Variational Flow Graphical Model

Shaogang Ren Belhal Karimi Ping Li

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 carefully 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 graphical inference on datasets. Multiple experiments on synthetic and real datasets show the benefits of our proposed method.

Key words: Variational Inference, Flow-based Models, Graphical Models; Generative Models, Tractable Inference

1 Introduction

There are two general approaches for graphical inference: *exact inference* and *approximate inference*. Exact inference [25, 13] resorts to an exact numerical calculation procedure of the quantity of interest. However, in most cases, exact inference is either *computationally involved* or simply *intractable*. Variational Inference (VI) is computationally efficient and has been applied to tackle the large scale inference problem [12, 10]. In Variational Inference, mean-field approximation [30] and variational message passing [3, 29] are two common approaches for graphical models. These approximation 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 [11], arithmetic circuits [5], and-or search spaces [20], multi-valued decision diagrams [6], cutset networks [21], sum-product nets [24], probabilistic sentential decision diagrams [17], and probabilistic circuits [4]. 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.

Apart from probabilistic inference, generative models have been developed to model high dimensional datasets and to learn the 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) [9] are widely applied to different categories of datasets. Flow-based models [8, 7, 22, 2] leverage invertable 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 lie in the stream of tractable neural networks that allow to perform inference on graphical structures. Sum-product networks [24] and probabilistic circuits [4] 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 approximation and consistency of nodes in graphical structures for tractable inference. Our contributions are summarized as follows.

Contributions. Dealing with high dimensional data using graphical models exacerbates this systemic inability to model the latent structure of the data efficiently, i.e., to sample from the posterior distribution of the latent variables given the observed data. To overcome these significant limitations, we propose a new framework, a variational hierarchical graphical flow model:

- **Hierarchical and Flow-Based:** Introducing the Variational Flow Graphical (VFG) model, we propose a novel graph 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.
- **Numerical Inference:** Aggregation nodes are introduced into our model in order to integrate hierarchical information through a variational forward-backward message passing scheme. We highlight the benefits of our VFG model on numerical applications: the missing values imputation problem and the numerical inference on graphical datasets. VFGs can achieve improved ev-

idence lower bound (ELBO) and likelihood values due to the implicitly invertible flow-based model structure.

- **Representation Learning:** We specifically show in our experiments that our model achieves to disentangle the factors of variation underlying the high dimensional data given as input.

Roadmap: Section 2 presents important concepts used in the paper. Section 3 introduces the Variational Flow Graphical (VFG) model to tackle the posterior distribution learning of high dimensional data. Section 4 provides the algorithms used to train VFG models. Section 5 discusses how to perform inference with a trained VFG model. Section 6 showcases the advantages of VFG on various tasks. Section 7 and Section 8 provide a discussion and conclusion of the paper.

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

2 Preliminaries

We introduce the general principles and notations of variational inference and flow-based models then explain how they can naturally be embedded with graphical models.

Variational Inference: Following the setting discussed above, the functional mapping $\mathbf{f}$: $\mathbf{x} \to \mathbf{z}$ can be viewed as an encoding process and the mapping $\mathbf{f}^{-1}$: $\mathbf{z} \to \mathbf{x}$ as a decoding one: $\mathbf{z} \sim p(\mathbf{z}), \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$. To learn the parameters θ, we maximize the following marginal log-likelihood $\log p_\theta(\mathbf{x}) = \log \int p(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}) d\mathbf{z}$. Direct optimization of the log-likelihood is usually not an option due to the intractable latent structure. Instead VI employs a parameterized family of so-called variational distributions $q_\phi(\mathbf{z}|\mathbf{x})$ to approximate the true posterior $p_\theta(\mathbf{z}|\mathbf{x}) \propto p(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z})$. The goal of VI is to minimize the distance, in terms of Kullback-Leibler ($\mathbf{KL}$) divergence, between the variational candidate and the true posterior $\mathbf{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$. This optimization problem can be shown to be equivalent to maximizing the following evidence lower bound (ELBO) objective, noted $\mathcal{L}(\mathbf{x}; \theta)$:

$$
\begin{aligned}
\log p_\theta(\mathbf{x}) &\geqslant \mathcal{L}(\mathbf{x}; \theta) \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] - \mathbf{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \\
&:= -\mathcal{F}(\theta, \phi)\,.
\end{aligned} \tag{2.1}
$$

In Variational Auto-Encoders (VAEs, [16, 23]), the calculation of the reconstruction term requires sampling from the posterior distribution along with using the reparameterization trick, i.e.,

$$
\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] \simeq \frac{1}{M} \sum_{m=1}^{M} \log p(\mathbf{x}|\mathbf{z}_m). \tag{2.2}
$$

Here M 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 [8, 7, 22, 2] correspond to a probability distribution transformation using a sequence of invertible and differentiable mappings, noted $\mathbf{f} : \mathcal{Z} \to \mathcal{X}$ between two random variables $z \in \mathcal{Z}$ of density $p(\mathbf{z})$ and $x \in \mathcal{X}$. The observed variable $\mathbf{x} \sim p_\theta(\mathbf{x})$ is assumed to be distributed according to an unknown distribution $p_\theta(\mathbf{x})$ parameterized by some parameter θ. 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:

$$
\begin{aligned}
\log p_\theta(\mathbf{x}) &= \log p(\mathbf{z}) + \log \left| \det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right) \right| \\
&= \log p(\mathbf{z}) + \sum_{i=1}^{L} \log \left| \det\left(\frac{\partial \mathbf{h}^i}{\partial \mathbf{h}^{i-1}}\right) \right|,
\end{aligned} \tag{2.3}
$$

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. (2.3) 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 (by sampling from the initial density and applying the transformations). [22] 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 negative EBLO is given by

$$
\begin{aligned}
\mathcal{F}(\theta, \phi) =& \mathbb{E}_{q_\phi}\big[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{x}, \mathbf{z})\big] \\
=& \mathbb{E}_{q_0}\big[\log q_0(\mathbf{z}_0|\mathbf{x}) - \log p_\theta(\mathbf{x}, \mathbf{z})\big] \\
&- \mathbb{E}_{q_0}\big[\sum_{k=1}^{K} \log \big|\det\big(\frac{\partial \mathbf{f}_k(\mathbf{z}_k; \psi_k)}{\partial \mathbf{z}_k}\big)\big|\big].
\end{aligned} \tag{2.4}
$$

Here $\mathbf{f}_k$ is the kth 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 parameters of 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 [28] 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 log-likelihood ratio asymmetry for causal-effect inference. Here, we propose a framework that generalizes flow-based models [8, 7, 22, 2] to graphical variable inference.

3 Variational Flow Graphical Model

Assume that there exist a relation between each data sample components and their corresponding variable in the latent space. Then, it is possible to define a graphical model using

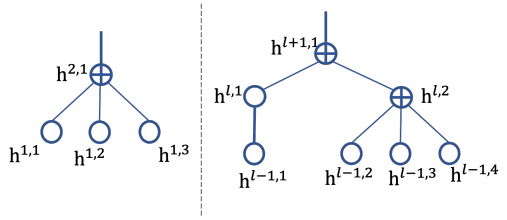

Figure 1: (Left) Node $\mathbf{h}^{2,1}$ connects its children with invertible functions. Messages from the children are aggregated at the parent node, $\mathbf{h}^{2,1}$; $\oplus$ is an aggregation node, and circles stand for non-aggregation nodes.(Right) An illustration of the latent structure from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

normalizing flows, as introduced Section 2, leading to exact latent variable inference and log-likelihood evaluation of the model. We call this model a *Variational Flow Graphical Model* (VFG) and introduce it in the sequel.

3.1 Evidence Lower Bound of Variational Flow Graphical Models A VFG model $\mathcal{G} = \{\mathcal{V}, \mathbf{f}\}$ consists of a set of nodes ($\mathcal{V}$) and a set of edges ($\mathbf{f}$). Figure 1-Right gives an illustration of a tree structure induced by a VFG model. 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 other nodes with a single flow function or an identity function. An aggregation node has multiple children, and it connects with each of them with an identity function. Unlike classical graphical models, a node in a VFG model may represent multiple variables. A node only belongs to one layer, and one layer can have multiple nodes. 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. We apply variational inference to assemble the layers of a VFG and learn parameters from data samples. Different from VAEs [16, 23], 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.

The hierarchical generative network comprises L layers, $\mathbf{h}^l$ denotes the latent variable in layer l, and θ is the vector of model parameters. We use $\mathbf{h}^{l,i}$ to denote the ith node's latent variable in layer l, and $\mathbf{h}^{(j)}$ to represent node j's latent variable without specification of the layer number, and j is the node index on a tree or graph. The joint distribution of the hierarchical model is given by:

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

where $\mathbf{h} = \{\mathbf{h}^1, \cdots, \mathbf{h}^L\}$ denotes the set of latent variables of the model. The hierarchical generative model is given by factorization $p(\mathbf{x}|\mathbf{h}^L) = \mathbf{\Pi}_{l=1}^{L-1} p(\mathbf{h}^l|\mathbf{h}^{l+1})p(\mathbf{x}|\mathbf{h}^1)$, and the prior distribution is $p(\mathbf{h}^L)$. Note that only the *root nodes* have *prior distributions*.

The probability density function $p(\mathbf{h}^{l-1}|\mathbf{h}^l)$ in the generative model is parameterized with one or multiple invertible flow-based functions. We follow the variational inference to approximate the posterior distribution of latent variables. The hierarchical posterior (recognition model) is factorized as

$$q_\theta(\mathbf{h}|\mathbf{x}) = q(\mathbf{h}^1|\mathbf{x})q(\mathbf{h}^2|\mathbf{h}^1)\cdots q(\mathbf{h}^L|\mathbf{h}^{L-1}). \tag{3.5}$$

Evaluation of the posterior(recognition model) equation 3.5 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 given by

$$\begin{aligned}\log p_\theta(\mathbf{x}) &\geqslant \mathcal{L}(\mathbf{x};\theta)\\ &= \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log p(\mathbf{x}|\mathbf{h}^{1:L})\big] - \sum_{l=1}^{L}\mathbf{KL}^l.\end{aligned} \tag{3.6}$$

Here $\mathbf{KL}^l$ is the Kullback-Leibler divergence between the posterior and generative model in layer l. The first term in (3.6) evaluates data reconstruction. When $1 \leqslant l \leqslant L$,

$$\mathbf{KL}^l = \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log q(\mathbf{h}^l|\mathbf{h}^{l-1}) - \log p(\mathbf{h}^l|\mathbf{h}^{l+1})\big]. \tag{3.7}$$

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 (3.6) 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:

$$\begin{aligned}\mathcal{L}(\mathbf{x};\theta) =& \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\big[\log p(\mathbf{x}|\mathbf{h})\big] - \sum_{i\in\mathcal{V}\setminus\mathcal{R}_\mathcal{G}}\mathbf{KL}^{(i)}\\ &- \sum_{i\in\mathcal{R}_\mathcal{G}}\mathbf{KL}\big(q(\mathbf{h}^{(i)}|\mathbf{h}^{ch(i)})||p(\mathbf{h}^{(i)})\big).\end{aligned} \tag{3.8}$$

Here $\mathbf{KL}^{(i)} = \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\big[\log q(\mathbf{h}^{(i)}|\mathbf{h}^{ch(i)}) - \log p(\mathbf{h}^{(i)}|\mathbf{h}^{pa(i)})\big]$. $\mathcal{R}_\mathcal{G}$ is the set of root or source nodes of DAG $\mathcal{G} = \{\mathcal{V}, \mathbf{f}\}$. Assuming there are k leaf nodes on a tree or a DAG model, corresponding to k sections of the input sample $\mathbf{x} = [\mathbf{x}^{(1)}, ..., \mathbf{x}^{(k)}]$, then the hidden variables in both (3.6) and (3.8) are computed with forward and backward message passing.

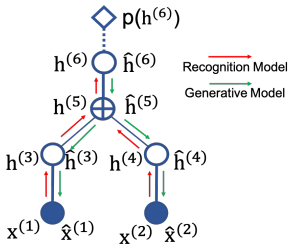

Figure 2: Recognition model consists of froward message from data to approximate the posterior distributions; the generative model is realized by backward message from the root node and generates the samples or reconstructions in each layer.

3.2 ELBO Calculation Maximizing the ELBO (3.6) equals to optimizing the parameters of the flows, θ. Similar to VAEs [16, 23], 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. VFGs are hierarchical aggregation model that requires information from all the leaf nodes aggregates at the root nodes.

For the following sections, we use $\mathbf{h}^l$ to represent the node state of layer l in the forward message, and $\widehat{\mathbf{h}}^l$ for the node state in the backward message. For all layer, both $\mathbf{h}^l$ and $\widehat{\mathbf{h}}^l$ are sampled from the posterior. At the rood nodes, we have $\widehat{\mathbf{h}}^{\mathcal{R}} = \mathbf{h}^{\mathcal{R}}$. The calculation of the data reconstruction term in equation 3.6 requires samples of $\mathbf{h}^l$ and $\widehat{\mathbf{h}}^l$ from the posterior. It corresponds to the encoding and decoding procedures in VAE model [16, 23] as given by Eq. (2.2). Specifically, we have

$$\mathbf{h}^l \sim q(\cdot|\mathbf{h}^{l-1}) \quad \text{where} \quad 1 \leqslant l \leqslant L\,., \tag{3.9}$$

$$\widehat{\mathbf{h}}^l \sim p(\cdot|\widehat{\mathbf{h}}^{l+1}) \quad \text{where} \quad 0 \leqslant l \leqslant L-1\,. \tag{3.10}$$

At the root node, we have $\widehat{\mathbf{h}}^L = \mathbf{h}^L \sim q(\cdot|\mathbf{h}^{L-1})$. The computation of $\mathbf{h}^l$s and $\widehat{\mathbf{h}}^l$s with (3.9) and (3.10) requires message passing via the flow functions shown in Figure 2. Latent node states in both (3.9) and (3.10) are directly obtained with flow functions.

The reconstruction term in ELBO (3.6) can be computed with the backward message from the generative model $p(\mathbf{x}|\widehat{\mathbf{h}}^1)$, i.e.,

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

For a VFG model, we set $M = 1$. In the last term, $p(\mathbf{x}|\widehat{\mathbf{h}}^1)$ is either Gaussian or Binary distribution parameterized with $\widehat{\mathbf{x}}$ generated via the flow function with $\widehat{\mathbf{h}}^1$ as the input. For any $l \in [L]$, the calculation of the $\mathbf{KL}^l$ term is done in a similar manner, and equation 3.7 requires both forward message from the posterior and backward message from the generative model, i.e.,

$$\begin{aligned}
\mathbf{KL}^l =& \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log q(\mathbf{h}^l|\mathbf{h}^{l-1}) - \log p(\mathbf{h}^l|\widehat{\mathbf{h}}^{l+1})\big] \\
&\simeq \log q(\mathbf{h}^l|\mathbf{h}^{l-1}) - \log p(\mathbf{h}^l|\widehat{\mathbf{h}}^{l+1}).
\end{aligned} \tag{3.11}$$

In a tree structured VFG, each layer may consist of multiple nodes. In layer l, each node's $\mathbf{KL}$ value can be computed individually, thus $\mathbf{KL}^l = \sum_{i\in l}\mathbf{KL}^{(i)}$, and here each i represents a node. For a DAG structure, we can individually evaluate each node's $\mathbf{KL}$ term.

3.3 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-Left 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, we observe that at node i

$$\begin{aligned}
\mathbf{h}^{(i)} &= \frac{1}{|ch(i)|}\sum_{j\in ch(i)}\mathbf{f}_{(j,i)}(\mathbf{h}^{(j)}), \\
\widehat{\mathbf{h}}^{(i)} &= \frac{1}{|pa(i)|}\sum_{j\in pa(i)}\mathbf{f}^{-1}_{(i,j)}(\widehat{\mathbf{h}}^{(j)})\,.
\end{aligned}$$

Notice that the above two equations hold even when node i has only one child or parent. In the sequel, we consider that all latent variables, noted $\mathbf{h}^{l,i}$, for all $l \in [L]$ and $i \in \mathbb{N}$, are distributed according to Laplace distributions. With the identity function between the parent and its children, there are *node consistency rules* regarding an *average aggregation node*: *(a)* the parent value is the mean of its children, i.e., $\mathbf{h}^{(i)} = \frac{1}{|ch(i)|}\sum_{j\in ch(i)}\mathbf{h}^{(j)}$; *(b)* the child node have the same reconstruction value with its parent, i.e., $\widehat{\mathbf{h}}^{(j)} = \widehat{\mathbf{h}}^{(i)}, \forall j \in ch(i)$.

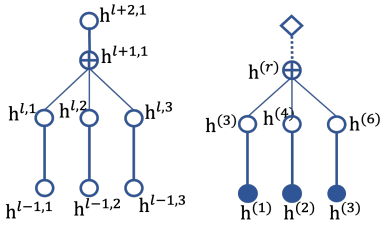

Figure 3: (Left) Aggregation node $\mathbf{h}^{l+1,1}$ has three children, $\mathbf{h}^{l,1}$, $\mathbf{h}^{l,2}$, and $\mathbf{h}^{l,3}$. (Right) A VFG model with one aggregation node, $\mathbf{h}^{(r)}$. Solid circles are nodes with observed values, and the diamond is the prior for the root node.

We use Figure 3-Right as an example to illustrate a simple model that has only one average aggregation node.

Then (3.6) yields the ELBO,

$$
\begin{aligned}
(3.12) \qquad & \log p(\mathbf{x}) \geqslant \mathcal{L}(\mathbf{x};\theta) \\
& = \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\big[\log p(\mathbf{x}|\widehat{\mathbf{h}}^1)\big] - \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\big[\log q(\mathbf{h}^1|\mathbf{x}) \\
& - \log p(\mathbf{h}^1|\widehat{\mathbf{h}}^2)\big] - \mathbf{KL}\big(q(\mathbf{h}^2|\mathbf{h}^1)|p(\mathbf{h}^2)\big).
\end{aligned}
$$

From Figure 3-Right, $\mathbf{h}^{(r)}$ is the root, and has k children, $\mathbf{h}^{(t)}, t = 1, ..., k$, and $k = 3$. With $\mathbf{f}_t$ as the flow function connecting $\mathbf{h}^{(t)}$ and $\mathbf{x}^{(t)}$, according to the aggregation rules, we get:

$$
\begin{aligned}
(3.13) \qquad & \mathbf{h}^{(t)} = \mathbf{f}_t(\mathbf{x}^{(t)}), \quad \widehat{\mathbf{h}}^{(r)} = \mathbf{h}^{(r)} = \frac{1}{k}\sum_{t=1}^{k}\mathbf{h}^{(t)}, \\
& \widehat{\mathbf{h}}^{(t)} = \widehat{\mathbf{h}}^{(r)},\ t = 1, ..., k\,.
\end{aligned}
$$

Assume the data to be normally distributed, then

$$
\log p(\mathbf{x}|\widehat{\mathbf{h}}^1) = -\sum_{t=1}^{k}\Bigg\{\underbrace{\frac{1}{2\sigma_{\mathbf{x}}^2}\big|\big|\mathbf{x}^{(t)} - \mathbf{f}_t^{-1}(\widehat{\mathbf{h}}^{(r)})\big|\big|_2^2}_{\text{By } \widehat{\mathbf{x}}^{(t)}=\mathbf{f}_t^{-1}(\widehat{\mathbf{h}}^{(t)})=\mathbf{f}_t^{-1}(\widehat{\mathbf{h}}^{(r)})}\Bigg\} + C
$$

(3.14)

$$
\log p(\mathbf{h}^1|\widehat{\mathbf{h}}^2) = -\sum_{t=1}^{k}\Big\{\underbrace{\big|\big|\mathbf{f}_t(\mathbf{x}^{(t)}) - \widehat{\mathbf{h}}^{(r)}\big|\big|_1}_{\text{By } \widehat{\mathbf{h}}^2=\widehat{\mathbf{h}}^{(r)},\ \mathbf{h}^{(t)}=\mathbf{f}_t(\mathbf{x}^{(t)})}\Big\} + C.
$$

Here $\sigma_{\mathbf{x}}$ is a constant standard deviation. We notice that maximizing the ELBO, or minimizing the $\mathbf{KL}$ term of an aggregation node implies the aggregation rule (b).

4 Algorithms and Implementation

In this section, we develop the training algorithm, see Algorithm 1, that outputs the fitted vector of parameters resulting from the maximization of the ELBO objective function (3.6) or (3.8) depending on what graph structure is used. In Algorithm 1, the inference of the latent variables is performed via forwarding message passing, cf. Line 6, and their reconstructions are computed in backward message passing, cf. Line 11. Unlike Variational Autoencoders (VAE), the variance of latent variables in a VFG is approximated rather than parameterized with neural networks. A VFG is a deterministic network passing latent variable values between nodes. The reconstruction (likelihood) terms in each layer are computed with forward and backward node states. Ignoring explicit neural network parameterized variances for all latent nodes enables us to use flow-based models as both the encoders and decoders. We thus obtain a deterministic ELBO objective (3.6)- (3.8) that can efficiently be optimized with standard stochastic optimizers.

4.1 Layer-wise Training 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

Algorithm 1 Inference model parameters with forward and backward message propagation

1: **Input:** Data distribution $\mathcal{D}$, $\mathcal{G} = \{\mathcal{V}, \mathbf{f}\}$
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 $i \in \mathcal{R}_{\mathcal{G}}$ or $i \in$ layer L;
9: **for** $i \in \mathcal{V}$ **do**
10: // backward message passing
11: $\widehat{\mathbf{h}}^{(i)} = \frac{1}{|pa(i)|}\sum_{j\in pa(i)}\mathbf{f}^{-1}_{(i,j)}(\widehat{\mathbf{h}}^{(j)})$;
12: **end for**
13: $\mathbf{h} = \{\mathbf{h}^{(t)}\big|t \in \mathcal{V}\}, \widehat{\mathbf{h}} = \{\widehat{\mathbf{h}}^{(t)}\big|t \in \mathcal{V}\}$;
14: Approximate the $\mathbf{KL}$ terms in ELBO for each layer with b samples;
15: Updating VFG model $\mathcal{G}$ 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**

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 (3.6) or (3.8) with the above algorithm, the *node consistency rules* in Section 3.3 are expected to be satisfied. We can improve the inference on sub-graphs by using the random masking method introduced in Section C.

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 $q(\mathbf{h}|\mathbf{x})$.

5 Inference on VFG Models

Given a VFG model, we aim to infer the state of any node given observed ones. Relations between variables at different nodes can also be inferred via our flow-based graphical model.

The hidden state of the parent node j in an aggregation model can be computed by the observed children as follows:

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

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 given node is updated via messages received from its children. 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:

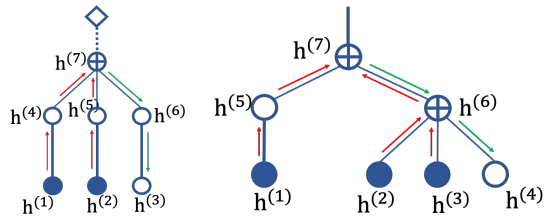

Figure 4: (Left) Inference on model with single aggregation node. Node 7 aggregates information from node 1 and 2, and pass down the updated state to node 3 for prediction. (Right) Inference on a tree model. Observed node states are gathered at node 7 to predict the state of node 4. Red and green lines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

LEMMA 5.1. *Let $\mathcal{G}$ be a trained variational flow graphical model (with a tree structure) with L layers, and i and j are two leaf nodes with a as the closest common ancestor. 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. Without loss generality, we assume that there are relationships among different data sections, and the value of one section can be imputed by other sections. According to the *node consistency rule (b)* discussed in section 3.3, at an aggregation node a, the latent value of a child node j has the same reconstruction value as the parent node. The reconstruction of the child node j can be calculated with the reconstruction of the parent node, i.e., $\widehat{\mathbf{h}}^{(j)} = \mathbf{f}_{(a,j)}(\widehat{\mathbf{h}}^{a})$. Recalling the reconstruction term in the ELBO (3.6), at each node we have $\mathbf{h}^{(a)} = \widehat{\mathbf{h}}^{(a)}$. Hence for node a's descendent j, we have $\widehat{\mathbf{h}}^{(j)} = \mathbf{f}_{(a,j)}(\mathbf{h}^{(a)})$, and $\mathbf{f}_{(a,j)}$ is the flow function path from a to j. The value of node a can be computed by the value of its descendent i, i.e., $\mathbf{h}^{(a)} = \mathbf{f}_{(i,a)}(\mathbf{h}^{(i)})$. Hence, we have $\widehat{\mathbf{x}}^{(j)} = \mathbf{f}_{(a,j)}(\mathbf{f}_{(i,a)}(\mathbf{x}^{(i)}))$. □

Considering the flow-based model (2.3), we have the following identity for each node of the graph structure:

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

Let O be the set of observed leaf nodes, j be an unobserved node, and a the closest ancestor of $O \cup \{a\}$. Then the state of j can be imputed with $\widehat{\mathbf{x}}^j = \mathbf{f}_{(a,j)}(\mathbf{f}_{(O,a)}(\mathbf{x}^{(i)}))$, where $\mathbf{f}_{(O,a)}$ is the flow function path from all nodes in O to a. Lemma 5.1 provides an approach to conduct inference on a tree and impute missing values in the dataset. It is easy to extend the inference method to DAG-VFG models.

6 Numerical Experiments

The first application we present is the imputation of missing values. We compare our method with several baseline models on a synthetic dataset. 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 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 experiments, the VFG structures are 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 [31, 28].

6.1 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 use the Mean Squared Error as the metric of reference in order to compare the different methods for the imputation task.

Baselines: We use the following for data imputation:

- *Mean Value:* Using training set mean values to replace the missing entries in the testing set.
- *Iterative Imputation:* A strategy for imputing missing values by modeling each feature with missing values as a function of other features in a Round-Robin fashion.
- *Multivariate Imputation by Chained Equation (MICE):* This method imputes the missing entries with multiple rounds of inference. This method can handle different types of data.

Synthetic dataset: In this set of experiments, we study the proposed model with synthetic datasets. 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

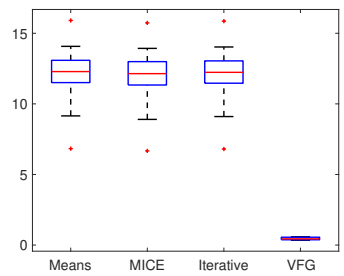

Figure 5: Synthetic datasets: MSE boxplots of VFG and baseline methods.

Table 1: 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 [22]	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	$\mathbf{80.80 \pm 0.76}$	$\mathbf{63.66 \pm 0.14}$	$\mathbf{67.26 \pm 0.53}$	$\mathbf{65.74 \pm 0.84}$	$\mathbf{80.16 \pm 0.73}$	$\mathbf{78.65 \pm 0.66}$

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 [8]. 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.

6.2 ELBO and Likelihood We further qualitatively compared VFG with existing methods on variational inference using three standard datasets. The evaluation datasets and setup are exactly following two standard flow-based variational models, sylvester normalizing flows [2] and [22]. We use the tree VFG structure as shown in Figure 6-Left for three datasets. We train the tree-VFG model using the following ELBO objective (equation 6.16). Empirically, a small β yields better ELBO and NLL values, and we set β around 0.1 in the experiments.

(6.16)

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

In Table 1, we provide the negative evidence lower bound (-ELBO) and the estimated negative likelihood (NLL) for baseline methods on three datasets, MNIST, Caltech101, and Omniglot. The results of the baseline methods are from [2]. These 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[8] 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 and NLL values as given in Table 1. The main reason why VFGs can achieve superior variational inference results (ELBOs and NLLs) in Table 1 is due to VFGs's approximately invertiable flow-based model structure. The intrinsic invertibility ensures the decoder or generative model in the VFG to achieve smaller reconstruction errors for data samples and hence smaller NLL values.

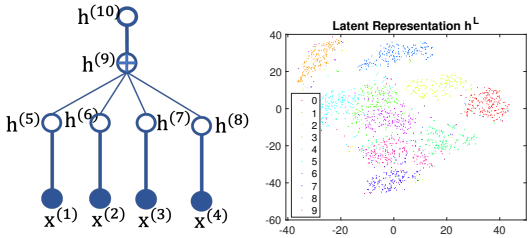

Figure 6: (Left) Tree structure for MNIST; (Right) MNIST: t-SNE plot of latent variables from VFG learned with labels.

6.3 Latent Representation Learning on MNIST In this set of experiments, we evaluate Variational Flow Graphical Models on latent representation learning of the MNIST dataset [18]. We construct a tree VFG model depicted in Figure 6-Left. 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. The latent dimension for this model is 196. Following [26], 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 regarding the 10 digits. The images in the second row of Figure 7 are reconstructions of MNIST samples extracted from the testing set, displayed in the first row of the same Figure, using our proposed VFG model.

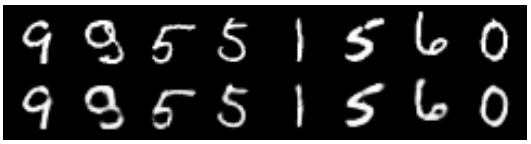

Figure 7: (Top) original MNIST digits. (Bottom) reconstructed images using VFG.

Figure 6-Right shows t-distributed stochastic neighbor embedding (t-SNE) [19] plot of $2,000$ testing images's latent variables learned with our model, and 200 for each digit.

Figure 6 illustrates that VFG can learn separated latent representations to distinguish different hand-written numbers.

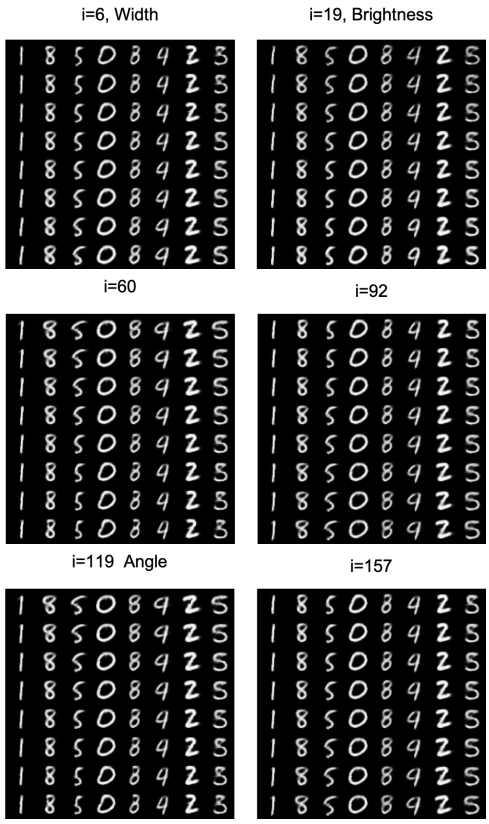

Figure 8: MNIST: Increasing each latent variable from a small value to a larger one.

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 8 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 $d = 196$ 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.

7 Discussion

One of the motivations for VFG is to develop a tractable model that can be used for inference and sampling on datasets. As long as the variable states in the aggregation nodes are consistent, we can always apply VFGs to missing value inference. We provide more discussion on the structures of VFGs below.

7.1 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 achieve 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.

7.2 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 can be used 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. 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 [31, 28].

VFGs rely on minimizing the KL term to achieve *consistency* among the nodes in an aggregation node. As long as the aggregation nodes retain consistency, the model always has a tight ELBO and can be applied for tractable posterior inference. According to the recent theoretical study [27], 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 training algorithm and thus a tight ELBO.

8 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 the latent representation of the input data through message passing between nodes in the model structure. The posterior inference, of the latent nodes given input observations, is facilitated by the careful embedding of flow functions in the general graph structure. Experiments on different datasets illustrate the effectiveness of the model. Future work includes applying our VFG model to fine grained data relational structure learning and reasoning.

References

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

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

[3] Christopher M Bishop, David Spiegelhalter, and John Winn. Vibes: A variational inference engine for bayesian networks. In *Advances in neural information processing systems*, pages 793–800, 2003.

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

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

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

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

[8] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *ArXiv*, abs/1605.08803, 2016.

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

[10] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

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

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

[13] David Kahle, Terrance Savitsky, Stephen Schnelle, and Volkan Cevher. Junction tree algorithm. *Stat*, 631, 2008.

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

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

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

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

[18] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[19] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[20] Radu Marinescu and Rina Dechter. And/or branch-and-bound for graphical models. In *IJCAI*, pages 224–229. Citeseer, 2005.

[21] Tahrima Rahman, Prasanna Kothalkar, and Vibhav Gogate. 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*, pages 630–645. Springer, 2014.

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

[23] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

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

[25] Scott Sanner and Ehsan Abbasnejad. Symbolic variable elimination for discrete and continuous graphical models. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[26] Peter Sorrenson, Carsten Rother, and Ullrich Köthe. Disentanglement by nonlinear ica with general incompressible-flow networks (gin). In *Ninth International Conference on Learning Representations*, 2020.

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

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

[29] John Winn and Christopher M Bishop. Variational message passing. *Journal of Machine Learning Research*, 6(Apr):661–694, 2005.

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

[31] Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. Dags with no tears: Continuous optimization for structure learning. In *NeurIPS*, 2018.

Supplement

The proposed variational flow graphical models assemble flow functions with tree or DAG structures via variational inference on aggregation nodes. In this supplemental file, we first present additional results, then we give more details on the methodology of VFG models. Section A provides additional experiments; section B gives added details on ELBO computation; section D presents other details on the aggregation node; ELBO derivation can be found in section E.

A Additional Results

All the experiments are conducted on NVIDIA-TITAN X (Pascal) GPUs. In the experiments, we use the same coupling block [8] 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 $\mathcal{B} \geqslant 2$. All latent variables, $\mathbf{h}^i, i \in \mathcal{V}$ are forced to be non-negative via Sigmoid or RELU functions. Non-negativeness can help the model to identify sparse structures of the latent space.

A.1 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. Here, 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 4 coupling blocks. We can see Table 2 that our model yields significantly better results than any other method in terms of prediction error.

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

Table 2: California Housing dataset: Imputation Mean Squared Error (MSE) results.

B ELBO Calculation

This section presents more details on ELBO calculation. The recognition model in a VFG is the neural network (encoder) used to approximate the posterior of latent variables. With invertible neural networks (flows), the recognition model and the generative model in a VFG share the same structure

and parameters. As shown in Figure 2, the recognition and generative models are realized with forward and backward message passing, respectively.

Maximize the ELBOs (3.6,3.8) requires evaluation of both the reconstruction and the $\mathbf{KL}$ terms. It involves node state samples from the posterior in both forward and backward messages.

B.1 Distributions of Latent Variables We first discuss additional details on the distributions of latent variables.

B.1.1 Generative Model In a tree VFG, the sample reconstruction in the generative model consists of layer-wise backward message passing, i.e., latent variable generation in each layer. For any $l, 0 \leq l \leq L-1$, latent variable backward state (reconstruction) $\widehat{\mathbf{h}}^l$ is propagated from layer $l+1$ via the flow function $\mathbf{f}_l$ between the two layers with $\widehat{\mathbf{h}}^l = \mathbf{f}_l^{-1}(\widehat{\mathbf{h}}^{l+1})$. The prior $p(\mathbf{h}^L)$ for the root latent variable $\mathbf{h}^L$ is Laplace(0,1). With a sample $\widehat{\mathbf{h}}^L$ from the posterior, i.e., $\widehat{\mathbf{h}}^L = \mathbf{h}^L \sim q(\cdot|\mathbf{h}^{L-1})$, the conditional distribution for latent variable in layer l is $p(\cdot|\widehat{\mathbf{h}}^{l+1}) :=$ Laplace$(\widehat{\mathbf{h}}^l, 1)$. Here the location parameter is generated from layer $l+1$, i.e., $\widehat{\mathbf{h}}^l = \mathbf{f}_l^{-1}(\widehat{\mathbf{h}}^{l+1})$. For a latent variable $\mathbf{h}^l$ sampling from the posterior, its log-likelihood regarding $p(\cdot|\widehat{\mathbf{h}}^{l+1})$ in (3.11) is given by

$$\log p(\mathbf{h}^l|\widehat{\mathbf{h}}^{l+1}) = -\|\mathbf{h}^l - \widehat{\mathbf{h}}^l\|_1 - d \cdot \log 2.$$

Here $d = dim(\mathbf{h}^l)$. Hence, minimizing $\mathbf{KL}$s is to minimize the ℓ_1 distance between latent variables and their reconstructions.

B.1.2 Recognition Model The forward message passing in the recognition model consists of layer-wise sample generation. In layer $l, 1 \leq l \leq L$, latent variable forward state $\mathbf{h}^l$ is propagated from layer $l-1$ via the flow function $\mathbf{f}_{l-1}$ between the two layers with $\mathbf{h}^l = \mathbf{f}_{l-1}(\mathbf{h}^{l-1})$.

We assume each entry of hidden variable $\mathbf{h}^l$ follows a Laplace distribution, i.e., $\mathbf{h}^l_j \sim$ Laplace(μ^l_j, s^l_j) for layer l's jth entry. Here μ^l_j is the location and s^l_j is the scale. Compared with other distributions, Laplace can introduce sparsity to the model and it works well in practice. At level $l \in [L]$, we set $q(\cdot|\mathbf{h}^{l-1}) :=$ Laplace$(\mu^l, \mathbf{s}^l)$ with

$$\text{(B.1)} \qquad \mu^l = \text{median}\big(H\big), \;\; \mathbf{s}^l = \frac{1}{B}\sum_{b=1}^{B} |\mathbf{h}^l(\mathbf{x}_b) - \mu^l| \; .$$

Here $H = \{\mathbf{h}^l(\mathbf{x}_b) | 1 \leqslant b \leqslant B\}$ is a batch of latent values generated from a batch of data samples with size B, i.e., $X_B = \{\mathbf{x}_b | 1 \leqslant b \leqslant B\}$. The median operation is performed element-wisely. For each $\mathbf{x}_b$, $\mathbf{h}^l(\mathbf{x}_b) = \mathbf{f}^{l-1}\big(\mathbf{h}^{l-1}(\mathbf{x}_b)\big)$.

In summary, we use Laplace distribution to model latent distributions. $q(\cdot|\mathbf{h}^{l-1})$ is a Laplace with location and scale equal to the median and scale of a batch of $\mathbf{h}^l$, respectively; $p(\cdot|\widehat{\mathbf{h}}^{l+1})$ is a Laplace parameterized with $(\widehat{\mathbf{h}}^l, 1.0)$ as the location and scale parameters. Hence with $\mathbf{h}^l$ we can compute the log-likelihoods on RHS of (3.11) and thus the $\mathbf{KL}^l$ value. We use Laplace distribution for latent variables because it may introduce sparsity in latent variables. We also tried Gaussian and Gamma distributions, and they also work well in most tasks. VFGs leverage the universal approximation power [27] of coupling flows, and tractable inference could be achieved with different latent distributions.

B.2 KL Term For any $l, 1 \leq l \leq L-1$, the calculation of the $\mathbf{KL}^l$ term (3.7) requires message passing and samples from both recognition and generative models, i.e.,

$$\text{(B.2)} \qquad \begin{aligned} \mathbf{KL}^l =& \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log q(\mathbf{h}^l|\mathbf{h}^{l-1}) - \log p(\mathbf{h}^l|\widehat{\mathbf{h}}^{l+1})\big] \\ \simeq& \log q(\mathbf{h}^l|\mathbf{h}^{l-1}) - \log p(\mathbf{h}^l|\widehat{\mathbf{h}}^{l+1}). \end{aligned}$$

Here $q(\cdot|\mathbf{h}^{l-1})$ is a Laplace with location and scale equal to the median and scale defined in equation B.1; $p(\cdot|\widehat{\mathbf{h}}^{l+1})$ is a Laplace parameterized with $(\widehat{\mathbf{h}}^l, 1.0)$ as discussed in B.1.1. Hence with $\mathbf{h}^l$ we can compute the log-likelihoods on RHS of (B.2) and thus the $\mathbf{KL}^l$ value. When $l = L$, $\mathbf{KL}^L = \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log q(\mathbf{h}^L|\mathbf{h}^{L-1}) - \log p(\mathbf{h}^L)\big] \simeq \log q(\mathbf{h}^L|\mathbf{h}^{L-1}) - \log p(\mathbf{h}^L)$.

Assume k leaf nodes on a tree or a DAG model, corresponding to k sections of input sample $\mathbf{x} = [\mathbf{x}^{(1)}, ..., \mathbf{x}^{(k)}]$. Then the hidden variables in both (3.6) and (3.8) are computed with forward and backward message passing.

In practice, we set $M = 1$ for efficiency. With a batch of training samples, the structure of flow functions make the forward and backward message passing very efficient, and thus the estimation of the ELBO.

B.3 Reconstruction Term The reconstruction term in ELBO (3.6) can be computed with the backward message from the generative model $p(\mathbf{x}|\widehat{\mathbf{h}}^1)$, i.e.,

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

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

C Additional Details on Algorithm

One import motivation of VFG is that we aim to develop a model that is tractable of inference on datasets. A very

simple application case with inference is to impute the missing values of data samples. It motivates us to use the imputation loss as the objective along with the KL terms from the ELBO as regularization terms.

Inference on a VFG model requires the aggregation node's state to be imputed from observed children's state, as shown in (5.15). Then, unobserved children's state can be inferred from their parent. The inference ability of VFG via imputation 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

$$\begin{aligned}(\text{C.3}) \quad & \mathcal{L}(\mathbf{x}, O_{\mathbf{x}}; \theta) \\ &= \sum_{t:1\leqslant t\leqslant k, t\notin O} \mathbb{E}_{q(\mathbf{h}^1|\mathbf{x}^{O_{\mathbf{x}}})}\Big[\log p(\mathbf{x}^{(t)}|\widehat{\mathbf{h}}^1)\Big] \\ &\quad - \sum_{l=1}^{L-1} \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\Big[\log q(\mathbf{h}^l|\mathbf{h}^{l-1}) - \log p(\mathbf{h}^l|\widehat{\mathbf{h}}^{l+1})\Big] \\ &\quad - \mathbf{KL}\big(q(\mathbf{h}^L|\mathbf{h}^{L-1})|p(\mathbf{h}^L)\big).\end{aligned}$$

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. Considering a minibatch of training samples, the objectives function in (3.6) and (C.3) can thus be optimized sequentially. The training with *random masking* is described in Algorithm 2.

Algorithm 2 Inference model parameters with random masking

1: **Input:** Data distribution $\mathcal{D}$, $\mathcal{G} = \{\mathcal{V}, \mathbf{f}\}$
2: **for** $s = 0, 1, ...$ **do**
3: Sample minibatch b samples $\{\mathbf{x}_1, ..., \mathbf{x}_b\}$ from $\mathcal{D}$;
4: Optimize (3.6) with Line 4 to Line 15 in Algorithm 1;
5: Sample a subset of the k data sections as data observation set $O_{\mathbf{x}}$; $O \leftarrow O_{\mathbf{x}}$;
6: **for** $i \in \mathcal{V}$ **do**
7: // forward message passing
8: $\mathbf{h}^{(i)} = \frac{1}{|ch(i)\cap O|}\sum_{j\in ch(i)\cap O} \mathbf{f}_{(j,i)}(\mathbf{h}^{(j)})$;
9: $O \leftarrow O \cup \{i\}$ if $ch(i) \cap O \neq \emptyset$;
10: **end for**
11: $\widehat{\mathbf{h}}^{(i)} = \mathbf{h}^{(i)}$ if $i \in \mathcal{R}_{\mathcal{G}}$ or $i \in$ layer L;
12: **for** $i \in \mathcal{V}$ **do**
13: // backward message passing
14: $\widehat{\mathbf{h}}^{(i)} = \frac{1}{|pa(i)|}\sum_{j\in pa(i)} \mathbf{f}^{-1}_{(i,j)}(\widehat{\mathbf{h}}^{(j)})$;
15: **end for**
16: $\mathbf{h} = \{\mathbf{h}^{(t)}|t \in \mathcal{V}\cap O\}$, $\widehat{\mathbf{h}} = \{\widehat{\mathbf{h}}^{(t)}|t \in \mathcal{V}\}$;
17: Approximate the $\mathbf{KL}$ terms in ELBO for each layer with b samples;
18: Updating VFG with gradient of (C.3): $\theta_{\mathbf{f}}^{(s+1)} = \theta_{\mathbf{f}}^{(s)} + \nabla_{\theta_{\mathbf{f}}} \frac{1}{b}\sum_{i=1}^{b} \mathcal{L}(\mathbf{x}_b, O_{\mathbf{x}}; \theta_{\mathbf{f}}^{(s)})$,
19: **end for**

In practice, we use Algorithm 2 along with Algorithm 1 to train a VFG model. Training with Algorithm 1 is to improve the data fitting and the *consistency* of aggregation nodes. Geometrically speaking, training with Algorithm 2 is to enhance the learned relational knowledge among leaf nodes. Though Algorithm 2 improves the model's inference capability in a heuristic way, it works in practice.

D Aggregation Node

We present additional details on aggregation nodes here. 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, at an aggregation node i that has multiple ($|ch(i)|$) children, its latent variable in forward message passing is the mean of all children's output, i.e.,

$$(\text{D.4}) \qquad \mathbf{h}^{(i)} = \frac{1}{|ch(i)|}\sum_{j\in ch(i)} \mathbf{f}_{(j,i)}(\mathbf{h}^{(j)}).$$

On the other hand, if node i in a DAG has multiple parents, the reconstruction of its latent variable is the mean of all parents' output , i.e.,

$$(\text{D.5}) \qquad \widehat{\mathbf{h}}^{(i)} = \frac{1}{|pa(i)|}\sum_{j\in pa(i)} \mathbf{f}^{-1}_{(i,j)}(\widehat{\mathbf{h}}^{(j)}).$$

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

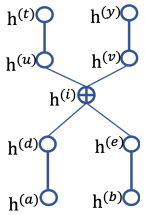

Figure 9: Aggregation node on a DAG.

Besides averaging, the aggregation nodes also ensure the latent variable on the two ends of an identity function are *consistent*. We use node i in the DAG presented in Figure 9 as an example. 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 (D.4) and (D.5), 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.,

$$(\text{D.6}) \qquad \mathbf{h}^{(d)} = \mathbf{h}^{(e)} = \widehat{\mathbf{h}}^{(i)}.$$

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

$$(\text{D.7}) \qquad \widehat{\mathbf{h}}^{(u)} = \widehat{\mathbf{h}}^{(v)} = \mathbf{h}^{(i)}.$$

We utilize the $\mathbf{KL}$ term in the ELBOs to ensure (D.6) and (D.7) can be satisfied during parameter updating. The $\mathbf{KL}$ term regarding node i is

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

Here

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

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 equation D.7.

Similarly, $\mathbf{KL}$s of i's children intend to realize consistency given in equation D.6. We use node d as an example. The $\mathbf{KL}$ term regarding d is

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

With

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

minimizing $\mathbf{KL}^{(d)}$ is to minimize $\|\mathbf{h}^{(d)} - \widehat{\mathbf{h}}^{(i)}\|_1$ that targets at equation D.6. In summary, by maximizing the ELBO of a VFG, the aggregation consistency can be attained along with fitting the model to the data.

E Derivation of the ELBOs for Trees and DAGs

E.1 ELBO of Tree Models 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 10. Then the marginal likelihood term of the data reads

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

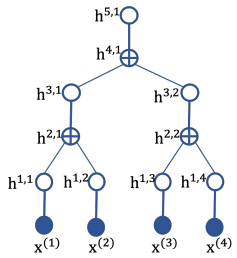

Figure 10: A tree VFG with $L = 5$ and three aggregation nodes.

The hierarchical generative model is given by factorization

$$p(\mathbf{h}) = p(\mathbf{h}^L)\mathbf{\Pi}_{l=1}^{L-1}p(\mathbf{h}^l|\mathbf{h}^{l+1}). \tag{E.8}$$

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

$$q_\theta(\mathbf{h}|\mathbf{x}) = q(\mathbf{h}^1|\mathbf{x})q(\mathbf{h}^2|\mathbf{h}^1)\cdots q(\mathbf{h}^L|\mathbf{h}^{L-1}). \tag{E.9}$$

Draw samples from the generative model (E.8) involves sequential conditional sampling from the top of the tree to the bottom, and computation of the recognition model (E.9) 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

$$\begin{aligned}&q(\mathbf{h}^{l:L}|\mathbf{h}^{l-1}) = q(\mathbf{h}^l|\mathbf{h}^{l-1})q(\mathbf{h}^{l+1:L}|\mathbf{h}^l\mathbf{h}^{l-1}) \\ &= q(\mathbf{h}^l|\mathbf{h}^{l-1})q(\mathbf{h}^{l+1:L}|\mathbf{h}^l)\end{aligned} \tag{E.10}$$

$$p(\mathbf{h}^{l:L}) = p(\mathbf{h}^l|\mathbf{h}^{l+1:L})p(\mathbf{h}^{l+1:L}) = p(\mathbf{h}^l|\mathbf{h}^{l+1})p(\mathbf{h}^{l+1:L}) \tag{E.11}$$

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

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

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

$$\begin{aligned}
&\log p(\mathbf{x}) \geqslant \mathcal{L}(x;\theta) \\
=&\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log p(\mathbf{x}|\mathbf{h}^{1:L}) - \log q(\mathbf{h}^{1:L}|\mathbf{x}) + \log p(\mathbf{h}^{1:L})\big] \\
=&\underbrace{\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log p(\mathbf{x}|\mathbf{h}^{1:L})\big]}_{\text{(a) Reconstruction of the data}} \\
&- \underbrace{\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log q(\mathbf{h}^{1:L}|\mathbf{x}) - \log p(\mathbf{h}^{1:L})\big]}_{\mathbf{KL}^{1:L}}
\end{aligned} \tag{E.12}$$

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 (E.12) can be further expanded as

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

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

$$\begin{aligned}
\mathbf{KL}^{1:L} =&\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log q(\mathbf{h}^1|\mathbf{x}) - \log p(\mathbf{h}^1|\mathbf{h}^2) \\
&+ \log q(\mathbf{h}^{2:L}|\mathbf{h}^1) - \log p(\mathbf{h}^{2:L})\big] \\
=&\underbrace{\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log q(\mathbf{h}^1|\mathbf{x}) - \log p(\mathbf{h}^1|\mathbf{h}^2)\big]}_{\mathbf{KL}^1} \\
&+ \underbrace{\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log q(\mathbf{h}^{2:L}|\mathbf{h}^1) - \log p(\mathbf{h}^{2:L})\big]}_{\mathbf{KL}^{2:L}}.
\end{aligned} \tag{E.13}$$

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

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

With (E.10) and (E.11), it is easy to show that

$$\begin{aligned}
\mathbf{KL}^{l:L} =&\underbrace{\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log q(\mathbf{h}^l|\mathbf{h}^{l-1}) - \log p(\mathbf{h}^l|\mathbf{h}^{l+1})\big]}_{\mathbf{KL}^l} \\
&+ \underbrace{\mathbb{E}_{q(\mathbf{h}^{l:L}|\mathbf{x})}\big[\log q(\mathbf{h}^{l+1:L}|\mathbf{h}^l) - \log p(\mathbf{h}^{l+1:L})\big]}_{\mathbf{KL}^{l+1:L}}.
\end{aligned} \tag{E.14}$$

The ELBO (E.12) can be written as

$$\mathcal{L}(\mathbf{x};\theta) = \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log p(\mathbf{x}|\mathbf{h}^{1:L})\big] - \sum_{l=1}^{L-1}\mathbf{KL}^l - \mathbf{KL}^L. \tag{E.15}$$

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

$$\mathbf{KL}^l = \mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log q(\mathbf{h}^l|\mathbf{h}^{l-1}) - \log p(\mathbf{h}^l|\mathbf{h}^{l+1})\big]. \tag{E.16}$$

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 (E.15) with the default assumption that all latent variables are generated regarding data sample $\mathbf{x}$. Therefore the ELBO (E.15) can be simplified as

$$\mathcal{L}(\mathbf{x};\theta) = \mathbb{E}_{q(\mathbf{h}^1|\mathbf{x})}\big[\log p(\mathbf{x}|\widehat{\mathbf{h}}^1)\big] - \sum_{l=1}^{L}\mathbf{KL}^l. \tag{E.17}$$

The $\mathbf{KL}$ term (E.16) becomes

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

When $l = L$,

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

E.2 Improve ELBO Estimation with Flows In this paper we follow the approach in [22, 15, 2] using normalizing flows to further improve posterior estimation on a tree VFG model. At each layer, minimizing $\mathbf{KL}$ term is to is to optimize the parameters of the network so that the posterior is closer to the prior. As shown in Figure 2, for layer l, we can take the encoding-decoding procedures (discussed in section B) as transformation of the posterior distribution from layer l to $l+1$, and then transform it back. By counting in the transformation difference [22, 15, 2], the $\mathbf{KL}$ at layer l becomes

$$\begin{aligned}
\mathbf{KL}^l =&\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\left[\log q(\mathbf{h}^l|\mathbf{h}^{l-1}) + \log\left|\det\frac{\partial\mathbf{h}^l}{\partial\mathbf{h}^{l+1}}\right|\right. \\
&\left.+ \log\left|\det\frac{\partial\widehat{\mathbf{h}}^{l+1}}{\partial\widehat{\mathbf{h}}^l}\right| - \log p(\mathbf{h}^l|\widehat{\mathbf{h}}^{l+1})\right] \\
\simeq&\frac{1}{M}\sum_{m=1}^{M}\left[\log q(\mathbf{h}_m^l|\mathbf{h}^{l-1}) + \log\left|\det\frac{\partial\mathbf{h}_m^l}{\partial\mathbf{h}_m^{l+1}}\right|\right. \\
&\left.+ \log\left|\det\frac{\partial\widehat{\mathbf{h}}_m^{l+1}}{\partial\widehat{\mathbf{h}}_m^l}\right| - \log p(\mathbf{h}_m^l|\widehat{\mathbf{h}}_m^{l+1})\right].
\end{aligned}$$

E.3 ELBO of DAG Models 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.

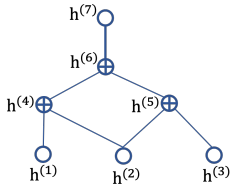

Figure 11: A DAG with inverse topology order { {1,2,3}, {4,5}, {6}, {7} }, and they correspond to layers 0 to 3.

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 (E.12) we develop the ELBO as

(E.18)

$$\begin{aligned}\log p(\mathbf{x}) \geqslant \mathcal{L}(x;\theta) &= \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{h})}{q(\mathbf{h}|\mathbf{x})}\right] \\ = \underbrace{\mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\Big[\log p(\mathbf{x}|\mathbf{h})\Big]}_{\text{Reconstruction of the data}} &- \underbrace{\mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\Big[\log q(\mathbf{h}|\mathbf{x}) - \log p(\mathbf{h})\Big]}_{\mathbf{KL}}.\end{aligned}$$

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

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

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

$$\begin{aligned}q(\mathbf{h}^{l:L}|\mathbf{h}^{1:l-1}) &= q(\mathbf{h}^{l}|\mathbf{h}^{1:l-1})q(\mathbf{h}^{l+1:L}|\mathbf{h}^{l}) \qquad \text{(E.19)}\\ &= q(\mathbf{h}^{l}|\mathbf{h}^{1:l-1})q(\mathbf{h}^{l+1:L}|\mathbf{h}^{1:l})\end{aligned}$$

$$p(\mathbf{h}^{l:L}) = p(\mathbf{h}^{l}|\mathbf{h}^{l+1:L})p(\mathbf{h}^{l+1:L}) \qquad \text{(E.20)}$$

Following (E.14) and with (E.19-E.20), we have

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

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,

$$\mathbf{KL}^{l:L} = \underbrace{\mathbb{E}_{q(\mathbf{h}^{1:L}|\mathbf{x})}\big[\log q(\mathbf{h}^{l}|\mathbf{h}^{ch(l)}) - \log p(\mathbf{h}^{l}|\mathbf{h}^{pa(l)})\big]}_{\mathbf{KL}^{l}} + \mathbf{KL}^{l+1:L} \qquad \text{(E.21)}$$

For nodes in layer l,

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

Recurrently applying (E.21) to (E.18) yields

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

For node i,

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