Learning Disentangled Representations with Semi-Supervised Deep Generative Models

N. Siddharth
University of Oxford
nsid@robots.ox.ac.uk

Brooks Paige
Alan Turing Institute
University of Cambridge
bpaige@turing.ac.uk

Jan-Willem Van de Meent
Northeastern University
j.vandemeent@northeastern.edu

Alban Desmaison
University of Oxford
alban@robots.ox.ac.uk

Frank Wood
University of Oxford
fwood@robots.ox.ac.uk

Noah D. Goodman
Stanford University
ngoodman@stanford.edu

Pushmeet Kohli *
Deepmind
pushmeet@google.com

Philip H.S. Torr
University of Oxford
philip.torr@eng.ox.ac.uk

Abstract

Variational autoencoders (VAEs) learn representations of data by jointly training a probabilistic encoder and decoder network. Typically these models encode all features of the data into a single variable. Here we are interested in learning disentangled representations that encode distinct aspects of the data into separate variables. We propose to learn such representations using model architectures that generalize from standard VAEs, employing a general graphical model structure in the encoder and decoder. This allows us to train partially-specified models that make relatively strong assumptions about a subset of interpretable variables and rely on the flexibility of neural networks to learn representations for the remaining variables. We further define a general objective for semi-supervised learning in this model class, which can be approximated using an importance sampling procedure. We evaluate our framework's ability to learn disentangled representations, both by qualitative exploration of its generative capacity, and quantitative evaluation of its discriminative ability on a variety of models and datasets.

1 Introduction

Learning representations from data is one of the fundamental challenges in machine learning and artificial intelligence. Characteristics of learned representations can depend on their intended use. For the purposes of solving a single task, the primary characteristic required is suitability for that task. However, learning separate representations for each and every such task involves a large amount of wasteful repetitive effort. A representation that has some factorisable structure, and consistent semantics associated to different parts, is more likely to generalise to a new task.

Probabilistic generative models provide a general framework for learning representations: a model is specified by a joint probability distribution both over the data and over latent random variables, and a representation can be found by considering the posterior on latent variables given specific data. The learned representation — that is, inferred values of latent variables — depends then not just on the data, but also on the generative model in its choice of latent variables and the relationships between the latent variables and the data. There are two extremes of approaches to constructing generative models. At one end are fully-specified probabilistic graphical models [18, 21], in which a practitioner decides on all latent variables present in the joint distribution, the relationships between them, and the functional form of the conditional distributions which define the model. At the other end are deep generative models [7, 16, 19, 20], which impose very few assumptions on the structure of the model, instead employing neural networks as flexible function approximators that can be used to train a conditional distribution on the data, rather than specify it by hand.

*Author was at Microsoft Research during this project.

The tradeoffs are clear. In an explicitly constructed graphical model, the structure and form of the joint distribution ensures that latent variables will have particular semantics, yielding a *disentangled* representation. Unfortunately, defining a good probabilistic model is hard: in complex perceptual domains such as vision, extensive feature engineering (e.g. Siddharth et al. [30], Berant et al. [1]) may be necessary to define a suitable likelihood function. Deep generative models completely sidestep the difficulties of feature engineering. Although they address learning representations which then enable them to better reconstruct data, the representations themselves do not always exhibit consistent meaning along axes of variation: they produce *entangled* representations. While such approaches have considerable merit, particularly when faced with the absence of any side information about data, there are often situations when aspects of variation in data can be, or are desired to be characterised.

Bridging this gap is challenging. One way to enforce a disentangled representation is to hold different axes of variation fixed during training [20]. Johnson et al. [13] combine a neural net likelihood with a conjugate exponential family model for the latent variables. In this class of models, efficient marginalization over the latent variables can be performed by learning a projection onto the same conjugate exponential family in the encoder. Here we propose a more general class of *partially-specified* graphical models: probabilistic graphical models in which the modeller only needs specify the exact relationship for some subset of the random variables in the model. Factors left undefined in the model definition are then learned, parametrized by flexible neural networks. This provides the ability to situate oneself at a particular point on a *spectrum*, by specifying precisely those axes of variations (and their dependencies) we have information about or would like to extract, and learning disentangled representations for them, while leaving the rest to be learned in an entangled manner.

A subclass of partially-specified models that is particularly common is that where we can obtain supervision data for some subset of the variables. In practice, there is often variation in the data which is (at least conceptually) easy to explain, and therefore annotate, whereas other variation is less clear. For example, consider the MNIST dataset of handwritten digits: the images vary both in terms of content (which digit is present), and style (how the digit is written), as is visible in the right-hand side of Figure 1. Having an explicit "digit" latent variable captures a meaningful and consistent axis of variation, independent of style; using a partially-specified graphical model means we can define a "digit" variable even while leaving unspecified the semantics of the different styles, and the process of rendering a digit to an image. In a fully unsupervised learning procedure there is generally no guarantee that inference on a model with 10 classes will in fact recover the 10 digits. However, given a small amount of labeled examples, this task becomes significantly easier. Beyond the ability to encode variation along some particular axes, we may also want to interpret the same data in different ways. For example, when considering images of people's faces, we might wish to capture the person's identity in one context, and the lighting conditions on the faces in another.

In this paper we introduce a recipe for learning and inference in partially-specified models, a flexible framework that learns disentangled representations of data by using graphical model structures to encode constraints to interpret the data. We present this framework in the context of *variational auto-encoders* (VAEs), developing a generalised formulation of semi-supervised learning with DGMs that enables our framework to automatically employ the correct factorisation of the objective for any given choice of model and set of latents taken to be observed. In this respect our work extends previous efforts to introduce supervision into variational auto-encoders [17, 31, 23]. We introduce a novel variational objective which is applicable to a more general class of models, allowing us to consider graphical-model structures with arbitrary dependencies between latents, continuous-domain latents, and those with dynamically changing dependencies. We provide a characterization of how to compile partially-supervised generative models into stochastic computation graphs, suitable for end-to-end training. This approach allows us also *amortise* inference [6, 22, 28, 33], learning simultaneously learning a network that performs approximate inference over representations at the same time we learn the unknown factors of the model itself. We demonstrate the efficacy of our framework on a variety of tasks, involving classification, regression, and predictive synthesis, including its ability to encode latents of variable dimensionality.

2 Framework and Formulation

VAEs [16, 27] are a class of deep generative models that simultaneously train both a probabilistic encoder and decoder for an element of a data set $\mathbf{x} \in \mathcal{D}$. The central analogy is that an encoding $\mathbf{z}$ can be considered a latent variable, casting the decoder as a conditional probability density $p_\theta(\mathbf{x}|\mathbf{z})$.

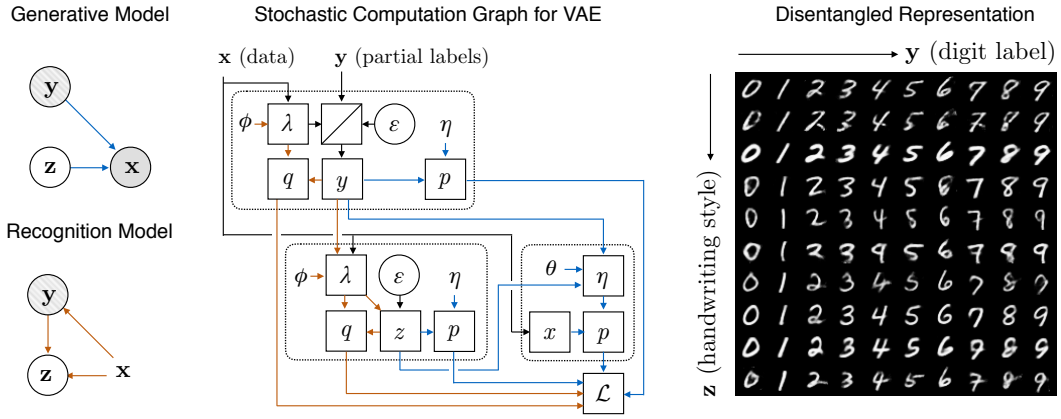


Figure 1: Semi-supervised learning in structured variational autoencoders, illustrated on MNIST digits. *top-left*: Generative model. *bottom-left:* Recognition model. *middle*: Stochastic computation graph, showing expansion of each node to its corresponding network module. Dependencies associated with the generative model are shown in blue and dependencies for the recognition model are shown in orange. *right:* learned representation. See main text for a more detailed explanation.

The parameters $\eta_\theta(\mathbf{z})$ of this distribution are the output of a deterministic neural network with parameters θ (most commonly MLPs or CNNs) which takes $\mathbf{z}$ as input. By placing a weak prior over $\mathbf{z}$, the decoder defines a posterior and joint distribution $p_\theta(\mathbf{z} \mid \mathbf{x}) \propto p_\theta(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})$.

Inference in VAEs can be performed using a variational method that approximates the posterior distribution $p_\theta(\mathbf{z} \mid \mathbf{x})$ using an encoder $q_\phi(\mathbf{z} \mid \mathbf{x})$, whose parameters $\lambda_\phi(\mathbf{x})$ are the output of an of a network (with parameters ϕ) that is referred to as an "inference network" or a "recognition network". The generative and inference networks are trained jointly by performing stochastic gradient ascent on the *evidence lower bound* (ELBO) $\mathcal{L}(\phi, \theta; \mathcal{D}) \leq \log p_\theta(\mathcal{D})$,

$$\mathcal{L}(\phi, \theta; \mathcal{D}) = \sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\phi, \theta; \mathbf{x}^i) = \sum_{\mathbf{x}^i \in \mathcal{D}} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^i)}\big[\log p_\theta(\mathbf{x}^i, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}^i)\big]. \tag{1}$$

In this paper we are interested in defining VAE architectures in which a subset of variables $\mathbf{y}$ are interpretable. For these variables, we assume that supervision labels are available for some fraction of the data. The VAE will additionally retain some set of variables $\mathbf{z}$ for which inference is performed in a fully unsupervised manner. In the running example for MNIST, $\mathbf{y}$ would correspond to the classification label, whereas $\mathbf{z}$ would be a vector that captures all other features, such as the pen type and handwriting style.

We will consider models in which both the generative model $p_\theta(\mathbf{x}, \mathbf{y}, \mathbf{z})$ and the approximate posterior $q_\phi(\mathbf{y}, \mathbf{z} \mid \mathbf{x})$ can have arbitrary conditional dependency structures. This class of models generalizes both the models in the work by Kingma et al. [17], who consider three specific model designs, and those presented by Johnson et al. [14], where $p(\mathbf{y}, \mathbf{z})$ is a conjugate exponential-family model.

To perform semi-supervised learning in this class of models, we need to (1) define an objective that is suitable to general dependency graphs and (2) define a method for constructing a stochastic computation graph [29] that incorporates both the conditional dependence structure in the generative model and that of the recognition model into this objective.

2.1 Objective Function

Previous work on semi-supervised learning for deep generative models [17] defines an objective

$$\mathcal{L}(\theta, \phi; \mathcal{D}, \mathcal{D}^{\text{sup}}) = \sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\theta, \phi; \mathbf{x}^i) + \alpha \sum_{(\mathbf{x}^j, \mathbf{y}^j) \in \mathcal{D}^{\text{sup}}} \left(\mathcal{L}^{\text{cond}}(\theta, \phi; \mathbf{x}^j, \mathbf{y}^j) + \beta \mathcal{L}^{\text{ml}}(\phi; \mathbf{x}^j, \mathbf{y}^j)\right). \tag{2}$$

For unsupervised examples $\mathbf{x}^i \in \mathcal{D}$, this objective evaluates the ELBO defined in Equation (1). For labelled examples $(\mathbf{x}^j, \mathbf{y}^j) \in \mathcal{D}^{\text{sup}}$, this objective evaluates a conditional ELBO $\mathcal{L}^{\text{cond}}(\theta, \phi; \mathbf{x}^j, \mathbf{y}^j)$

and a maximum likelihood objective $\mathcal{L}^{\text{ml}}(\phi; \mathbf{x}^j, \mathbf{y}^j)$, which are defined as,

$$\mathcal{L}^{\text{cond}}(\theta, \phi; \mathbf{x}^j, \mathbf{y}^j) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^j, \mathbf{y}^j)}\left[\log \frac{p_\theta(\mathbf{x}^j, \mathbf{y}^j, \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x}^j, \mathbf{y}^j)}\right], \quad \mathcal{L}^{\text{ml}}(\phi; \mathbf{x}^j, \mathbf{y}^j) = \log q_\phi(\mathbf{y}^j \mid \mathbf{x}^j). \tag{3}$$

This objective assumes that it is possible to evaluate the point-wise probabilities of the marginal $q_\phi(\mathbf{y}|\mathbf{x}) = \int d\mathbf{z}\ q_\phi(\mathbf{y}, \mathbf{z}|\mathbf{x})$ and conditional $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$. This is indeed the case for the model structures considered in [17], which assume a factorization $q_\phi(\mathbf{y}, \mathbf{z}|\mathbf{x}) = q_{\phi_\mathbf{z}}(\mathbf{z}|\mathbf{x}, \mathbf{y}) q_{\phi_\mathbf{y}}(\mathbf{y}|\mathbf{x})$.

Here we propose an alternative objective. We extend the model with an auxiliary variable $\tilde{\mathbf{y}}$ with likelihood $p(\tilde{\mathbf{y}} \mid \mathbf{y}) = \delta_{\tilde{\mathbf{y}}}(\mathbf{y})$ to define densities

$$\begin{aligned} p(\tilde{\mathbf{y}}, \mathbf{y}, \mathbf{z}, \mathbf{x}) &= p(\tilde{\mathbf{y}} \mid \mathbf{y}) p_\theta(\mathbf{x} \mid \mathbf{y}, \mathbf{z}) p(\mathbf{y}, \mathbf{z}), \\ q(\tilde{\mathbf{y}}, \mathbf{y}, \mathbf{z} \mid \mathbf{x}) &= p(\tilde{\mathbf{y}} \mid \mathbf{y}) q_\phi(\mathbf{y}, \mathbf{z} \mid \mathbf{x}). \end{aligned}$$

When we marginalize the ELBO for this model over $\tilde{\mathbf{y}}$, we recover the expression in Equation (1). Treating $\tilde{\mathbf{y}} = \mathbf{y}^j$ as observed results in the supervised objective

$$\begin{aligned} \mathcal{L}\big(\theta, \phi; \mathbf{x}^j, \mathbf{y}^j\big) &= \mathbb{E}_{q_\phi(\mathbf{z}, \mathbf{y} \mid \mathbf{x}^j)}\left[\delta_{\mathbf{y}^j}(\mathbf{y}) \log \frac{p_\theta\big(\mathbf{x}^j \mid \mathbf{z}, \mathbf{y}\big) p(\mathbf{z}, \mathbf{y})}{q_\phi(\mathbf{z}, \mathbf{y} \mid \mathbf{x}^j)}\right], \\ &= q_\phi(\mathbf{y}^j|\mathbf{x}^j)\, \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^j, \mathbf{y}^j)}\left[\log \frac{p_\theta(\mathbf{x}^j, \mathbf{y}^j, \mathbf{z})}{q_\phi(\mathbf{y}^j, \mathbf{z}|\mathbf{x}^j)}\right]. \end{aligned} \tag{4}$$

The objective for the full dataset is then

$$\mathcal{L}(\theta, \phi; \mathcal{D}, \mathcal{D}^{\text{sup}}) = \sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\theta, \phi; \mathbf{x}^i) + \alpha \sum_{(\mathbf{x}^j, \mathbf{y}^j) \in \mathcal{D}^{\text{sup}}} \mathcal{L}(\theta, \phi; \mathbf{x}^j, \mathbf{y}^j). \tag{5}$$

In this objective we incorporate a hyper-parameter α, which we refer to as the supervision scale factor. This parameter controls the relative weight of the labelled examples relative to the unlabelled examples in the data. Operationally, we either alternate between batches from $\mathcal{D}$ and $\mathcal{D}^{\text{sup}}$ during gradient ascent, or sample mixed batches at each step. The α parameter equates to sampling examples or batches from $\mathcal{D}$ with probability $|\mathcal{D}|/(|\mathcal{D}| + \alpha|\mathcal{D}|^{\text{sup}})$ and examples from $\mathcal{D}^{\text{sup}}$ with probability $\alpha|\mathcal{D}|^{\text{sup}}/(|\mathcal{D}| + \alpha|\mathcal{D}|^{\text{sup}})$. Note that the objective from [17] implicitly sets $\alpha = 1$, but contains an additional degree of freedom β, which controls the relative strength of the $\mathcal{L}^{\text{cond}}$ and $\mathcal{L}^{\text{ml}}$ terms.

As with the objective in [17], this objective assumes that we can evaluate the marginal $q_\phi(\mathbf{y}^j|\mathbf{x}^j)$ and generate samples $\mathbf{z}^{j,s} \sim q(\mathbf{z}|\mathbf{x}^j, \mathbf{y}^j)$. This is in fact the case for the models in all experiments that we evaluate in the next section. That said, the objective above also extends to models in which $\mathbf{y}$ conditionally depends on $\mathbf{z}$. For such models, we can calculate a Monte Carlo estimate using a sequential importance sampling procedure analogous to the one commonly used in probabilistic programming systems such as [8, 35]. This procedure samples unobserved variables $\mathbf{z}^{j,s}$ and $\mathbf{y}^{j,s}$ (when needed) from the conditional distributions in the recognition model and calculates an importance weight $w^{j,s}$ according to the conditional probability of observed values $\mathbf{y}^j$ at each node,

$$\mathcal{L}^{1:S}(\theta, \phi; \mathbf{x}^j, \mathbf{y}^j) \simeq \hat{\mathcal{L}}^{1:S}(\theta, \phi; \mathbf{x}^j, \mathbf{y}^j) = \frac{1}{S} \sum_{s=1}^{S} w^{j,s} \log \frac{p_\theta(\mathbf{x}^j, \mathbf{y}^j, \mathbf{z}^{j,s})}{q_\phi(\mathbf{y}^j, \mathbf{z}^{j,s}|\mathbf{x}^j)}. \tag{6}$$

This estimate can be derived by replacing $q_\phi(\mathbf{y}^j|\mathbf{x}^j)$ with an importance sampling estimate, which cancels out against the normalization term of the importance sampling estimate over $q_\phi(\mathbf{z} \mid \mathbf{x}^j, \mathbf{y}^j)$.

2.2 Construction of the Stochastic Computation Graph

In order to perform gradient ascent on the objective in Equation (5), we map the graphical models for $p_\theta(\mathbf{x}, \mathbf{y}, \mathbf{z})$ and $q_\phi(\mathbf{y}, \mathbf{z}|\mathbf{x})$ onto a stochastic computation graph in which each variable node is expanded to a sub-graph that we refer to as a module. Figure 1 shows this expansion for a simple VAE for MNIST digits. In this model $\mathbf{y}$ is a discrete variable that represents the digit, for which we have partial supervision data. $\mathbf{z}$ is a Gaussian-distributed vector that represents the hand-writing style, which is unobserved. In the generative model (Figure 1 top-left) we assume a factorization $p_\theta(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p_\theta(\mathbf{x} \mid \mathbf{y}, \mathbf{z}) p(\mathbf{y}) p(\mathbf{z})$ in which $\mathbf{y}$ and $\mathbf{z}$ are independent under the

(a) (b) (c) (d)

Figure 2: (a) Visual analogies for the MNIST data, with inferred style latent variable fixed and the label varied. (b) Exploration in "style" space for a 2D latent gaussian random variable. Visual analogies for the SVHN data when (c) fully supervised, and (d) partially supervised with just 100 labels per digit.

prior. In the recognition model (Figure 1 bottom-left), we use a conditional dependency structure $q_\phi(\mathbf{y}, \mathbf{z} \mid \mathbf{x}) = q_{\phi_\mathbf{z}}(\mathbf{z} \mid \mathbf{y}, \mathbf{x}) q_{\phi_\mathbf{y}}(\mathbf{y} \mid \mathbf{x})$ to induce a disentangled representation (Figure 1 right) that separates the digit label $\mathbf{y}$ from the handwriting style $\mathbf{z}$.

The generative and recognition model are jointly translated to a stochastic computation graph (Figure 1 centre) that contains a module for each variables. Modules can correspond to fully supervised, partially supervised and unsupervised variables. This graph contains one module of each type:

- For the fully supervised variable $\mathbf{x}$, the module simply calculates the likelihood p under the generative model, that is $p_\theta(\mathbf{x} \mid \mathbf{y}, \mathbf{z}) = \mathcal{N}(\mathbf{x}\,;\, \eta_\theta(\mathbf{y}, \mathbf{z}))$. Here $\eta_\theta(\mathbf{y}, \mathbf{z})$ is a neural net with parameters θ that outputs the parameters of a normal distribution (i.e. the mean vector and a diagonal covariance).
- For the unobserved variable $\mathbf{z}$, we calculate both the prior probability $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}\,;\, \eta_\mathbf{z})$, and the conditional probability $q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{y}) = \mathcal{N}(\mathbf{z}\,;\, \lambda_{\phi_\mathbf{z}}(\mathbf{x}, \mathbf{y}))$. Here the usual reparametrization is used to sample $\mathbf{z}$ from $q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{y})$ by first sampling $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ using the usual reparametrization trick $\mathbf{z} = g(\epsilon, \lambda_\phi(\mathbf{x}, \mathbf{y}))$.
- For the partially observed variable $\mathbf{y}$, we also calculate probabilities $p(\mathbf{y}) = \text{Discrete}(\mathbf{y}; \eta_\mathbf{y})$ and $q_{\phi_\mathbf{y}}(\mathbf{y} \mid \mathbf{x}) = \text{Discrete}(\mathbf{y}; \lambda_{\phi_\mathbf{z}}(\mathbf{x}))$. The value $\mathbf{y}$ is either sampled from $q_{\phi_\mathbf{y}}(\mathbf{y} \mid \mathbf{x})$ using a Gumbel-softmax [12, 24] relaxation or treated as observed when available.

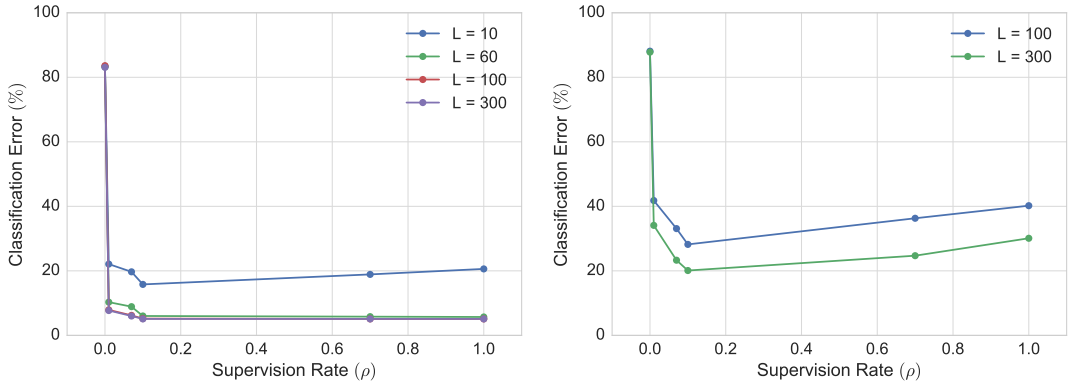
The example in Figure 1 illustrates a general framework for defining VAEs with arbitrary dependency structures. We begin by defining a module for each random variable. For each module we must now specify a distribution type and parameter function η, which determine how the probability under the generative model depends on the other variables in the network. This function can be a constant, fully deterministic, or a neural net whose parameters are learned from the data. For each unsupervised and semi-supervised variable we must similarly specify a mapping λ that determines the dependency graph in the recognition model, along with a procedure for sampling from the conditional distribution.

Given this specification of a computation graph, we can now calculate a Monte Carlo estimate of the lower bound in Equation (5) by simply running the network forward to obtain samples from $q(\cdot \mid \lambda)$ for all unobserved variables, and then calculating $p_\theta(\mathbf{x}, \mathbf{y}, \mathbf{z})$, $q_\phi(\mathbf{y} \mid \mathbf{x})$ and $q_\phi(\mathbf{y}, \mathbf{z} \mid \mathbf{x})$. This estimate can then be optimized with respect to the variables θ and ϕ to train the autoencoder.

If we wish to calculate the importance sampling estimate in Equation (6), then we can run the network forward to generate samples from the conditional distributions $q(\cdot \mid \lambda)$ at each unsupervised node. After running the network forward we can now calculate the importance weight w as the product over all outputs q from semi-supervised nodes for which labels are available. This procedure is fully general and in fact recovers the expression in Equation (4) in models where we can directly evaluate $q_\phi(\mathbf{y} \mid \mathbf{x})$. For this reason, all experiments in the next section follow this implementation.

3 Experiments

We evaluate our framework along a number of different axes pertaining to its ability to learn disentangled representations through the provision of partial graphical-model structures for the latents



	MNIST		SVHN	
L	Ours	Kingma et al. [17]	Ours	Kingma et al. [17]
10	15.3 ($\pm$ 1.18)	11.97 ($\pm$ 1.71)	—	—
60	5.78 ($\pm$ 0.71)	4.94 ($\pm$ 0.13)	—	—
100	5.01 ($\pm$ 0.62)	3.60 ($\pm$ 0.56)	28.71 ($\pm$ 2.38)	36.02 ($\pm$ 0.10)
300	4.02 ($\pm$ 0.71)	3.92 ($\pm$ 0.63)	19.13 ($\pm$ 1.03)	—

Figure 3: (Top) Classification error graphs over different labelled set (per class) sizes and supervision rates for MNIST (left) and SVHN (right). Note the steep drop in error rate with just a handful of labels per class ($L = |\mathcal{D}^{\text{sup}}| \leq 300, U = |\mathcal{D}| = O(10^4)$), that are just slightly over-represented during training ($\rho = \alpha L/(U + \alpha L) \leq 0.1$). (Bottom) Classification error rates for different (per-class) labelled-set sizes L over different runs.

and weak supervision. In particular, we evaluate its ability to (i) function as a classifier/regressor for particular latents under the given dataset, (ii) learn the generative model in a manner that preserves the semantics of the latents with respect to the data generated, and (iii) perform these tasks, in a flexible manner, for a variety of different models and data.

All experiments, save the one involving Street-View House Numbers [25] (SVHN), were run using a 2 layer MLP with 512 nodes and using a Bernoulli loss function. For the SVHN, we employed a two stage convolutional and a 2 stage deconvolutional network to extract features for the standard MLP model for the recognition network and the generative model respectively; training the entire network end-to-end. For learning, we used AdaM [15] with a learning rate and momentum-correction terms set to their default values. As for the minibatch sizes, they varied from 80-500 depending on the dataset being used and its size. All of the above, including further details of precise parameter values and the source code, including our library for specifying arbitrary graphical models in the VAE framework, will be made available shortly.

3.1 MNIST and SVHN

We begin by conducting a similar experiment to Kingma et al. [17], to compare and contrast the performance of our generalised framework and objective against the domain-specific formulation used therein. To this end, we explore our model's ability to learn disentangled representations in the standard MNIST and Google Street-View House Numbers (SVHN) datasets. Figure 1(left) shows the structure of the generative and recognition models used, where the "digit" label is partially specified (and partially supervised) and the "style" factor is assumed to be an unspecified variable.

Figure 2(a) and (c) show the effect of first transforming a given input (leftmost column) into the disentangled latent space, and with the style latent variable fixed, manipulating the digit through the generative model to generate data with expected visual characteristics. These were both derived with full supervision over a 50 and 100 dimensional Gaussian latent space for the styles, respectively. Figure 2(b) shows the transformation for a fixed digit, when the style latent is varied, and Figure 2(d) shows disentangling in the latent space with just 1000 labelled data points (out of 73000).

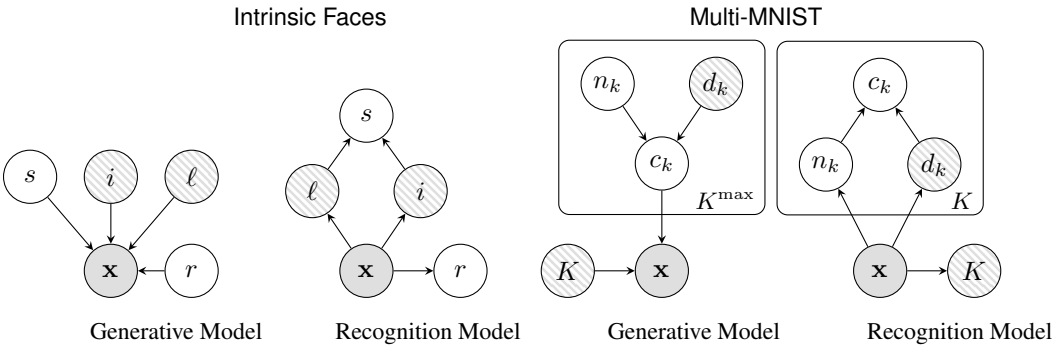


Figure 4: Generative and recognition models for the intrinsic-faces and multi-MNIST experiments.

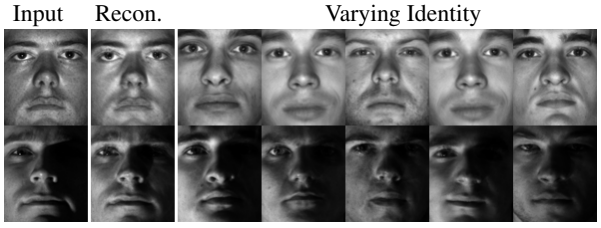
We compute the classification accuracy of the label-prediction task with this model for both datasets, and the results are reported in the bottom of Figure 3. The results are compared to those reported in Kingma et al. [17]. For MNIST, we compare against model M2 as we run directly on the data, without performing a preliminary feature-extraction step. For SVHN, we compare against model M1+M2 even though we run directly on the data, as we use a CNN to simultaneously learn to extract features. Confidence estimates for both of our estimates were computed off of 10 runs.

The results in Figure 3 are similar to those in [17], which serves as a basic validation of our framework. Better quantitative performance could be achieved by using alternate factorisations, as in Maaløe et al. [23] and innovations in neural-network architectures, as in Sønderby et al. [32], Rasmus et al. [26].

Supervision rate: We additionally explore the effect of the hyper-parameter α, that controls the relative weight of labelled vs. unlabelled data on the objective in Figure 3. We use $L = |\mathcal{D}^{\text{sup}}|$ and $U = |\mathcal{D}|$ to refer to the number of labeled and unlabeled examples. We report classification errors as a function of the supervision rate $\rho = \alpha L/(U + \alpha L)$ and the total number of labeled examples L. In situations when the labelled data is very sparse, we observe that over-representing the labeled examples during training aids generalisation. However, as is to be expected, over-fitting occurs when ρ is increased beyond a certain point.

3.2 Intrinsic Faces

We next move to a more complex domain involving generative models of faces. The stronger dependencies between the latents, compared to the models in Section 3.1, adds additional structural constraints for the recognition network to consider. We use the "Yale B" dataset [5] as processed by Jampani et al. [11] for the results in Figure 5. The primary tasks we are interested in here to demonstrate interpretability and disentangled representations are (i) classification of person identity, and (ii) regression for lighting direction.



	Identity	Lighting
Ours (Full Supervision)	4.2 ($\pm$ 0.84)	13.1 ($\pm$ 1.05)
Ours (Semi-Supervised)	10.2 ($\pm$ 2.31)	18.2 ($\pm$ 2.23)
Jampani et al. [11] (plot asymptotes)	≈ 30	≈ 10

Figure 5: (Left) Exploring the generative capacity of the model showing the input image, its reconstruction, and reconstructions with fixed (inferred) lighting and varying identities. (Right) Classification and regression error rates for the identity and lighting latent variables, fully-supervised, and semi-supervised with 20 distinct labelled example per variation axis (60 total). Classification is a direct 1-out-of-38 choice, whereas for the comparison, error is a nearest-neighbour loss based on the inferred reflectance. Regression loss for lighting is measured as cosine angle distance.

Supervision		Classification
size (L)	rate (ρ)	error rate (%)
0	0.00	32.25 ($\pm$ 12.97)
500	0.01	6.42 ($\pm$ 2.15)
500	0.10	4.21 ($\pm$ 1.29)
1000	0.01	4.72 ($\pm$ 1.60)
1000	0.10	2.98 ($\pm$ 0.93)

Figure 6: (Left) Example input multi-MNIST images and reconstructions. (Right) Count accuracy for the multi-MNIST dataset over different supervised set sizes L and supervision rates ρ.

Figure 5 presents both qualitative and quantitative evaluation of the framework to jointly learn both the structured recognition model, and the generative model parameters, denoting identity i, lighting l, shading s, and reflectance r. Note that in the recognition model (Figure 4), the lighting l is a latent variable with *continuous* domain, and one that we partially supervise. We encode identity i as a categorical random variable. Having partially specified these two variables of interest (and how they relate to each other), we can leave the rest of the model unspecified.

Previous work [11] assumed a generative relationship $(n \cdot l) \times r + \epsilon$ for the pixel data. Here we specify an approximate structure in which all variables are independent under the prior, allowing for the neural-network learn this relationship from data. This allows us to address the task of directly predicting identity, instead of applying surrogate evaluation methods (e.g. nearest-neighbour classification based on inferred reflectance).

3.3 Multi-MNIST

The previous examples explored some ways in which one could incorporate structure, particularly into the recognition network, the structures were themselves static. So far, the size and dimensionality of the latent variables has been fixed in all models. Here, we explore the ability of our framework to handle models that induce latent representations of *variable* dimension. These correspond to graphical models with dynamically changing (stochastic) number of latent variables.

We extend the models from the MNIST experiment by composing it with a *stochastic sequence generator*, to test the ability of our framework to *count* the number of digits in a given input image, given its ability to encode and reconstruct the digits in isolation as evidenced in Section 3.1. The recognition network is structured as a *recurrent neural network* (RNN) with disentangled state, with the generative model attending to each digit sequentially as shown in Figure 4. Here, we denote the recurrent index k, digit d_k, style n_k, and canvas c_k combining digit and style. This recurrent architecture is somewhat similar to those used in DRAW [10], recurrent VAEs [2], and AIR [4].

In the absence of a canonical multi-MNIST dataset, we created our own from the MNIST dataset by manipulating the scale and positioning of the standard digits into a combined canvas, evenly balanced across the counts (1-3) and digits. The count accuracy errors across different supervised set sizes and supervision rates, as well as the reconstructions for a random set of inputs are shown in Figure 6. We observe that not only are we able to reliably infer the *counts* of the digits in the given images, we are able to simultaneously reconstruct the input images using the predicted count of digits.

4 Discussion and Conclusion

In this paper we introduce a framework for learning disentangled representations of data using partially-specified graphical model structures and semi-supervised learning schemes in the domain of variational autoencoders (VAEs). This is accomplished by defining hybrid generative models which incorporate both structured graphical models and unstructured random variables in the same latent space. We demonstrate the flexibility of this approach by applying it to a variety of different tasks in the visual domain, and evaluate its efficacy at learning disentangled representations in a semi-supervised manner, showing strong performance. Such partially-specified models yield recognition networks that make predictions in an interpretable and disentangled space, constrained by the structure provided by the graphical model and the weak supervision.

The framework is implemented as a Torch library [3], enabling the construction of stochastic computation graphs which encode the requisite structure and computation. This provides another direction to explore in the future — the extension of the stochastic computation graph framework to probabilistic programming [9, 34, 35]. Probabilistic programs go beyond the presented framework to permit more expressive models than can be expressed as graphical models, incorporating recursive structures and higher-order functions. The combination of such frameworks with neural networks has recently been studied in Le et al. [22] and Ritchie et al. [28], indicating a promising avenue for further exploration.

References

[1] Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. Modeling biological processes for reading comprehension. In *EMNLP*, 2014.

[2] 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*, pages 2980–2988, 2015.

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

[4] S. M. Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, Koray Kavukcuoglu, and Geoffrey. E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. *arXiv preprint arXiv:1603.08575*, 2016.

[5] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.

[6] Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In *CogSci*, 2014.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[8] N. D. Goodman and A. Stuhlmüller. *The Design and Implementation of Probabilistic Programming Languages*. 2015.

[9] ND Goodman, VK Mansinghka, D Roy, K Bonawitz, and JB Tenenbaum. Church: A language for generative models. In *Uncertainty in Artificial Intelligence*, pages 220–229, 2008.

[10] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1462–1471, 2015.

[11] Varun Jampani, S. M. Ali Eslami, Daniel Tarlow, Pushmeet Kohli, and John Winn. Consensus message passing for layered graphical models. In *International Conference on Artificial Intelligence and Statistics*, pages 425–433, 2015.

[12] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[13] 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 *Advances in Neural Information Processing Systems*, pages 2946–2954, 2016.

[14] Matthew J. Johnson, David K. Duvenaud, Alex B. Wiltschko, Sandeep R. Datta, and Ryan P. Adams. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems*, 2016.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

[16] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations*, 2014.

[17] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

[18] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[19] Tejas D Kulkarni, Pushmeet Kohli, Joshua B Tenenbaum, and Vikash Mansinghka. Picture: A probabilistic programming language for scene perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4390–4399, 2015.

[20] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2530–2538, 2015.

[21] Steffen L Lauritzen and David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 157–224, 1988.

[22] Tuan Anh Le, Atilim Gunes Baydin, and Frank Wood. Inference compilation and universal probabilistic programming. *arXiv preprint arXiv:1610.09900*, 2016.

[23] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016.

[24] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[25] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, number 2, page 5, 2011.

[26] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and Raiko. T. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3532–3540, 2015.

[27] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1278–1286, 2014.

[28] Daniel Ritchie, Paul Horsfall, and Noah D Goodman. Deep amortized inference for probabilistic programs. *arXiv preprint arXiv:1610.05735*, 2016.

[29] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, pages 3510–3522, 2015.

[30] N. Siddharth, A. Barbu, and J. M. Siskind. Seeing what you're told: Sentence-guided activity recognition in video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 732–39, June 2014.

[31] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3465–3473, 2015.

[32] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, 2016.

[33] Andreas Stuhlmüller, Jacob Taylor, and Noah Goodman. Learning stochastic inverses. In *Advances in neural information processing systems*, pages 3048–3056, 2013.

[34] David Wingate, Andreas Stuhlmueller, and Noah D Goodman. Lightweight implementations of probabilistic programming languages via transformational compilation. In *International Conference on Artificial Intelligence and Statistics*, pages 770–778, 2011.

[35] Frank Wood, Jan Willem van de Meent, and Vikash Mansinghka. A new approach to probabilistic programming inference. In *Artificial Intelligence and Statistics*, pages 1024–1032, 2014.