Bidirectional Inference Networks: A Class of Deep Bayesian Networks for Health Profiling

Hao Wang,[1] **Chengzhi Mao,**[2] **Hao He,**[1] **Mingmin Zhao,**[1] **Tommi S. Jaakkola,**[1] **Dina Katabi**[1]
[1]MIT CSAIL, Cambridge, MA [2]Columbia University, New York, NY
{hwang87, haohe, mingmin}@mit.edu, cm3797@columbia.edu, {tommi, dina}@csail.mit.edu

Abstract

We consider the problem of inferring the values of an arbitrary set of variables (e.g., risk of diseases) given other observed variables (e.g., symptoms and diagnosed diseases) and high-dimensional signals (e.g., MRI images or EEG). This is a common problem in healthcare since variables of interest often differ for different patients. Existing methods including Bayesian networks and structured prediction either do not incorporate high-dimensional signals or fail to model conditional dependencies among variables. To address these issues, we propose *bidirectional inference networks* (BIN), which stich together multiple probabilistic neural networks, each modeling a conditional dependency. Predictions are then made via iteratively updating variables using backpropagation (BP) to maximize corresponding posterior probability. Furthermore, we extend BIN to *composite BIN* (CBIN), which involves the iterative prediction process in the training stage and improves both accuracy and computational efficiency by adaptively smoothing the optimization landscape. Experiments on synthetic and real-world datasets (a sleep study and a dermatology dataset) show that CBIN is *a single model* that can achieve state-of-the-art performance and obtain better accuracy in most inference tasks than *multiple models each specifically trained for a different task.*

Introduction

In healthcare, it is often desirable to infer an arbitrary set of variables (e.g., the probability of having a particular disease) given other known or observable variables (e.g., coughing, itching, or already diagnosed diseases) and high-dimensional signals (e.g., ECG, EEG or CT) (Sesen et al. 2013). Which variables are known and which need to be inferred typically vary across patients. Such inference problems are important to assist clinicians in making more informed treatment decisions in clinical settings where uncertainty is ubiquitous.

Traditionally, Baysian networks (BN) are used for this task since they naturally reason with uncertain domain knowledge. However, although BN can model relationship among variables of interest, they do not model complex high-dimensional data such as MRI images or EEG time series. Deep neural networks (NN), on the other hand, are able to process such high-dimensional signals. Recently, there has been work that combines structured prediction and NN (Belanger and McCallum 2016; Belanger, Yang, and McCallum 2017) to take into account the relationship among variables of interest during prediction. However, (1) they are designed to predict all variables at once without the ability to infer an arbitrary subset of variables given others; (2) they do not model conditional dependencies among variables.

In this paper, we propose bidirectional inference networks (BIN) as a kind of probabilistic NN that can get the best of all worlds: (1) it can handle high-dimensional data (e.g., EEG, CT); (2) it can infer an arbitrary subset V_S of variables in V given $V \setminus V_S$; (3) it can capture complex conditional dependencies among variables of interest.

Note that a naive way to infer arbitrary V_S is to train different NNs for different V_S. However, this needs $O(2^N)$ NNs when $|V| = N$ and ignores the relationship among variables to be predicted. Instead, BIN tries to model the factorized joint distribution of V by connecting multiple probabilistic NNs, each modeling a factor (conditional distribution). After it is jointly trained, predictions are made via first *initializing V_S with a feedforward (FF) pass* and then *iteratively updating variables using backpropagation (BP) to maximize corresponding posterior probability*. BIN is then extended to CBIN, which involves the iterative prediction process in the training stage, consequently alleviating the problem of local optima and improving the performance.

We evaluate our model on two datasets (a large-scale sleep study (Quan et al. 1997) and a dermatology dataset (Lichman 2013)). Experiments show that *a single BIN* can predict V_S given $V \setminus V_S$ with performance comparable to or better than training $O(2^N)$ separate models for $O(2^N)$ different V_S when there are N variables in V. We can also outperform state-of-the-art structured prediction models (Belanger and McCallum 2016; Belanger, Yang, and McCallum 2017) adapted for our task.

In summary, our contributions are as follows:

- We propose BIN, a probabilistic neural network model that can perform bidirectional inference, handle high-dimensional data, and effectively predict an arbitrary subset V_S of variables in V given $V \setminus V_S$.
- We enhance BIN with composite likelihood to derive composite BIN (CBIN), which not only improves the inference accuracy but also reduces the number of iterations

Copyright © 2019, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

needed during inference.
- Experiments show that BIN and CBIN can outperform state-of-the-art structured prediction models adapted for our task and achieve performance comparable to or even better than training $O(2^N)$ separate models for $O(2^N)$ different V_S.
- We show that it is possible to predict health status including physical and emotional well-being scores using EEG, ECG, and breathing signals.

Related Work

Combination of Probabilistic Graphical Models and Deep Neural Networks Our work is related to a recent trend of combining probabilistic graphical models (PGM) and deep NN (Wang and Yeung 2016). In particular, (Johnson et al. 2016) builds on variational autoencoders (VAE) (Kingma and Welling 2013; Sohn, Lee, and Yan 2015) to propose *structured VAE* (SVAE), where they leverage conjugacy to update some variational parameters and use BP to update other parameters without conjugate structures. (Lin, Khan, and Hubacher 2018) generalizes SVAE to cover PGMs with non-conjugate factors and improve its performance. In addition, some methods focus on incorporating VAE into state-space models (as a kind of PGMs) (Fraccaro et al. 2016; Krishnan, Shalit, and Sontag 2015; Archer et al. 2015) or recurrent neural networks (RNN) (Chung et al. 2015; Gregor et al. 2015; Bayer and Osendorfer 2014). For example, deep Kalman filters (Krishnan, Shalit, and Sontag 2015) use variational distributions parameterized by multi-layer perceptrons or RNN to approximate the posterior distributions of the hidden states in the state-space models. Besides the VAE-based models above, there are also models based on other forms of probabilistic NN (Wang, Wang, and Yeung 2015; Zhang et al. 2016; Wang, Shi, and Yeung 2017).

These methods focus more on generative modeling than conditional structured prediction tasks. Besides, they usually need different inference networks to infer different subsets of variables, which means that $O(2^N)$ networks are needed for general inference of N variables (each variable can be in various forms, e.g., scalars and vectors). In our work, $O(N)$ subnetworks are sufficient to support general inference of N variables. Note that one feasible way to avoid $O(2^N)$ networks in the SVAE framework is to combine it with our method to enable BP-based inference. This is used as one of our baselines. Note that our work is also different from probabilistic NNs such as (Larochelle and Murray 2011; Germain et al. 2015) (though they also model conditional distributions using NN), which fail to infer an arbitrary subset of variables and are often restricted to binary variables (see the Supplement for detailed comparison).

Using Backpropagation as Inferential Procedures In our method, both FF and BP are involved during inference. This is different from most previous works, where only FF is involved during inference and BP is used to update parameters during training. However, the idea of using BP to (iteratively) compute predictions has been useful in some applications. For example, BP has been used to generate adversarial data points that NN would misclassify with high confidence (Goodfellow, Shlens, and Szegedy 2014; Szegedy et al. 2013), to generate embeddings for documents (Le and Mikolov 2014), and to perform texture synthesis or style transfer for images (Gatys, Ecker, and Bethge 2016; Gatys et al. 2017). In (Belanger and McCallum 2016; Belanger, Yang, and McCallum 2017), the authors propose variants of structured prediction energy networks (SPEN), which utilize BP to perform structured predictions. However, (1) SPEN is designed to predict all variables of interest at once given the input $\mathbf{X}$ and cannot perform inference on an arbitrary subset of variables given others (which is the focus of our method). (2) Unlike our method, even if SPEN is adapted to perform different inference cases, for most cases it does not have the proper 'prior distributions' to provide good initialization of the target variables V_S (for example, when inferring v_2 given $\mathbf{X}$, v_1, and v_3, SPEN does not have the prior $p(v_2|\mathbf{X}, v_1)$ to provide initialization for v_2 and can only rely on the general prior $p(v_2|\mathbf{X})$, which does not incorporate information of v_1), which is important for both accuracy and efficiency, as shown in our experiments. (3) Relationship among variables in SPEN is modeled using a global energy function while BIN models the relationship in a way similar to Bayesian networks. Hence BIN can handle conditional dependencies, more easily incorporate domain knowledge, and have better interpretability.

Natural-Parameter Networks Our model use *natural-parameter networks* (NPN) as a building block (Wang, Shi, and Yeung 2016). Different from vanilla NN which usually takes deterministic input, NPN is a probabilistic NN taking distributions as input. The input distributions 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. As a simple example, in a vanilla linear NN $f_w(x) = wx$ takes a scalar x as input and computes the output based on a scalar parameter w; a corresponding Gaussian NPN would assume w is drawn from a Gaussian distribution $\mathcal{N}(w_m, w_s)$ and that x is drawn from $\mathcal{N}(x_m, x_s)$ (x_s is set to 0 when the input is deterministic). With $\theta = (w_m, w_s)$ as a learnable parameter pair, NPN 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 (bias terms are ignored for clarity) as:

$$\mu_\theta(x_m, x_s) = E[wx] = x_m w_m, \quad (1)$$

$$s_\theta(x_m, x_s) = D[wx] = x_s w_s + x_s w_m^2 + x_m^2 w_s, \quad (2)$$

Hence the output of this Gaussian NPN is a tuple $(\mu_\theta(x_m, x_s), s_\theta(x_m, x_s))$ representing a Gaussian distribution instead of a single value. Input variance x_s to NPN is set to 0 in the next section. Note that since $s_\theta(x_m, 0) = x_m^2 w_s$, w_m and w_s can still be learned even if $x_s = 0$ for all data points (see the Supplement for generalization of NPN to handle vectors and matrices).

Bidirectional Inference Networks

Unlike typical inference networks that perform inference by NN's feedforward pass (Kingma and Welling 2013), our proposed bidirectional inference networks (BIN) use the

feedforward pass and/or the backpropagration pass to perform inference. Such a design enables the same network to (1) predict the output given the input, and (2) infer the input from the output. In this section we introduce BIN's learning and inference process. Note that though we assume Gaussian distributions for simplicity when necessary, *our framework applies to any exponential-family distributions* (see the Supplement for details).

Notation and Motivation

Notation and Problem Formulation We use $\mathbf{X}$ to denote the high-dimensional information (e.g., EEG) of each subject and $V = \{v_n\}_{n=1}^N$ to denote the subject's N attributes[1] of interest. For any index set $S \subseteq \{1, 2 \dots N\}$, we define $V_S = \{v_n | n \in S\}$ and $V_{\setminus S} = V \setminus V_S = \{v_n | n \notin S\}$. We use V_k to denote a set $\{v_n\}_{n=1}^k$, with $V_0 = \emptyset$. We are interested in learning a general network which is able to predict an arbitrary subset $V_S \subseteq V$ given other attributes $V_{\setminus S} = V \setminus V_S$ and $\mathbf{X}$, the simplest case being predicting all attributes V given $\mathbf{X}$. Input variance (corresponding to x_s in Eqn. 1-2) to NPN is set to 0 in this section.

Motivating Example Consider modeling the relations among $\mathbf{X}$ and variables $V_N = V_{N-1} \cup \{v_N\}$. Naively, one can learn a neural network $f(\cdot)$ with $(\mathbf{X}, V_{N-1})$ as input and v_N as output using the loss function $\mathcal{L} = \|f(\mathbf{X}, V_{N-1}) - v_N\|_2^2$. Such a network has no problem predicting v_N given $\mathbf{X}$ and V_{N-1}. However, if we want to infer $V_S \in V_{N-1}$ given $f(\cdot)$, $\mathbf{X}$, and $V_N \setminus V_S$, we need to (1) randomly initialize V_S, and (2) iteratively compute the gradient $\frac{\partial \mathcal{L}}{\partial V_S}$ and update V_S until $\mathcal{L}$ is minimized. Unfortunately this does not work because: (1) With random initialization, V_S can easily get trapped in a poor local optimum. (2) There are many adversarial gradient directions that can decrease $\mathcal{L}$ and lead V_S far from the ground truth, especially in large and complex networks (Zhang et al. 2017).

Model Formulation and Learning

To alleviate the problems above, we propose to construct BIN as a 'deep Bayesian network' over V so that we can properly *initialize* and *regularize* V_S for the iterative updates during inference (details of *initialization* and *regularization* of V_S during inference will be introduced later). We first factorize the conditional joint distribution $p(V|\mathbf{X})$ as:

$$p(V|\mathbf{X}) = \prod_{n=1}^{N} p(v_n|\mathbf{X}, V_{n-1}). \tag{3}$$

Note that here we assume full factorization using the chain rule as in (Larochelle and Murray 2011) only for simplicity (since usually we do not have prior knowledge on the relationship among variables, as is the case in our experiments); *BIN is actually general enough to handle arbitrary factorization (i.e., Bayesian network structure)*. Besides the difference in structure flexibility, as shown in the following sections, both learning and inference of BIN are also substantially different from (Larochelle and Murray 2011) and its variants. For example, BIN performs inference mainly with backpropagation, while (Larochelle and Murray 2011) performs inference the usual way with feedforward; BIN naturally parameterizes each conditional with an NPN, while (Larochelle and Murray 2011) uses parameter sharing to parameterize different conditionals (see the Supplement for detailed comparison). The large performance gap in Table 2∼4 also empirically verifies significance of these differences.

As mentioned above, we can naturally use N NPN networks with parameters $\boldsymbol{\theta}_n$ to model each of the conditional distribution $p(v_n|\mathbf{X}, V_{n-1})$. The negative joint log-likelihood for a given V can be written as:

$$\mathcal{L}(V|\mathbf{X}; \boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(v_n|\mathbf{X}, V_{n-1}; \boldsymbol{\theta}_n), \tag{4}$$

where each term corresponds to an NPN subnetwork:

$$-\log p(v_n|\mathbf{X}, V_{n-1}; \boldsymbol{\theta}_n) = \frac{\|\mu_{\theta_n}(\mathbf{X}, V_{n-1}) - v_n\|_2^2}{2 s_{\theta_n}(\mathbf{X}, V_{n-1})} + \frac{1}{2} \log s_{\theta_n}(\mathbf{X}, V_{n-1}), \tag{5}$$

where we assume Gaussian NPN in Eqn. 5 and $\boldsymbol{\theta} = \{\boldsymbol{\theta}_n\}_{n=1}^N$. $\mu_{\theta_n}(\cdot)$ and $s_{\theta_n}(\cdot)$ are the output mean and variance of the n-th NPN (similar to Eqn. 1∼2 with $x_s = 0$). Our model is trained by minimizing the negative log-likelihood (Eqn. 4∼5) of all M training samples.

Inference

General Inference Once the model is trained, it can be used to predict an arbitrary subset V_S given all other attributes $V_{\setminus S}$ and $\mathbf{X}$ using the maximum a posteriori (MAP) estimates of $p(V_S|\mathbf{X}, V_{\setminus S})$:

$$\begin{aligned} \operatorname*{argmax}_{V_S} p(V_S|\mathbf{X}, V_{\setminus S}; \boldsymbol{\theta}) &= \operatorname*{argmax}_{V_S} p(V_S, V_{\setminus S}|\mathbf{X}; \boldsymbol{\theta}) \\ &= \operatorname*{argmin}_{V_S} \mathcal{L}(V_S, V_{\setminus S}|\mathbf{X}; \boldsymbol{\theta}), \end{aligned} \tag{6}$$

where $\mathcal{L}(V_S, V_{\setminus S}|\mathbf{X}; \boldsymbol{\theta})$ is identical to Eqn. 4. We use Adam (Kingma and Ba 2014) as an adaptive gradient update procedure to minimize $\mathcal{L}(V_S, V_{\setminus S}|\mathbf{X}; \boldsymbol{\theta})$ with respect to V_S while *fixing* $\mathbf{X}$*,* $V_{\setminus S}$ *and* $\boldsymbol{\theta}$.

On the Motivating Example: Echoing the *motivating example* in previous text, note that during the inference of $v_n \in V_S$, the term $\frac{\|\mu_{\theta_n}(\mathbf{X}, V_{n-1}) - v_n\|_2^2}{2 s_{\theta_n}(\mathbf{X}, V_{n-1})}$ in Eqn. 5 provides both the *initialization* (i.e., initialize v_n as $\mu_{\theta_n}(\mathbf{X}, V_{n-1})$) and *adaptive prior (regularization)* while the term $\sum_{k=n+1}^{N} (\frac{\|\mu_{\theta_k}(\mathbf{X}, V_{k-1}) - v_k\|_2^2}{2 s_{\theta_k}(\mathbf{X}, V_{k-1})} + \frac{\log s_{\theta_k}(\mathbf{X}, V_{k-1})}{2})$ provides the main part of $\frac{\partial \mathcal{L}}{\partial v_n}$ to update v_n (since $v_n \in V_{k-1}$ when $k > n$). The trade-off between the prior and main gradient terms then depends on predicted variance terms $s_{\theta_k}(\mathbf{X}, V_{k-1})$ where $k \geq n$.

A Toy Example: As a more concrete example, assume $N = 2$ and the structure $p(v_1|\mathbf{X})p(v_2|\mathbf{X}, v_1)$. When inferring v_1's MAP estimate given $\{\mathbf{X}, v_2\}$: $p(v_1|\mathbf{X})$ is described

[1] Note that though in this paper v_i is a scalar value, our methods are general enough to handle vectors.

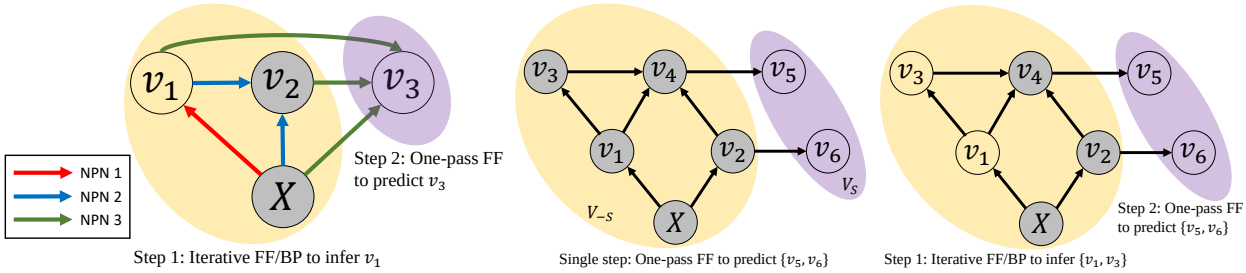


Figure 1: Transparent circles and shaded circles represent V_S and V_{-S}, respectively. Left: An example for *hybrid inference* when $V_S = \{v_1, v_3\}$ and $V_{-S} = \{v_2\}$. Edges in different colors correspond to different *probabilistic* neural networks (NPN). Best viewed in color. Middle: An example for *forward prediction* of a more general BN structure. Right: An example for *hybrid inference* of a more general BN structure.

by *sufficient statistics* $(\mu_{\theta_1}(\mathbf{X}), s_{\theta_1}(\mathbf{X}))$ produced by NPN and provides regularization $\frac{\|\mu_{\theta_1}(\mathbf{X}) - v_1\|_2^2}{2s_{\theta_1}(\mathbf{X})}$ for v_1's MAP estimate; $p(v_2|\mathbf{X}, v_1)$ however depends on the *actual value* of v_1, not the NPN representation of v_1, and therefore also guides the MAP estimate of v_1 (see the Supplement for an illustrative figure on this example). Overall, MAP inference is performed by minimizing the following objective $\mathcal{L}_{toy}$ w.r.t. v_1 given $\{\mathbf{X}, v_2\}$:

$$\mathcal{L}_{toy} = \frac{\|\mu_{\theta_1}(\mathbf{X}) - v_1\|_2^2}{2s_{\theta_1}(\mathbf{X})} + \frac{1}{2}\log s_{\theta_1}(\mathbf{X}) + \frac{\|\mu_{\theta_2}(\mathbf{X}, v_1) - v_2\|_2^2}{2s_{\theta_2}(\mathbf{X}, v_1)} + \frac{1}{2}\log s_{\theta_2}(\mathbf{X}, v_1)$$

Inference in Special Cases *In general*, inference can be performed by iterative FF and BP to jointly find V_S using Eqn. 6. *In some special cases* where $v_N \notin V_{-S}$, inference can be sped up by leveraging the structure of the conditional dependencies among variables. Next we consider two such cases (where $v_N \notin V_{-S}$):

Case 1 (Forward Prediction): When the task is to predict $V_S = V \setminus V_k$ given $\mathbf{X}$ and $V_{-S} = V_k$. We can use one-pass FF to infer V_S by greedy maximization as follows:

$$\widehat{v}_n = \underset{v_n}{\operatorname{argmax}}\, p(v_n|\mathbf{X}, V_{n-1}) = \mu_{\theta_n}(\mathbf{X}, \widehat{V}_{n-1}), \quad (7)$$

where $n = k+1, \ldots, N$. Note that although $\widehat{v}_n$ may not be the global optimum due to the terms $\frac{1}{2}\log s_{\theta_n}(\mathbf{X}, V_{n-1})$, in our preliminary experiments, we find that using $\widehat{v}_n$ as initialization and then finetuning jointly V_S to minimize $\mathcal{L}$ (according to Eqn. 6) has very similar performance. In Theorem 1 below, we show that under some assumptions on V, our forward prediction based on greedy maximization can achieve the global optimum, which provides some insight on the similar performance.

Theorem 1. *Assume our ground-truth joint distribution $p(V_N|\mathbf{X})$ is an elliptically unimodal distribution and BIN converges to its optimal.* ***Forward prediction*** *based on greedy maximization in Eqn. 7 achieves the global optimum.*

Case 2 (Hybrid Inference): When $v_N \notin V_{-S}$ and $V_{-S} \neq V_k$ for any k, one can first perform the (joint) general inference for variables $\{v_n|v_n \in V_S, n < q\}$, where v_q is the last variable (with the largest index q) in V_{-S}, by *iterative* FF and BP, and then perform *one-pass* forward prediction for variables $\{v_n|v_n \in V_S, n > q\}$. Doing this could significantly cut down the time needed for predicting $\{v_n|v_n \in V_S, n > q\}$, since no BP and iterative process is needed for them. For example, if $V_S = \{v_1, v_3\}$ and $V_{-S} = \{v_2\}$, one can first perform general inference for v_1 and then perform forward prediction for v_3 (see Fig. 1(left)). Similar to Theorem 1, we show in Theorem 2 that under some assumptions on V, hybrid inference can also achieve the global optimum.

Theorem 2. *Assume our ground-truth joint distribution $p(V_N|\mathbf{X})$ is an elliptically unimodal distribution and BIN converges to its optimal.* ***Hybrid inference*** *can achieve the global optimum for V_S if the backward part achieves the global optimum for $\{v_n|v_n \in V_S, n < q\}$.*

Elliptically unimodal distributions are a broad class of distributions. We provide the definition and some properties as well as the proof of the theorems above in the Supplement.

Remark: Empirically BIN works well even though the conditions in the theorems above do not necessarily hold. Also note that although Case 1 and Case 2 assume the BN structure in Eqn. 3, they can be naturally generalized to other facetorization (BN structure). For example, in a general BN structure, if there exists a topological order of variables in V such that $V_{-S} = V_k$ (i.e., all variables in V_S are descendents of V_{-S}), Case 1 can be applied similarly. See Fig. 1 for some examples of both cases.

Composite Bidirectional Inference Networks

From BIN to Composite BIN: During *inference* of BIN we may need to iteratively update V_S via BP. To reflect this effect during *training* and tailor the model in a manner that works well with the proposed inferential procedure (to further improve the optimization landscape and speed up inference), *composite BIN* (CBIN) augments Eqn. 4 with composite likelihood (CL) terms $\mathcal{L}_j$ and uses the following training objective:

$$\mathcal{L}_{all} = \mathcal{L}(V|\mathbf{X};\boldsymbol{\theta}) + \lambda_c \sum_{j=1}^{J} \mathcal{L}_j,$$

$$\mathcal{L}_j = \mathcal{L}(\widehat{V}_{S_j}, V_{-S_j}|\mathbf{X};\boldsymbol{\theta}) - \mathcal{L}(V_{-S_j}|\mathbf{X};\boldsymbol{\theta}), \quad (8)$$

$$\widehat{V}_{S_j} \approx \underset{V_{S_j}}{\operatorname{argmin}}\, \mathcal{L}(V_{S_j}, V_{-S_j}|\mathbf{X};\boldsymbol{\theta}), \quad (9)$$

where $\mathcal{L}_j$ integrates the inference task of predicting V_{S_j} (S_j is a subset of $\{1,\ldots,N\}$ specified by users) given $(\mathbf{X}, V_{\text{-}S_j})$ into the training process. $\widehat{V}_{S_j}$ is computed in an inner loop during each epoch by iteratively updating V_{S_j} through BP. λ_c is a hyperparameter. Note that Eqn. 9 is an approximation because the inner loop may not yield global optima. To gain more insight, $\mathcal{L}_j$ can be written as

$$\begin{aligned}\mathcal{L}_j &= \mathcal{L}(\widehat{V}_{S_j}|\mathbf{X};\boldsymbol{\theta}) + \mathcal{L}(V_{\text{-}S_j}|\mathbf{X},\widehat{V}_{S_j};\boldsymbol{\theta}) \\ &\quad - \mathcal{L}(V_{\text{-}S_j}|\mathbf{X};\boldsymbol{\theta}), \qquad (10)\\ \mathcal{L}(V_{\text{-}S_j}|\mathbf{X};\boldsymbol{\theta}) &= -\log p(V_{\text{-}S_j}|\mathbf{X}) \\ &= -\log \int p(V_{\text{-}S_j}|\mathbf{X},V_{S_j})p(V_{S_j}|\mathbf{X})dV_{S_j}.\end{aligned}$$

Intuition for the Objective Function: The training process of CBIN is summarized in Algorithm 1. Interestingly, the last two terms in Eqn. 10 can be seen as the difference between (1) the negative log-likelihood of $V_{\text{-}S_j}$ given $\mathbf{X}$ and the inferred $\widehat{V}_{S_j}$ (which is smaller since $\widehat{V}_{S_j}$ contains information from V_{S_j}) and (2) the negative log-likelihood of $V_{\text{-}S_j}$ given only $\mathbf{X}$ (which is larger and serves as a baseline). Hence minimizing the last two terms makes the network aware that $\widehat{V}_{S_j}$ contains additional information and consequently decrease the loss. On the other hand, the first term in Eqn. 10 would update the network so that the prior and initialization provided to infer V_{S_j} can be closer to $\widehat{V}_{S_j}$. From another perspective, $\mathcal{L}_j$ can be seen as adapting the augmented data $(\widehat{V}_{S_j}, V_{\text{-}S_j})$ according to the training process to make the optimization landscape of V_{S_j} more friendly to inference via BP (see the next section for more details and experiments). It is also worth noting that in Eqn. 8, $\mathcal{L}_j = \mathcal{L}(\widehat{V}_{S_j}|\mathbf{X}, V_{\text{-}S_j};\boldsymbol{\theta})$ can be considered as a composite likelihood (a generalized version of pseudolikelihood (Besag 1974)) term (Lindsay 1988) given $\mathbf{X}$ if $\widehat{V}_{S_j}$ is replaced with V_{S_j}. It is proven that adding CL terms does not bias the learning of parameters (Lindsay 1988).

Configuration of V_{S_j}: One challenge, however, 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**: The inner loop inferring $\widehat{V}_j$ given $V \setminus V_j$ can be seen as searching for the best path for correcting $\widehat{V}_j$. (2) **Optimization landscape**: The generated $\widehat{V}_j$ is used as input for $N-j$ subnetworks; hence these $N-1$ extra terms are sufficient to cover N subnetworks and improve their optimization landscape. For more details please refer to the Supplement.

Remark: As shown in the experiments below, the inclusion of CL terms in CBIN not only improves the accuracy and generalization but also leads to faster inference. We attribute this to the preference of CBIN to learn smoother (and consequently more generalizable) optimization landscape w.r.t. V (see the optimization landscape in the Supplement and Fig. 2). Note that the marginal negative log-likelihood $\mathcal{L}(V_{\text{-}S_j}|\mathbf{X};\boldsymbol{\theta})$ in Eqn. 8 can be approximated efficiently leveraging the properties of NPN (see the Supplement for details).

In the Supplement, we prove that for single-layer NPN subnetworks, the approximation process can obtain the exact mean and variance (diagonal entries of the covariance matrix) of $p(V_{\text{-}S_j}|\mathbf{X};\boldsymbol{\theta})$.

Algorithm 1 Learning CBIN

Input: Data $\mathcal{D} = \{(\mathbf{X}^{(i)}, V^{(i)}\}_{i=1}^{M}$, training iterations T_t, warmup iterations T_w, inference iterations T_{in}, learning rate ρ_t, step size γ_t, initialized model parameter $\boldsymbol{\theta} = \{(\mu_{\theta_n}(\cdot), s_{\theta_n}(\cdot))\}_{n=1}^{N}$. A minibatch of size K for each iteration is denoted as $\{(\mathbf{X}^{(k)}, V^{(k)})\}_{k\in\{i_1,\ldots,i_K\}}$.
for $t = 1 : T_w$ **do**
  Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\rho_t}{K}\sum_k \nabla_{\boldsymbol{\theta}}\mathcal{L}(V^{(k)}|\mathbf{X}^{(k)};\boldsymbol{\theta})$ with $\mathcal{L}$ in Eqn. 4 as a loss function.
end for
for $t = 1 : T_t$ **do**
  for $j = 1 : J$ **do**
    Initialize $\{\widehat{V}_{S_j}^{(k)}\}$ of the current minibatch for the CL term $\mathcal{L}_j$ using Eqn. 7.
    for $t_{in} = 1 : T_{in}$ **do**
      Update $\{\widehat{V}_{S_j}^{(k)}\}$ via FF and BP: $\widehat{V}_{S_j}^{(k)} \leftarrow \widehat{V}_{S_j}^{(k)} - \gamma_{t_{in}}\nabla_{\widehat{V}_{S_j}^{(k)}}\mathcal{L}(\widehat{V}_{S_j}^{(k)}, V_{\text{-}S_j}^{(k)}|\mathbf{X}^{(k)};\boldsymbol{\theta})$.
    end for
  end for
  Update parameters: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\rho_t}{K}\sum_k \nabla_{\boldsymbol{\theta}}\mathcal{L}_{all}$ with the total loss $\mathcal{L}_{all}$ defined in Eqn. 8.
end for

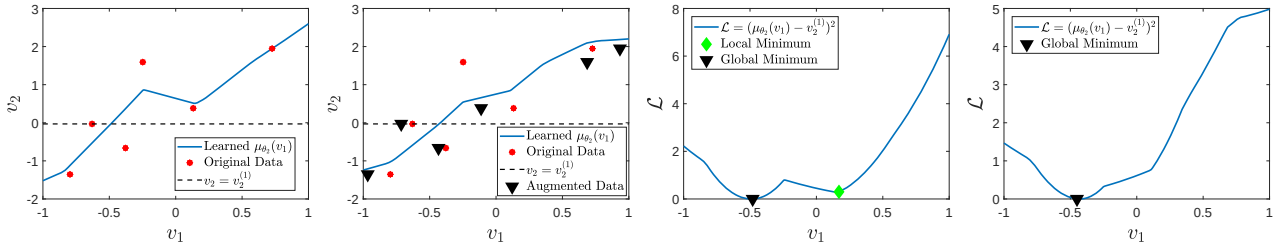
Experiments

In this section, we first compare BIN and CBIN in toy datasets to gain more insight about our models and then evaluate variants of BIN/CBIN and other state-of-the-art methods on two real-world datasets. In all tables below, we mark the best results without retraining new models in **bold** and the best results with or without retraining new models by underlining.

Toy Inference Tasks

We start with toy datasets and toy inference tasks to show that the composite likelihood terms in CBIN help shape the function we learned to be *smoother* and consequently reduce the number of local optima during inference. In the first toy dataset, $\mathbf{X}$ is ignored and $V = \{v_1, v_2\}$. We generate 6 data points $\{(v_1^{(i)}, v_2^{(i)})\}_{i=1}^{6}$ according to $v_2 = 3v_1 + 1 + \epsilon$, where $\epsilon \sim \mathcal{N}(0,1)$ and v_1 is sampled from a uniform distribution $\mathcal{U}(0,1)$. We train BIN according to Eqn. 4 and CBIN according to Eqn. 8 with $J = 1$ and $V_{S_1} = \{v_1\}$ (see the Supplement for details).

Fig. 2(a) and Fig. 2(b) show the $\mu_{\theta_2}(v_1)$ learned by BIN and CBIN, respectively, with the original training data points. Correspondingly, Fig. 2(c) and Fig. 2(d) show the loss surface of $\mathcal{L}$ with respect to $V_S = \{v_1\}$ when inferring v_1 given $v_2^{(1)}$ ($v_2^{(1)}$ corresponds to the dashed line).



(a) BIN (b) CBIN (c) BIN (d) CBIN

Figure 2: (a) and (b): $\mu_{\theta_2}(v_1)$ learned by BIN and CBIN. (c) and (d): corresponding loss surface of $\mathcal{L}$ with respect to $\{v_1\}$ when inferring v_1 given $v_2^{(1)}$ in BIN and CBIN.

Table 1: 8 variables in the *SHHS2* dataset.

v_1	Physical functioning
v_2	Role limitation due to physical health
v_3	General health
v_4	Role limitation due to emotional problems
v_5	Energy/fatigue
v_6	Emotional well being
v_7	Social functioning
v_8	Pain

During inference, BIN searches for the lowest point in the loss surface of Fig. 2(c) and may be trapped in the poor local optimum (e.g., the diamond point in Fig. 2(c)). In contrast, CBIN can alleviate this problem, as shown in Fig. 2(d), where there is no poor local optimum. It would be interesting to inspect $\widehat{v}_1$ generated in the inner loop when training CBIN. Fig. 2(b) plots augmented points $(\widehat{v}_1, v_2)$ generated in the last inner loop iteration as triangles. As can be seen, the augmented points concentrate near a straight line and hence help learn a smoother function $\mu_{\theta_2}(v_1)$ from v_1 to v_2, leading to a *smoother loss surface* that is more friendly to gradient-based inference (shown in Fig. 2(d)). Besides this toy inference task, we also examine a more complex toy dataset where $\mathbf{X}$ is also considered and obtain similar results (see the Supplement for details).

Experiments on the *SHHS2* Dataset

Besides synthetic datasets, we also evaluate BIN and CBIN on the sleep heart health study 2 (*SHHS2*) dataset. *SHHS2* contains full-night Polysomnography (PSG) from 2,651 subjects. Available PSG signals include Electroencephalography (EEG), Electrocardiography (ECG), and breathing signals (airflow, abdomen, and thorax). For each subject, the dataset also includes the 36-Item Short Form Health Survey (SF-36) (Ware Jr and Sherbourne 1992). SF-36 is a standard survey that is widely used to extract 8 health variables (as shown in Table 1). Each of the 8 variables is represented using a score in $[0, 100]$. In the experiments, we consider PSG as high-dimensional information $\mathbf{X}$ and the 8 scores $\{v_n\}_{n=1}^8$ as attributes of interest. Since the scores are based on subjects' self-reported results and intrinsically noisy, preprocessing is necessary. In particular, we use the mean of each score over all subjects as the threshold to binarize the scores into $\{0, 1\}$, where 0 indicates 'unhealthy' and 1 indicates 'healthy'. Here $\mathbf{X}$ is dense, high-dimensional, variable-length signals that consist of EEG spectrograms $\mathbf{X}_e \in \mathbb{R}^{64\times l^{(i)}}$, ECG spectrograms $\mathbf{X}_c \in \mathbb{R}^{22\times l^{(i)}}$, and breathing $\mathbf{X}_b \in \mathbb{R}^{3\times 10l^{(i)}}$, where $l^{(i)}$ is the number of seconds for the i-th subject (in the range $7278 \sim 45448$).

We compare our models with the following baselines: **'Prior Only' (PO)** refers to using only the prior part of the trained BIN to directly output predictions without iterative inference during testing (e.g., use $\mu_{\theta_1}(\mathbf{X})$ from the first subnetwork as predictions when $V_S = \{v_1\}$ and $V_{\setminus S} = \{v_2\}$). **'Random Initialization' (RI)** refers to randomly initializing V_S before iterative inference instead of using the prior's output during testing (e.g., during the inference for $V_S = \{v_1\}$ given $V_{\setminus S}\{v_2\}$, use random initialization rather than $\mu_{\theta_1}(\mathbf{X})$). **SPEN** refers to the structured prediction energy networks and **eSPEN** is its end-to-end variant (Belanger and McCallum 2016; Belanger, Yang, and McCallum 2017). **SVAE** refers to combining SVAE (Johnson et al. 2016; Kingma and Welling 2013) and our method to enable BP-based inference and avoid $O(2^N)$ networks (see the Supplement for details). **DNADE** is the orderless and deep Neural Autoregressive Distribution Estimation (NADE) (Larochelle and Murray 2011) proposed in (Uria et al. 2016). It is combined with the real-valued NADE (Uria, Murray, and Larochelle 2013; Uria et al. 2016) for the regression task. **'Retrain'** means retraining a model for that specific inference task (i.e., retraining an end-to-end NPN with $\mathbf{X}$ and $V_{\setminus S}$ as input and V_S as output). Note that the original SPEN and eSPEN *can only predict* V *given* $\mathbf{X}$. We adapt them for different inference tasks by resetting $V_{\setminus S}$ to the given values in each inference iteration (essentially learn a density estimator $f(V, \mathbf{X})$ and optimize $\max_{V_S} f(V_S, V_{\setminus S}, \mathbf{X})$). Please refer to the Supplement for details on hyperparameters and model training of BIN, CBIN, and all baselines.

Table 2 shows the accuracy of predicting different V_S given $V_{\setminus S}$ when $V = V_8$ (see the Supplement for additional results when $V = V_3$) for BIN, CBIN, and the baselines, where the accuracy is averaged across all inferred variable. As we can see: (1) BIN significantly outperforms 'Prior Only', verifying the effectiveness of the iterative inference process. (2) BIN also outperforms 'Random Initialization', which verifies the effectiveness of the initialization provided by the prior part (e.g., $\mu_{\theta_1}(\mathbf{X})$ when $V_S = \{v_1\}$ and $V_{\setminus S} = \{v_2\}$). (3) Furthermore, CBIN consistently out-

Table 2: Accuracy (%) for predicting V_S given $\mathbf{X}$ and $V_{\setminus S} = \{v_n\}_{n=1}^{8} \setminus V_S$ in the *SHHS2* dataset.

V_S	$\{v_1, v_3\}$	$\{v_4, v_5\}$	$\{v_1, v_3, v_6, v_7\}$	$\{v_2, v_6, v_7\}$	$\{v_3, v_5, v_8\}$	$\{v_4, v_5, v_6\}$	$\{v_4, v_6, v_7\}$
SPEN	63.78	70.21	64.29	65.72	59.84	66.51	70.26
eSPEN	64.39	71.13	64.22	65.63	61.17	66.92	69.95
SVAE	62.07	69.59	62.51	65.13	59.73	66.03	68.32
DNADE	69.83	74.32	67.58	68.88	64.77	68.15	72.39
PO	68.39	76.29	70.98	75.00	71.52	69.75	74.00
RI	68.08	68.06	66.26	67.22	63.48	65.80	66.62
BIN	75.31	79.07	73.48	75.00	72.55	**74.36**	76.11
CBIN	**77.16**	**80.22**	**75.21**	**75.87**	**72.68**	73.85	**76.31**
Retrain	75.92	79.65	74.94	76.04	72.24	73.58	75.74

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

V_S	$\{v_1\}$	$\{v_2\}$	$\{v_1, v_2\}$	$\{v_1, v_3\}$
SPEN	0.0973	0.1310	0.1163	0.2464
eSPEN	0.0944	0.1243	0.1138	0.2401
SVAE	0.0998	0.1373	0.1236	0.2489
DNADE	0.0828	0.1179	0.1122	0.2321
PO	0.0979	0.1071	0.1113	0.2395
RI	0.0787	0.1301	0.1248	0.2333
BIN	0.0691	0.1087	0.1069	0.2292
CBIN	**0.0643**	**0.1062**	**0.1011**	**0.2130**
Retrain	0.0714	0.1059	0.1058	0.2271

Table 4: RMSE when $V_S = V$ for the *Dermatology* dataset.

V_S	$\{v_1\}$	$\{v_1, v_2\}$	$\{v_1, v_2, v_3\}$
SPEN	-	0.1135	0.2148
eSPEN	-	0.1120	0.2109
SVAE	-	0.1132	0.2086
DNADE	-	0.1185	0.2161
BIN	-	0.1118	0.2010
CBIN	-	**0.1098**	**0.1967**
Retrain	0.0950	0.1144	0.2059

performs BIN, which means the CL terms (and inner loops during training) are helpful to shape the optimization landscape with respect to V_S so that the iterative inference process reaches better local optima (or even global optima). (4) Interestingly in most cases, CBIN can even outperform a new model trained for the specific V_S and $V_{\setminus S}$ possibly because CBIN (and BIN) can take into account the conditional dependency among different variables while the specifically retrained model cannot. (5) SVAE, DNADE, SPEN, and eSPEN perform poorly since they are not designed for arbitrary inference tasks or fail to properly model conditional dependencies. Besides, we also compare the accuracy of different methods in *forward prediction* cases where $V_S = V$ with different V. We find that SVAE, SPEN, and eSPEN achieve similar or slightly better accuracy than the retrained specific models while BIN and CBIN outperform all the above baselines (see the Supplement for more experimental results).

Fig. 3(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. 3(right)

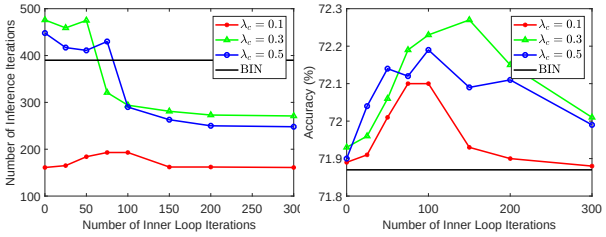


Figure 3: Left: Number of inference iterations needed during testing versus number of inner loop iterations during training (T_{in} in Algorithm 1) with different λ_c. The horizontal line the number for BIN (without $\mathcal{L}_j$). Right: Accuracy versus T_{in}. Similarly the horizontal line shows the accuracy of the corresponding BIN.

shows the corresponding accuracy versus T_{in}. The horizontal lines show the number of inference iterations and accuracy 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. 3.

Experiments on the *Dermatology* Dataset

Besides classification tasks, we also evaluate our methods on regression tasks using the *Dermatology* dataset, which contains 12 clinical features (e.g., itching, erythema, age, etc.) and 21 histopathological attributes (e.g., saw-tooth appearance of retes) (Lichman 2013) of 366 subjects. In the experiments, we use all 12 clinical features as $\mathbf{X}$ with 'vacuolisation and damage of basal layer', 'saw-tooth appearance of retes', and 'elongation of the rete ridges' as attributes of interest (v_1, v_2, and v_3). For details on hyperparameters and model training, please refer to the Supplement.

Similar to the *SHHS2* experiments, Table 3 shows the Root Mean Square Error (RMSE) of predicting v_1, v_2, and v_3 with different V_S for BIN, CBIN, baselines, and retraining a model for that specific inference task. Table 4 shows the RMSE in *forward prediction* cases where $V_S = V$ with different V. The results and conclusions are consistent with those in the *SHHS2* experiments.

Conclusion

In this paper, we propose BIN to connect multiple probabilistic neural networks in an organized way so that each network models a conditional dependency among variables. We further extend BIN to CBIN, involving the iterative inference process in the training stage and improving both accuracy and computational efficiency. Experiments on real-world healthcare datasets demonstrate that BIN/CBIN can achieve state-of-the-art performance in the arbitrary inference tasks with *a single model*. As future work it would be interesting to evaluate different factorizations for the joint likelihood of variables and different distributions (e.g., gamma distributions) beyond Gaussians. It would also be interesting to extend the models to handle hidden variables and missing values.

Acknowledgments

The authors thank Xingjian Shi, Vikas Garg, Jonas Mueller, Hongyi Zhang, Guang-He Lee, Yonglong Tian, other members of NETMIT and CSAIL, and the reviewers for their insightful comments and helpful discussion.

References

Archer, E.; Park, I. M.; Buesing, L.; Cunningham, J.; and Paninski, L. 2015. Black box variational inference for state space models. *CoRR* abs/1511.07367.

Bayer, J., and Osendorfer, C. 2014. Learning stochastic recurrent networks. *CoRR* abs/1411.7610.

Belanger, D., and McCallum, A. 2016. Structured prediction energy networks. In *ICML*, 983–992.

Belanger, D.; Yang, B.; and McCallum, A. 2017. End-to-end learning for structured prediction energy networks. In *ICML*, 429–439.

Besag, J. 1974. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society B* 192–236.

Chung, J.; Kastner, K.; Dinh, L.; Goel, K.; Courville, A. C.; and Bengio, Y. 2015. A recurrent latent variable model for sequential data. In *NIPS*, 2980–2988.

Fraccaro, M.; Sønderby, S. K.; Paquet, U.; and Winther, O. 2016. Sequential neural models with stochastic layers. In *NIPS*, 2199–2207.

Gatys, L. A.; Ecker, A. S.; Bethge, M.; Hertzmann, A.; and Shechtman, E. 2017. Controlling perceptual factors in neural style transfer. In *CVPR*, 3730–3738.

Gatys, L. A.; Ecker, A. S.; and Bethge, M. 2016. Image style transfer using convolutional neural networks. In *CVPR*, 2414–2423.

Germain, M.; Gregor, K.; Murray, I.; and Larochelle, H. 2015. MADE: masked autoencoder for distribution estimation. In *ICML*, 881–889.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *CoRR* abs/1412.6572.

Gregor, K.; Danihelka, I.; Graves, A.; Rezende, D. J.; and Wierstra, D. 2015. DRAW: A recurrent neural network for image generation. In *ICML*, 1462–1471.

Johnson, M.; Duvenaud, D. K.; Wiltschko, A.; Adams, R. P.; and Datta, S. R. 2016. Composing graphical models with neural networks for structured representations and fast inference. In *NIPS*, 2946–2954.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *CoRR* abs/1312.6114.

Krishnan, R. G.; Shalit, U.; and Sontag, D. 2015. Deep kalman filters. *CoRR* abs/1511.05121.

Larochelle, H., and Murray, I. 2011. The neural autoregressive distribution estimator. In *AISTATS*, 29–37.

Le, Q. V., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *ICML*, 1188–1196.

Lichman, M. 2013. UCI machine learning repository.

Lin, W.; Khan, M. E.; and Hubacher, N. 2018. Variational message passing with structured inference networks. In *ICLR*.

Lindsay, B. G. 1988. Composite likelihood methods. *Contemporary Mathematics* 80(1):221–239.

Quan, S. F.; Howard, B. V.; Iber, C.; Kiley, J. P.; Nieto, F. J.; O'connor, G. T.; Rapoport, D. M.; Redline, S.; Robbins, J.; Samet, J. M.; et al. 1997. The sleep heart health study: design, rationale, and methods. *Sleep* 20(12):1077–1085.

Sesen, M. B.; Nicholson, A. E.; Banares-Alcantara, R.; Kadir, T.; and Brady, M. 2013. Bayesian networks for clinical decision support in lung cancer care. *PloS one* 8(12):e82349.

Sohn, K.; Lee, H.; and Yan, X. 2015. Learning structured output representation using deep conditional generative models. In *NIPS*, 3483–3491.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2013. Intriguing properties of neural networks. *CoRR* abs/1312.6199.

Uria, B.; Côté, M.; Gregor, K.; Murray, I.; and Larochelle, H. 2016. Neural autoregressive distribution estimation. *JMLR* 17:205:1–205:37.

Uria, B.; Murray, I.; and Larochelle, H. 2013. RNADE: the real-valued neural autoregressive density-estimator. In *NIPS*, 2175–2183.

Wang, H., and Yeung, D. 2016. Towards Bayesian deep learning: A framework and some existing methods. *TKDE* 27(5):1343–1355.

Wang, H.; Shi, X.; and Yeung, D.-Y. 2016. Natural-parameter networks: A class of probabilistic neural networks. In *NIPS*, 118–126.

Wang, H.; Shi, X.; and Yeung, D. 2017. Relational deep learning: A deep latent variable model for link prediction. In *AAAI*, 2688–2694.

Wang, H.; Wang, N.; and Yeung, D. 2015. Collaborative deep learning for recommender systems. In *SIGKDD*, 1235–1244.

Ware Jr, J. E., and Sherbourne, C. D. 1992. The MOS 36-item short-form health survey (SF-36): I. conceptual framework and item selection. *Medical care* 473–483.

Zhang, F.; Yuan, N. J.; Lian, D.; Xie, X.; and Ma, W. 2016. Collaborative knowledge base embedding for recommender systems. In *KDD*, 353–362.

Zhang, H.; Cissé, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017. mixup: Beyond empirical risk minimization. *CoRR* abs/1710.09412.