Supplementary Materials: Bidirectional Inference Networks with Application to Health Profiling

1 Proof of Theorem 1 and Theorem 2 in the main paper

In this section, we explain the detail of Theorem 1 and Theorem 2 in the main paper. We start with giving the definition and important properties of unimodal distribution and elliptical distribution. Then we introduce distribution families of our interest, elliptically unimodal distribution. Finally we prove the arguments in Theorem 1 and Theorem 2 in main paper.

1.1 Unimodal Distribution

We first briefly explain the concept of convex unimodal for probability distributions. We refer readers to the book [7] for more detailed discussion on unimodality of distributions, especially in high dimensional space. Some of our definitions directly borrow from [7].

Definition 1. *A set S is called symmetric about center c if, for all x, $c + x \in S \Rightarrow c - x \in S$. A distribution with density p is called symmetric about center c if, for all x, $p(c + x) = p(c - x)$.*

Definition 2. *A distribution on $\mathcal{R}^n$ is said to be **convex unimodal** if it has a density p such that, for every $\eta > 0$, the set $\{x : p(x) > \eta\}$ is convex. Further if for every $\eta > 0$, the set $\{x : p(x) > \eta\}$ is convex and symmetric, then this distribution is called **symmetric convex unimodal**.*

Lemma 1. *Marginal distributions of symmetric convex unimodal distribution is symmetric convex unimodal.*

Proof. To prove our lemma, we refer the theorem in book [7] which states that marginal distributions of central convex unimodal distribution is central convex unimodal. Basically, **central convex unimodal** distribution is symmetric convex unimodal distribution whose symmetric center is origin. Consider a random vector $Y = (Y_1, Y_2)$ has a symmetric convex unimodal distribution with the symmetric center $\mu = (\mu_1, \mu_2)$. Since $Y - \mu$ has a central convex unimodal distribution, $Y_1 - \mu_1$ as the marginal distribution is also central convex unimodal distribution. Thus the distribution of Y_1 is symmetric convex unimodal with the symmetric center μ_1. □

Lemma 2. *Condition distributions of convex unimodal distribution is convex unimodal.*

Proof. Consider a convex unimodal distribution $p(V)$ and its conditional distribution $p(V_S|V_{\text{-}S} = \widetilde{V}_{\text{-}S})$ given a subset variables $V_{\text{-}S} = \widetilde{V}_{\text{-}S}$. For any $\eta > 0$, consider following sets $C(\eta) = \{V_S : p(V_S|V_{\text{-}S} = \widetilde{V}_{\text{-}S}) > \eta\}$, $A = \{V : p(V) > \eta p(V_{\text{-}S} = \widetilde{V}_{\text{-}S})\}$ and $B = \{V : V_{\text{-}S} = \widetilde{V}_{\text{-}S}\}$. Convex unimodality of $p(V)$ implies that A is convex. Since B is also convex, $A \cap B = \{V = (V_S, V_{\text{-}S}) : V_{\text{-}S} = \widetilde{V}_{\text{-}S}$ and $p(V_S|V_{\text{-}S} = \widetilde{V}_{\text{-}S}) = \frac{p(V_S, V_{\text{-}S} = \widetilde{V}_{\text{-}S})}{p(V_{\text{-}S} = \widetilde{V}_{\text{-}S})} > \eta\}$ is convex. Thus, $C(\eta)$, as the low dimensional projection of $A \cap B$, is convex. Since $C(\eta)$ is convex for any $\eta > 0$, by definition, distribution $p(V_S|V_{\text{-}S} = \widetilde{V}_{\text{-}S})$ is convex unimodal. □

1.2 Elliptical Distribution

Further, we introduce elliptical distribution. [8] gives a good introduction of elliptical distribution. We refer readers to this paper for more properties of elliptical distribution and the proofs of lemmas we use here.

Definition 3. *A random vector Y in space $\mathbb{R}^d$ has an **elliptical distribution** if its characteristic function has the form,*

$$t \mapsto \phi(t) = \exp(it'\mu)\varphi(t'\Sigma t),\ t \in \mathbb{R}^d. \tag{1}$$

where $\mu \in \mathbb{R}^d$, $\Sigma \in \mathbb{R}^{d\times d}$ is symmetric and semi-definite and $\varphi : \mathbb{R}^+ \mapsto \mathbb{R}$ is function called characteristic generator. And we denote this distribution as $\mathcal{E}_d(\mu, \Sigma, \phi)$.

Lemma 3 in the following describes the symmetricity of elliptical distributions. This lemma implies that the parameter μ of the distribution $\mathcal{E}_d(\mu, \Sigma, \phi)$ is actually the mean of the random vector having this distribution. Also distribution $\mathcal{E}_d(\mu, \Sigma, \phi)$ is symmetric about μ.

Lemma 3. *A random vector Y have an elliptical distribution $\mathcal{E}_d(\mu, \Sigma, \phi)$ iff $Y \stackrel{d}{=} \mu + RAU$ [1] where R is non-negative random variable, A is matrix that satisfies $A'A = \Sigma$, U is a random vector uniformly distributed over $k = rank(\Sigma)$ dimensional unit sphere i.e. $\{u \in \mathbb{R}^k : |u|_2 = 1\}$. And R and U are independent.*

In the following lemmas[2], we introduce the properties of the marginal and conditional distributions of elliptical distribution. Before that, we introduce more notions in our setting. Let $Y \sim \mathcal{E}_d(\mu, \Sigma, \phi)$ where $\mu = (\mu_1, \mu_2) \in \mathbb{R}^d$, the matrix $\Sigma \in \mathbb{R}^{d\times d}$ is positive semidefinate with $rank(\Sigma) = r$. Let

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \tag{2}$$

with sub-matrices $\Sigma_{11} \in \mathbb{R}^{k\times k}$, $\Sigma_{12} \in \mathbb{R}^{k\times(d-k)}$, $\Sigma_{21} \in \mathbb{R}^{(d-k)\times k}$, $\Sigma_{22} \in \mathbb{R}^{(d-k)\times(d-k)}$. Further, let $Y = (Y_1, Y_2)$ where Y_1 is k dimensional sub-vector of Y.

Lemma 4. *Distributions of Y_1 and Y_2 as marginal distributions of $\mathcal{E}_d(\mu, \Sigma, \phi)$ are $\mathcal{E}_d(\mu_1, \Sigma_{11}, \phi_1)$ and $\mathcal{E}_d(\mu_2, \Sigma_{22}, \phi_1)$ respectively which are also elliptical.*

Lemma 5. *Conditional distribution for $Y_2|Y_1 = y_1$ is elliptical distribution $\mathcal{E}_d(\mu_{2|1}, \Sigma_{2|1}, \phi_{2|1})$, where*

$$\mu_{2|1} = \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(y_1 - \mu_1) \tag{3}$$

$$\Sigma_{2|1} = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12} \tag{4}$$

1.3 Elliptically Unimodal Distribution

Definition 4. *A distribution is said to be **elliptically unimodal** if it is elliptical, convex unimodal and has a density p with an unique maximum (mode).*

Lemma 6. *Elliptically unimodal distribution $\mathcal{EU}_d(\mu, \Sigma, \phi)$ is **symmetric convex unimodal**. Furthermore, the mean vector μ is the symmetric center and also the unique mode.*

Proof. According to Lemma 3, the distribution $\mathcal{EU}_d(\mu, \Sigma, \phi)$ being elliptical implies that it is symmetric about μ, i.e., its density p satisfies $p(\mu + x) = p(\mu - x), \forall x$. For every $\eta > 0$, set $C(\eta) = \{x : p(x) > \eta\}$ is symmetric about μ, since $\mu + x \in C(\eta) \Rightarrow p(\mu + x) > \eta \Rightarrow p(\mu - x) > \eta \Rightarrow \mu - x \in C(\eta)$. Since we already know set $C(\eta)$ is convex due to the fact that the distribution $\mathcal{EU}_d(\mu, \Sigma, \phi)$ is convex unimodal. Thus it is symmetric convex unimodal. We further show that mean vector μ is the mode. Assume the mode is $\mu + \delta$ ($\delta \neq 0$), then there is another vector $\mu - \delta$ such that $p(\mu - \delta) = p(\mu + \delta)$, which violates the uniqueness of the mode. Thus any vector other than μ can not be the mode of the density function. □

Lemma 7. *Consider $p(V)$ is a elliptically unimodal distribution with the mean vector μ. Its marginal distribution $p(V_S)$ is elliptically unimodal distribution with the mean vector μ_S.*

[1] The distribution equal sign, $\stackrel{d}{=}$, means the random vectors on both sides have the same distribution.

[2] The derivations of Lemma 4 and Lemma 5 can be found in section 1.2.4 and 1.2.5 of [8].

Proof. By Lemma 4, we know marginal distribution $p(V_S)$ is elliptical and have the mean vector μ_S. By Lemma 1, we know the marginal distribution $p(V_S)$ is symmetric convex unimodal. Combining together, we have that the marginal distribution $p(V_S)$ is elliptically unimodal distribution with mean vector μ_S. □

Lemma 8. *Consider $p(V)$, a elliptically unimodal distribution with mean vector μ. Its conditional distribution $p(V_S|V_{\text{–}S} = \widetilde{V}_{\text{–}S})$ is elliptically unimodal distribution. Furthermore, if $\widetilde{V}_{\text{–}S} = \mu_{\text{–}S}$, the mean vector of conditional distribution $p(V_S|V_{\text{–}S} = \mu_{\text{–}S})$ is μ_S.*

Proof. By Lemma 5, we know the conditional distribution $p(V_S|V_{\text{–}S} = \widetilde{V}_{\text{–}S})$ is elliptical. By Lemma 2, we know the conditional distribution $p(V_S|V_{\text{–}S} = \widetilde{V}_{\text{–}S})$ is convex unimodal. Thus the conditional distribution $p(V_S|V_{\text{–}S} = \widetilde{V}_{\text{–}S})$ is also elliptically unimodal. Furthermore, if $\widetilde{V}_{\text{–}S} = \mu_{\text{–}S}$, according to the given Eqn. 3 in Lemma 5, we can derive that the mean vector for $p(V_S|V_{\text{–}S} = \widetilde{V}_{\text{–}S})$ is μ_S. □

1.4 Proof of the Theorem

In the following, we restate the arguments of Theorem 1 and Theorem 2 in main paper give the proofs separately.

Theorem 1. *In the forward inference case, we predict a set of variables V based on the distribution $p(V|\mathbf{X})$ conditioned on given variables $\mathbf{X}$. Our prediction $\widehat{V}$ satisfies following greedy property.*

$$\widehat{v}_n = \underset{v_n}{\operatorname{argmax}}\, p(v_n|\mathbf{X}, \widehat{V}_n), \quad n = 1, \ldots, N \tag{5}$$

The arguemnt of this theorem is: if the distribution $p(V|\mathbf{X})$ is elliptically unimodal, our prediction is actually global optimum, i.e., $\widehat{V} = \operatorname{argmax}_V p(V|\mathbf{X})$

Proof. Since $p(V|\mathbf{X})$ is elliptically unimodal, say its mode is V^*. By Lemma 7, we know for every n, the marginal distribution $p(V_n|\mathbf{X})$ is also elliptically unimodal with mode V_n^*. Further by Lemma 8, $p(V_n|\mathbf{X})$'s conditional distribution $p(v_n|\mathbf{X}, V_{n-1} = V_{n-1}^*)$ is elliptically unimodal with mode v_n^*. Thus at first step in forward inference, our prediction for variable v_1, $\widehat{v}_1$, satisfies $\widehat{v}_1 = \operatorname{argmax}_{v_1} p(v_n|\mathbf{X}) = v_1^*$. For the following steps, the prediction $\widehat{v}_n$ satisfies $\widehat{v}_n = \operatorname{argmax}_{v_n} p(v_n|\mathbf{X}, V_{n-1} = \widehat{V}_{n-1} = V_{n-1}^*) = v_n^*$. Thus in total, our prediction $\widehat{V} = V^*$. □

Theorem 2. *In hybrid inference case, we predict a subset of variables V_S based on the distribution $p(V_S|\mathbf{X}, V_{\text{–}S} = \widetilde{V}_{\text{–}S})$ with given extra variable $\mathbf{X}$ and $V_{\text{–}S}$. We divide the target variables V_S into two parts: $V_B = V_S \cap \{v_i : i < L\}$ and $V_B = V_S \cap \{v_i : i > L\}$, where L is largest index of variable in $V_{\text{–}S}$, i.e., $L = \max_{i \in -S} i$. We first do backward inference and assume the result $\widehat{V}_B$ satisfies $\widehat{V}_B = \operatorname{argmax}_{V_B} p(V_B|\mathbf{X}, V_{\text{–}S} = \widetilde{V}_{\text{–}S})$. Then we use forward prediction to get the prediction $\widehat{V}_F$. The arguemnt of this theorem is: if the distribution $p(V_S|\mathbf{X}, V_{\text{–}S} = \widetilde{V}_{\text{–}S})$ is elliptically unimodal, our prediction is actually global optimum, i.e., $\widehat{V}_S = (\widehat{V}_B, \widehat{V}_F) = \operatorname{argmax}_{V_S} p(V_S|\mathbf{X}, V_{\text{–}S} = \widetilde{V}_{\text{–}S})$.*

Proof. We denote the mode of elliptically unimodal distribution $p(V_S|\mathbf{X}, V_{\text{–}S} = \widetilde{V}_{\text{–}S})$ as V_S^*. By Lemma 7, the marginal distribution $p(V_B|\mathbf{X}, V_{\text{–}S} = \widetilde{V}_{\text{–}S})$ is an elliptically unimodal distribution with mode V_B^*. By Lemma 8, the conditional distribution $p(V_F|\mathbf{X}, V_{\text{–}S} = \widetilde{V}_{\text{–}S}, V_B = V_B^*)$ is an elliptically unimodal distribution with mode V_F^*. By our assumption, the prediction $\widehat{V}_B = \operatorname{argmax}_{V_B} p(V_B|\mathbf{X}, V_{\text{–}S} = \widetilde{V}_{\text{–}S}) = V_B^*$. Further, since $p(V_F|\mathbf{X}, V_{\text{–}S} = \widetilde{V}_{\text{–}S}, V_B = V_B^*)$ is elliptically unimodal, by Theorem 1 we can conclude that the forward prediction actually gives global optimum i.e. $\widehat{V}_F = \operatorname{argmax}_{V_F} p(V_F|\mathbf{X}, V_{\text{–}S} = \widetilde{V}_{\text{–}S}, V_B = V_B^*) = V_F^*$. Thus in total our prediction result $\widehat{V}_S = (\widehat{V}_B, \widehat{V}_F)$ equals to the global optimum $V_S^* = (V_B^*, V_F^*)$. □

2 Marginal Likelihood of $V_{\text{–}S}$

In the paper, we propose to approximate the marginal negative log-likelihood $\mathcal{L}(V_{\text{–}S_j}|\mathbf{X}; \boldsymbol{\theta})$ efficiently and effectively by leveraging the properties of NPN. The process is as follows:

$$\mathcal{L}(V_{\text{–}S_j}|\mathbf{X}; \boldsymbol{\theta}) \approx \sum_{v_n \in \text{–}S_j} -\log p(v_n|\mathbf{X}),$$

where $-\log p(v_n|\mathbf{X})$ can be computed recursively as follows:

- $-\log p(v_1|\mathbf{X}) = \frac{\|\mu_{\theta_1}(\mathbf{X}) - v_1\|_2^2}{2s_{\theta_1}(\mathbf{X})} + \frac{1}{2}\log s_{\theta_1}(\mathbf{X})$.
- For $k > 1$,

$$-\log p(v_k|\mathbf{X}) = \frac{\|\mu_{\theta_k}(\mathbf{X}, \widehat{U}_{k-1}) - v_i\|_2^2}{2s_{\theta_k}(\mathbf{X}, \widehat{U}_{k-1})} + \frac{1}{2}\log s_{\theta_k}(\mathbf{X}, \widehat{U}_{k-1}),$$

where $\widehat{U}_k = \{\widehat{u}_1, \ldots, \widehat{u}_k\}$ with $\widehat{u}_k$ as the estimated mean and variance (output by NPN) of v_k given $\mathbf{X}$ (note that NPN can take mean-variance pairs as input):

$$\widehat{u}_k = (\mu_{\theta_k}(\mathbf{X}, \widehat{U}_{k-1}), s_{\theta_k}(\mathbf{X}, \widehat{U}_{k-1}))$$

In this section, we justify this approximation by showing that if each NPN subnetwork has a single layer, the process above computes the mean and variance V_S exactly (note that since NPN assumes diagonal covariance matrices for the output, the our process can only compute the diagonal entries of the covariance matrix for V_S exactly and ignores the off-diagonal entries).

2.1 More Background on NPN

Different from vanilla neural networks which usually take deterministic input, NPN is a probabilistic neural network which takes distributions as input. The input distributions will go through layers of linear and nonlinear transformation to produce output distributions. In NPN, all hidden neurons and weights are also distributions expressed in closed form. Specifically, in a vanilla neural network $f_w(x)$ will take x is input and compute the output based on parameters w. A corresponding Gaussian NPN would assume w is drawn from a Gaussian distribution $p_\theta(w)$ parameterized by θ and that x is drawn from $\mathcal{N}(x_m, x_s)$ (x_s is set to 0 when the input is deterministic). It will then compute the mean and variance of the output Gaussian distribution $\mu_\theta(x_m, x_s)$ and $s_\theta(x_m, x_s)$ in closed form, where $\mu_\theta(\cdot,\cdot)$ and $s_\theta(\cdot,\cdot)$ share parameters θ in a sophisticated way so that:

$$E[f_w(x)] \approx \mu_\theta(x_m, x_s)$$
$$E[f_w^2(x)] \approx s_\theta(x_m, x_s) + \mu_\theta^2(x_m, x_s),$$

where the expectations are taken over $x \sim \mathcal{N}(x_m, x_s)$ and $w \sim p_\theta(w)$.

In a linear NPN layer, if the input $\mathbf{a}$ is drawn from a distribution $p(\mathbf{a}|\mathbf{a}_m, \mathbf{a}_s)$ (*not* necessarily Gaussian) with the mean $\mathbf{a}_m$ and the variance $\mathbf{a}_s$ and the weights $\mathbf{W}$ (we ignore biases in this section for simplicity) are drawn from a distribution $p(\mathbf{W}|\mathbf{W}_m, \mathbf{W}_s)$ with the mean $\mathbf{W}_m$ and the variance $\mathbf{W}_s$ (NPN assumes diagonal covariance matrix for hidden neurons and parameters), the mean and variance of the output $\mathbf{o}$, denoted as $\mathbf{o}_m$ and $\mathbf{o}_s$, can be computed as:

$$\mathbf{o}_m = \mu_\theta(\mathbf{a}_m, \mathbf{a}_s) = \mathbf{a}_m\mathbf{W}_m, \tag{6}$$
$$\mathbf{o}_s = s_\theta(\mathbf{a}_m, \mathbf{a}_s) = \mathbf{a}_s\mathbf{W}_s + \mathbf{a}_s(\mathbf{W}_m \circ \mathbf{W}_m) + (\mathbf{a}_m \circ \mathbf{a}_m)\mathbf{W}_s, \tag{7}$$

where $\mathbf{o}_m$ and $\mathbf{o}_s$ are the mean and variance (diagonal entries of the covariance matrix) of the following distribution:

$$p(\mathbf{o}|\mathbf{a}_m, \mathbf{a}_s, \mathbf{W}_m, \mathbf{W}_s) = \int p(\mathbf{a}|\mathbf{a}_m, \mathbf{a}_s)p(\mathbf{W}|\mathbf{W}_m, \mathbf{W}_s)p(\mathbf{o}|\mathbf{a}, \mathbf{W})d\mathbf{a}d\mathbf{W}, \tag{8}$$

where $p(\mathbf{o}|\mathbf{a}, \mathbf{W})$ is a Dirac delta distribution centered at $\mathbf{aW}$. These properties turn out to be the key to efficient computation of marginal negative log-likelihood $\mathcal{L}(V_{\_S_j}|\mathbf{X}; \boldsymbol{\theta})$.

2.2 Proof on the Process of Computing $\mathcal{L}(V_{\_S_j}|\mathbf{X}; \boldsymbol{\theta})$

Notation: In the following, we denote a vector of scalar variable $(v_1, v_2, \ldots, v_k)$ as $\mathbf{v}_k$ and assume the high-dimensional context information is a vector $\mathbf{x} \in \mathbb{R}^C$. We will prove that if each NPN subnetwork has one layer and output the correct mean and variance, chaining the N networks using the process mentioned above will produce the correct mean and variance of the joint distribution of $\mathbf{v}_N$ (given $\mathbf{x}$). Since $V_{\_S} \subseteq V$, the process can also give the correct mean and variance for the vector $(v_n)_n$ where $v_n \in V_{\_S}$. We assume that the n-th NPN subnetwork use the mean and variance of $(\mathbf{x}, \mathbf{v}_{n-1})$ as input and output the mean and variance of v_n. Specifically, $p(\mathbf{x}|\boldsymbol{\phi}_x)$ is the distribution over $\mathbf{x}$, and

$p(\mathbf{w}_n|\boldsymbol{\theta}_n)$ is the distribution over the weights $\mathbf{w}_n$ of the n-th NPN subnetwork. Here $\boldsymbol{\phi}_x$ and $\boldsymbol{\theta}_n$ are the parameters for corresponding distributions. We further define the shorthand $\mathbf{W}_k = \{\mathbf{w}_i\}_{i=1}^k$ and $\boldsymbol{\Theta}_k = \{\boldsymbol{\theta}_i\}_{i=1}^k$ for convenience. Vectors such as $\mathbf{v}_0$ are $\boldsymbol{\psi}_0$ empty vectors, which can be ignored during derivation. $(\cdot,\cdot)$ is used to denote concatenation of vectors. To prevent clutter, we omit all biases b in the network parameters (note that the theorem still holds with the biases).

If each NPN subnetwork has only one linear layer, Eqn. 6 and Eqn. 7 for the n-th network can be written as (omitting the bias terms):

$$\mu_\theta(\mathbf{x}_m, \mathbf{v}_{n-1,m}) = (\mathbf{x}_m, \mathbf{v}_{n-1,m})\mathbf{w}_m^T, \tag{9}$$

$$s_\theta(\mathbf{x}_m, \mathbf{v}_{n-1,m}) = (\mathbf{x}_s, \mathbf{v}_{n-1,s})\mathbf{w}_s^T + (\mathbf{x}_s, \mathbf{v}_{n-1,s})(\mathbf{w}_m \circ \mathbf{w}_m)^T + ((\mathbf{x}_m, \mathbf{v}_{n-1,m}) \circ (\mathbf{x}_m, \mathbf{v}_{n-1,m}))\mathbf{w}_s^T, \tag{10}$$

where $(\mathbf{x}_m, \mathbf{x}_s)$ and $(\mathbf{w}_m, \mathbf{w}_s)$ are the mean-variance pairs of the distributions $p(\mathbf{x}|\boldsymbol{\phi}_x)$ and $p(\mathbf{w}_n|\boldsymbol{\theta}_n)$, respectively. $(\mathbf{v}_{n-1,m}, \mathbf{v}_{n-1,s})$ is the mean-variance pair for $\mathbf{v}_{n-1}$.

Theorem 3. *Assume all n single-layer NPN subnetworks are correct, namely, the output $(\mu_{\theta_n}(\boldsymbol{\phi}_x, \boldsymbol{\psi}_{n-1}), s_{\theta_n}(\boldsymbol{\phi}_x, \boldsymbol{\psi}_{n-1}))$ of the n-th subnetwork is the mean and variance of the following distribution (where $\boldsymbol{\phi}_x$ can be the mean and variance of $\mathbf{x}$ and $\boldsymbol{\psi}_{n-1}$ can be the mean and variance of $\mathbf{v}_{n-1}$):*

$$p(v_n|\boldsymbol{\phi}_x, \boldsymbol{\psi}_{n-1}, \boldsymbol{\theta}_n) = \int p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{v}_{n-1}|\boldsymbol{\psi}_{n-1})p(\mathbf{w}_n|\boldsymbol{\theta}_n)p(v_n|\mathbf{x}, \mathbf{v}_{n-1}, \mathbf{W}_n)d\mathbf{x}d\mathbf{v}_{n-1}d\mathbf{w}_n, 1 \le n \le N \tag{11}$$

where $p(v_n|\mathbf{x}, \mathbf{v}_{n-1}, \mathbf{W}_n)$ a Dirac delta distribution centered at $v_n = (\mathbf{x}, \mathbf{v}_{n-1})\mathbf{w}_n^T$ (computed recursively). Consider the following recursive process (also mentioned at the start of Sec. 2):

- *Define the concatenation of tuples $\widehat{\mathbf{u}}_k = (\widehat{u}_1, \ldots, \widehat{u}_k)$ with each tuple $\widehat{u}_k$ as the estimated mean and variance (output by NPN) of v_k given $\mathbf{x}$ (note that NPN can take mean-variance pairs as input).*
- *Let $\widehat{u}_1 = (\mu_{\theta_1}(\mathbf{x}), s_{\theta_1}(\mathbf{x}))^T$.*
- *For $1 < k \le N$, let $\widehat{u}_k = (\mu_{\theta_k}(\mathbf{x}, \widehat{\mathbf{u}}_{k-1}), s_{\theta_k}(\mathbf{x}, \widehat{\mathbf{u}}_{k-1}))^T$.*

Then the computed $\widehat{\mathbf{u}}_N$ contains the mean and variance (diagonal entries of the covariance matrix) of the joint distribution of all variables $\mathbf{v}_N$:

$$p(\mathbf{v}_N|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_N) = \int p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{W}_N|\boldsymbol{\Theta}_N)p(\mathbf{v}_N|\mathbf{x}, \mathbf{W}_N)d\mathbf{x}d\mathbf{W}_N, \tag{12}$$

where $p(\mathbf{v}_N|\mathbf{x}, \mathbf{W}_N)$ a Dirac delta distribution centered at $\mathbf{v}_N = (v_n)_{n=1}^N$ and $v_n = (\mathbf{x}, \mathbf{v}_{n-1})\mathbf{w}_n^T$ (computed recursively). Specifically $\widehat{\mathbf{u}}_{N,1*} = (\widehat{u}_{n,1})_{n=1}^N$ (concatenate the first entries of all tuples to a vector) is the mean, and similarly $\widehat{\mathbf{u}}_{N,2*} = (\widehat{u}_{n,2})_{n=1}^N$ is the variance.

Proof. We first focus on the **mean** of $p(\mathbf{v}_N|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_N)$ and prove it by induction. For the base case, according to Eqn. 11, $\widehat{u}_1 = (\mu_{\theta_1}(\mathbf{x}), s_{\theta_1}(\mathbf{x}))$ is the mean and variance of the distribution:

$$p(\mathbf{v}_1|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_1) = \int p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{W}_1|\boldsymbol{\Theta}_1)p(\mathbf{v}_1|\mathbf{x}, \mathbf{W}_1)d\mathbf{x}d\mathbf{W}_1 \tag{13}$$

$$= \int p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{w}_1|\boldsymbol{\theta}_1)p(v_1|\mathbf{x}, \mathbf{w}_1)d\mathbf{x}d\mathbf{w}_1. \tag{14}$$

Assume $\widehat{\mathbf{u}}_{n-1,1*}$ is the mean of the distribution

$$p(\mathbf{v}_{n-1}|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_{n-1}) = \int p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{W}_{n-1}|\boldsymbol{\Theta}_{n-1})p(\mathbf{v}_{n-1}|\mathbf{x}, \mathbf{W}_{n-1})d\mathbf{x}d\mathbf{W}_{n-1}. \tag{15}$$

Then the mean of the distribution

$$p(\mathbf{v}_n|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_n) = \int p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{W}_n|\boldsymbol{\Theta}_n)p(\mathbf{v}_n|\mathbf{x}, \mathbf{W}_n)d\mathbf{x}d\mathbf{W}_n \tag{16}$$

can be written as

$$\begin{aligned}
&\int (\mathbf{v}_{n-1}, (\mathbf{x}, \mathbf{v}_{n-1})\mathbf{w}_n^T)\ p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{W}_n|\boldsymbol{\Theta}_n)p(\mathbf{v}_n|\mathbf{x}, \mathbf{W}_n)d\mathbf{x}d\mathbf{W}_n \qquad (17)\\
=&\int (\mathbf{v}_{n-1}, (\mathbf{x}, \mathbf{v}_{n-1})\mathbf{w}_n^T)\ p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{w}_n|\boldsymbol{\theta}_n)p(\mathbf{W}_{n-1}|\boldsymbol{\Theta}_{n-1})p(\mathbf{v}_{n-1}|\mathbf{x}, \mathbf{W}_{n-1})d\mathbf{x}d\mathbf{W}_n\\
=&\int (\mathbf{v}_{n-1}, (\mathbf{x}, \mathbf{v}_{n-1})\mathbf{w}_n^T)\ p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{w}_n|\boldsymbol{\theta}_n)(\int p(\mathbf{x}|\boldsymbol{\phi}_x)d\mathbf{x})p(\mathbf{W}_{n-1}|\boldsymbol{\Theta}_{n-1})p(\mathbf{v}_{n-1}|\mathbf{x}, \mathbf{W}_{n-1})d\mathbf{W}_{n-1}d\mathbf{x}d\mathbf{w}_n d\mathbf{v}_{n-1}\\
=&\int (\mathbf{v}_{n-1}, (\mathbf{x}, \mathbf{v}_{n-1})\mathbf{w}_n^T)\ p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{w}_n|\boldsymbol{\theta}_n)(\int p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{W}_{n-1}|\boldsymbol{\Theta}_{n-1})p(\mathbf{v}_{n-1}|\mathbf{x}, \mathbf{W}_{n-1})d\mathbf{x}d\mathbf{W}_{n-1})d\mathbf{x}d\mathbf{w}_n d\mathbf{v}_{n-1}\\
=&\int (\mathbf{v}_{n-1}, (\mathbf{x}, \mathbf{v}_{n-1})\mathbf{w}_n^T)\ p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{w}_n|\boldsymbol{\theta}_n)p(\mathbf{v}_{n-1}|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_{n-1})d\mathbf{x}d\mathbf{w}_n d\mathbf{v}_{n-1}
\end{aligned}$$

According to the case of $(n-1)$ in Eqn. 15, we have the mean of $p(\mathbf{v}_{n-1}|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_{n-1})$ as

$$\begin{aligned}
&\int \mathbf{v}_{n-1}\ p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{w}_n|\boldsymbol{\theta}_n)p(\mathbf{v}_{n-1}|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_{n-1})d\mathbf{x}d\mathbf{w}_n d\mathbf{v}_{n-1}\\
=&\int (\int \mathbf{v}_{n-1}\ p(\mathbf{v}_{n-1}|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_{n-1})d\mathbf{v}_{n-1})p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{w}_n|\boldsymbol{\theta}_n)d\mathbf{x}d\mathbf{w}_n\\
=&\int \widehat{\mathbf{u}}_{n-1,1*}p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{w}_n|\boldsymbol{\theta}_n)d\mathbf{x}d\mathbf{w}_n\\
=&\widehat{\mathbf{u}}_{n-1,1*}\int p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{w}_n|\boldsymbol{\theta}_n)d\mathbf{x}d\mathbf{w}_n\\
=&\widehat{\mathbf{u}}_{n-1,1*}.
\end{aligned}$$

Besides, we have

$$\begin{aligned}
&\int (\mathbf{x}, \mathbf{v}_{n-1})\mathbf{w}_n^T\ p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{w}_n|\boldsymbol{\theta}_n)p(\mathbf{v}_{n-1}|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_{n-1})d\mathbf{x}d\mathbf{w}_n d\mathbf{v}_{n-1}\\
=&(\mathbf{x}_m, \mathbf{v}_{n-1,m})\mathbf{w}_m^T\\
=&\mu_\theta(\mathbf{x}_m, \mathbf{v}_{n-1,m})\\
=&\widehat{u}_{n,1}.
\end{aligned}$$

Hence the mean of the distribution $p(\mathbf{v}_n|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_n)$ is:

$$\int (\mathbf{v}_{n-1}, (\mathbf{x}, \mathbf{v}_{n-1})\mathbf{w}_n^T)\ p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{W}_n|\boldsymbol{\Theta}_n)p(\mathbf{v}_n|\mathbf{x}, \mathbf{W}_n)d\mathbf{x}d\mathbf{W}_n = \widehat{\mathbf{u}}_{n,1*},$$

meaning that $\widehat{\mathbf{u}}_{N,1*}$ is the mean of $p(\mathbf{v}_N|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_N)$.

Next we use similar techniques to prove that $\widehat{\mathbf{u}}_{N,2*}$ is the **variance** of $p(\mathbf{v}_N|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_N)$. For the base case, according to Eqn. 11, $\widehat{u}_1 = (\mu_{\theta_1}(\mathbf{x}), s_{\theta_1}(\mathbf{x}))$ is the mean and variance of the distribution:

$$\begin{aligned}
p(\mathbf{v}_1|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_1) &= \int p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{W}_1|\boldsymbol{\Theta}_1)p(\mathbf{v}_1|\mathbf{x}, \mathbf{W}_1)d\mathbf{x}d\mathbf{W}_1 \qquad (18)\\
&= \int p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{w}_1|\boldsymbol{\theta}_1)p(v_1|\mathbf{x}, \mathbf{w}_1)d\mathbf{x}d\mathbf{w}_1. \qquad (19)
\end{aligned}$$

Assume $\widehat{\mathbf{u}}_{n-1,1*}$ and $\widehat{\mathbf{u}}_{n-1,2*}$ are the mean and variance of the distribution

$$p(\mathbf{v}_{n-1}|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_{n-1}) = \int p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{W}_{n-1}|\boldsymbol{\Theta}_{n-1})p(\mathbf{v}_{n-1}|\mathbf{x}, \mathbf{W}_{n-1})d\mathbf{x}d\mathbf{W}_{n-1}. \qquad (20)$$

Then the covariance matrix of the distribution

$$p(\mathbf{v}_n|\boldsymbol{\phi}_x, \boldsymbol{\Theta}_n) = \int p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{W}_n|\boldsymbol{\Theta}_n)p(\mathbf{v}_n|\mathbf{x}, \mathbf{W}_n)d\mathbf{x}d\mathbf{W}_n \qquad (21)$$

can be written as

$$\int((\mathbf{v}_{n-1},(\mathbf{x},\mathbf{v}_{n-1})\mathbf{w}_n^T)-\widehat{\mathbf{u}}_{N,1*})^T((\mathbf{v}_{n-1},(\mathbf{x},\mathbf{v}_{n-1})\mathbf{w}_n^T)-\widehat{\mathbf{u}}_{N,1*})\,p(\mathbf{x}|\boldsymbol{\phi}_x)p(\mathbf{W}_n|\boldsymbol{\Theta}_n)p(\mathbf{v}_n|\mathbf{x},\mathbf{W}_n)d\mathbf{x}d\mathbf{W}_n, \quad (22)$$

$$=\int(\mathbf{v}_{n-1},(\mathbf{x},\mathbf{v}_{n-1})\mathbf{w}_n^T)^T(\mathbf{v}_{n-1},(\mathbf{x},\mathbf{v}_{n-1})\mathbf{w}_n^T)\,p(\mathbf{x}|\boldsymbol{\phi}_x)\,p(\mathbf{W}_n|\boldsymbol{\Theta}_n)p(\mathbf{v}_n|\mathbf{x},\mathbf{W}_n)d\mathbf{x}d\mathbf{W}_n-\widehat{\mathbf{u}}_{n,1*}^T\widehat{\mathbf{u}}_{n,1*} \quad (23)$$

$$=E[diag(\mathbf{v}_{n-1}^2,((\mathbf{x},\mathbf{v}_{n-1})\mathbf{w}_n^T)^2)]-E^2[diag(\mathbf{v}_{n-1},(\mathbf{x},\mathbf{v}_{n-1})\mathbf{w}_n^T)]+\mathbf{A}_n \quad (24)$$

$$=diag((\widehat{\mathbf{u}}_{n-1,2*},\widehat{u}_{n,2}))+\mathbf{A}_n \quad (25)$$

$$=diag(\widehat{\mathbf{u}}_{n,2*})+\mathbf{A}_n, \quad (26)$$

where $(\mathbf{v}_{n-1},(\mathbf{x},\mathbf{v}_{n-1})\mathbf{w}_n^T)^T(\mathbf{v}_{n-1},(\mathbf{x},\mathbf{v}_{n-1})\mathbf{w}_n^T)$ is an n-by-n matrix, and $\widehat{\mathbf{u}}_{n,1*}$ is the mean of $p(\mathbf{v}_n|\boldsymbol{\phi}_x,\boldsymbol{\Theta}_n)$. $\mathbf{v}_{n-1}^2$ is the element-wise square of $\mathbf{v}_{n-1}$. $diag(\mathbf{a})$ denotes the a diagonal matrix with values in the vector $\mathbf{a}$ as the diagonal entries. Eqn. 25 is due to Eqn. 20 and the fact the assumption that $\widehat{u}_{n,2}$ is the variance of the distribution in Eqn. 11. In Eqn. 24, The expectation is over the distribution $p(\mathbf{W}_n|\boldsymbol{\Theta}_n)p(\mathbf{v}_n|\mathbf{x},\mathbf{W}_n)$, and entries of $\mathbf{A}_n\in\mathbb{R}^{n\times n}$ can be computed recursively as:

$$\mathbf{A}_{n,ij}=\begin{cases}0 & 1\le i=j\le n\\ \mathbf{A}_{n-1,ij} & i<n \text{ and } j<n \text{ and } i\neq j\\ \widehat{u}_{i,2}\mathbf{w}_{n,m}^{(C+i)} & i<n \text{ and } j=n\\ \widehat{u}_{j,2}\mathbf{w}_{n,m}^{(C+j)} & i=n \text{ and } j<n,\end{cases}$$

where $\mathbf{w}_{n,m}^{(k)}$ is the k-th entry of $\mathbf{w}_{n,m}$. Hence the diagonal entries of the covariance matrix for the distribution in Eqn. 21 is $\widehat{u}_{n,2}$, which completes the proof for the variance part. □

Remark: Note that the theorem above is general since we do not assume Gaussian distributions. And since $\widehat{u}_{N,1*}$ and $\widehat{u}_{N,2*}$ are the mean and variance for the joint distribution Eqn. 12 (corresponding to V), respectively, the process in Theorem 3 also computes the correct mean and variance for the marginal distribution for V_{-S}. Theorem 3 can also be extended to the case where there is nonlinearity after the linear layer, as long as the mean and variance of the nonlinear layer can be computed exactly, as in most cases of NPN [20].

2.3 Marginal MAP

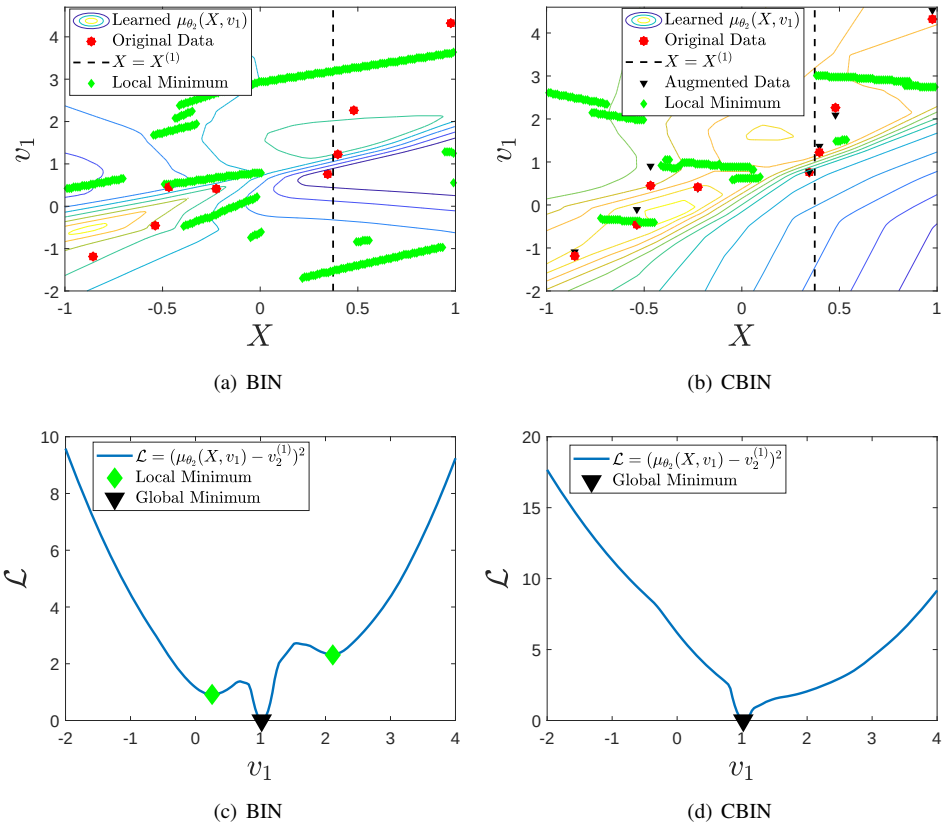
Note that using similar techniques above, BIN can be extended to get all marginal MAPs. For example, to get the marginal MAP of $p(v_2|\mathbf{X},v_3)$ during inference, we need to marginalize out v_1 and compute $p(v_2,v_3|\mathbf{X})=p(v_2|\mathbf{X})p(v_3|\mathbf{X},v_2)$. To do this, instead of using a deterministic v_1, we use the output mean and variance $(\mu_{\theta_1}(\mathbf{X}),s_{\theta_1}(\mathbf{X}))$ of the first subnetwork $p(v_1|\mathbf{X})$ as the input to subnetwork $p(v_2|\mathbf{X},v_1)$ to get the marginal $p(v_2|\mathbf{X})$. Marginal $p(v_3|\mathbf{X},v_2)$ can be obtained from subnetwork $p(v_3|\mathbf{X},v_1,v_2)$ similarly. Maximizing $p(v_2|\mathbf{X})p(v_3|\mathbf{X},v_2)$ with a fixed v_3 produces the MAP for v_2.

3 More Experimental Results

3.1 Toy Inference Tasks

Besides the toy inference task in the main paper, we examine a slightly more complex toy dataset where $\mathbf{X}$ is also considered (here $\mathbf{X}$ is a scalar for simplicity). We generate 8 data points $\{(\mathbf{X}^{(i)},v_1^{(i)},v_2^{(i)})\}_{i=1}^8$ according to $v_1=3\mathbf{X}+1+\epsilon_1$ and $v_2=0.5v_1-\mathbf{X}+1+\epsilon_2$, where ϵ_1 and ϵ_2 are sampled from $\mathcal{N}(0,1)$. $\mathbf{X}$ is sampled from a uniform distribution $\mathcal{U}(-1,1)$. We use similar hyperparameters as the first toy task (see the Supplement for details). Again, we train BIN according to Eqn. 4 in the paper and CBIN according to Eqn. 8 ($J=1$ and $V_{S_1}=\{v_1\}$) in the paper. The inference task is to infer $V_S=\{v_1\}$ givein $\mathbf{X}$ and $V_{-S}=\{v_2\}$.

Fig. 1(a) and Fig. 1(b) show the contours of $\mu_{\theta_2}(\mathbf{X},v_1)$ (predicted mean of v_2) learned by BIN and CBIN, respectively, with the original training data points. As we can see in Fig. 1, with a given $\mathbf{X}$ there are usually more than 1 local minima of $\mu_{\theta_2}(\mathbf{X},v_1)$ with respect to v_1 (there are even 5 when $\mathbf{X}$ is around -0.4) for BIN while there are much fewer for CBIN. Correspondingly, Fig. 1(c) and Fig. 1(d) show the loss surface of $\mathcal{L}$ with respect to $V_S=\{v_1\}$ when inferring v_1 given $(\mathbf{X}^{(1)},v_2^{(1)})$. As expected, BIN is easier to get trapped in poor local optima (shown as green



(a) BIN (b) CBIN

(c) BIN (d) CBIN

Figure 1: (a) and (b): $\mu_{\theta_2}(\mathbf{X}, v_1)$ learned by BIN and CBIN. (c) and (d): corresponding loss surface of $\mathcal{L}$ with respect to $\{v_1\}$ when inferring v_1 given $\mathbf{X}^{(1)}$ and $v_2^{(1)}$.

Table 1: Accuracy (%) for predicting V_S given $\mathbf{X}$ and $V_{\setminus S} = \{v_n\}_{n=1}^3 \setminus V_S$ in the *SHHS2* dataset.

V_S	$\{v_1\}$	$\{v_2\}$	$\{v_1, v_2\}$	$\{v_1, v_3\}$
SPEN	68.66	67.38	68.33	64.24
eSPEN	69.43	68.31	68.87	64.85
SVAE	67.75	66.77	66.95	62.86
PO	69.74	76.85	64.80	68.13
RI	70.36	70.16	64.42	65.78
BIN	78.19	77.54	71.87	72.53
CBIN	**78.50**	**78.77**	**72.27**	**73.91**
Retrain	78.86	78.31	71.45	73.72

Table 2: Accuracy (%) when $V_S = V$ for the *SHHS2* dataset.

V_S	$\{v_1\}$	$\{v_1, v_2\}$	$\{v_1, v_2, v_3\}$	$\{v_n\}_{n=1}^8$
SPEN	-	67.26	65.28	65.35
eSPEN	-	68.56	65.54	66.13
SVAE	-	68.14	65.71	65.90
BIN	-	68.64	66.05	**66.32**
CBIN	-	**69.04**	**66.35**	66.26
Retrain	71.21	68.11	65.63	66.09

diamonds) than CBIN. Note that since $\mathcal{L}$ in Fig. 1(c) is a quadratic function of $\mu_{\theta_2}(\mathbf{X}, v_1)$ in Fig. 1(a), local minima in Fig. 1(a) do *not* correspond to local opitma in Fig. 1(c).

3.2 Experiments on the *SHHS2* Dataset

Table 2 shows the accuracy in forward inference cases where $V_S = V$ with different V. We can see that SVAE, SPEN, and eSPEN achieve similar or slightly better accuracy than the retrained specific models which essentially assumes conditional independence between variables in V given $\mathbf{X}$. Compared to retrained models, BIN and CBIN consider also the conditional dependence among variables, making the predictions more accurate.

Fig. 2(left) shows the number of inference iterations needed to predict $V_S = \{v_1, v_2\}$ given $V_{\setminus S} = \{v_3\}$ versus number of inner loop iterations during training (T_{in} in Algorithm 1 of the main paper) with different λ_c. Fig. 2(right) shows the corresponding accuracy versus T_{in}. The horizontal lines show the number of inference iterations and accuracy

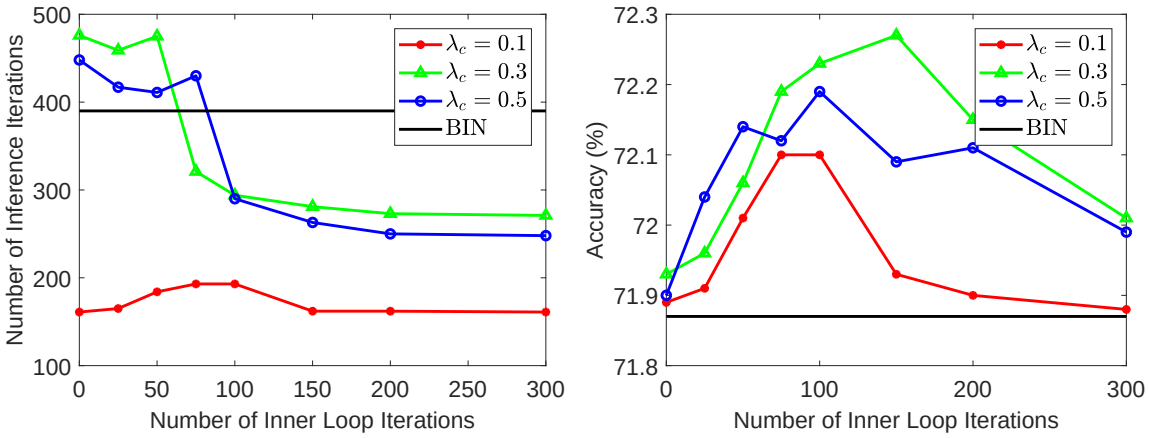


Figure 2: Left: Number of inference iterations needed during testing versus number of inner loop iterations during training (T_{in} in Algorithm 1) with different λ_c. The horizontal line the number for BIN (without $\mathcal{L}_j$). Right: Accuracy versus T_{in}. Similarly the horizontal line shows the accuracy of the corresponding BIN.

Table 3: Standard error for accuracy (%) of predicting V_S given $\mathbf{X}$ and $V_{-S} = \{v_n\}_{n=1}^8 \setminus V_S$ in the *SHHS2* dataset.

V_S	$\{v_1, v_3\}$	$\{v_4, v_5\}$	$\{v_1, v_3, v_6, v_7\}$	$\{v_2, v_6, v_7\}$	$\{v_3, v_5, v_8\}$	$\{v_4, v_5, v_6\}$	$\{v_4, v_6, v_7\}$
SVAE	0.16	0.19	0.11	0.13	0.13	0.13	0.20
DNADE	0.19	0.10	0.20	0.09	0.19	0.21	0.08
PO	0.18	0.11	0.16	0.17	0.10	0.17	0.19
RI	0.08	0.09	0.08	0.11	0.11	0.20	0.10
BIN	0.15	0.17	0.15	0.16	0.12	0.11	0.14
CBIN	0.10	0.08	0.20	0.09	0.08	0.10	0.11
Retrain	0.14	0.15	0.12	0.13	0.18	0.13	0.17

Table 4: Standard error for RMSE of predicting V_S given $\mathbf{X}$ and $V_{-S} = \{v_n\}_{n=1}^3 \setminus V_S$ in the *Dermatology* dataset.

V_S	$\{v_1\}$	$\{v_2\}$	$\{v_1, v_2\}$	$\{v_1, v_3\}$
SVAE	0.0108	0.0040	0.0147	0.0140
DNADE	0.0054	0.0025	0.0047	0.0066
PO	0.0142	0.0137	0.0011	0.0141
RI	0.0144	0.0121	0.0074	0.0145
BIN	0.0049	0.0044	0.0095	0.0113
CBIN	0.0083	0.0039	0.0051	0.0105
Retrain	0.0068	0.0019	0.0065	0.0091

Table 5: Standard error for RMSE when $V_S = V$ for the *Dermatology* dataset.

V_S	$\{v_1\}$	$\{v_1, v_2\}$	$\{v_1, v_2, v_3\}$
SVAE	-	0.0130	0.0103
DNADE	-	0.0136	0.0117
BIN	-	0.0138	0.0049
CBIN	-	0.0099	0.0112
Retrain	0.0052	0.0115	0.0023

for BIN. As we can see: (1) CBIN needs much fewer iterations during testing if T_{in} is large enough to get better estimates of V_S during training. (2) CBIN consistently outperforms BIN in a wide range of T_{in}. Results for other V_S (and V) are consistent with Fig. 2.

3.3 Standard Errors

In this section we provide the standard errors (of three trials of different random seeds) in Table 3, Table 4, and Table 5 corresponding to results in the main paper. As we can see, most differences between BIN/CBIN and baselines in accuracy/RMSE are larger than three times the standard errors.

4 Intuition and an Illustrative Example for CBIN

As an illustrative example, assume we want to learn a function $v_2 = f_\theta(v_1)$ (ignoring $\mathbf{X}$ for simplicity), given 4 data points $\{v_1^{(i)}, v_2^{(i)}\}_{i=1}^4$. Fig. 3 shows the model $v_2 = f_\theta(v_1)$ in the current epoch and the original 4 data points. In the T_i inner loops of Algorithm 1, $\widehat{v}_1^{(i)}$ will be inferred given $v_2^{(i)}$. For example, given $v_2^{(1)}$, Algorithm 1 (in the main paper)

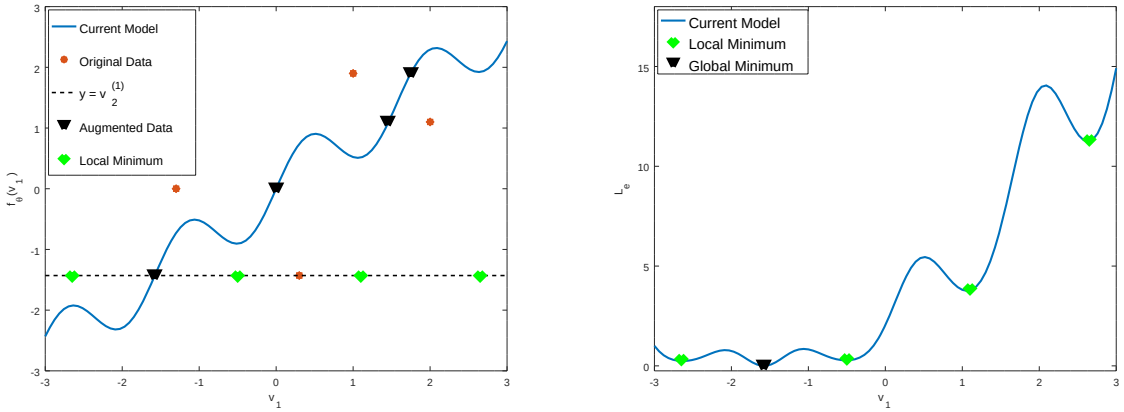


Figure 3: Left: The model $v_2 = f_\theta(v_1)$ in the current epoch, original data points, and augmented data points given by CL terms. Right: Loss surface of $\mathcal{L}_e$ when inferring $\widehat{v}_1^{(1)}$ given $v_2^{(1)}$ and the current model $f_\theta(\cdot)$. The T_i inner loops will try to find the global minimum according to $\frac{\partial L_c}{\partial v_1}$ and use it as $\widehat{v}_1^{(1)}$. If T_i is too small, it is more possible to get trapped in local minima.

will infer $\widehat{v}_1^{(1)}$ by iteratively computing $\frac{\partial \mathcal{L}_e}{\partial v_1}$ and updating v_1, where $\mathcal{L}_e = (f_\theta(v_1) - v_2^{(1)})^2$, as shown in Fig. 3(right). As we can see, the current model has many local optima when inferring $\widehat{v}_1$. Since the resulting $(\widehat{v}_1^{(i)}, v_2^{(i)})$ can be viewed as augmented data points, if $(\widehat{v}_1^{(i)}, v_2^{(i)})$ is closer to the current $v_2 = f_\theta(v_1)$ curve, it can make $v_2 = f_\theta(v_1)$ smoother after updating θ in the current epoch, leading to a loss surface $\mathcal{L}_e$ more friendly to backward inference (for v_1). For instance, we will have the augmented data points as shown in Fig. 3(left) if all $\widehat{v}_1^{(i)}$ can reach the global minima in the inner loops. Incorporating these points (in black) can make $v_2 = f_\theta(v_1)$ much smoother after updating θ.

As mentioned in Eqn. 9 of the paper, the augmented data $\widehat{V}_{S_j}$ is an approximation instead of the true minimizer. Hence in practice the augmented data points may not be exactly on the $v_2 = f_\theta(v_1)$ curve as shown in Fig. 3(left). Empirically however, we did not find this to be an issue because $\widehat{V}_{S_j}$ tends to be close enough to the curve if we have sufficient number of inner loop iterations T_{in}, as shown in Fig. 2(b) of the main paper and in Fig. 1(b).

Besides local optima, it is worth noting that when V_S has more than 1 element, the gradient-based optimization can also be affected by saddle points [6]. Potentially this problem can also be alleviated by extending BIN to CBIN, because the optimization landscape of the objective $\mathcal{L}$ with respect to V_S can be improved if $\widehat{V}_S$ (inferred using the inner loops of Algorithm 1 in the paper) can successfully escape the saddle points in certain iterations during training.

5 BIN/CBIN for Other Types of Distributions

As mentioned in Sec. of the paper, our model naturally generalizes to arbitrary exponential-family distributions (e.g., gamma distributions), due to the properties of NPN. In this section we briefly introduce BIN/CBIN with gamma distributions and Poisson distributions. Note that it is also possible to have a hybrid BIN/CBIN, where subnetworks belongs to different types of NPNs (e.g., some are Gaussian NPNs and others are gamma NPNs).

5.1 Gamma BIN/CBIN

Essentially gamma BIN/CBIN would replace the Gaussian NPN used in the main paper with gamma NPN. Specifically, the negative log-likelihood in Sec. of the paper:

$$-\log p(v_n|\mathbf{X}, V_{n-1}; \boldsymbol{\theta}_n) = \frac{\|\mu_{\theta_n}(\mathbf{X}, V_{n-1}) - v_n\|_2^2}{2s_{\theta_n}(\mathbf{X}, V_{n-1})} + \frac{1}{2}\log s_{\theta_n}(\mathbf{X}, V_{n-1}) \tag{27}$$

becomes

$$-\log p(v_n|\mathbf{X}, V_{n-1}; \boldsymbol{\theta}_n) = \log\Gamma(c) - c\log d - (c-1)\log v_n + dv_n, \tag{28}$$

where

$$c = \frac{\mu_{\theta_n}(\mathbf{X}, V_{n-1})^2}{s_{\theta_n}(\mathbf{X}, V_{n-1})}, \quad d = \frac{\mu_{\theta_n}(\mathbf{X}, V_{n-1})}{s_{\theta_n}(\mathbf{X}, V_{n-1})}.$$

The computation of the hidden layers in gamma NPN is the same as that in [20].

5.2 Poisson BIN/CBIN

Besides gamma distributions, BIN/CBIN can also model counts (e.g., word counts in documents). Similar to gamma BIN/CBIN, Eqn. 27 in Poisson BIN/CBIN becomes:

$$-\log p(v_n|\mathbf{X}, V_{n-1}; \boldsymbol{\theta}_n) = -v_n \log c + c + \log(v_n!), \tag{29}$$

where v_n is a nonnegative integer and

$$c = \frac{1}{4}(2\mu_{\theta_n}(\mathbf{X}, V_{n-1}) - 1 + \sqrt{(2\mu_{\theta_n}(\mathbf{X}, V_{n-1}) - 1)^2 + 8s_{\theta_n}(\mathbf{X}, V_{n-1})}).$$

In practice we need to relax the nonnegative integer v_n to be a nonnegative real value and replace $v_n!$ with $\Gamma(v_n + 1)$ to enable BP. During inference, the predicted real-valued v_n is then transformed back to a nonnegative integer in the end.

6 More Experiment Details and Hyperparameters

We use the published code of [1, 2] implemented in Torch 7 [5] for experiments on SPEN and eSPEN, while our models BIN/CBIN, 'Retrain' (NPN), and other baselines are implemented using PyTorch[3].

6.1 Toy Inference Tasks

For the first toy inference task, models are trained for 100 epochs with a minibatch size of 1. We use Adam with a learning rate of 0.005. For CBIN, we set $\lambda_c = 1$, the number of inner loop iterations $T_{in} = 10$, and the number of warmup epochs $T_w = 10$. For the second toy inference task, we use Adam [12] with a learning rate of 0.01 in the training loop and a learning rate of 0.05 in the inner loop. For CBIN, we set $\lambda_c = 1$, the number of inner loop iterations $T_{in} = 20$, and the number of warmup epochs $T_w = 10$. For simplicity, $s_{\theta_2}(\cdot)$ is ignored (set to a constant 1) in both tasks. We use multi-layer perceptrons (MLP) with 2 hidden layers of 64 neurons.

6.2 Experiments on Real-world Datasets

In the experiments, all subnetworks (e.g., $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$) share the same encoder which encodes $\mathbf{X}$ into a 512-dimensional (fixed-length) vector. We use 75% of the dataset for training and the rest for testing. Cross validation is performed to determine the best network structures and hyperparameters (see the Supplement for details). We use NPN subnetworks with one hidden layer of 50 neurons after the encoder. For iterative inference on the test set, we perform the same type of inference on the validation set to decide the number of inference iterations. Note that for the inference in the inner loop (with T_{in} iterations) of Algorithm 1 (in the main paper), one can dramatically speed up the computation by treating all data points as one single minibatch.

For the experiments on *SHHS2*, the fixed-length encodings (256, 128, and 128 dimensions for breathing, EEG, and ECG respectively) produced by three encoders, along with V_k, are concatenated to a $(512 + k)$-dimensional vector, followed by a hidden NPN layer of 50 neurons and an output NPN layer that produces the mean and variance of v_n. Note that the encoder part is shared across different NPN subnetworks and that the model is trained in an end-to-end fashion. For both real-world datasets, we use the default ordering of variables in the datasets except that we move forward the variable 'general health' of *SHHS2* from the last to the third, for convenience of evaluation when $V = \{v_n\}_{n=1}^3$. For *SHHS2* we tried different ordering when $V = \{v_n\}_{n=1}^8$, and the results are very similar and consistent to Table 1 in the paper. For *Dermatology*, v_1, v_2, and v_3 are 'vacuolisation and damage of basal layer', 'saw-tooth appearance of retes', and 'elongation of the rete ridges'[4].

[3] `https://github.com/pytorch/pytorch`

[4] We choose 3 histopathological attributes with the the largest average covariance.

Table 6: Network structure for encoders of EEG $\mathbf{X}_e$ and ECG $\mathbf{X}_c$.

Kernel	Stride	Channel In	Channel Middle	Channel Out	Type	Number
5	1	C_{in}	128	128	ResBlock	1
5	1	128	64	128	ResBlock	3
5	3	128	64	256	ResBlock	1
-	-	256	-	256	SRU	1
3	2	256	128	512	ResBlock	1
-	-	512	-	512	SRU	1
3	2	512	256	512	ResBlock	1
-	-	512	-	512	SRU	1

Table 7: Additional layers for encoders of breathing signals $\mathbf{X}_b$.

Kernel	Stride	Channel In	Channel Middle	Channel Out	Type	Number
11	5	C_{in}	-	64	Conv	1
5	1	64	32	64	ResBlock	3
5	2	64	32	64	ResBlock	1

For the experiments on *Dermatology*, we use 80% of the data for training and the rest for testing. Cross validation is performed to decide hyperparameters. Since $\mathbf{X}$ in *Dermatology* is low-dimensional features, Gaussian NPNs with one hidden layer of 50 neurons are used as subnetworks and no shared encoder is needed. As preprocessing, all attributes are normalized into $[0, 1]$.

In the experiments, the number of warmup epochs $T_w = 1$ and the number of inner loop iterations $T_{in} = 150$. We use Adam with a learning rate of $1 \times e^{-4}$ during training and $1 \times e^{-3}$ during inference. The minibatch size is set to 4. For fairness in the number of free parameters, we also try a larger number of hidden neurons ($50 \sim 400$) in the hidden layer of 'Retrain', SVAE, DNADE, SPEN, and eSPEN, and the best performance is used in the tables of the main papers (50 is the best choice most of the time). As mentioned in the paper, we assume Gaussian distributions for the variables in both datasets. For the classification task in *SHHS2*, we use a threshold of 0.5 to process the predicted values.

6.3 Network Architecture for Encoders

Table 6 shows the neural network architecture for the encoders of EEG $\mathbf{X}_e$ and ECG $\mathbf{X}_c$. Since the breathing signal $\mathbf{X}_b$ is 10 times the length of $\mathbf{X}_e$ and $\mathbf{X}_c$, additional layers (as shown in Table 7) are needed to align $\mathbf{X}_b$ with $\mathbf{X}_e$ and $\mathbf{X}_c$. We use 1D convolution since $\mathbf{X}$ is time series with multiple channels. 'ResBlock' refers to the ResNet block as used in [10]. We use simple recurrent units [3, 15] as a simplified version of gated recurrent units [4] as our recurrent neural network (RNN) components. 'Number' in the tables indicates the number of corresponding blocks stacked in the network. The output of the last SRU will then go through a self-attention layer [16] to output a fix-length encoding, which later are shared as inputs by all NPN subnetworks of BIN/CBIN. The network is trained in an end-to-end fashion.

6.4 Details on the SVAE Baseline

As a baseline in the experiments, we combine SVAE [11, 13] and our method to enable BP-based inference and avoid $O(2^N)$ networks. Specifically, we train the model by maximizing

$$\begin{aligned}\log p(V|\mathbf{X}) &= \log \int p_{enc}(\mathbf{z}|\mathbf{X}) p_{dec}(V|\mathbf{z}) d\mathbf{z} \\ &\geq \int p_{enc}(\mathbf{z}|\mathbf{X}) \log p_{dec}(V|\mathbf{z}) d\mathbf{z} \\ &= \mathbb{E}_{p_{enc}(\mathbf{X}|\mathbf{z})}[\log p_{dec}(V|\mathbf{z})],\end{aligned}$$

where $\mathbf{z}$ is the latent variable, $p_{enc}(\mathbf{z}|\mathbf{X})$ can be seen as the prior on $\mathbf{z}$ using $\mathbf{X}$ as input, and $V = V_S \cup V_{\setminus S}$ is the set of all variables. Similar to [11, 13], $p_{enc}(\mathbf{z}|\mathbf{X})$ and $p_{dec}(V|\mathbf{z})$ can be learned using BP and the reparameterization trick. Note that since we have the prior $p_{enc}(\mathbf{z}|\mathbf{X})$ on $\mathbf{z}$, the recognition model $q(\mathbf{z}|V)$ is not needed here.

During the inference phase, using MAP and BP as in BIN, one can infer any subset V_S from V and $V_{\setminus S}$ by finding

$$\operatorname*{argmax}_{V_S, \mathbf{z}} p_{enc}(\mathbf{z}|\mathbf{X}) p_{dec}(V_S, V_{\setminus S}|\mathbf{z}),$$

where $p_{enc}(\mathbf{z}|\mathbf{X})$ provides regularization when updating $\mathbf{z}$ during inference, and $p_{dec}(V_S, V_{-S}|\mathbf{z})$ provides the main gradient, similar to the inferential procedure of BIN.

Note that a learned inferential procedure $q(\mathbf{z}|\mathbf{X}, V_{-S})$ would be *subset-specific*, meaning $O(2^N)$ *models* of $q(\mathbf{z}|\mathbf{X}, V_{-S})$ for $O(2^N)$ *possible subsets*; so a feasible method would have to involve an MAP inference over $\mathbf{z}$ as well as V_S, as shown above. This "augmentation" step within MAP can introduce looseness, however. In fact, the same idea is used in sampling methods precisely to avoid getting stuck in local rigid configurations. In contrast, the Bayesian network formulation in BIN captures even rigid interactions very directly.

7 Configuration of V_{S_j}

As mentioned in the paper, for CBIN, one challenge is that there are $2^N - 1$ configurations of V_{S_j}, including all $2^N - 1$ terms of $\mathcal{L}_j$ during training is obviously impractical. In our experiments, we let $J = N - 1$ and $V_{S_j} = V_j = \{v_n\}_{n=1}^{j}$. Doing this has the effect of both self-correction and improving the optimization landscape: (1) **Self-correction**: For example, when $N = 3$ and $j = 2$, we have $\widehat{V}_{S_j} = \{\widehat{v}_1, \widehat{v}_2\}$ and $V_{-S_j} = \{v_3\}$. Since (a) $(\widehat{v}_1, \widehat{v}_2)$ is different from (v_1, v_2), and (b) $(\widehat{v}_1, \widehat{v}_2)$ is the current best estimate for (v_1, v_2) given the true v_3, using $(\widehat{v}_1, \widehat{v}_2, v_3)$ as 'augmented data' to train the $N = 3$ subnetworks ($p(v_1|\mathbf{X})$, $p(v_2|\mathbf{X}, v_1)$, and $p(v_3|\mathbf{X}, v_1, v_2)$) in CBIN has the effect of guiding the subnetworks to perform self-correction collaboratively. Here the inner loop inferring $(\widehat{v}_1, \widehat{v}_2)$ can be seen as searching for the best path for self-correction. (2) **Optimization landscape**: The effect of improving the optimization landscape comes from using inner loops to infer $\widehat{V}_j$ given $V \setminus V_j$ and using $(\widehat{V}_j, V \setminus V_j)$ during training. Note that the generated $\widehat{V}_j$ is used as input for $N - j$ subnetworks (e.g., subnetworks $p(v_2|\mathbf{X}, v_1)$, and $p(v_3|\mathbf{X}, v_1, v_2)$ both use $\widehat{v}_1$ as input). Hence using $N - 1$ extra terms is sufficient to cover N subnetworks.

8 Alternative Building Blocks and Related Work

It may be tempting to use probabilistic NN such as [14, 9] as building blocks. However: (1) These methods are designed for binary variables (and assume Bernoulli distributions) while NPN [20] can handle different kinds of variables (e.g., binary variables or continuous variables) and arbitrary exponential-family distributions. (2) More importantly, they do not output the prediction variance, which is crucial to naturally trade off the influence of the prior terms and the terms that provide the main gradient as mentioned in Sec. of the main paper. Besides the choice about building blocks, note that [14, 9] model the joint distributions of binary variables in a deterministic way (models such as [19] can be seen as extension of [14, 9] for real values). Hence they can only predict variables using a feedforward pass and cannot perform inference for the values of an arbitrary set of variables. The best [14, 9, 19] can do is to predict $V \setminus V_k$ given $\mathbf{X}$ and V_k, which covers only $N - 1$ of the $2^N - 1$ cases.

Note that BIN and NADE-based models (including the orderless and real-valued version [18, 17]) are substantially different: (1) BIN performs inference mainly with backpropagation, while NADEs perform inference the usual way with feedforward. (2) NADEs use parameter sharing to parameterize different conditionals with a single network while BIN parameterizes each conditional with an NPN; (3) BIN supports arbitrary Bayesian Network structures while NADEs do not; this allows convenient incorporation of domain knowledge; (4) BIN can be extended to CBIN to improve accuracy/efficiency. Finally, the large performance gap in Table 2∼4 empirically verifies these differences.

9 Figures in the Paper

In this section we provide additional illustrative figures in Fig. 4 (an example inference process of BIN) and larger versions of some figures in the main paper for readers' convenience in Fig. 5.

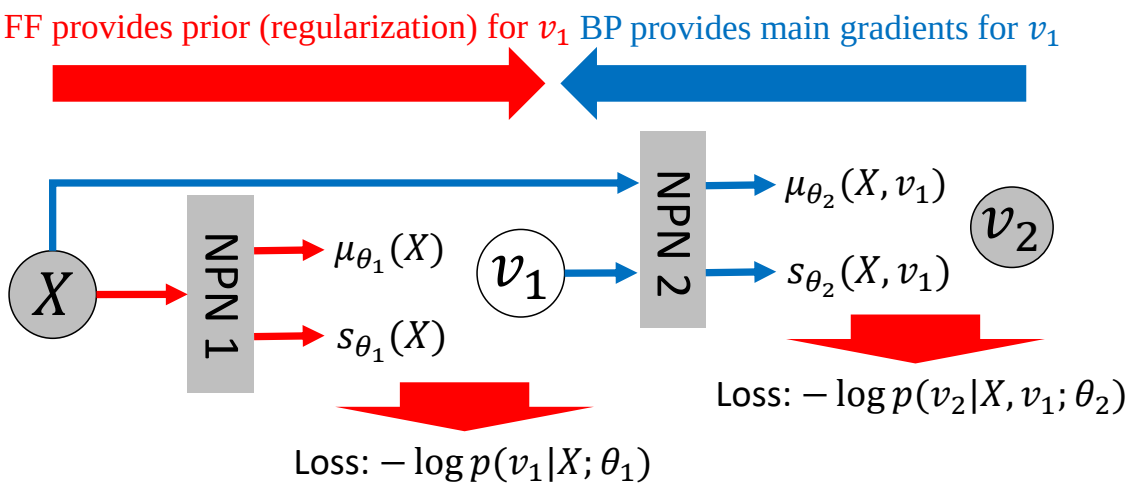
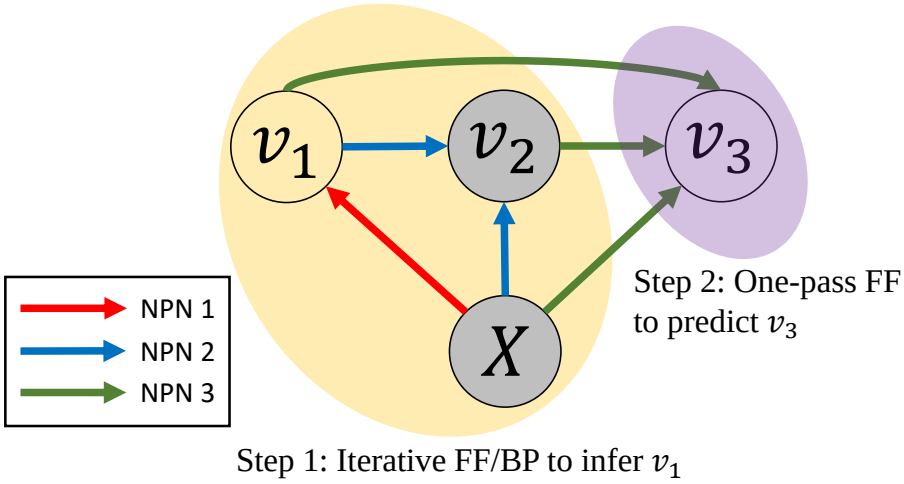
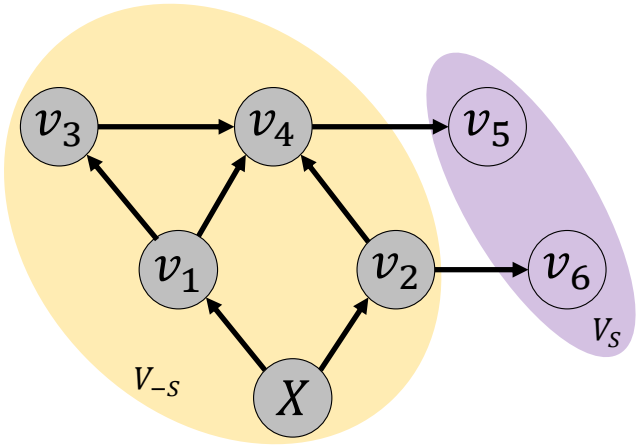


Figure 4: Illustration of the inference process of BIN when $V_S = \{v_1\}$ and $V_{\neg S} = \{v_2\}$. FF provides prior (regularization) for v_1 and BP provides the main gradients to update v_1. Circles and rectangles in grey indicates observed variables and fixed networks parameters, respectively.





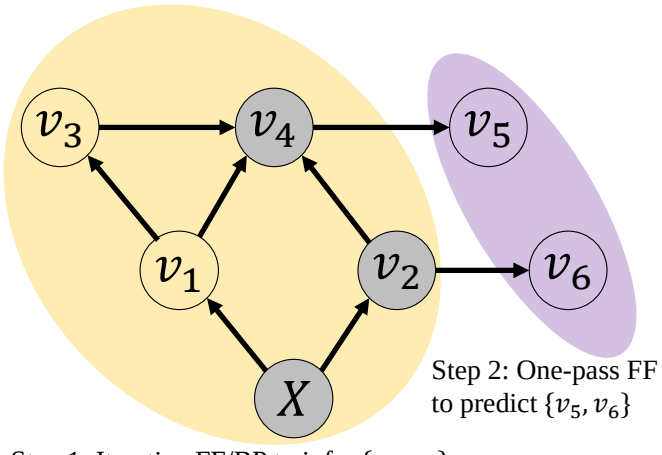


Figure 5: Top: An example for *hybrid inference* when $V_S = \{v_1, v_3\}$ and $V_{-S} = \{v_2\}$. Edges in different colors correspond to different NPN subnetworks. Best viewed in color. Middle: An example for *forward prediction* of a more general BN structure. Bottom: An example for *hybrid inference* of a more general BN structure.

References

[1] David Belanger and Andrew McCallum. Structured prediction energy networks. In *ICML*, pages 983–992, 2016.

[2] David Belanger, Bishan Yang, and Andrew McCallum. End-to-end learning for structured prediction energy networks. In *ICML*, pages 429–439, 2017.

[3] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. *CoRR*, abs/1611.01576, 2016.

[4] Kyunghyun Cho, Bart van Merrienboer, **c**Caglar Gül**c**cehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[5] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS workshop*, 2011.

[6] Yann N. Dauphin, Razvan Pascanu, **c**Caglar Gül**c**cehre, KyungHyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NIPS*, pages 2933–2941, 2014.

[7] Sudhakar Dharmadhikari and Kumar Joag-Dev. *Unimodality, convexity, and applications*. Elsevier, 1988.

[8] Gabriel Frahm. *Generalized elliptical distributions: theory and applications*. PhD thesis, Universität zu Köln, 2004.

[9] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: masked autoencoder for distribution estimation. In *ICML*, pages 881–889, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[11] Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In *NIPS*, pages 2946–2954, 2016.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[13] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[14] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *AISTATS*, pages 29–37, 2011.

[15] Tao Lei, Yu Zhang, and Yoav Artzi. Training RNNs as fast as CNNs. *CoRR*, abs/1709.02755, 2017.

[16] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130, 2017.

[17] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *JMLR*, 17:205:1–205:37, 2016.

[18] Benigno Uria, Iain Murray, and Hugo Larochelle. RNADE: the real-valued neural autoregressive density-estimator. In *NIPS*, pages 2175–2183, 2013.

[19] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, pages 1747–1756, 2016.

[20] Hao Wang, Xingjian Shi, and Dit-Yan Yeung. Natural-parameter networks: A class of probabilistic neural networks. In *NIPS*, pages 118–126, 2016.