VARIATIONAL AUTOENCODERS WITH JOINTLY OPTIMIZED LATENT DEPENDENCY STRUCTURE

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a method for learning the dependency structure between latent variables in deep latent variable models. Our general modeling and inference framework combines the complementary strengths of deep generative models and probabilistic graphical models. In particular, we express the latent variable space of a variational autoencoder (VAE) in terms of a Bayesian network with a learned, flexible dependency structure. The network parameters, variational parameters as well as the latent topology are optimized simultaneously with a single variational objective. Inference is formulated via a sampling procedure that produces expectations over latent variable structures and incorporates top-down and bottom-up reasoning over latent variable values. We validate our framework in extensive experiments on MNIST, Omniglot, and CIFAR-10. Comparisons to state-of-the-art structured variational autoencoder baselines show improvements in terms of the expressiveness of the learned model.

1 INTRODUCTION

Deep latent variable models offer an effective method for automatically learning structure from data. By explicitly modeling the data distribution using latent variables, these models are capable of learning compressed representations that are then relevant for downstream tasks. Such models have been applied across a wide array of domains, such as images (Gregor et al., 2014; Kingma & Welling, 2014; Rezende et al., 2014; Gregor et al., 2015), audio (Chung et al., 2015; Fraccaro et al., 2016), video (He et al., 2018; Yingzhen & Mandt, 2018), and text (Bowman et al., 2016; Krishnan et al., 2017). However, despite their success, latent variable models are often formulated with simple (e.g. Gaussian) distributions, making independence assumptions among the latent variables. That is, each latent variable is sampled independently. Ignoring the dependencies between latent variables limits the flexibility of these models, negatively impacting the model's ability to fit the data.

Latent dependencies can be incorporated both during inference as well as in the model itself. For instance, normalizing flows (Rezende & Mohamed, 2015) accounts for latent dependencies during inference by learning a mapping from a simple distribution to a more complex distribution that contains these dependencies. On the modeling side, one can directly add dependencies by constructing a *hierarchical* latent representation (Dayan et al., 1995). These structures consist of conditional (*empirical*) priors, in which one latent variable forms a prior on another latent variable. While this conditional distribution may take a simple form, marginalizing over the parent variable can result in an arbitrarily complex distribution. Models with these more flexible latent dependency structures have been shown to result in improved performance (Sønderby et al., 2016; Burda et al., 2016; Kingma et al., 2016). However, despite the benefits of including additional structure in these models, latent dependency structures have been predefined, potentially limiting the performance of this approach.

In this work, we propose a method for learning dependency structures in latent variable models. Structure learning is a difficult task with a long history in the graphical models community (Koller & Friedman, 2009). Over the years, it has been tackled from several perspectives, including constraint-based approaches (Cheng et al., 2002; Lehmann & Romano, 2008), optimization of structure scores (Kass & Raftery, 1995; Heckerman et al., 1995; Barron et al., 1998), Bayesian model averaging (Heckerman et al., 1999; Koivisto & Sood, 2004), and many more. Unfortunately, the underlying objectives are often limited to graphs of a particular form (e.g., limited tree width),

prohibitively expensive, or difficult to integrate with the gradient-based optimization techniques of modern neural networks. Here, we discuss a variational end-to-end approach for general graph structures introducing minimal complexity overhead. In particular, we introduce a set of binary global variables to gate the latent dependencies. The whole model (including its structure) is jointly optimized with a single stochastic variational inference objective. In our experimental validation, we show that the learned dependency structures contribute to a more accurate representation of the true generative distribution, outperforming several other variants of variational autoencoders.

2 BACKGROUND

2.1 VARIATIONAL INFERENCE & VARIATIONAL AUTOENCODERS

A latent variable model, defined by the joint distribution, $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$, models each data example, $\mathbf{x}$, using a local latent variable, $\mathbf{z}$, and global parameters, θ. $p_\theta(\mathbf{x}|\mathbf{z})$ denotes the conditional likelihood, and $p_\theta(\mathbf{z})$ denotes the prior. Latent variable models are capable of capturing the structure present in data, with $\mathbf{z}$ forming a compressed representation of each data example. Unfortunately, inferring the posterior, $p_\theta(\mathbf{z}|\mathbf{x})$, is typically computationally intractable, prompting the use of approximate inference techniques. Variational inference (Jordan et al., 1999) introduces an approximate posterior, $q_\phi(\mathbf{z}|\mathbf{x})$, and optimizes variational parameters, ϕ, to minimize the KL-divergence to the true posterior, $\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$. As this quantity cannot be evaluated directly, the following relation is used:

$$\log p_\theta(\mathbf{x}) = \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) + \mathcal{L}(\mathbf{x}; \theta, \phi), \tag{1}$$

where $\mathcal{L}(\mathbf{x}; \theta, \phi)$ is the evidence lower bound (ELBO), defined as

$$\mathcal{L}(\mathbf{x}; \theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})). \tag{2}$$

In Eq. (1), $\log p_\theta(\mathbf{x})$ is independent of ϕ, so we can minimize the KL divergence term, i.e. perform approximate inference, by maximizing $\mathcal{L}(\mathbf{x}; \theta, \phi)$ w.r.t. $q_\phi(\mathbf{z}|\mathbf{x})$. Further, because KL divergence is non-negative, $\mathcal{L}(\mathbf{x}; \theta, \phi)$ is a lower bound on $\log p_\theta(\mathbf{x})$, meaning we can then learn the model parameters by maximizing $\mathcal{L}(\mathbf{x}; \theta, \phi)$ w.r.t. θ.

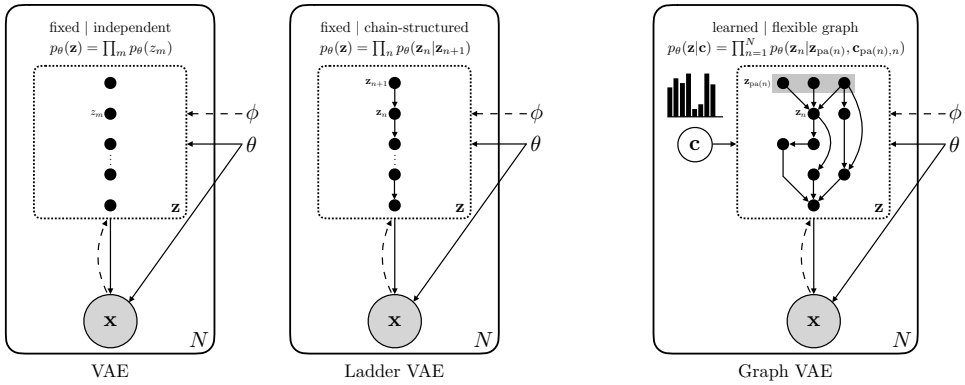
Variational autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014) amortize inference optimization across data examples by parameterizing $q_\phi(\mathbf{z}|\mathbf{x})$ as a separate inference model, then jointly optimizing the model parameters θ and ϕ. VAEs instantiate both the inference model and latent variable model with deep networks, allowing them to scale to high-dimensional data. However, VAEs are typically implemented with basic graphical structures and simple, unimodal distributions (e.g. Gaussians). For instance, the dimensions of the prior are often assumed to be independent, $p_\theta(\mathbf{z}) = \prod_m p_\theta(z_m)$, with a common assumption being a fixed standard Gaussian: $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. Similarly, approximate posteriors often make the mean field assumption, $q_\phi(\mathbf{z}|\mathbf{x}) = \prod_m q_\phi(z_m|\mathbf{x})$. Independence assumptions such as these may be overly restrictive, thereby limiting modeling capabilities.

2.2 EMPIRICAL PRIORS THROUGH LATENT DEPENDENCY STRUCTURE

One technique for improving the expressive capacity of latent variable models is through incorporating dependency structure among the latent variables, forming a hierarchy (Dayan et al., 1995; Rezende et al., 2014). These dependencies provide *empirical* priors, learned priors that are conditioned on other latent variables. With M latent dimensions, the full prior takes the following auto-regressive form:

$$p_\theta(\mathbf{z}) = \prod_{m=1}^{M} p_\theta(z_m|\mathbf{z}_{\mathrm{pa}(m)}), \tag{3}$$

where $\mathbf{z}_{\mathrm{pa}(m)}$ denotes the vector of latent variables constituting the parents of z_m. Each conditional distribution can be parameterized by deep networks that output the parameters of distributions, e.g. mean and variance of a Gaussian. While these conditional distributions may be relatively simple, the marginal empirical prior, $p_\theta(z_m) = \int p_\theta(z_m|\mathbf{z}_{\mathrm{pa}(m)})p_\theta(\mathbf{z}_{\mathrm{pa}(m)})\mathrm{d}\mathbf{z}_{\mathrm{pa}(m)}$, can be arbitrarily complex. By using more flexible priors, models with latent dependency structure are less restrictive in their latent representations, hopefully capturing more information and enabling a better fit to the data.



(a) Traditional Latent Variable Models (b) Proposed Model

Figure 1: **Overview: Model Comparison.** We show the graphical representations of **(a)** traditional latent variable models (VAE, ladder VAE) and **(b)** the proposed graph VAE. Solid lines denote generation, dashed lines denote inference, and the dotted area indicates the latent space governed by variational parameters ϕ and generative parameters θ. Both VAE and ladder VAE use a fixed graph structure with limited expressiveness (VAE: independent; ladder VAE: chain-structured). In contrast, graph VAE jointly optimizes a distribution over latent structures $\mathbf{c}$ and model parameters (ϕ,θ), allowing test-time sampling of a flexible, data-driven latent structure.

With the added latent dependencies in the model, the independence assumption in the approximate posterior is even less valid in this setting. While normalizing flows (Rezende & Mohamed, 2015) offers one technique for overcoming the mean field assumption, a separate line of work has investigated the use of *structured* approximate posteriors, particularly in the context of models with empirical priors (Johnson et al., 2016). This technique introduces dependencies between the dimensions of the approximate posterior, often mirroring the dependency structure of the latent variable model. An explanation for this was provided by Marino et al. (2018): optimizing the approximate posterior requires knowledge of the prior, which is especially relevant in models with empirical priors where the prior can vary with the data. Ladder VAE (Sønderby et al., 2016) incorporates these prior dependencies by using a structured approximate posterior of the form

$$q_\phi(\mathbf{z}|\mathbf{x}) = \prod_{m=1}^{M} q_\phi(z_m|\mathbf{x}, \mathbf{z}_{\text{pa}(m)}). \tag{4}$$

Unlike the mean field approximate posterior, which conditions each dimension only on the data example, $\mathbf{x}$, the distributions in Eq. (4) account for latent dependencies by conditioning on samples from the parent variables. Ladder VAE performs this conditioning by reusing the empirical prior during inference, forming the approximate posterior by combining a "bottom-up" recognition distribution and the "top-down" prior.

While Eqs. (3) and (4) permit separate latent dependencies for each individual latent dimension, the dimensions are typically partitioned into a set of nodes, with dimensions within each node sharing the same parents. This improves computational efficiency by allowing priors and approximate posteriors within each node to be calculated in parallel. Using $\mathbf{z}_n$ to denote latent node n of N and $\mathbf{z}_{\text{pa}(n)}$ to denote the concatenation of its parent nodes, we can write the ELBO (Eq. (2)) as

$$\mathcal{L}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \sum_{n=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_\phi(\mathbf{z}_n|\mathbf{x}, \mathbf{z}_{\text{pa}(n)})}{p_\theta(\mathbf{z}_n|\mathbf{z}_{\text{pa}(n)})}\right]. \tag{5}$$

Note that the KL divergence term in the ELBO can no longer be evaluated analytically, now requiring a sampling-based estimate of the expectation (Kingma & Welling, 2014). While this can lead to higher variance in ELBO estimates and the resulting gradients, models with latent dependencies still tend to empirically outperform models with independence assumptions (Burda et al., 2016; Sønderby et al., 2016). However, by increasing the number of nodes, the burden of devising a suitable dependency structure falls upon the experimental practitioner. This is non-trivial, as the structure may depend on the data and other model hyperparameters, such as the number of layers

in the deep networks, non-linearities, latent distributions, etc. Rather than relying on pre-defined fully-connected structures (Kingma et al., 2016) or chain structures (Sønderby et al., 2016), we seek to automatically learn the latent dependency structure as part of the variational optimization process. A comparison of these approaches is visualized in Fig. 1.

3 Variational Optimization of Latent Structures

A major difficulty in the context of variational optimization of latent structures is the fact that, unlike the model parameters (ϕ, θ), which are optimized over a continuous domain, the latent dependency structure is *discrete*, without a clear ordering. While the discrete nature of the latent space's topological structure introduces discontinuities in the optimization landscape, complicating the learning process, there is (unlike the related setting of neural architecture search (Zoph & Le, 2016)) fortunately only a finite number of possible dependency structures over a fixed number of latent dimensions: In a directed graphical model, a fully-connected directed acyclic graph (DAG) models all possible dependencies. In this model, an ordering is induced over the latent nodes, and the parents of node n (of N) are given as $\mathbf{z}_{\text{pa}(n)} = \{\mathbf{z}_{n+1}, \ldots, \mathbf{z}_N\}$.[1] Thus, to learn a latent dependency structure, we can maintain all dependencies in a fully-connected DAG, modifying their presence or absence during training. Naively, one might assume that a fully-connected DAG is the most general structure, and, therefore, current models should implicitly learn to ignore unnecessary dependencies. However, latent variable models are highly prone to local optima (Bowman et al., 2016; Burda et al., 2016), and as we show empirically in Section 4, modifying entire dependencies can yield improved performance.

3.1 Gated Dependencies

To control the dependency structure of the model, we introduce a set of binary global variables, $\mathbf{c} = \{c_{i,j}\}_{i,j}$, which *gate* the latent dependencies. The element $c_{i,j}$ denotes the gate variable from $\mathbf{z}_i$ to $\mathbf{z}_j$ $(i > j)$, specifying the presence or absence of this latent dependency. We treat each $c_{i,j}$ as an independent random variable, sampled from a Bernoulli distribution with mean $\mu_{i,j}$, i.e. $c_{i,j} \sim \mathcal{B}(\mu_{i,j})$. We denote the set of these Bernoulli means as $\boldsymbol{\mu}$. Because each element of $\mathbf{c}$ takes values in $\{0, 1\}$, dependencies can be preserved or removed simply through (broadcasted) element-wise multiplication with the corresponding parent nodes. Removing a dependency entails multiplying the corresponding input parent node by 0.

With this addition, the model is now expressed as $p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{c}) = p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{c}) p_\theta(\mathbf{z}|\mathbf{c}) p(\mathbf{c})$. Whereas the model parameters, θ, are point estimates, the dependency gates are governed by Bernoulli distributions, with each sampling from $p(\mathbf{c})$ inducing a corresponding dependency structure. Similar to Eq. (3), the prior can now be expressed as

$$p_\theta(\mathbf{z}|\mathbf{c}) = \prod_{n=1}^{N} p_\theta(\mathbf{z}_n | \mathbf{z}_{\text{pa}(n)}, \mathbf{c}_{\text{pa}(n),n}), \tag{6}$$

where $\mathbf{c}_{\text{pa}(n),n}$ denotes the gate variables associated with the dependencies between node $\mathbf{z}_n$ and its parents, $\mathbf{z}_{\text{pa}(n)}$. Note that $\mathbf{z}_{\text{pa}(n)}$ denotes the set of all *possible* parents of node $\mathbf{z}_n$ in the fully-connected DAG, i.e. $\mathbf{z}_{\text{pa}(n)} = \{\mathbf{z}_{n+1}, \ldots, \mathbf{z}_N\}$. To give a concrete example of the gating procedure, consider the case in which $p_\theta(\mathbf{z}_n | \mathbf{z}_{\text{pa}(n)}, \mathbf{c}_{\text{pa}(n),n})$ is given by a Gaussian density with parameters $\widehat{\boldsymbol{\psi}}_n = (\widehat{\boldsymbol{\mu}}_n, \widehat{\boldsymbol{\Sigma}}_n)$. We obtain these parameters recursively by multiplying samples of node $\mathbf{z}_n$'s parent variables $\mathbf{z}_{\text{pa(n)}}$ with samples of their corresponding gating variables $\boldsymbol{c}_{\text{pa(n),n}}$ and inject a concatenation of the results of this operation into a multi-layer perceptron $\text{MLP}_n^{(TD)}$ predicting $\widehat{\boldsymbol{\psi}}_n$ (see Appendix A.2 for additional details on the MLP architecture). The top-down recursion ends at the root node $\mathbf{z}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. An illustration of this process is shown in black in Fig. 2.

The approximate posterior, $q_\phi(\mathbf{z}|\mathbf{x})$, must approximate $p_\theta(\mathbf{z}|\mathbf{x})$, which is marginalized across $\mathbf{c}$. In effect, $q_\phi(\mathbf{z}|\mathbf{x})$ should account for the entire distribution of dependency structures, rather than any

[1] We follow convention (Dayan et al., 1995; Rezende et al., 2014; Sønderby et al., 2016), with parent nodes having a larger index than their children.

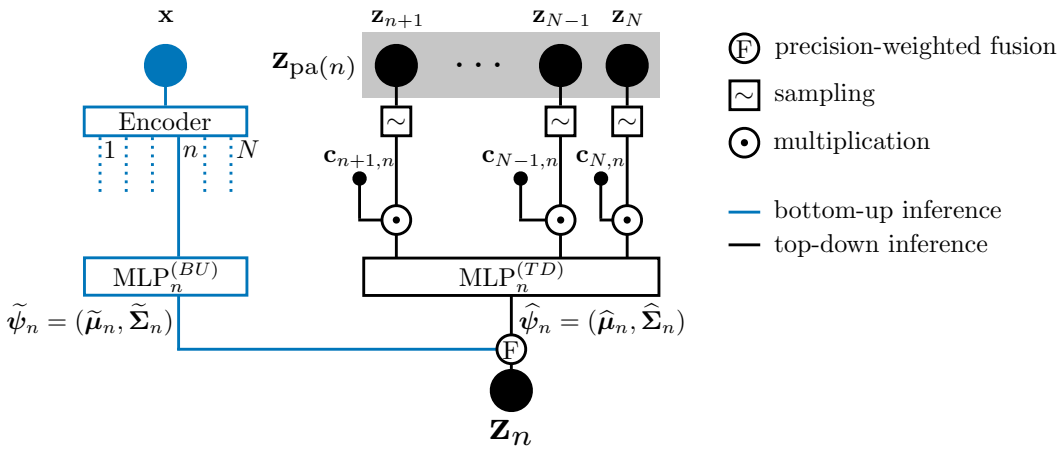


Figure 2: **Local Distributions.** We illustrate the parametrization of a local variable $\mathbf{z}_n$ in our structured representation. The local prior (Eq. (6)) is defined in terms of a top-down process (in black) predicting the node's parameters $\widehat{\psi}_n$ from a gate-modulated sample of $\mathbf{z}_n$'s parents $\mathbf{z}_{\text{pa}(n)}$. The local approximate posterior (Eq. (7)) additionally performs a precision-weighted fusion of these parameters with the result of a bottom-up process using a node-specific MLP to predict input-conditioned parameters $\widetilde{\psi}_n$ from a generic encoding of $\mathbf{x}$.

particular structure. Thus, we express the approximate posterior as:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \prod_{n=1}^{N} q_\phi(\mathbf{z}_n|\mathbf{x}, \mathbf{z}_{\text{pa}(n)}), \tag{7}$$

where $\mathbf{z}_{\text{pa}(n)}$ denotes the parent nodes, marginalized across the dependency gate variables, $\mathbf{c}$. In practice, this involves sampling dependency gating variables to estimate the top-down input for inference. With a single sample, corresponding to a single dependency structure, this exactly corresponds to the inference procedure from Sønderby et al. (2016). Parameter prediction for the local factors $q_\phi(\mathbf{z}_n|\mathbf{x}, \mathbf{z}_{\text{pa}(n)})$ consists of a precision-weighted fusion of the top-down prediction $\widehat{\psi}_n$ described above and a bottom-up prediction $\widetilde{\psi}_n$. The latter is obtained by encoding $\mathbf{x}$ into a generic feature that is used as an input to a node-specific multi-layer perceptron MLP$_n^{(BU)}$ predicting $\widetilde{\psi}_n$. This is shown in blue in Fig. 2.

Introducing the dependency gating variables modifies the variational objective (Eq. (2)), as we must now marginalize over these additional variables. In Appendix A.1, we derive the corresponding lower bound, $\widetilde{\mathcal{L}}$, which can be expressed as

$$\widetilde{\mathcal{L}} = \mathbb{E}_{p(\mathbf{c})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{c})\right] - \text{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{c}))\right] = \mathbb{E}_{p(\mathbf{c})}\left[\mathcal{L}_\mathbf{c}\right], \tag{8}$$

where $\mathcal{L}_\mathbf{c}$ is the ELBO for a given value of dependency gating variables, $\mathbf{c}$. Thus, $\widetilde{\mathcal{L}}$ can be interpreted as the expected ELBO under the distribution of dependency structures induced by $p(\mathbf{c})$. We form a Monte Carlo estimate of $\widetilde{\mathcal{L}}$ by sampling $\mathbf{c} \sim p(\mathbf{c})$ and evaluating $\mathcal{L}_\mathbf{c}$. The training procedure is outlined in Algorithm 1.

3.2 Learning the Discrete Gating Variable Distributions

For a given latent dependency structure, gradients for the parameters θ and ϕ can be estimated using Monte Carlo samples and the reparameterization trick (Kingma & Welling, 2014; Rezende et al., 2014). To obtain gradients for the gate means, $\boldsymbol{\mu}$, we make use of recent advances in differentiating through discrete operations (Maddison et al., 2017; Jang et al., 2017), allowing us to differentiate through the sampling of the dependency gating variables, $\mathbf{c}$. Specifically, we recast the gating variables using the Gumbel-Softmax estimator from Jang et al. (2017), re-expressing $c_{i,j}$ as:

$$c_{i,j} = \frac{\exp((\log(\mu_{i,j}) + \epsilon_1)/\tau)}{\exp((\log(\mu_{i,j}) + \epsilon_2)/\tau) + \exp((\log(1 - \mu_{i,j}) + \epsilon_3)/\tau)}, \tag{9}$$

Algorithm 1 Optimizing VAEs with Latent Dependency Structure

Require: Data $\mathbf{x}$, number of latent nodes N, number of dimensions per node N'.
1: Initialize θ, ϕ, $\boldsymbol{\mu}$.
2: **repeat**
3: Sample $\mathbf{c}$ using Eq. (9) and determine $\mathbf{z}_{\text{pa}(n)}$ for each $\mathbf{z}_n$ based on the sampled structure.
4: For each node, compute $q_\phi(\mathbf{z}_n|\mathbf{x}, \mathbf{z}_{\text{pa}(n)})$ using Eq. (7).
5: Sample $\mathbf{z}$ from $q_\phi(\mathbf{z}|\mathbf{x})$ using Eq. (6) and compute $p_\theta(\mathbf{x}|\mathbf{z})$.
6: Update $\theta, \phi, \boldsymbol{\mu}$ based on the gradients derived from Eq. (8).
7: **until** Convergence.

where ϵ_1, ϵ_2, and ϵ_3 are i.i.d samples drawn from a Gumbel$(0, 1)$ distribution and τ is a temperature parameter. The Gumbel-Softmax distribution is differentiable for $\tau > 0$, allowing us to estimate the derivative $\frac{\partial c_{i,j}}{\partial \mu_{i,j}}$. For large values of τ, we obtain smoothed versions of $\mathbf{c}$, essentially interpolating between different dependency structures. As $\tau \to 0$, we recover binary values for $\mathbf{c}$, yielding the desired discrete sampling of dependency structures at the cost of high-variance gradient estimates. Thus, we anneal τ during training to learn the dependency gate means, $\boldsymbol{\mu}$, eventually arriving at the discrete setting.

4 Evaluation

We evaluate the proposed model and its learned latent dependency structure on three challenging datasets: MNIST (Lecun et al., 1998; Larochelle & Murray, 2011), Omniglot (Lake et al., 2013), and CIFAR-10 (Krizhevsky, 2009). MNIST and Omniglot are binarized and modeled with Bernoulli distributions; CIFAR-10 is modeled with Gaussian densities.

Our baselines consist of classic variational autoencoders (Kingma & Welling, 2014; Rezende et al., 2014) as well as its popular variants, including ladder VAEs (Sønderby et al., 2016) and VAEs with fully-connected latent dependency structure (FC-VAEs; cf. (Kingma et al., 2016)). Together, these baselines cover the full spectrum of latent dependency structures, ranging from the strongest possible independence assumptions (fully independent in classic VAEs) to linear dependencies (chain-structured in ladder VAEs) to the most expressive representation (fully connected in FC-VAEs). Our experiments show that none of those predefined extremes is optimal. Instead, we demonstrate that learning a data-driven distribution over graph structures that is jointly optimized with the variational and generative model parameters leads to a superior representation of the underlying test-time distribution. An ablation study provides further insights into our learning process, hyperparameter selection, and the nature of the inferred structures.

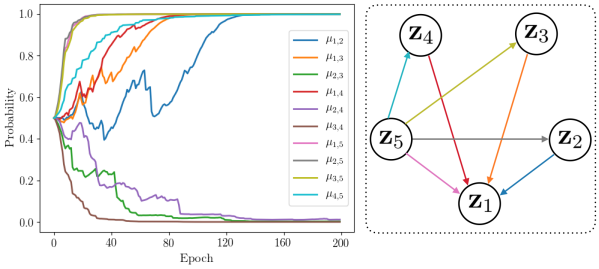
We note that our approach is orthogonal and could be complemented by a number of other approaches attempting to overcome the mean field assumptions of classic VAEs: Similar to ladder VAEs, (Zhao et al., 2017) also introduces chain-structured latent dependencies but focuses on learning a disentangled representation. VAEs with normalizing flows (Rezende & Mohamed, 2015) gradually increase the complexity of the approximate posterior through a series of invertible transformations.

4.1 Experimental Setup

To guarantee a fair comparison, the encoders of all structured methods use the same MLP architecture with batch normalization (Ioffe & Szegedy, 2015) and ReLU non-linearities (Nair & Hinton, 2010) in all experiments.[2] Decoder structures are the reverse of the encoders. Likewise, the number of latent dimensions is the same in all experiments/models ($M = 80$). As discussed in section 3.1, all latent dependencies are modelled by non-linear MLPs as well.

For MNIST and Omniglot, we model $p_\theta(\mathbf{x}|\mathbf{z})$ as a Bernoulli distribution, using a sigmoid non-linearity on the output layer to produce the mean of this distribution. For CIFAR-10, we model

[2] Classic VAE uses a more complex encoder to match the number of parameters of the structured methods. All baselines use the same or more parameters than graph VAE.



(a) (b)

Figure 3: **Structure Learning.** In (**a**), we show training of the Bernoulli parameters $\mu_{i,j}$ governing the distribution over graph structures in architecture space. All edges are color-coded and can be located in (**b**), where we show a random sample from the resulting steady-state distribution with the same color scheme.

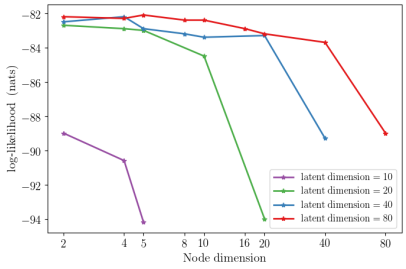


Figure 4: **Ablation Study on MNIST.** For a fixed latent dimension M (color-coded), we report the log-likelihood of all possible factorizations of node dimension N' (x-axis) and number of nodes $N = M/N'$.

$p_\theta(\mathbf{x}|\mathbf{z})$ with a Gaussian density, with mean and log-variance predicted by sigmoid and linear functions, respectively. Further implementation details, including the exact model architectures and training criteria, can be found in Appendix A.2.

4.2 Qualitative Analysis

An example of the structure learning process on MNIST with $N = 5$ nodes and a total latent dimension of $M = 80$ (i.e., $N' = M/N = 16$ dimensions per node) is visualized in Fig. 3. In Fig. 3a, we show the evolution of the parameters $\boldsymbol{\mu}$ of the gating variables $\mathbf{c}$. The network actively drops $3/10$ edges during the learning process, while the remaining parameters eventually converge to 1. We observed that the resulting distribution is stable across multiple training runs with different seeds for weight initialization and minibatch sampling. A sample of the learned distribution is shown in Fig. 3b.

Fig. 4 reports an ablation study conducted on MNIST. In particular, we investigate the influence of the total latent dimension M and, for fixed latent dimension, the trade-off between number of nodes N and node dimension $N' = M/N$. Models with the same total latent dimension M are shown in the same color. We observe two trends: (1) The performance of our model increases with latent dimension. This is an expected result and a likely explanation for this behaviour is the additional flexibility of a higher-dimensional latent space; (2) For a fixed number of latent dimensions, less node dimensions (and thus more nodes and the potential for a more complex dependency structure) typically performs better, highlighting the importance of a learned, non-trivial dependency structure.

4.3 Quantitative Comparison

We evaluate the performance of all models using their test-time log-likelihood $\log p_\theta(\mathbf{x})$ (Table 1). All values were estimated using $5,000$ importance-weighted samples. Following standard practice, we report $\log p_\theta(\mathbf{x})$ in *nats* on MNIST/Omniglot and in *bits/input dimension* on CIFAR-10. Our proposed model with a jointly optimized, learned dependency structure consistently outperforms both models with less expressive (VAE, ladder VAE) and more expressive (FC-VAE) predefined structures. We discuss potential reasons in section 4.4. To provide further insights into the training objective, Table 1 also reports $D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$ and the full ELBO objective (Eq. (8)) of the trained models.

4.4 Discussion

We want to briefly discuss a number of observations that we have made during training and application of the proposed model and hope they provide the starting point for further research on structure learning in deep generative models.

Dataset	Method								
	MNIST			Omniglot			CIFAR-10		
	LL	KL	ELBO	LL	KL	ELBO	LL	KL	ELBO
VAE	−89.1	29.0	−92.1	−110.9	30.5	−120.4	−6.63	0.110	−6.69
Ladder VAE	−84.8	**24.3**	−87.8	−106.4	**27.9**	−112.5	−6.47	0.082	−6.50
FC VAE	−83.0	28.9	−84.8	−104.8	29.9	−106.6	−6.44	0.077	−6.46
Graph VAE	**−82.1**	27.8	**−84.1**	**−103.4**	29.1	**−105.2**	**−6.40**	**0.074**	**−6.41**

Table 1: **Quantitative Analysis.** We show test-time log-likelihoods (LL), $D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$ (KL), and ELBO of the proposed graph VAE model (last row) and compare it to baselines on 3 popular datasets.

Performance vs. Speed/Memory. As shown in Fig. 4, the number of latent nodes has a big impact on the performance of our model. While allowing more complex dependency structures through a low M/N-ratio is typically beneficial, it also has an adverse effect on the training time and memory consumption. Fortunately, the ability to freely select this ratio allows a simple adaption to the available processing power, hardware constraints, and application scenarios.

Optimization Order. It is worth noting that the learning process optimizes the model parameters $(\mathbf{c}, \phi, \theta)$ in a clear temporal order: While the latent structure, governed by $\mathbf{c}$, converges during the first ≈ 200 epochs (Fig. 3a), it takes over $10\times$ as long until the variational and generative parameters (ϕ, θ) converge. There is no external force incentivizing this behaviour, indicating that the loss can initially most easily be decreased by limiting the latent structure to the complexity prescribed by the observed training data.

Improvement over Fully-Connected VAEs. By definition, the latent spaces of FC-VAEs make no conditional independence assumptions and are therefore more expressive and more flexible than the other models in this comparison, including any latent structure resulting from the proposed learning process. The fact that graph VAE outperforms FC-VAE across all datasets is thus interesting, but not entirely surprising: It is, for instance, well-known that latent variable models are prone to local optima (Bowman et al., 2016; Burda et al., 2016) and it is reasonable to assume that complex dependencies only add to the optimization challenges. The ability to modify entire latent dependencies allows a data-driven simplification of the underlying structure and, potentially, the optimization task itself.

5 Conclusion

We presented a novel method for optimizing variational autoencoders jointly with their latent dependency structures. Our experiments showed that the learned latent dependency structure improves the generative performance of latent variable models. By introducing an additional set of binary structure indicator variables and optimizing a structure-dependent evidence lower bound, our model is able to learn a better representation of the underlying generative distribution than baselines with a predetermined structure.

We believe that the proposed model is useful beyond the applications shown in this work and can be extended in many interesting ways. Examples include structure learning on undirected Markov random fields, dynamic structure prediction in time-series models, and joint optimization with the neural encoder-decoder architecture.

REFERENCES

A. Barron, J. Rissanen, and B. Yu. The Minimum Description Length Principle in Coding and Modeling. In *IEEE Transactions on Information Theory*, 1998.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 10–21, 2016.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *International Conference on Learning Representations*, 2016.

Jie Cheng, Russell Greiner, Jonathan Kelly, David Bell, and Weiru Liu. Learning Bayesian Networks from Data: An Information-theoretic Approach. In *Artificial Intelligence*, 2002.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2980–2988, 2015.

Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in neural information processing systems*, pp. 2199–2207, 2016.

Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. In *International Conference on Machine Learning*, pp. 1242–1250, 2014.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *International Conference on Machine Learning*, pp. 1462–1471, 2015.

Jiawei He, Andreas Lehrmann, Joseph Marino, Greg Mori, and Leonid Sigal. Probabilistic video generation using holistic attribute control. In *European Conference on Computer Vision*, 2018.

D. Heckerman, C. Meek, and G. Cooper. A Bayesian Approach to Causal Discovery. In *Computation, Causation, Discovery*, 1999.

David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. In *Machine Learning*, 1995.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations*, 2017.

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 D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 2946–2954. Curran Associates, Inc., 2016.

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

R. Kass and A. Raftery. Bayes Factors. In *Journal of the American Statistical Association*, 1995.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pp. 4743–4751, 2016.

M. Koivisto and K. Sood. Exact Bayesian Structure Discovery in Bayesian Networks. In *Journal of Machine Learning Research*, 2004.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models*. MIT Press, 2009.

Rahul G Krishnan, Dawen Liang, and Matthew Hoffman. On the challenges of learning with inference networks on sparse, high-dimensional data. *arXiv preprint arXiv:1710.06085*, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Brenden M Lake, Ruslan R Salakhutdinov, and Josh Tenenbaum. One-shot learning by inverting a compositional causal process. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26*, pp. 2526–2534. Curran Associates, Inc., 2013.

Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 29–37, 2011.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

Erich L. Lehmann and Joseph P. Romano. *Testing Statistical Hypotheses*. Springer, 2008.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *International Conference on Learning Representations*, 2017.

Joseph Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *International Conference on Machine Learning*, pp. 3400–3409, 2018.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pp. 1278–1286, 2014.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pp. 3738–3746, 2016.

Li Yingzhen and Stephan Mandt. Disentangled sequential autoencoder. In *International Conference on Machine Learning*, pp. 5656–5665, 2018.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Learning hierarchical features from deep generative models. In *International Conference on Machine Learning*, pp. 4091–4099, 2017.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

A SUPPLEMENTARY MATERIAL

A.1 LOWER BOUND DERIVATION

Introducing the dependency gating variables, $\mathbf{c}$, modifies the evidence lower bound (ELBO) from Eq. (2), as we now have an additional set of random variables over which to marginalize. To see this, we can start by re-expressing Eq. (2) as

$$\mathcal{L} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x},\mathbf{z})\right] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log q_\phi(\mathbf{z}|\mathbf{x})\right]. \tag{10}$$

$p_\theta(\mathbf{x},\mathbf{z})$ can be expressed as a marginalization over the gating variables, effectively averaging over an ensemble of models with a distribution of dependency structures:

$$p_\theta(\mathbf{x},\mathbf{z}) = \int p_\theta(\mathbf{x},\mathbf{z},\mathbf{c})d\mathbf{c} = \int p_\theta(\mathbf{x},\mathbf{z}|\mathbf{c})p(\mathbf{c})d\mathbf{c} = \mathbb{E}_{p(\mathbf{c})}\left[p_\theta(\mathbf{x},\mathbf{z}|\mathbf{c})\right]. \tag{11}$$

Plugging this into Eq. (10):

$$\mathcal{L} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \mathbb{E}_{p(\mathbf{c})}\left[p_\theta(\mathbf{x},\mathbf{z}|\mathbf{c})\right]\right] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log q_\phi(\mathbf{z}|\mathbf{x})\right]. \tag{12}$$

Using Jensen's inequality, we can bring the $\log$ inside of the expectation $\mathbb{E}_{p(\mathbf{c})}[\cdot]$, yielding

$$\mathcal{L} \geq \widetilde{\mathcal{L}} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\mathbb{E}_{p(\mathbf{c})}\left[\log p_\theta(\mathbf{x},\mathbf{z}|\mathbf{c})\right]\right] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log q_\phi(\mathbf{z}|\mathbf{x})\right], \tag{13}$$

where we have defined $\widetilde{\mathcal{L}}$ as a lower bound on $\mathcal{L}$ (and therefore still a lower bound on $\log p_\theta(\mathbf{x})$). Swapping the order of expectation, we can rewrite $\widetilde{\mathcal{L}}$ as

$$\widetilde{\mathcal{L}} = \mathbb{E}_{p(\mathbf{c})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x},\mathbf{z}|\mathbf{c})\right]\right] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log q_\phi(\mathbf{z}|\mathbf{x})\right], \tag{14}$$

and because the second term is independent of $p(\mathbf{c})$, we can include both terms inside of a single outer expectation:

$$\begin{aligned}\widetilde{\mathcal{L}} &= \mathbb{E}_{p(\mathbf{c})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x},\mathbf{z}|\mathbf{c}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right]\right] \\ &= \mathbb{E}_{p(\mathbf{c})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z},\mathbf{c})\right] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{c}))\right] \\ &= \mathbb{E}_{p(\mathbf{c})}\left[\mathcal{L}_\mathbf{c}\right],\end{aligned} \tag{15}$$

where we have defined $\mathcal{L}_\mathbf{c}$ as the ELBO for a given dependency structure. Thus, during training, we can estimate $\widetilde{\mathcal{L}}$ by sampling gating variables, $\mathbf{c} \sim p(\mathbf{c})$, and evaluating Monte Carlo estimates of $\mathcal{L}_\mathbf{c}$.

A.2 IMPLEMENTATION DETAILS

A.2.1 NETWORK ARCHITECTURE

We document the network architectures used in our experiments. We use the same network architectures for all datasets. Input size (pixels per image) is the only difference across datasets. The input_size is 28×28 for MNIST and Omniglot, and $3 \times 32 \times 32$ for CIFAR-10.

Encoders:

fc(input_size, 512) → Batch Norm → ELU → fc(512, 512) → Batch Norm → ELU → fc(512, 512) → Batch Norm → ELU → fc(512, 256)

Latent Dependencies: Latent dependencies are modelled by non-linear MLPs. Note that the top-down architecture is shared between inference model and generative model. But the MLPs are optimized independently.

bottom-up: For each node with N' dimensions, the local potential is predicted by a mapping from the encoded feature:

fc(256, 256) → Batch Norm → ReLU → fc(256, N'). The output feature is then mapped to μ and $\log$ var with two independent fc(N', N') respectively.

top-down: For each node (N' dimensions) with a set of parent nodes, the top-down inference/generation is implemented as:

fc(sum of parent nodes' dimension, 256) → Batch Norm → ReLU → fc(256, N') The output feature is then mapped to μ and $\log$ var with two independent fc(N', N') respectively.

Decoders:

fc(512, 512) → Batch Norm → ReLU → fc(512, 512) → Batch Norm → ReLU → fc(512, 512) → Batch Norm → ReLU → fc(512, input_size) → out_put function()

out_put function for MNIST and Omniglot is sigmoid() which predicts the μ of Bernoulli observations; and sigmoid predicting μ, fc(input_size, input_size) predicting $\log$ var of Gaussion observations for CIFAR-10.

A.2.2 TRAINING

All models were implemented with PyTorch (Paszke et al., 2017) and trained using the Adam (Kingma & Ba, 2015) optimizer with a mini-batch size of 128, and learning rate of $3e^{-4}$. The Gumbel-softmax temperature were initialized as 1, and decrease to 0.999^{epoch} at each epoch. MNIST and Omniglot took 2000 epochs to converge, and CIFAR took 3000 epochs to converge.