Variational Flow Graphical Model

Shaogang Ren
Belhal Karimi
{renshaogang, belhal.karimi}@gmail.com
Cognitive Computing Lab, Baidu Research
10900 NE 8th St. Bellevue, WA 98004, USA

Dingcheng Li
Ping Li
{dingchengli, pingli98}@gmail.com
Cognitive Computing Lab, Baidu Research
10900 NE 8th St. Bellevue, WA 98004, USA

ABSTRACT
This paper introduces a novel approach embedding flow-based models in 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, our model provides a new approach for distribution modeling and numerical inference on datasets. Multiple experiments on synthetic and real-world datasets show the benefits of our proposed method and potentially broad applications.

CCS CONCEPTS
• Computing methodologies → Learning latent representations; Neural networks; Bayesian network models; Probabilistic reasoning.

KEYWORDS
variational inference, flow-based model, generative model

ACM Reference Format:
Shaogang Ren, Belhal Karimi, Dingcheng Li, and Ping Li. 2022. Variational Flow Graphical Model. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’22), August 14–18, 2022, Washington, DC, USA. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539450

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 [5, 25, 45]. 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), stemmed from statistical physics, is computationally efficient and is applied to tackle large-scale inference problems [1, 7, 14, 15, 18, 20]. In variational inference, mean-field approximation [1, 18, 49] and variational message passing [6, 47] 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). They include probabilistic decision graphs [19], arithmetic circuits [10], and-or search spaces [29], multi-valued decision diagrams [11], sum-product nets [38], probabilistic sentential decision diagrams [24], and probabilistic circuits (PCs) [9]. 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. The recent GFlowNets [4] also target tractable probabilistic inference on different structures.

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 underlying distributions. Variational Auto-Encoders (VAEs) [23, 37] and Generative Adversarial Networks (GAN) [2, 16, 34, 50, 54] are widely applied to different categories of datasets. Flow-based models [12, 13, 35, 36, 43] leverage invertible neural networks and can be used to estimate the density values of data samples. Energy-based models (EBMs) [17, 27, 31, 48, 51, 53, 55] define an unnormalized probability density function of data, which is the exponential of the negative energy function. Unlike TPMs, it is often rather 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 are applicable for inference on graphical structures. Sum-product networks [38] and probabilistic circuits [9] are falling into this type of models as well. Sum-product networks and probabilistic circuits leverage mixture models and probabilistic factorization in graphical structures for tractable inference. Whereas, VFGs rely on the consistency of aggregation nodes in graphical structures to achieve tractable inference.

Summary of our contributions. Dealing with high-dimensional data using graph structures exacerbates the systemic inability for effective distribution modeling and efficient inference. To overcome the limitations, VFG is proposed 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 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 entity 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 training algorithms for VFGs. Section 6 discusses how to perform inference with a VFG. Section 7 showcases the advantages of VFG on various tasks. Section 8 and Section 9 provide a discussion and conclusion of the paper.

2 PRELIMINARIES

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

Notation: We use \([L]\) to denote the set \([1, \cdots, L]\), for all \(L > 1\). \(\text{KL}(p||q) := \int_{\mathcal{Z}} p(z) \log(p(z)/q(z)) \, dz\) 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 \(m > 0\).

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

\[
\log p(x) \geq \mathcal{L}(x; \theta, \phi) = \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)] - \text{KL}(q_\phi(z|x)||p(z)).
\]

In Variational Auto-Encoders (VAEs, [23, 37]), the calculation of the reconstruction term requires sampling from the posterior distribution along with using the reparameterization trick, i.e.,

\[
\mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)] \approx \frac{1}{U} \sum_{u=1}^{U} \log p_\theta(x|z_u).
\]  

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

Flow-based Models: Flow-based models [12, 13, 36, 43] correspond to a probability distribution transformation using a sequence of invertible and differentiable mappings, noted \(f: \mathcal{Z} \rightarrow \mathcal{X}\). By defining the invertible mappings \(\{f_i\}_{i=1}^{L}\) and by the chain rule and inverse function theorem, the variable \(x = f(z)\) has a tractable probability density function (pdf) given as:

\[
\log p_\theta(x) = \log p(z) + \sum_{i=1}^{L} \log \left| \det \left( \frac{\partial f_i}{\partial h_i^{-1}} \right) \right|, \tag{2}
\]

where we have \(h^0 = x\) and \(h^L = z\) for conciseness. The scalar value \(\log |\det(\partial h_i/\partial h_i^{-1})|\) is the logarithm of the absolute value of the determinant of the Jacobian matrix \(\partial h_i/\partial h_i^{-1}\), also called the log-determinant. Eq. (2) 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. [36] 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 \(\mathcal{N}(z_0|\mu(x), \sigma(x))\). With \(K\) flows, latent variable \(z_0\) is transformed to \(z_k\). The reformulated ELBO is given by

\[
\mathcal{L}(x; \theta, \phi) = \mathbb{E}_{q_\phi} \left[ \log p_\theta(x, z) - \log q_\phi(z|x) \right] = \\
\mathbb{E}_{q_\phi} \left[ \log p_\theta(x, z) - \log q_\phi(z_0|x) \right] + \mathbb{E}_{q_\phi} \left[ \sum_{k=1}^{K} \log \left| \det \left( \frac{\partial f_k(z_k; \psi_k)}{\partial z_k} \right) \right| \right].
\]

Here \(f_k\) is the \(k\)-th flow with parameter \(\psi_k\), i.e., \(z_{k} = f_k \circ \cdots \circ f_2 \circ f_1(z_0)\). The flows are functions of data sample \(x\), 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 [46] learns a DAG structure from the input data under sparsity penalty and maximum likelihood estimation. The bivariate causal discovery method [21] relies on autoregressive structure of flow-based models and the asymmetry of log-likelihood ratio for cause-effect pairs. In this paper, VFGs generalize flow-based models [12, 13, 36, 43] to graphical variable inference.

3 VARIATIONAL FLOW GRAPHICAL MODEL

Assume \(K\) sections in the data samples, i.e. \(x = [x^{(1)}, \cdots, x^{(K)}]\), 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 \(\mathcal{G} = (\mathcal{V}, f)\) consists of a node set \(\mathcal{V}\) and an edge set \(\mathcal{f}\). 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

![Figure 1: (Left) Node \(h^{i,1}\) connects its children with invertible functions. Messages from the children are aggregated at the parent node, \(h^{i+1,1}\). (Right) Illustration of latent structure from layer \(l - 1\) to \(l + 1\). Thin lines are identity functions, and thick lines are flow functions. ⊕ is an aggregation node and circles for non-aggregation nodes.](image-url)
sections of this paper, identity function is considered as a special case of flow functions.

### 3.1 Evidence Lower Bound of VFGs

We apply variational inference to learn model parameters $\theta$ from data samples. Different from VAEs [23, 37], 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.

![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.](image)

When $l = L$, $KLL^L = \mathbb{E}_{q(h^{L}|x)}[\log q(h^{L}|h^{L-1}) - \log p(h^{L})]$. It is easy to extend the computation of the ELBO (4) 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:

$$L(x; \theta) = \mathbb{E}_{q(h|x)}[\log p(x|h)] - \sum_{i \in V \setminus R_G} KL(i) - \sum_{i \in R_G} KL(q(h^{i}||h^{ch(i)})).$$

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

Maximizing the ELBO (4) or (5) equals to optimizing the parameters of the flows, $\theta$. Similar to VAEs, 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 $h^i$ to represent node $i$’s state in the forward message, and $h^i$ for node $i$’s state in the backward message. For all nodes, both $h^i$ and $h^i$ are sampled from the posterior. At the root nodes, we have $h^R = h^R$.

### 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 $\Omega$. Let $f_{i,j}(\cdot)$ be the direct edge (function) from node $i$ to node $j$, and $f_{i,(j)}$ or $f_{j,(i)}$ defined as its inverse function. Then, the aggregation operation at node $i$ reads

$$h^{(i)} = \frac{1}{|ch(i)|} \sum_{j \in ch(i)} f_{i,(j)}(h^{(j)}), \quad \hat{h}^{(i)} = \frac{1}{|pa(i)|} \sum_{j \in pa(i)} f_{j,(i)}(\hat{h}^{(j)}).$$

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

With the identity functions between parents and their 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., $\hat{h}^{(i)} = \frac{1}{|ch(i)|} \sum_{j \in ch(i)} h^{(j)}$; (b) a child node’s forward state equals the mean of its parents’ backward states, i.e., $h^{(i)} = \frac{1}{|pa(i)|} \sum_{j \in pa(i)} \hat{h}^{(j)}$. Both rules empower VFGs with implicit invertibility.

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 (6), we have $h^{(i)} = (h^{(d)} + h^{(e)})/2$ and $\hat{h}^{(i)} = (\hat{h}^{(u)} + \hat{h}^{(v)})/2$. Here aggregation consistency means, for $i$’s children, their forward state should be...
consistent with i’s backward state, i.e.,
\[ h^{(d)} = h^{(e)} = \hat{h}^{(i)}. \] (7)
For i’s parents, their backward state should be consistent with i’s forward state, i.e.,
\[ \hat{h}^{(u)} = \hat{h}^{(o)} = h^{(i)}. \] (8)

We utilize the KL term in ELBO (5) to ensure (7) and (8) can be satisfied during parameter updating. The KL term regarding node i is
\[ \text{KL}^{(i)} = \mathbb{E}_{q(h|x)} \left[ \log q(h^{(i)}|h^{ch(i)}) - \log p(h^{(i)}|\hat{h}^{pa(i)}) \right] \]
\[ \simeq \log q(h^{(i)}|h^{ch(i)}) - \log p(h^{(i)}|\hat{h}^{pa(i)}). \] (9)

As the term \( \log q(h^{(i)}|h^{ch(i)}) \) involves node states that are deterministic according to (6), it is omitted in the computation of (9). With Laplace as the latent state distribution, here
\[ \log p(h^{(i)}|\hat{h}^{pa(i)}) = \frac{1}{2} \left( \log p(h^{(i)}|\hat{h}^{(u)}) + \log p(h^{(i)}|\hat{h}^{(o)}) \right) \]
\[ = \frac{1}{2} \left( -||h^{(i)} - \hat{h}^{(u)}||_1 - ||h^{(i)} - \hat{h}^{(o)}||_1 + 2m \cdot \log 2 \right). \]

Hence minimizing \( \text{KL}^{(i)} \) is equal to minimizing \( ||h^{(i)} - \hat{h}^{(u)}||_1 + ||h^{(i)} - \hat{h}^{(o)}||_1 \) which achieves the consistent objective in (8). Similarly, KL-term i’s children intend to realize consistency given in (7). We use node d as an example. The KL term regarding node d is
\[ \text{KL}^{(d)} = \mathbb{E}_{q(h|x)} \left[ \log q(h^{(d)}|h^{ch(d)}) - \log p(h^{(d)}|\hat{h}^{pa(d)}) \right] \]
\[ \simeq \log q(h^{(d)}|h^{ch(d)}) - \log p(h^{(d)}|\hat{h}^{pa(d)}). \]

The first term \( \log q(h^{(d)}|h^{ch(d)}) \) is omitted in the calculation of \( \text{KL}^{(d)} \) due to the deterministic relation with (6). Knowing that
\[ \log p(h^{(d)}|\hat{h}^{pa(d)}) = \log p(h^{(d)}|\hat{h}^{(i)}) = -||h^{(d)} - \hat{h}^{(i)}||_1 - m \cdot \log 2 \]
we notice that minimizing \( \text{KL}^{(d)} \) boils down to minimizing \( ||h^{(d)} - \hat{h}^{(i)}||_1 \), that targets (7). 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 (5) requires node states \( h^i \) and \( \hat{h}^i \) (\( i \in V \)) from the posterior. They correspond to the encoding and decoding procedures in VAE model as shown in Eq. (1). At the root node, we have \( \hat{h} = h \). The reconstruction terms in ELBO (5) can be computed with the backward message in the generative model \( p(x|h) \), i.e.,
\[ \mathbb{E}_{q(h|x)} \left[ \log p(x|h, \hat{h}) \right] \]
\[ \simeq 1 \sum_{u=1}^{U} \log p(x|h^{u}_a) = 1 \sum_{u=1}^{U} \log p(x|h^{pa(x)}_a). \]

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

4. UNIVERSAL APPROXIMATION PROPERTY
A universal approximation power of coupling-layer based flows has been highlighted in [41]. Following the analysis for flows [41], we prove that coupling-layer based VFGs have universal approximation as well. We first give several additional definitions regarding universal approximation. For a measurable mapping \( f : \mathbb{R}^m \rightarrow \mathbb{R}^n \) and a subset \( K \subset \mathbb{R}^m \), we define the following,
\[ ||f||_{p,K} = \left( \int_K ||f(x)||^p dx \right)^{1/p}. \]
Here \( ||.|| \) is the Euclidean norm of \( \mathbb{R}^n \) and \( ||f||_{sup,K} := \sup_{x \in K} ||f(x)|| \).

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

Definition 4.2. (Immersion and submanifold) \( g : \mathbb{R} \rightarrow \mathbb{R} \) is said to be an immersion if \( \text{rank}(g) = m = \dim(\mathbb{R}) \) everywhere. If \( g \) is injective (one-to-one) immersion, then \( g \) establish an one-to-one correspondence of \( \mathbb{R} \) and the subset \( \mathbb{R} = g(\mathbb{R}) \) of \( \mathbb{R} \). If we use this correspondence to endow \( \mathbb{R} \) with a topology and \( C^\infty \) structure, then \( \mathbb{R} = g(\mathbb{R}) \) is a submanifold (or immersed submanifold) and \( g : \mathbb{R} \rightarrow \mathbb{R} \) 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 \( g : U_g \rightarrow \mathbb{R} \), where \( U_g \subset \mathbb{R}^m \) is an open set \( \mathcal{C}^r \)-diffeomorphic to \( \mathbb{R} \), which may depend on \( g \) and \( \mathbb{R} \) is a submanifold of \( \mathbb{R}^n \).

We use \( m \) to represent the root node 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 \( \mathcal{C}^\infty_c(\mathbb{R}^{m-1}) \) as the set of all compactly-supported \( C^\infty \) mappings from \( \mathbb{R}^{m-1} \) to \( \mathbb{R} \). For a function set \( T \), we define \( T-\text{ACF} \) as the set of affine coupling flows [41] that are assembled with functions in \( T \), and we use \( \text{VFG}_{T-\text{ACF}} \) to represent the set of VFGs constructed using flows in \( T-\text{ACF} \).

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

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 \( f \) (f also represents the function/edge set) and the root as an aggregation node. The children divide the \( n \) input entries into \( \tau \equiv n/m \) even sections, and each section connects the aggregation node with a flow function.

Given an injective immersion \( g : \mathbb{R} \rightarrow \mathbb{R} \), function \( g \) can be represented with the concatenation of a set of functions, i.e., \( g = [g_1, g_2, \ldots, g_r] \), each invertible \( g_i \) has dimension \( m \). According to the function decomposition theory [26], its inverse can be represented as the summation of functions \( g_i^{-1} : 1 \leq i \leq r, \), i.e., \( g^{-1} = \frac{1}{r} \sum_{i=1}^{r} g_i^{-1} \).

For each \( g_i \) and \( \mathbb{R}_i = g_i(\mathbb{R}) \) is a submanifold in \( \mathbb{R} \), and it is
diffeomorphic to \( \mathcal{R} \). According to Theorem 2 in [41], \( \mathcal{H} - ACF \) is an universal approximator for each \( g_i, 1 \leq i \leq \tau \). Therefore, VFG \( f \) has universal approximation for immersion \( \mathcal{G} \rightarrow \mathcal{R} \).

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

The input coordinate index of \( g_r \) in \( \mathbb{R}^m \) is \( l_r = \lfloor 1, 2, \ldots, [(m + r)/2] \rfloor \), and the output index of \( g_r \) in \( \mathbb{R}^n \) is \( l_r + y = \lceil y + 1, y + 2, \ldots, [(m + r)/2] \rceil \), and \( y = (\tau - 1)m \). The input coordinate index of \( g_{r+1} \) in \( \mathbb{R}^m \) is \( l_{r+1} = \lfloor m - [(m + r)/2] + 1, \ldots, m - 1, m \rfloor \), and the output index of \( g_{r+1} \) in \( \mathbb{R}^n \) is \( l_{r+1} + y \). We can see that the \( m \) dimensions are divided into two sets, the overlapped set \( O = \lfloor m - [(m + r)/2] + 1, [(m + r)/2] \rfloor \), and the remaining set \( R \) containing the rest dimensions.

The mapping \( g : \mathcal{R} \rightarrow \mathcal{R} \) can be decomposed into \( \tau + 1 \) functions, i.e., \( g = [g_1, \ldots, g_\tau, g_{\tau+1}] \), and the inverse \( g^{-1} \) is adjusted here: \( g_j^{-1} = \frac{1}{\omega} \sum_{i=1}^{\tau} b_{ij} \). When \( j \in O, \omega = \tau + 1 \), and all \( g_j^{-1} \)'s will be involved; when \( j \in R, \omega = \tau \), and either \( g_j^{-1} \) or \( g_{\tau+1}^{-1} \) is omitted due to the missing of entry \( j \) in the function output. The mapping \( g_r \) is a diffeomorphism from manifold \( \mathcal{R}_r \) (\( \mathcal{R}_r \subset \mathcal{R} \)) to sub-manifold \( \mathcal{R}_r \) in \( \mathcal{R} \). Similarly \( g_{\tau+1} \) is a diffeomorphism from \( \mathcal{R}_{\tau+1} \) to manifold \( \mathcal{R}_{\tau+1} \). For each \( g_i, 1 \leq i \leq \tau + 1 \), it can be universally approximated with a function in \( \mathcal{H} - ACF \) [41]. Hence, we construct a VFG with universal approximation for any \( g \) in \( Q_1 \).

With the conditions in Theorem 4.4, VFG\( H - ACF \) is a distributional universal approximator as well [41].

5 ALGORITHM

In this section, we develop the training algorithm (Algorithm 1) to maximize the ELBO objective function (5). 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 states 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 (4)- (5) that can efficiently be optimized with standard stochastic optimizers.

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

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 a 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 (5) with the above algorithm, the node consistency rules in Section 3.2 are expected to be satisfied.

### Algorithm 1 Inference model parameters with forward and backward message propagation

1. **Input:** Data distribution \( \mathcal{D}, \mathcal{G} = \{\mathcal{V}, f\} \)
2. **for** \( s = 0, 1, \ldots \) **do**
3. Sample minibatch \( b \) samples \( \{x_1, \ldots, x_b\} \) from \( \mathcal{D} \);
4. **for** \( i \in \mathcal{V} \) **do**
5. **// forward message passing**
6. \( h^{(i)} = \frac{1}{|\mathcal{ch}(i)|} \sum_{j \in \mathcal{ch}(i)} f_{(i,j)}(h^{(j)}) \);
7. **end for**
8. \( \hat{h}^{(i)} = h^{(i)} \) if \( i \in \mathcal{O} \) or \( i \) in layer \( L \);
9. **for** \( i \in \mathcal{V} \) **do**
10. **// backward message passing**
11. \( \bar{h}^{(i)} = \frac{1}{|\mathcal{pa}(i)|} \sum_{j \in \mathcal{pa}(i)} f^{-1}_{(i,j)}(\bar{h}^{(j)}) \);
12. **end for**
13. \( h = [h^{(i)}]|i \in \mathcal{V}^\prime \), \( \hat{h} = [\hat{h}^{(i)}]|i \in \mathcal{V}^\prime \);
14. Approximate the KL terms in ELBO for each layer with \( b \) samples;
15. Updating VFG model \( G \) with gradient ascending: \( \theta_1^{(s+1)} = \theta_1^{(s)} + \nabla_{\theta_1} L(\{x_b, \theta_1^{(s)}\}) \).
16. **end for**

#### Figure 4: (Left) Inference on a VFG with single aggregation node.

Node 7 aggregates information from node 1 and 2, and passes down the update 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 (green) lines are forward (backward) messages.

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:

\[
\hat{h}^{(j)} = \frac{1}{|\mathcal{ch}(j)|} \sum_{i \in \mathcal{ch}(j) \cap \mathcal{O}} h^{(i)},
\]

where \( \mathcal{O} \) 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.

**Lemma 6.1.** Let \( \mathcal{G} \) 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 \( \hat{h}^{(i)} = f_{(i,a)}(f_{(i,a)}(x^{(i)})) \). Here \( f_{(i,a)} \) is the flow function path from node \( i \) to node \( a \).
We now focus on the task of imputing missing entries in a graph with graphical models that can perform inference on explicit graphs. The first application we present is missing value imputation. We introduce the main file, we also compared with KNN (k-nearest neighbor) method to impute this missing section. In addition to the three baselines in this set of experiments, we study different methods with synthetic datasets. The baselines for this set of experiments include mean value method (Mean), iterative imputation (Iterative) [8], and multivariate imputation by chained equation (MICE) [42]. Mean Squared Error as the metric of reference in order to compare the different methods for the imputation task. The experiments use the baseline implementations in [32].

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 $z_1 \sim N(0, 1.0^2)$ and $z_2 \sim N(1.0, 2.0^2)$ be the latent variables. For a sample $x$, 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 [13]. 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.

7.1 Evaluation on Inference with Missing Entries Imputation

We now focus on the task of imputing missing entries in a graph structure. 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.

### 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          |
first layer. Each flow function has $B = 4$ 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.

7.3 Comparison with Graphical Models. In this set of experiments, we use a synthetic Gaussian graphical model dataset from the blinearnet package [39] 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 $V = f(p_a(V), N_V)$. Here $p_a(V)$ 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. $f()$ is $V$’s generating function, and only linear functions are used in this dataset. All the noise terms are Normal distributions.

We take Bayesian network implementation [39] and sum-product network (SPN) package [30, 33] 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, i.e., VFGs are much more stable in this set of experiments.

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 |

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 [23], Planer [36], IAF [22], and SNF [43]. The evaluation datasets and setup are following two standard flow-based variational models: Sylvester Normalizing Flows [43] and [36]. 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 $\beta$ coefficient for the KL terms. Empirically, a small $\beta$ yields better ELBO and NLL values, and we set $\beta$ around 0.1 in the experiments.

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

Table 3 presents the negative evidence lower bound (ELBO) and the estimated negative likelihood (NLL) for all methods on 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 [13] number with $B = 4$, and following [43] we run multiple times to get the mean and standard derivation as well. VFG can achieve superior ELBO as well as NLL values on all three datasets compared against the baselines as given in Table 3. VFGs can achieve better variational inference and data distribution modeling results (ELBOs and NLLs) in Table 3 in part due to VFGs’ universal approximation power as given in Theorem 4.4. Also, 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 [28]. We construct a tree VFG model depicted in Figure 7. In the first layer, there are 4 flow functions, and each of them takes $14 \times 14$ image blocks as the input. Thus a $28 \times 28$ input image is divided into four $14 \times 14$ blocks as the input of VFG model. We use $B = 4$ for all the flows. The latent dimension for this model is $m = 196$. Following [40], the VFG model is trained with image labels to learn the latent representation of the input data. We set the parameters of $h^{1:L}$’s prior distribution as a function of image label, i.e., $\lambda(u)$, where $u$ denotes image label.

In practice, we use 10 trainable $\lambda$’s 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 6: Graph structure for Gaussian graphical model dataset.

Figure 7: MIST Tree structure.

Figure 8: (Top) original MNIST digits. (Bottom) reconstructed images using VFG.
There are advantages for the encoder and decoder to share parameters; see more discussion on the structures of VFGs in the sequel.

Variables do not impact the generation significantly. Latent variables that have obvious effects on images. Most of the \( m \) values are changed increasingly within a range centered at the value of the latent variable obtained from the last step. Figure 10 shows the change of images by increasing one latent variable from a small value to a larger one, presenting latent variables that have obvious effects on images. 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 brightness. \( i = 92 \), \( i = 157 \) and some variables not displayed here control the style of the generated digits.

7.3.1 Disentangled Representation. 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 within a range centered at the value of the latent variable obtained from the last step. Figure 10 shows the change of images by increasing one latent variable from a small value to a larger one, presenting latent variables that have obvious effects on images. 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 brightness. \( i = 92 \), \( i = 157 \) and some variables not displayed here control the style of the generated digits.

8 DISCUSSION

One motivation for proposing our VFG algorithm 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 to infer missing values; see more discussion on the structures of VFGs in the sequel.

8.1 Encoder-decoder Parameter Sharing

There are 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 ELBOs. Whereas without the intrinsic constraint of invertibility or any help or regularization from the encoder, VAE-based models have to learn an unsuited mapping function (decoder) to reconstruct all data samples with the latent variables, and there are always 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 [46, 52] and is left for future works.

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 [41], 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 experiments. Future work includes applying our VFG model to relational data structure learning and reasoning.
A ELBO OF TREE VFGS

The hierarchical generative model is given by factorization

\[ p = \prod_{i=1}^{L} p(h_i | h_{i+1}). \]

The probability density function \( p(h_i | h_{i+1}) \) in the generative model is modeled with one or multiple invertible normalizing flow functions. The hierarchical posterior (recognition network) is factorized as

\[ q_{\theta}(h|x) = q(h^1|x)q(h^2|h^1) \cdots q(h^L|h^{L-1}). \]

Let each data sample has \( k \) sections, i.e., \( x = [x^{(1)}, \ldots, x^{(k)}] \). VFGs are graphical models that can model each pair of connected nodes, the edge is an invertible flow function. The vector of parameters for all the edges is denoted by \( \theta \). The forward message passing starts from \( x \) and ends at \( h^L \), 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

\[ p(x|\theta) = \sum_{h^1, \ldots, h^L} p(h^L|\theta)p(h^{L-1}|h^L, \theta) \cdots p(h^1, \theta). \]

The hierarchical generative model is given by factorization

\[ p(h) = p(h^L)\prod_{i=1}^{L-1} p(h^i | h^{i+1}). \]  \hspace{1cm} (11)

Reconstruction of data

\[ KL^{1:L} \]  \hspace{1cm} (15)

With conditional independence in the hierarchical structure, we have

\[ q(h^{i:L}|x) = q(h^{2:L}|h^1|x)q(h^1|x) = q(h^{2:L}|h^1)q(h^1|x). \]

The second term of (15) can be further expanded as

\[ KL^{1:L} = \mathbb{E}_{q(h^{1:L}|x)} \left[ \log q(h^1|x) + \log q(h^{2:L}|h^1) - \log p(h^1|h^{2:L}) - \log p(h^{2:L}) \right]. \]  \hspace{1cm} (16)

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

\[ KL^{1:L} = \mathbb{E}_{q(h^{1:L}|x)} \left[ \log q(h^1|x) - \log p(h^1|h^2) + \log q(h^{2:L}|h^1) - \log p(h^{2:L}) \right] \]

\[ + \mathbb{E}_{q(h^{1:L}|x)} \left[ \log q(h^{2:L}|h^1) - \log p(h^{2:L}) \right]. \]

The ELBO (15) can be written as

\[ \mathcal{L}(x; \theta) = \mathbb{E}_{q(h^{1:L}|x)} \left[ \log p(x|h^{1:L}) \right] - \sum_{l=1}^{L-1} KL^l - KL^L. \]  \hspace{1cm} (18)

When \( 1 \leq l \leq L - 1 \)

\[ KL^l = \mathbb{E}_{q(h^{1:L}|x)} \left[ \log q(h^l|h^{l-1}) - \log p(h^l|h^{l+1}) \right]. \]  \hspace{1cm} (19)

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

\[ \log p(x) = \log \int p(x|h)p(h)\,dh \]

\[ = \log \int \frac{q(h|x)}{\theta(q|h)} p(x|h)p(h)\,dh \]

\[ \geq \mathbb{E}_{q(h|x)} \left[ \log p(x|h) - \log q(h|x) + \log p(h) \right] = \mathcal{L}(x; \theta). \]

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

\[ \log p(x) \geq \mathcal{L}(x; \theta) \]

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

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

\[ \mathcal{L}(x; \theta) = \mathbb{E}_{q(h^1|x)} \left[ \log p(x|h^1) \right] - \sum_{l=1}^{L} KL^l. \]  \hspace{1cm} (20)
The KL term (19) becomes
\[ KL^l = \mathbb{E}_{q(h^{l+1} | x)} \left[ \log q(h^l | h^{l-1}) - \log p(h^l | h^{l+1}) \right]. \]
When \( l = L \),
\[ KL^L = \mathbb{E}_{q(h^{L+1} | x)} \left[ \log q(h^L | h^{L-1}) - \log p(h^L) \right]. \]

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 \( h \) the vector of latent variables, then following (15) we develop the ELBO as
\[
\log p(x) \geq \mathcal{L}(x; \theta) = \mathbb{E}_{q(h|x)} \left[ \log p(x, h) \right] - \mathbb{E}_{q(h|x)} \left[ \log q(h|x) \right] - KL \]
Reconstruction of the data
Similarly the KL term can be expanded as in the tree structures. For nodes in layer \( l \)
\[
KL^{LL} = \mathbb{E}_{q(h^{L+1} | x)} \left[ \log q(h^L | h^{L-1}) - \log p(h^L) \right].
\]
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
\[
q(h^L | h^{L-1}) = q(h^L | h_{ch(l)}) q(h^L | h_{pa(l)}) \]
\[
= q(h^L | h_{ch(l)}) q(h^{L+1} | h_{pa(l)}) \]
\[
= q(h^L | h_{ch(l)}) p(h^{L+1}) \] (23)

Following (17) and with (22-23), we have
\[
KL^{LL} = \mathbb{E}_{q(h^{L+1} | x)} \left[ KL \right] + \mathbb{E}_{q(h^L | x)} \left[ \log q(h^L | h_{ch(l)}) - \log p(h^L | h_{pa(l)}) \right].
\]

Furthermore,
\[
q(h^L | h^{L-1}) = q(h^L | h_{ch(l)}) \]
\[ p(h^L | h^{L+1}) = p(h^L | h_{pa(l)}) \]
Hence,
\[
KL^{LL} = \mathbb{E}_{q(h^{L+1} | x)} \left[ KL \right] + \mathbb{E}_{q(h^L | x)} \left[ KL \right] \]

For nodes in layer \( l \),
\[
KL^l = \sum_{i \in ll} \mathbb{E}_{q(h^{L+1} | x)} \left[ KL \right] = \sum_{i \in ll} \mathbb{E}_{q(h^{L+1} | x)} \left[ KL \right]
\]

Recursively applying (24) to (21) yields
\[
\mathcal{L}(x; \theta) = \mathbb{E}_{q(h|x)} \left[ \log p(x) \right] - \sum_{i \in ll} \mathbb{E}_{q(h^{L+1} | x)} \left[ KL \right]
\]
For node \( i \),
\[
KL^{(i)} = \mathbb{E}_{q(h^{i} | x)} \left[ \log q(h^{i} | h_{ch(i)}) - \log p(h^{i} | h_{pa(i)}) \right].
\]

C IMPROVE TRAINING OF VFG

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
\[
\mathcal{L}(x, O_{x}; \theta) = \sum_{i \in ll} \mathbb{E}_{q(h^{i} | x)} \left[ \log p(x) \right] - \sum_{i \in ll} \mathbb{E}_{q(h^{L+1} | x)} \left[ KL \right]
\]
where \( O_{x} \) is the index set of leaf nodes with observation, and \( x^{O_{x}} \) is the union of observed data sections. The random-masking training procedure for objective (25) 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 \( \theta \) with the gradient of (25) to ensure the distribution learning running well.

**Algorithm 2 Inference model parameters with random masking**

1. **Input**: Data distribution \( D \), \( \mathcal{O} = \{ V, f \} \)
2. for \( s = 0, 1, \ldots \) do
3. Sample minibatch \( b \) samples \( \{ x_1, \ldots, x_b \} \) from \( D \); optimize (4) with Line 4 to Line 15 in Algorithm 1;
4. Sample a subset of the \( k \) data sections as data observation set \( O_{x}; O \leftarrow O \cup O_{x}; \)
5. for \( i \in V \) do
6. // forward message passing
7. \( h^{(i)} = \frac{1}{|ch(i)|} \sum_{j \in ch(i) \cap O} f_{i,j}(h^{(j)}); \)
8. \( O \leftarrow O \cup \{ i \} \) if \( ch(i) \cap O \neq \emptyset; \)
9. end for
10. \( h^{(i)} = h^{(i)} \) if \( i \in \mathcal{R}_O \) or \( i \) is in layer \( L; \)
11. for \( i \in V \) do
12. // backward message passing
13. \( h^{(i)} = \frac{1}{|pa(i)|} \sum_{j \in pa(i)} f^{-1}_{i,j}(h^{(j)}); \)
14. end for
15. \( h = \{ h^{(i)} \}_{i \in V \cap O}, O = \{ h^{(i)} \}_{i \in V}; \)
16. Approximate the KL terms in ELBO for each layer with \( b \) samples;
17. Updating VFG with gradient of (25): \( \theta_{f}^{(s+1)} = \theta_{f}^{(s)} + \nabla_{\theta_{f}} \frac{1}{b} \sum_{i=1}^{b} \mathcal{L}(x_{b}, O_{x}; \theta_{f}^{(s)}), \)
18. end for