 = z_2$, and $x_7 = x_8 = z_2^2$. In the testing dataset, x_3, x_4, x_7, and x_8 are missing. We use a VFG model with a single average aggregation node that has four children, and each child connects the parent with a flow function consisting of 3 coupling layers [9]. Each child takes 2 variables as input data section, and the latent dimension of the VFG is 2. We compare, Figure 5, our VFG method with the baselines described above using boxplots on obtained MSE values for those 10 simulated datasets. We can see that the proposed VFG model performs much better than mean value, iterative, and MICE methods. Figure 5 shows that VFGs also demonstrates more performance robustness compared against other methods.

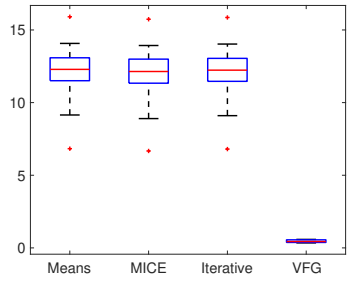

Figure 5: Synthetic datasets: MSE boxplots of VFG and baseline methods.

7.1.2 California Housing dataset. We further investigate the method on a real dataset. The California Housing dataset has 8 feature entries and 20 640 data samples. We use the first 20 000 samples for training and 100 of the rest for testing. We get 4 data sections, and each section contains 2 variables. In the testing set, the second section is assumed missing for illustration purposes, as the goal is to impute this missing section. In addition to the three baselines in introduced the main file, we also compared with KNN (k-nearest neighbor) method. Again, we use the implementations from [23] for the baselines in this set of experiments.

Table 1: California Housing dataset: Imputation Mean Squared Error (MSE) results.

Methods	*Imputation MSE*
Mean Value	1.993
MICE	1.951
Iterative Imputation	1.966
KNN (k=5)	1.969
VFG	**1.356**

The VFG structure is designed heuristically. We construct a tree structure VFG with 2 layers. The first layer has two aggregation nodes, and each of them has two children. The second layer consists of one aggregation node that has two children connecting with the first layer. Each flow function has $\mathcal{B} = 4$ coupling blocks. Table 1 shows that our model yields significantly better results than any

other method in terms of prediction error. It indicates that with the help of universal approximation power of neural networks, VFGs have superior inference capability.

7.1.3 Comparison with Graphical Models. In this set of experiments, we use a synthetic Gaussian graphical model dataset from the bnlearn package [29] to evaluate the proposed model. The data graph structure is given. The dataset consists of 7 variables and 5 000 samples. Sample values at each node are generated according to a structured causal model with a diagram given by Figure 6. Each node represents a variable generated with a function of its parent nodes. For instance, node V is generated with $V = \mathbf{f}(pa(V), N_V)$. Here $pa(V)$ is the set of V's parents, and N_V is a noise term for V. A node without any parent is determined only by the noise term. $\mathbf{f}()$ is V's generating function, and only linear functions are used in this dataset. All the noise terms are Normal distributions.

Table 2: Gaussian graphical model dataset: Imputation Mean Squared Error (MSE) and Variance results.

Methods	Bayesian Net	SPN	VFG
Imputation MSE	1.059	0.402	**0.104**
Imputation Variance	2.171	0.401	**0.012**

We take Bayesian network implementation [29] and sum-product network (SPN) package [22, 24] as experimental baselines. 4 500 samples are used for training, and the rest 500 samples are for testing. The structure of VFG is designed based on the directed graph given by Figure 6. In the imputation task, we take Node 'F' as the missing entry, and use the values of other node to impute the missing entry. Table 2 gives the imputation results from the three methods. We can see that VFG achieves the smallest prediction error. Besides the imputation MSE, Table 2 also gives the prediction error variance. Compared against Bayesian net and SPN, VFG achieves much smaller performance variance. It means VFGs are much more stable in this set of experiments.

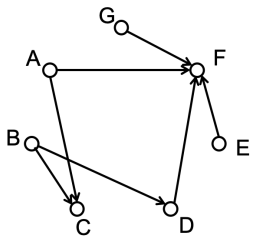

Figure 6: Graph structure for Gaussian graphical model dataset.

7.2 ELBO and Likelihood

We further qualitatively compare our VFG model with existing methods on data distribution learning and variational inference using three standard datasets. The baselines we compare in this experiment are VAE [16], Planer [26], IAF [15], and SNF [2]. The evaluation datasets and setup are following two standard flow-based variational models, Sylvester Normalizing Flows [2] and [26]. We use a tree VFG with structure as shown in Figure 7 for three datasets. We train the tree VFG with the following ELBO objective that incorporate a β coefficient for the **KL** terms. Empirically, a small β yields better ELBO and NLL values, and we set β around 0.1 in the experiments.

$$\text{ELBO} = \mathcal{L}(\mathbf{x};\theta) = \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{h}^{1:L})\right] - \beta \sum_{l=1}^{L} \mathbf{KL}^l.$$

Table 3 presents the negative evidence lower bound (-ELBO) and the estimated negative likelihood (NLL) for all methods on the three datasets, MNIST, Caltech101, and Omniglot. The baseline methods are VAE based methods enhanced with normalizing flows. They use 16 flows to improve the posterior estimation. SNF is orthogonal sylvester flow method with a bottleneck of M = 32. We set the VFG coupling block[9] number with $\mathcal{B} = 4$, and following [2] we run multiple times to get the mean and standard derivation as well. VFG can achieve superior EBLO as well as NLL values on all three datasets compared against the baselines as given in Table 3. The first reason why VFGs can achieve better variational inference and data distribution modeling results (ELBOs and NLLs) in Table 3 is due to VFGs' universal approximation power as given in Theorem 4.4. Secondly, the intrinsic approximate invertible property of VFGs ensures the decoder or generative model in a VFG to achieve smaller reconstruction errors for data samples and hence smaller NLL values.

7.3 Latent Representation Learning on MNIST

In this set of experiments, we evaluate VFGs on latent representation learning of the MNIST dataset [19]. We construct a tree VFG model depicted in Figure 7. In the first layer, there are 4 flow functions, and each of them takes 14×14 image blocks as the input. Thus a 28 × 28 input image is divided into four 14 × 14 blocks as the input of VFG model. We use $\mathcal{B} = 4$ for all the flows. The latent dimension for this model is $m = 196$. Following [30], the VFG model is trained with image labels to learn the latent representation of the input data. We set the parameters of $\mathbf{h}^L$'s prior distribution as a function of image label, i.e., $\lambda^L(u)$, where u denotes the image label. In practice, we use 10 trainable λ^Ls

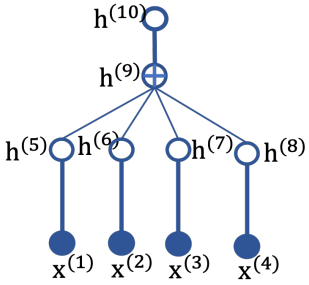

Figure 7: MIST Tree structure.

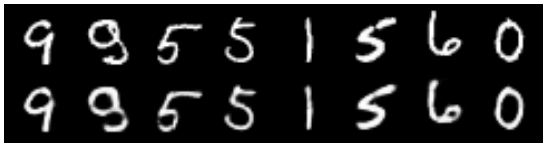

Figure 8: (Top) original MNIST digits. (Bottom) reconstructed images using VFG.

regarding the 10 digits. The images in the second row of Figure 8 are reconstructions of MNIST samples extracted from the testing set, displayed in the first row of the same Figure, using our proposed VFG model.

Figure 9-Left shows t-distributed stochastic neighbor embedding (t-SNE) [20] plot of 2, 000 testing images' latent variables learned

Table 3: Negative log-likelihood and free energy (negative evidence lower bound) for static MNIST, Caltech101, and Omniglot.

Model	MNIST		Caltech101		Omniglot	
	-ELBO	NLL	-ELBO	NLL	-ELBO	NLL
VAE [16]	86.55 ± 0.06	82.14 ± 0.07	110.80 ± 0.46	99.62 ± 0.74	104.28 ± 0.39	97.25 ± 0.23
Planer [26]	86.06 ± 0.31	81.91 ± 0.22	109.66 ± 0.42	98.53 ± 0.68	102.65 ± 0.42	96.04 ± 0.28
IAF [15]	84.20 ± 0.17	80.79 ± 0.12	111.58 ± 0.38	99.92 ± 0.30	102.41 ± 0.04	96.08 ± 0.16
SNF [2]	83.32 ± 0.06	80.22 ± 0.03	104.62 ± 0.29	93.82 ± 0.62	99.00 ± 0.04	93.77 ± 0.03
VFG	**80.80 ± 0.76**	**63.66 ± 0.14**	**67.26 ± 0.53**	**65.74 ± 0.84**	**80.16 ± 0.73**	**78.65 ± 0.66**

with our model, and 200 for each digit. Figure 9-Left illustrates that VFG can learn separated latent representations to distinguish different hand-written numbers. For comparison, we also present the results of a baseline model. The baseline model (coupling-based flow) is constructed using the same coupling block and similar number of parameters as VFGs but with 28×28 as the input and latent dimension. Figure 9-Right gives the baseline coupling-layer-based flow training and testing with the same procedures. The results show that coupling-based flow cannot give a clear division between some digits, e.g., 1 and 2, 7 and 9 due to the bias introduced by the high-dimensional redundant latent variables.

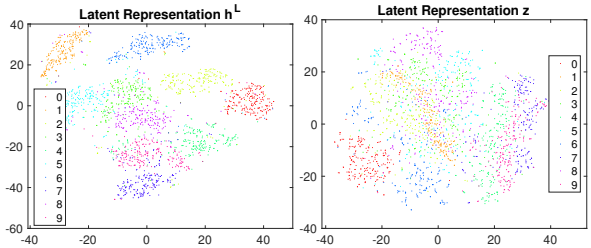

Figure 9: t-SNE plot of latent variables for VFG (Left) and coupling-layer based flow (Right) on MNIST.

To provide a description of the learned latent representation, we first obtain the root latent variables of a set of images through forward message passing. Each latent variable's values are changed increasingly within a range centered at the value of the latent variable obtained from last step. This perturbation is performed for each image in the set. Figure 10 shows the change of images by increasing one latent variable from a small value to a larger one. The figure presents some of the latent variables that have obvious effects on images, and most of the $m = 196$ variables do not impact the generation significantly. Latent variables $i = 6$ and $i = 60$ control the digit width. Variable $i = 19$ affects the brightness. $i = 92$, $i = 157$ and some of the variables not displayed here control the style of the generated digits.

8 DISCUSSION

One of the motivations for VFG is to develop a tractable model that can be used for distribution learning and posterior inference. As long as the node states in the aggregation nodes are consistent, we can always apply VFGs in order to infer missing values. We provide more discussion on the structures of VFGs in the sequel.

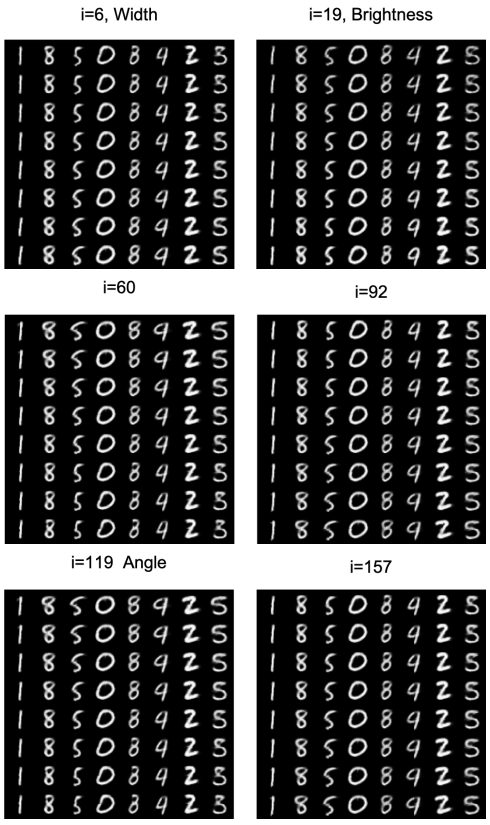

Figure 10: MNIST: Increasing each latent variable from a small value to a larger one.

8.1 Benefits of Encoder-decoder Parameter Sharing

There are several advantages for the encoder and decoder to share parameters. Firstly, it makes the network's structure simple. Secondly, the training and inference can be simplified with concise and simple graph structures. Thirdly, by leveraging invertible flow-based functions, VFGs obtain tighter ELBOs in comparison with VAE based models.The intrinsic invertibility introduced by flow functions ensures the decoder or generative model in a VFG achieves smaller reconstruction errors for data samples and hence smaller NLL values and tighter ELBO. Whereas without the intrinsic constraint of invertibility or any help or regularization from the encoder, VAE-based models have to learn an unassisted mapping function (decoder) to reconstruct all data samples with the latent

variables, and there are always some discrepancy errors in the reconstruction that lead to relatively larger NLL values and hence inferior ELBOs.

8.2 Structures of VFGs

In the experiments, the model structures have been chosen heuristically and for the sake of numerical illustrations. A tree VFG model can be taken as a dimension reduction model that is available for missing value imputation as well. Variants of those structures will lead to different numerical results and at this point, we can not claim any generalization regarding the impact of the VFG structure on the outputs. Meanwhile, learning the structure of VFG is an interesting research problem and is left for future works. VFG structures could be learned through the regularization of DAG structures [35, 38].

VFGs rely on minimizing the KL term to achieve *consistency* in aggregation nodes. As long as the aggregation nodes retain consistency, the model always has a tight ELBO and can be applied to tractable posterior inference. According to [32], coupling-based flows are endowed with the universal approximation power. Hence, we believe that the consistency of aggregation nodes on a VFG can be attained with a tight ELBO.

9 CONCLUSION

In this paper, we propose VFG, a variational flow graphical model that aims at bridging the gap between flow-based models and the paradigm of graphical models. Our VFG model learns data distribution and latent representation through message passing between nodes in the model structure. We leverage the power of invertible flow functions in any general graph structure to simplify the inference step of the latent nodes given some input observations. We illustrate the effectiveness of our variational model through a set of experiments on various datasets. Future work includes applying our VFG model to relational data structure learning and reasoning.

REFERENCES

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

[2] Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. 2018. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649* (2018).

[3] Christopher M Bishop, David Spiegelhalter, and John Winn. 2003. VIBES: A variational inference engine for Bayesian networks. In *NeurIPS*. 793–800.

[4] Samuel F Buck. 1960. A method of estimation of missing values in multivariate data suitable for use with an electronic computer. *Journal of the Royal Statistical Society: Series B (Methodological)* 22, 2 (1960), 302–306.

[5] YooJung Choi, Antonio Vergari, and Guy Van den Broeck. 2020. *Probabilistic circuits: A unifying framework for tractable probabilistic models.* Technical Report. Technical report.

[6] Adnan Darwiche. 2003. A differential approach to inference in Bayesian networks. *Journal of the ACM (JACM)* 50, 3 (2003), 280–305.

[7] Rina Dechter and Robert Mateescu. 2007. AND/OR search spaces for graphical models. *Artificial intelligence* 171, 2-3 (2007), 73–106.

[8] Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516* (2014).

[9] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using Real NVP. *ArXiv* abs/1605.08803 (2016).

[10] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS*.

[11] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *JMLR* 14, 1 (2013), 1303–1347.

[12] Manfred Jaeger, Jens D Nielsen, and Tomi Silander. 2006. Learning probabilistic decision graphs. *International Journal of Approximate Reasoning* 42, 1-2 (2006), 84–100.

[13] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. 1999. An introduction to variational methods for graphical models. *Machine learning* 37, 2 (1999), 183–233.

[14] Ilyes Khemakhem, Ricardo Monti, Robert Leech, and Aapo Hyvarinen. 2021. Causal autoregressive flows. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 3520–3528.

[15] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934* (2016).

[16] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[17] Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. 2014. Probabilistic sentential decision diagrams. In *Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

[18] F Kuo, I Sloan, Grzegorz Wasilkowski, and Henryk Woźniakowski. 2010. On decompositions of multivariate functions. *Mathematics of computation* 79, 270 (2010), 953–966.

[19] Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. (2010). http://yann.lecun.com/exdb/mnist/

[20] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR* 9, Nov (2008), 2579–2605.

[21] Radu Marinescu and Rina Dechter. 2005. AND/OR branch-and-bound for graphical models. In *IJCAI*. Citeseer, 224–229.

[22] Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Pranav Subramani, Nicola Di Mauro, Pascal Poupart, and Kristian Kersting. 2019. SPFlow: An Easy and Extensible Library for Deep Probabilistic Learning using Sum-Product Networks. arXiv:arXiv:1901.03704

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[24] Hoifung Poon and Pedro Domingos. 2011. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, 689–690.

[25] Tahrima Rahman, Prasanna Kothalkar, and Vibhav Gogate. 2014. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of Chow-Liu trees. In *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 630–645.

[26] Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770* (2015).

[27] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*. PMLR, 1278–1286.

[28] Raquel Sánchez-Cauce, Iago París, and Francisco Javier Díez. 2021. Sum-product networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[29] Marco Scutari. 2009. Learning Bayesian networks with the bnlearn R package. *arXiv preprint arXiv:0908.3817* (2009).

[30] Peter Sorrenson, Carsten Rother, and Ullrich Köthe. 2020. Disentanglement by Nonlinear ICA with General Incompressible-flow Networks (GIN). In *ICLR*.

[31] Takeshi Teshima, Isao Ishikawa, Koichi Tojo, Kenta Oono, Masahiro Ikeda, and Masashi Sugiyama. 2020. Coupling-based Invertible Neural Networks Are Universal Diffeomorphism Approximators. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 3362–3373.

[32] Takeshi Teshima, Isao Ishikawa, Koichi Tojo, Kenta Oono, Masahiro Ikeda, and Masashi Sugiyama. 2020. Coupling-based invertible neural networks are universal diffeomorphism approximators. *arXiv preprint arXiv:2006.11469* (2020).

[33] Stef Van Buuren and Karin Groothuis-Oudshoorn. 2011. mice: Multivariate imputation by chained equations in R. *Journal of statistical software* 45 (2011), 1–67.

[34] Martin J Wainwright and Michael Irwin Jordan. 2008. *Graphical models, exponential families, and variational inference.* Now Publishers Inc.

[35] Antoine Wehenkel and Gilles Louppe. 2021. Graphical normalizing flows. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 37–45.

[36] John Winn and Christopher M Bishop. 2005. Variational message passing. *JMLR* 6, Apr (2005), 661–694.

[37] Eric P Xing, Michael I Jordan, and Stuart Russell. 2012. A generalized mean field algorithm for variational inference in exponential families. *arXiv:1212.2512* (2012).

[38] Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. 2018. DAGs with NO TEARS: Continuous Optimization for Structure Learning. In *NeurIPS*. 9492–9503.

A ELBO OF TREE VFGS

Let each data sample has k sections, i.e., $\mathbf{x} = [\mathbf{x}^{(1)}, ..., \mathbf{x}^{(k)}]$. VFGs are graphical models that can integrate different sections or components of the dataset. We assume that for each pair of connected nodes, the edge is an invertible flow function. The vector of parameters for all the edges is denoted by θ. The forward message passing starts from $\mathbf{x}$ and ends at $\mathbf{h}^L$, and backward message passing in the reverse direction. We start with the hierarchical generative tree network structure illustrated by an example in Figure 11-Left. Then the marginal likelihood term of the data reads

$$p(\mathbf{x}|\theta) = \sum_{\mathbf{h}^1,...,\mathbf{h}^L} p(\mathbf{h}^L|\theta)p(\mathbf{h}^{L-1}|\mathbf{h}^L,\theta)\cdots p(\mathbf{x}|\mathbf{h}^1,\theta)\,.$$

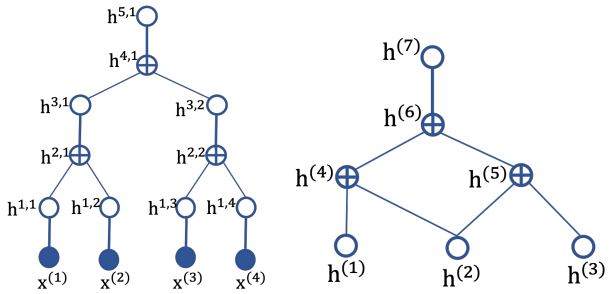

Figure 11: (Left) A tree VFG with $L = 5$ and three aggregation nodes. (Right) A DAG with inverse topology order { {1,2,3}, {4,5}, {6}, {7} }, and they correspond to layers 0 to 3.

The hierarchical generative model is given by factorization

$$p(\mathbf{h}) = p(\mathbf{h}^L)\Pi_{l=1}^{L-1} p(\mathbf{h}^l|\mathbf{h}^{l+1}). \tag{13}$$

The probability density function $p(\mathbf{h}^{l-1}|\mathbf{h}^l)$ in the generative model is modeled with one or multiple invertible normalizing flow functions. The hierarchical posterior (recognition network) is factorized as

$$q_\theta(\mathbf{h}|\mathbf{x}) = q(\mathbf{h}^1|\mathbf{x})q(\mathbf{h}^2|\mathbf{h}^1)\cdots q(\mathbf{h}^L|\mathbf{h}^{L-1}). \tag{14}$$

Draw samples from the generative model (13) involves sequential conditional sampling from the top of the tree to the bottom, and computation of the recognition model (14) takes the reverse direction. Notice that

$$q(\mathbf{h}|\mathbf{x}) = q(\mathbf{h}^1|\mathbf{x})q(\mathbf{h}^{2:L}|\mathbf{h}^1)\,.$$

With the hierarchical structure of a tree, we further have

$$q(\mathbf{h}^{l:L}|\mathbf{h}^{l-1}) = q(\mathbf{h}^l|\mathbf{h}^{l-1})q(\mathbf{h}^{l+1:L}|\mathbf{h}^l\mathbf{h}^{l-1}) = q(\mathbf{h}^l|\mathbf{h}^{l-1})q(\mathbf{h}^{l+1:L}|\mathbf{h}^l) \tag{15}$$

$$p(\mathbf{h}^{l:L}) = p(\mathbf{h}^l|\mathbf{h}^{l+1:L})p(\mathbf{h}^{l+1:L}) = p(\mathbf{h}^l|\mathbf{h}^{l+1})p(\mathbf{h}^{l+1:L}) \tag{16}$$

By leveraging the conditional independence in the chain structures of both recognition and generative models, the derivation of trees' ELBO becomes easier.

$$\begin{aligned}\log p(\mathbf{x}) &= \log \int p(\mathbf{x}|\mathbf{h})p(\mathbf{h})d\mathbf{h}\\ &= \log \int \frac{q(\mathbf{h}|\mathbf{x})}{q(\mathbf{h}|\mathbf{x})}p(\mathbf{x}|\mathbf{h})p(\mathbf{h})d\mathbf{h}\\ &\geqslant \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{h}) - \log q(\mathbf{h}|\mathbf{x}) + \log p(\mathbf{h})\right]\\ &= \mathcal{L}(x;\theta).\end{aligned}$$

The last step is due to the Jensen inequality. With $\mathbf{h} = \mathbf{h}^{1:L}$,

$$\begin{aligned}&\log p(\mathbf{x}) \geqslant \mathcal{L}(x;\theta)\\ =&\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{h}^{1:L}) - \log q(\mathbf{h}^{1:L}|\mathbf{x}) + \log p(\mathbf{h}^{1:L})\right]\\ =&\underbrace{\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{h}^{1:L})\right]}_{\text{(a) Reconstruction of the data}}\\ &\underbrace{-\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log q(\mathbf{h}^{1:L}|\mathbf{x}) - \log p(\mathbf{h}^{1:L})\right]}_{\mathbf{KL}^{1:L}}\end{aligned} \tag{17}$$

With conditional independence in the hierarchical structure, we have

$$q(\mathbf{h}^{1:L}|\mathbf{x}) = q(\mathbf{h}^{2:L}|\mathbf{h}^1\mathbf{x})q(\mathbf{h}^1|\mathbf{x}) = q(\mathbf{h}^{2:L}|\mathbf{h}^1)q(\mathbf{h}^1|\mathbf{x}).$$

The second term of (17) can be further expanded as

$$\begin{aligned}\mathbf{KL}^{1:L} =&\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log q(\mathbf{h}^1|\mathbf{x}) + \log q(\mathbf{h}^{2:L}|\mathbf{h}^1)\right.\\ &\left.- \log p(\mathbf{h}^1|\mathbf{h}^{2:L}) - \log p(\mathbf{h}^{2:L})\right].\end{aligned} \tag{18}$$

Similarly, with conditional independence of the hierarchical latent variables, $p(\mathbf{h}^1|\mathbf{h}^{2:L}) = p(\mathbf{h}^1|\mathbf{h}^2)$. Thus

$$\begin{aligned}\mathbf{KL}^{1:L} =&\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log q(\mathbf{h}^1|\mathbf{x}) - \log p(\mathbf{h}^1|\mathbf{h}^2)\right.\\ &\left.+ \log q(\mathbf{h}^{2:L}|\mathbf{h}^1) - \log p(\mathbf{h}^{2:L})\right]\\ =&\underbrace{\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log q(\mathbf{h}^1|\mathbf{x}) - \log p(\mathbf{h}^1|\mathbf{h}^2)\right]}_{\mathbf{KL}^1}\\ &+\underbrace{\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log q(\mathbf{h}^{2:L}|\mathbf{h}^1) - \log p(\mathbf{h}^{2:L})\right]}_{\mathbf{KL}^{2:L}}.\end{aligned}$$

We can further expand the $\mathbf{KL}^{2:L}$ term following similar conditional independent rules regarding the tree structure. At level l, we get

$$\mathbf{KL}^{l:L} = \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log q(\mathbf{h}^{l:L}|\mathbf{h}^{l-1}) - \log p(\mathbf{h}^{l:L})\right].$$

With (15) and (16), it is easy to show that

$$\begin{aligned}\mathbf{KL}^{l:L} =&\underbrace{\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log q(\mathbf{h}^l|\mathbf{h}^{l-1}) - \log p(\mathbf{h}^l|\mathbf{h}^{l+1})\right]}_{\mathbf{KL}^l}\\ &+\underbrace{\mathbb{E}_{q(\mathbf{h}^{l:L}|\mathbf{x})}\left[\log q(\mathbf{h}^{l+1:L}|\mathbf{h}^l) - \log p(\mathbf{h}^{l+1:L})\right]}_{\mathbf{KL}^{l+1:L}}.\end{aligned} \tag{19}$$

The ELBO (17) can be written as

$$\mathcal{L}(\mathbf{x};\theta) = \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{h}^{1:L})\right] - \sum_{l=1}^{L-1}\mathbf{KL}^l - \mathbf{KL}^L. \tag{20}$$

When $1 \leqslant l \leqslant L-1$

$$\mathbf{KL}^l = \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log q(\mathbf{h}^l|\mathbf{h}^{l-1}) - \log p(\mathbf{h}^l|\mathbf{h}^{l+1})\right]. \tag{21}$$

According to conditional independence, the expectation regarding variational distribution layer l just depends on layer $l-1$. We can simplify the expectation each term of (20) with the default

assumption that all latent variables are generated regarding data sample $\mathbf{x}$. Therefore the ELBO (20) can be simplified as

$$\mathcal{L}(\mathbf{x};\theta) = \mathbb{E}_{q(\mathbf{h}^1|\mathbf{x})}\left[\log p(\mathbf{x}|\widehat{\mathbf{h}}^1)\right] - \sum_{l=1}^{L} \mathbf{KL}^l. \tag{22}$$

The $\mathbf{KL}$ term (21) becomes

$$\mathbf{KL}^l = \mathbb{E}_{q(\mathbf{h}^l|\mathbf{h}^{l-1})}\left[\log q(\mathbf{h}^l|\mathbf{h}^{l-1}) - \log p(\mathbf{h}^l|\widehat{\mathbf{h}}^{l+1})\right].$$

When $l = L$,

$$\mathbf{KL}^L = \mathbb{E}_{q(\mathbf{h}^L|\mathbf{h}^{L-1})}\left[\log q(\mathbf{h}^L|\mathbf{h}^{L-1}) - \log p(\mathbf{h}^L)\right].$$

B ELBO OF DAG VFGS

Note that if we reverse the edge directions in a DAG, the resulting graph is still a DAG graph. The nodes can be listed in a topological order regarding the DAG structure as shown in Figure 11-Right.

By taking the topology order as the layers in tree structures, we can derive the ELBO for DAG structures. Assume the DAG structure has L layers, and the root nodes are in layer L. We denote by $\mathbf{h}$ the vector of latent variables, then following (17) we develop the ELBO as

$$\log p(\mathbf{x}) \geqslant \mathcal{L}(x;\theta) = \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{h})}{q(\mathbf{h}|\mathbf{x})}\right] \tag{23}$$

$$= \underbrace{\mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{h})\right]}_{\text{Reconstruction of the data}} \underbrace{- \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\left[\log q(\mathbf{h}|\mathbf{x}) - \log p(\mathbf{h})\right]}_{\mathbf{KL}}.$$

Similarly the KL term can be expanded as in the tree structures. For nodes in layer l

$$\mathbf{KL}^{l:L} = \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log q(\mathbf{h}^{l:L}|\mathbf{h}^{1:l-1}) - \log p(\mathbf{h}^{l:L})\right].$$

Note that $ch(l)$ may include nodes from layers lower than $l-1$, and $pa(l)$ may include nodes from layers higher than l. Some nodes in l may not have parent. Based on conditional independence with the topology order of a DAG, we have

$$q(\mathbf{h}^{l:L}|\mathbf{h}^{1:l-1}) = q(\mathbf{h}^l|\mathbf{h}^{1:l-1})q(\mathbf{h}^{l+1:L}|\mathbf{h}^l) \tag{24}$$

$$= q(\mathbf{h}^l|\mathbf{h}^{1:l-1})q(\mathbf{h}^{l+1:L}|\mathbf{h}^{1:l})p(\mathbf{h}^{l:L}) = p(\mathbf{h}^l|\mathbf{h}^{l+1:L})p(\mathbf{h}^{l+1:L}) \tag{25}$$

Following (19) and with (24-25), we have

$$\mathbf{KL}^{l:L} = \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log q(\mathbf{h}^l|\mathbf{h}^{1:l-1}) - \log p(\mathbf{h}^l|\mathbf{h}^{l+1:L})\right] + \underbrace{\mathbb{E}_{q(\mathbf{h}^{l:L}|\mathbf{x})}\left[\log q(\mathbf{h}^{l+1:L}|\mathbf{h}^{1:l}) - \log p(\mathbf{h}^{l+1:L})\right]}_{\mathbf{KL}^{l+1:L}}.$$

Furthermore,

$$q(\mathbf{h}^l|\mathbf{h}^{1:l-1}) = q(\mathbf{h}^l|\mathbf{h}^{ch(l)}), \qquad p(\mathbf{h}^l|\mathbf{h}^{l+1:L}) = p(\mathbf{h}^l|\mathbf{h}^{pa(l)}).$$

Hence,

$$\mathbf{KL}^{l:L} = \underbrace{\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log q(\mathbf{h}^l|\mathbf{h}^{ch(l)}) - \log p(\mathbf{h}^l|\mathbf{h}^{pa(l)})\right]}_{\mathbf{KL}^l} + \mathbf{KL}^{l+1:L} \tag{26}$$

For nodes in layer l,

$$\mathbf{KL}^l = \sum_{i\in l} \underbrace{\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log q(\mathbf{h}^{(i)}|\mathbf{h}^{ch(i)}) - \log p(\mathbf{h}^{(i)}|\mathbf{h}^{pa(i)})\right]}_{\mathbf{KL}^{(i)}}.$$

Recurrently applying (26) to (23) yields

$$\mathcal{L}(\mathbf{x};\theta) = \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{h})\right] - \sum_{i\in\mathcal{V}\setminus\mathcal{R}_{\mathbb{G}}} \mathbf{KL}^{(i)} - \sum_{i\in\mathcal{R}_{\mathbb{G}}} \mathbf{KL}\big(q(\mathbf{h}^{(i)}|\mathbf{h}^{ch(i)})||p(\mathbf{h}^{(i)})\big).$$

For node i,

$$\mathbf{KL}^{(i)} = \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\left[\log q(\mathbf{h}^{(i)}|\mathbf{h}^{ch(i)}) - \log p(\mathbf{h}^{(i)}|\mathbf{h}^{pa(i)})\right].$$

C ADDITIONAL DETAILS ON ALGORITHM

The inference ability of VFG can be reinforced by masking out some sections of the training samples. The training objective can be changed to force the model to impute the value of the masked sections. For example in a tree model, the alternative objective function reads

$$\mathcal{L}(\mathbf{x}, O_{\mathbf{x}};\theta) = \sum_{t:1\leqslant t\leqslant k, t\notin O} \mathbb{E}_{q(\mathbf{h},\widehat{\mathbf{h}}|\mathbf{x}^{O_{\mathbf{x}}})}\left[\log p(\mathbf{x}^{(t)}|\widehat{\mathbf{h}}^1)\right] - \sum_{l=1}^{L-1} \mathbb{E}_{q(\mathbf{h},\widehat{\mathbf{h}}|\mathbf{x})}\left[\log q(\mathbf{h}^l|\mathbf{h}^{l-1}) - \log p(\mathbf{h}^l|\widehat{\mathbf{h}}^{l+1})\right] - \mathbf{KL}\big(q(\mathbf{h}^L|\mathbf{h}^{L-1})|p(\mathbf{h}^L)\big). \tag{27}$$

where $O_{\mathbf{x}}$ is the index set of leaf nodes with observation, and $\mathbf{x}^{O_{\mathbf{x}}}$ is the union of observed data sections. The random-masking training procedure for objective(27) is described in Algorithm 2. In practice, we use Algorithm 2 along with Algorithm 1 to enhance the training of a VFG model. However, we only occasionally update the model parameter θ with the gradient of (27) to ensure the distribution learning running well.

Algorithm 2 Inference model parameters with random masking

Input: Data distribution $\mathcal{D}$, $\mathbb{G} = \{\mathcal{V}, \mathbf{f}\}$
for $s = 0, 1, \ldots$ **do**
  Sample minibatch b samples $\{\mathbf{x}_1, ..., \mathbf{x}_b\}$ from $\mathcal{D}$;
  Optimize (5) with Line 4 to Line 15 in Algorithm 1;
  Sample a subset of the k data sections as data observation set $O_{\mathbf{x}}$; $O \leftarrow O_{\mathbf{x}}$;
  for $i \in \mathcal{V}$ **do**
    // forward message passing
    $\mathbf{h}^{(i)} = \frac{1}{|ch(i)\cap O|}\sum_{j\in ch(i)\cap O} \mathbf{f}_{(j,i)}(\mathbf{h}^{(j)})$;
    $O \leftarrow O \cup \{i\}$ if $ch(i) \cap O \neq \emptyset$;
  end for
  $\widehat{\mathbf{h}}^{(i)} = \mathbf{h}^{(i)}$ if $i \in \mathcal{R}_{\mathbb{G}}$ or $i \in$ layer L;
  for $i \in \mathcal{V}$ **do**
    // backward message passing
    $\widehat{\mathbf{h}}^{(i)} = \frac{1}{|pa(i)|}\sum_{j\in pa(i)} \mathbf{f}^{-1}_{(i,j)}(\widehat{\mathbf{h}}^{(j)})$;
  end for
  $\mathbf{h} = \{\mathbf{h}^{(t)} | t \in \mathcal{V}\cap O\}$, $\widehat{\mathbf{h}} = \{\widehat{\mathbf{h}}^{(t)} | t \in \mathcal{V}\}$;
  Approximate the $\mathbf{KL}$ terms in ELBO for each layer with b samples;
  Updating VFG with gradient of (27): $\theta_{\mathbf{f}}^{(s+1)} = \theta_{\mathbf{f}}^{(s)} + \nabla_{\theta_{\mathbf{f}}} \frac{1}{b}\sum_{i=1}^{b} \mathcal{L}(\mathbf{x}_b, O_{\mathbf{x}}; \theta_{\mathbf{f}}^{(s)})$,
end for