

A Factorial Mixture Prior for Compositional Deep Generative Models

Ulrich Paquet
DeepMind

Sumedh K. Ghaisas
DeepMind

Olivier Tielemans
DeepMind

Abstract

We assume that a high-dimensional datum, like an image, is a compositional expression of a set of properties, with a complicated non-linear relationship between the datum and its properties. This paper proposes a factorial mixture prior for capturing latent properties, thereby adding structured compositionality to deep generative models. The prior treats a latent vector as belonging to Cartesian product of subspaces, each of which is quantized separately with a Gaussian mixture model. Some mixture components can be set to represent properties as observed random variables whenever labeled properties are present. Through a combination of stochastic variational inference and gradient descent, a method for learning how to infer discrete properties in an unsupervised or semi-supervised way is outlined and empirically evaluated.

1. Introduction

An image \mathbf{x} is often compositional in its properties or attributes. We might generate a random portrait \mathbf{x} by first picking discrete properties that are simple to describe, like whether the person should be male, whether the person should be smiling, whether the person should wear glasses, and maybe whether the person should have one of a discrete set of hair colors. In the same vein, we might include discrete properties that are harder to describe, but are still present in typical portraits. These are properties that we wish to discover in an unsupervised way from data, and may represent background textures, skin tone, lighting, or any property that we didn't explicitly enumerate.

This paper proposes a factorial mixture prior for capturing such properties, and presents a method for learning how to infer discrete properties in an unsupervised or semi-supervised way. The outlined framework adds structured compositionality to deep generative models.

To make the example of learning to render a portrait \mathbf{x} concrete, let each property $i = 1, \dots, I$, be it one that we

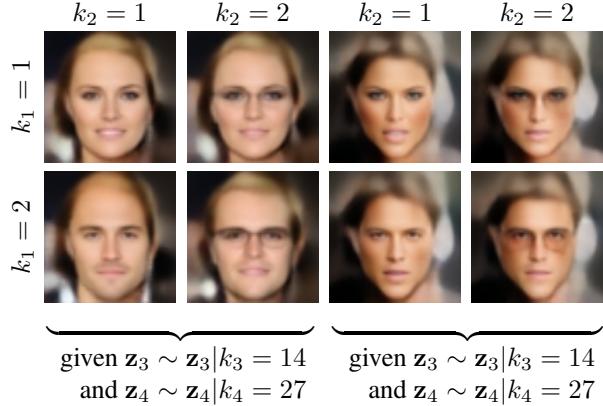


Figure 1. Samples from a deep generative model $p_\theta(\mathbf{x}|\mathbf{z})$. For the left and right blocks of faces, we sampled \mathbf{z}_3 and \mathbf{z}_4 once, and constructed \mathbf{z} as a Cartesian product $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4]$. The remaining \mathbf{z}_1 and \mathbf{z}_2 vectors were iteratively filled in by sampling $\mathbf{z}_1|k_1$ and $\mathbf{z}_2|k_2$ for all (k_1, k_2) settings. With background, skin tone and other properties being kept constant in the left and right blocks, we mark a visible change in gender in the two settings of latent variable k_1 . The presence or absence of glasses on a portrait \mathbf{x} is captured by latent variable k_2 . Following the discussion in Section 2.5, with semi-supervision, mixture components $(k_1 = 2, k_2 = 2)$ are responsible for rendering (male, glasses) portraits.

can describe or not, take values $k_i \in \{1, \dots, K_i\}$. Given a sampled latent value k_i , we generate a real-valued vector $\mathbf{z}_i \in \mathbb{R}^D$ from a mixture component k_i in a different K_i -component mixture model for each property i . The \mathbf{z}_i -vectors are concatenated into a latent vector $\mathbf{z} = [\mathbf{z}_1, \dots, \mathbf{z}_I]$, from which \mathbf{x} is generated according to any deep generative model $p_\theta(\mathbf{x}|\mathbf{z})$ that is parameterized by θ .

The parameters of the aforementioned generative process can be trained wholly unsupervised, and a factorial structure of clustered latent properties will naturally emerge if it exists in the training data. However, unsupervised learning provides no guarantee that some \mathbf{z}_i encodes and models a labelled property of \mathbf{x} . Some semi-supervised learning is required to associate a named property with some \mathbf{z}_i . A few training data points might be accompanied with sets of labels, an example set being $\mathbf{y} = \{\text{female}, \text{no_glasses}\}$. We may choose which labels in the set to map to properties

$i = 1, 2, \dots$, and then when present in a training example, use these labels in a semi-supervised way.

Continuing the illustrative example, let k_1 index the gender of portrait \mathbf{x} and k_2 whether glasses appear in \mathbf{x} or not, to yield two semi-supervised latent clusters with $K_1 = K_2 = 2$ for \mathbf{z}_1 and \mathbf{z}_2 . In Figure 1, we let factors $i = 3, 4$ be free to capture other combinations of variation in \mathbf{x} . For a given unsupervised cluster pair (k_3, k_4) of latent properties, Figure 1 illustrates how portraits \mathbf{x} of males and females with and without glasses are sampled by iterating over k_1 and k_2 .

Despite the naive simplicity of the generative model, inferring the posterior distribution of \mathbf{z} and the properties' latent cluster assignments for a given \mathbf{x} is far from trivial. Next to inference stands parameter learning. Section 2 is devoted to the model, inference and learning. Section 3 traces the influences on this model, and related work from the body of literature on “untangled”, structured latent representations. Empirical evaluations follow in Section 4.

2. A Structured, Factorial Mixture Model

We observe data $\mathbf{X} = \{\mathbf{x}^{(n)}\}_{n=1}^N$, and model each $\mathbf{x}^{(n)}$ with a likelihood $p_\theta(\mathbf{x}^{(n)}|\mathbf{z}^{(n)})$, a deep and flexible function with parameters θ . A few data points might have labels, and we denote their indices by set Ω and labels by $\mathbf{Y} = \{\mathbf{y}^{(n)}\}_{n \in \Omega}$. We will ignore the labels for the time being, and return to them in Section 2.5 when we incorporate them as a semi-supervised signal to aid the prior factors in corresponding to named properties. Additionally, when clear from the context, we will omit superscripts n .

2.1. Factorial Mixture Prior on $\mathbf{z}^{(n)}$

We present a prior that decomposes \mathbf{z} in a discrete, structured way. With a subdivision of \mathbf{z} into sub-vectors \mathbf{z}_i , let each factor \mathbf{z}_i be generated by a separate Gaussian mixture model with K_i components with non-negative mixing weights π_i that sum to one, $\sum_{k=1}^{K_i} \pi_{ik} = 1$. Let factor i 's component k have a mean $\boldsymbol{\mu}_{ik}$ and diagonal precision matrix $\text{diag}(\boldsymbol{\alpha}_{ik})$, and define $\xi = \{\pi_i, \{\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik}\}_{k=1}^{K_i}\}_{i=1}^I$ to denote all prior parameters as random variables, so that $p(\mathbf{z}|\xi) = \prod_i \sum_k \pi_{ik} \mathcal{N}(\mathbf{z}_i; \boldsymbol{\mu}_{ik}, \text{diag}(\boldsymbol{\alpha}_{ik})^{-1})$. Each \mathbf{z}_i is augmented with a cluster assignment vector $\mathbf{r}_i \in \mathbb{R}^{K_i}$, a one-hot encoding of an assignment index at $k = k_i$, producing $\mathbf{r} = [\mathbf{r}_1, \dots, \mathbf{r}_I]$.

Seen differently, the indices (k_1, \dots, k_I) form a discrete product-quantized (PQ) code for $\mathbf{z} \in \mathbb{R}^{DI}$. PQ codes have been used with great success for indexing in computer vision (Jegou et al., 2011). It decomposes the vector space \mathbb{R}^{DI} into a Cartesian product of low dimensional subspaces \mathbb{R}^D , and quantizes each subspace separately, to represent a vector by its subspace quantization indices.

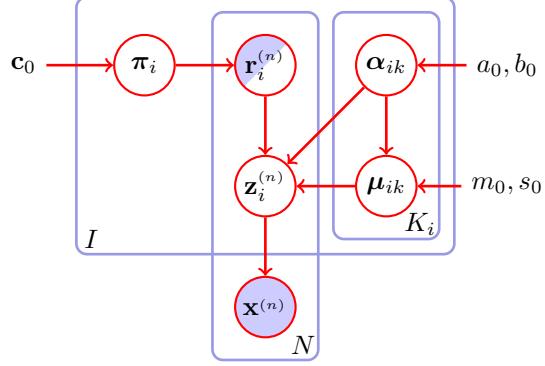


Figure 2. The joint density in (2), shown as a probabilistic graphical model. In the semi-supervised setting, some of the $\mathbf{r}_i^{(n)}$ random variables are also observed; see Section 2.5.

We let \mathbf{r}_i be drawn from a multinomial distribution that is parameterized by π_i , to yield the joint prior distribution

$$p(\mathbf{r}|\xi) p(\mathbf{z}|\mathbf{r}, \xi) = \prod_i \prod_k \pi_{ik}^{r_{ik}} \mathcal{N}(\mathbf{z}_i; \boldsymbol{\mu}_{ik}, \text{diag}(\boldsymbol{\alpha}_{ik})^{-1})^{r_{ik}}. \quad (1)$$

Our hope is that (k_1, \dots, k_I) provides a factorial representation of \mathbf{x} , with the ability to choose and combine discrete concepts k_i , then generate \mathbf{z}_i 's from those discrete latent variables, and finally to model \mathbf{x} from the concatenation of the \mathbf{z}_i 's.

2.2. Hierarchical Model

In (1), the prior is written as $p(\mathbf{r}, \mathbf{z}|\xi)$, where ξ are random variables (Johnson et al., 2016). As random variables, the hyperprior $p(\xi)$ is judiciously chosen from a conjugate exponential family of distributions, and the rich history of variational Bayes from the 1990's and 2000's is drawn on to additionally infer the posterior distribution of ξ (Johnson et al., 2016). An alternative, not pursued here, would have been to treat ξ as model parameters, writing the prior as $p_\xi(\mathbf{r}, \mathbf{z})$ (Fraccaro et al., 2017).

With ξ in (1) being random variables (that are integrated out when any marginal distributions are determined), they are *a priori* modeled with a hyperprior $p(\xi)$. In particular, each Gaussian mean and precision is independently governed by a conjugate Normal-Gamma hyperprior, $p(\mu_{ikd}, \alpha_{ikd}) = \mathcal{N}(\mu_{ikd}; m_0, (s_0 \alpha_{ikd})^{-1}) \mathcal{G}(\alpha_{ikd}; a_0, b_0)$, for $d = 1, \dots, D$. Our implementation uses $m_0 = 0$, $s_0 = 1$, and $a_0 = b_0 = 0.01$. The choice of a conjugate hyperprior becomes apparent when ξ is inferred from data using stochastic variational inference (Hoffman et al., 2013; Paquet & Koenigstein, 2013). For a similar reason, each mixing weights vector π_i is drawn from a Dirichlet distribution $p(\pi_i) = \mathcal{D}(\pi_i; \mathbf{c}_0)$ with pseudo-counts $\mathbf{c}_0 = \mathbf{1}$. The joint density over the observed random variables \mathbf{X} and hidden

random variables $\mathbf{Z} = \{\mathbf{z}^{(n)}\}_{n=1}^N$, $\mathbf{R} = \{\mathbf{r}^{(n)}\}_{n=1}^N$ and $\boldsymbol{\xi}$ is

$$p(\mathbf{X}, \mathbf{Z}, \mathbf{R}, \boldsymbol{\xi}) = \prod_n p_{\theta}(\mathbf{x}^{(n)} | \mathbf{z}^{(n)}) p(\mathbf{z}^{(n)} | \mathbf{r}^{(n)}, \boldsymbol{\xi}) p(\mathbf{r}^{(n)} | \boldsymbol{\xi}) \\ \cdot \prod_i p(\boldsymbol{\pi}_i) \prod_k p(\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik}), \quad (2)$$

and is illustrated as a probabilistic graphical model in Figure 2. Equation (2) contains local factors and global random variables, and the posterior approximation in Section 2.3 will contain local and global factors.

The model’s prior is related to an independent component analysis (ICA) prior on \mathbf{z} , after which a non-linear mixing function is applied to yield \mathbf{x} . It is also worth noting that $I = 1$ yields Johnson et al. (2016)’s Gaussian mixture model (GMM) structured variational auto-encoder (SVAE), and we generalise here from a single GMM to ICA-style component-wise mixtures. When $\boldsymbol{\xi}$ is treated as parameters (and not random variables) in $p_{\boldsymbol{\xi}}(\mathbf{r}, \mathbf{z})$, and I set to one, we obtain Jiang et al. (2017)’s variational deep embedding (VaDE) model. Further connections to related work are made in Section 3.

2.3. Posterior Approximation

We approximate the posterior with a fully factorized distribution

$$p(\mathbf{Z}, \mathbf{R}, \boldsymbol{\xi} | \mathbf{X}) \approx \underbrace{\prod_n q_{\phi}(\mathbf{z}^{(n)} | \mathbf{x}^{(n)})}_{q_{\phi}(\mathbf{Z} | \mathbf{X})} \underbrace{\prod_n q_{\gamma}(\mathbf{r}^{(n)})}_{q_{\gamma}(\mathbf{R})} \\ \cdot \underbrace{\prod_i q_{\lambda}(\boldsymbol{\pi}_i) \prod_k q_{\lambda}(\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik})}_{q_{\lambda}(\boldsymbol{\xi})}. \quad (3)$$

An “encoding network” or “inference network” $q_{\phi}(\mathbf{z} | \mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\phi}(\mathbf{x}), \text{diag } \boldsymbol{\sigma}_{\phi}^2(\mathbf{x}))$ amortizes inference for \mathbf{z} , and maps $\mathbf{x}^{(n)}$ to a Gaussian distribution on $\mathbf{z}^{(n)}$; both $\boldsymbol{\mu}_{\phi}(\mathbf{x})$ and $\boldsymbol{\sigma}_{\phi}^2(\mathbf{x})$ are deep neural networks (Kingma & Welling, 2014). We approximate $q_{\gamma}(\mathbf{R})$ and $q_{\lambda}(\boldsymbol{\xi})$ with a “structured VAE” approach (Johnson et al., 2016).

There is a local approximation $q_{\gamma}(\mathbf{r}_i^{(n)}) = \prod_k [\gamma_{ik}^{(n)}]^{r_{ik}^{(n)}}$ for every data point, yielding a “responsibility” for block i through $\sum_k \gamma_{ik} = 1$. Furthermore, $q_{\gamma}(\mathbf{r}^{(n)}) = \prod_i q_{\gamma}(\mathbf{r}_i^{(n)})$. Define $\boldsymbol{\gamma} = \{\{\gamma_i^{(n)}\}_{i=1}^I\}_{n=1}^N$ as parameters of $q(\mathbf{R})$. Maximization for the local factor $q_{\gamma}(\mathbf{r}^{(n)})$ is semi-amortized; we’ll see in (5) that it can be obtained as an analytic, closed form expression. In (5) it is a softmax over cluster assignments, and could be interpreted as another “encoding network”, this time for the one-hot discrete representations \mathbf{r}_i .¹

¹ Here, there is room for splitting hairs. We opt for uncluttered notation $q_{\gamma}(\mathbf{r}_i^{(n)})$, following the traditional convention in the variational Bayes (VB) literature. According to (5), the ELBO is locally

To express $q_{\lambda}(\boldsymbol{\xi})$, we parameterize each of the factors

$$q_{\lambda}(\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik}) = \prod_d q(\mu_{ikd}, \alpha_{ikd}) \\ = \prod_d \mathcal{N}(\mu_{ikd}; m_{ikd}, (s_{ikd}\alpha_{ikd})^{-1}) \mathcal{G}(\alpha_{ikd}; a_{ikd}, b_{ikd})$$

as a product of Normal-Gamma distributions, in the same form as the prior. Lastly, let $q_{\lambda}(\boldsymbol{\pi}_i)$ be a Dirichlet distribution parameterized by its pseudo-counts vector \mathbf{c}_i .

We’ve stated the factors in terms of their *mean parameters* $\{\{\mathbf{m}_{ik}, \mathbf{s}_{ik}, \mathbf{a}_{ik}, \mathbf{b}_{ik}\}_{k=1}^{K_i}, \mathbf{c}_i\}_{i=1}^I$ here because we require the mean parameters in (6). However, instead of doing gradient descent in the mean parameterization, we will view $q_{\lambda}(\boldsymbol{\xi})$ through the lens of its *natural parameters* $\boldsymbol{\lambda}$. This parameterization has pleasing properties, as it allows gradients to be conveniently expressed as natural gradients; a gradient step of length one gives an update to a local minimum (for a batch) (Paquet, 2015). This is also the framework required for stochastic variational inference (SVI) (Hoffman et al., 2013). Here the exponential family representation of $q_{\lambda}(\boldsymbol{\xi})$ is minimal and there exists a one-to-one mapping between the mean parameters and natural parameters $\boldsymbol{\lambda}$ (Wainwright & Jordan, 2008); we state the mapping in Appendix B.

2.4. Inference and Learning

The posterior approximation depends on $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, $\boldsymbol{\lambda}$, and $\boldsymbol{\gamma}$. They will be found by maximizing a variational lower bound to $\log p_{\theta}(\mathbf{X}) \geq \mathbb{E}_q[\log p_{\theta}(\mathbf{X}, \mathbf{Z}, \mathbf{R}, \boldsymbol{\xi}) - \log q_{\phi}(\mathbf{Z} | \mathbf{X}) q_{\gamma}(\mathbf{R}) q_{\lambda}(\boldsymbol{\xi})] = \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\lambda}, \boldsymbol{\gamma})$, also referred to as the “evidence lower bound” (ELBO):

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\lambda}, \boldsymbol{\gamma}) &= \sum_n \left\{ \mathbb{E}_{q_{\phi}(\mathbf{z}^{(n)} | \mathbf{x}^{(n)})} [\log p_{\theta}(\mathbf{x}^{(n)} | \mathbf{z}^{(n)})] \right. \\ &\quad \left. - \sum_{i,k} \underbrace{\mathbb{E}_{q_{\gamma}(\mathbf{r}_i^{(n)})} [r_{ik}^{(n)}]}_{\gamma_{ik}^{(n)} \text{ from (5) at max}} \mathbb{E}_{q_{\lambda}(\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik})} [\dots \right. \\ &\quad \left. \dots \text{KL}\left(q_{\phi}(\mathbf{z}_i^{(n)} | \mathbf{x}^{(n)}) \parallel p(\mathbf{z}_i^{(n)} | \boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik})\right)] \right\} \end{aligned} \quad (4a)$$

$$(4b) - \sum_i \mathbb{E}_{q_{\lambda}(\boldsymbol{\pi}_i)} [\text{KL}\left(q_{\gamma}(\mathbf{r}_i^{(n)}) \parallel p(\mathbf{r}_i^{(n)} | \boldsymbol{\pi}_i)\right)] \quad (4c)$$

$$- \frac{1}{N} \sum_{i,k} \text{KL}\left(q_{\lambda}(\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik}) \parallel p(\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik})\right) \quad (4d)$$

$$- \frac{1}{N} \sum_i \text{KL}\left(q_{\lambda}(\boldsymbol{\pi}_i) \parallel p(\boldsymbol{\pi}_i)\right) \} . \quad (4e)$$

The full derivation is provided in Appendix A. The contribution of the hyperpriors is equally divided over all data points

maximized at an expression for $q_{\gamma}(\mathbf{r}_i^{(n)})$ that depends on $\boldsymbol{\phi}$, $\boldsymbol{\lambda}$ and $\mathbf{x}^{(n)}$. One might also make the “amortized” nature of these local factors clearer by writing the factors as $q_{\phi\lambda}(\mathbf{r}_i^{(n)} | \mathbf{x}^{(n)})$.

through $\sum_n \frac{1}{N}$ in lines (4d) and (4e), so that the outer sum is over n . This aids stochastic gradient descent, and appropriately weighs $q_{\lambda}(\xi)$'s gradients as part of mini-batches of \mathbf{X} . The objective in (4a) to (4e) forms a structured VAE (Johnson et al., 2016).

There are four Kullback-Leibler (KL) divergences in \mathcal{L} . For each data point and factor i , (4b) contains the KL divergence between $q_{\phi}(\mathbf{z}_i|\mathbf{x})$ and $p(\mathbf{z}_i|\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik})$. This is the familiar local KL divergence between the encoder and the prior. Because ξ is inferred, the KL is averaged over the (approximate) posterior uncertainty $q_{\lambda}(\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik})$. With the allocation of \mathbf{x} to clusters (k_1, \dots, k_I) also being inferred, the smoothed KL is further weighted in a convex combination over $\mathbb{E}_q[r_{ik}]$. A second local KL cost is incurred in line (4c), where discrete random variables \mathbf{r} encode information about \mathbf{x} . Finally, lines (4d) and (4e) penalize the global posterior approximation $q_{\lambda}(\xi)$ for moving from the prior $p(\xi)$. The KL divergence between two Normal-Gamma distributions is given in Appendix C.1.

Objective \mathcal{L} can be optimized using three different techniques, spanning the last three decades of progress in variational inference. This combination has already been employed by a number of authors recently (Johnson et al., 2016; Lin et al., 2018). Given a mini-batch \mathcal{B} of data points n at iteration τ , the following parameter updates are followed:

1. \mathcal{L} is locally maximized with respect to $\gamma_i^{(n)}$ with an analytic, closed form expression. In an Expectation-Maximization (EM) algorithm, this is the variational Bayes E-step (Attias, 1999; Waterhouse et al., 1996).
2. Given a batch's $\{\gamma_i^{(n)}\}$, a stochastic gradient descent step updates θ and ϕ , for example using Kingma & Ba (2015)'s optimizer. When line (4b) is expressed analytically using (6), the reparameterization trick can be readily used (Kingma & Welling, 2014).
3. After recomputing $\{\gamma_i^{(n)}\}$ at their new local optima using the updated θ and ϕ , a SVI natural gradient step is taken on λ , using a decreasing Robbins-Monro step size (Hoffman et al., 2013). The updates on θ , ϕ and λ together form a stochastic M-step.

In practice, we initialize θ and ϕ by first running a number of gradient updates, using an ELBO with a $\mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$ prior, possibly with an annealed KL term. That gives an initial encoder $p_{\phi}(\mathbf{z}|\mathbf{x})$. Parameters λ are then initialized by repeating steps 1 and 3, to seed a basic factorized clustering of the encoded \mathbf{z} 's. We consider steps 1–3 separately below:

2.4.1. LOCALLY MAXIMIZING OVER $\gamma^{(n)}$

The local maximum of \mathcal{L} over $q_{\gamma}(\mathbf{r}_i^{(n)})$ is the softmax

$$\gamma_{ik} = \tilde{\gamma}_{ik} / \sum_k \tilde{\gamma}_{ik} \quad (5)$$

$$\begin{aligned} \tilde{\gamma}_{ik} = \exp\{ & \mathbb{E}_{q_{\phi}(\mathbf{z}_i|\mathbf{x}) q_{\lambda}(\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik})} [\log p(\mathbf{z}_i|\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik})] \\ & + \mathbb{E}_{q_{\lambda}(\boldsymbol{\pi}_i)} [\log \pi_{ik}] \} \end{aligned}$$

(dropping superscripts n). If the encoding network $q_{\phi}(\mathbf{z}|\mathbf{x})$ yields a Gaussian distribution, we can determine the above expectation required for (5) analytically:

$$\begin{aligned} & \mathbb{E}_{q_{\phi}(\mathbf{z}_i|\mathbf{x}) q_{\lambda}(\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik})} [\log \mathcal{N}(\mathbf{z}_i; \boldsymbol{\mu}_{ik}, \text{diag}(\boldsymbol{\alpha}_{ik})^{-1})] \\ &= \frac{1}{2} \sum_d \left(\psi(a_{ikd}) - \log b_{ikd} - \log(2\pi) - \frac{1}{s_{ikd}} \right. \\ & \quad \left. - \frac{a_{ikd}}{b_{ikd}} ((\boldsymbol{\mu}_{\phi}(\mathbf{x})_{d'} - m_{ikd})^2 + \sigma_{\phi}^2(\mathbf{x})_{d'}) \right) \quad (6) \end{aligned}$$

where index $d' = d + (i-1)D$ is the index into the matching sub-vector in \mathbf{z} , and $\psi(\cdot)$ indicates the digamma function. Furthermore, $\mathbb{E}_{q_{\lambda}(\boldsymbol{\pi}_i)} [\log \pi_{ik}] = \psi(c_{ik}) - \psi(\sum_k c_{ik})$. The expression in (6) appears in line (4b), so that the KL divergence is expressed exactly when stochastic gradients over θ and ϕ are computed.

Aside from showing that the maximum of \mathcal{L} with respect to $q_{\gamma}(\mathbf{r}_i^{(n)})$ is analytically tractable, (5) is a softmax classification into expected cluster allocations. Semi-supervised labels could be used to guide $q_{\phi}(\mathbf{z}_i|\mathbf{x})$ —since it appears in analytic form in (6)—towards encodings for which the expected cluster allocations correspond to labelled properties or attributes; we turn to this theme in (8) in Section 2.5.

2.4.2. STOCHASTIC GRADIENTS OF θ, ϕ

Lines (4a) and (4b) depend on θ and ϕ , and the “reparameterization trick” can be used with a stochastic gradient optimizer (Kingma & Ba, 2015; Kingma & Welling, 2014).

2.4.3. CONJUGATE, NATURAL GRADIENTS OF λ

We provide only a sketch of natural gradients and their use in SVI here, and provide the derivations and detailed expressions of the gradients in Appendices B and D.

At iteration τ , for mini-batch \mathcal{B} of data points, \mathcal{L} is analytically minimized with respect to λ to yield λ^* . If $\lambda^{(\tau)}$ denotes the natural parameters at iteration τ , then the natural gradient is $\widehat{\nabla}_{\lambda^{(\tau)}} \mathcal{L}(\lambda) = \lambda^* - \lambda^{(\tau)}$ (Paquet, 2015). We update $\lambda^{\tau+1} \leftarrow \lambda^{\tau} + \rho_{\tau} \widehat{\nabla}_{\lambda^{\tau}} \mathcal{L}(\lambda)$, which is equivalent to updating it to the convex combination between the currently tracked minimizer $\lambda^{(\tau)}$ and the batch's minimizer λ^* ,

$$\lambda^{(\tau+1)} \leftarrow (1 - \rho_{\tau}) \lambda^{(\tau)} + \rho_{\tau} \lambda^*. \quad (7)$$

This parameter update is a rewritten form of updating $\lambda^{(\tau)}$ with the natural gradient $\widehat{\nabla}_{\lambda^{(\tau)}} \mathcal{L}(\lambda)$ that is rescaled by ρ_{τ} (Hoffman et al., 2013). The step sizes should satisfy $\sum_{\tau=1}^{\infty} \rho_{\tau} = \infty$ and $\sum_{\tau=1}^{\infty} \rho_{\tau}^2 < \infty$. In our results $\rho_{\tau} = (\tau_0 + \tau)^{-\kappa}$ was used, with forgetting rate $\kappa \in (\frac{1}{2}, 1]$ and delay $\tau_0 \geq 0$. The updated mean parameters of $q_{\lambda}(\xi)$ are obtained from the updated natural parameters $\lambda^{(\tau+1)}$.

2.5. Semi-supervised Learning

For some data points $n \in \Omega$, we have access to labels $\mathbf{y}^{(n)}$. With appropriate preprocessing, we let $\mathbf{y}_i^{(n)}$ correspond to a one-hot encoding for property i , and let $i \in \Pi(n)$ denote the observed properties for data point n . Where present, we treat the labels as *observed random variables* $\mathbf{r}_i^{(n)} = \mathbf{y}_i^{(n)}$ in the probabilistic graphical model in (2) and Figure 2. The posterior approximation in (3), which is used in (4b) and (4c), hence includes pre-set delta functions $q_\gamma(\mathbf{r}_i^{(n)}) = \delta(\mathbf{r}_i^{(n)} - \mathbf{y}_i^{(n)})$. Certain marginals in the posterior approximation are thus clamped.

Merely clamping marginals may not be sufficient for ensuring that \mathbf{z}_1 encodes, for example, latent male or female properties in Figure 1. The encoder $q_\phi(\mathbf{z}|\mathbf{x})$ might still distribute these properties through *all* of \mathbf{z} , and decoder $p_\theta(\mathbf{x}|\mathbf{z})$ find parameters that recover \mathbf{x} from that \mathbf{z} . We would like the encoder to emit two linearly separable clusters for block \mathbf{z}_1 for male and female inputs, and define the expected cluster allocation $g_{ik}^{(n)} = \tilde{\gamma}_{ik}^{(n)} / \sum_{k'} \tilde{\gamma}_{ik'}^{(n)}$ like (5). To encourage a distributed representation of male and female primarily in \mathbf{z}_1 , we incorporate a cross entropy classification loss to $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\lambda}, \boldsymbol{\gamma})$,

$$\mathcal{F}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\lambda}, \boldsymbol{\gamma}) = \mathcal{L} + \Delta \sum_{n \in \Omega} \sum_{i \in \Pi(n)} \sum_k g_{ik}^{(n)} \log g_{ik}^{(n)}, \quad (8)$$

with a tunable scalar knob $\Delta \geq 0$.

On closer inspection, the ‘‘logits’’ of the $g_{ik}^{(n)}$ ’s are differentiable functions of the encoder, using the analytic expression in (6). For the parameters of the subset of $q_\lambda(\boldsymbol{\pi}_i)$ and $q_\lambda(\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik})$ factors that influence the semi-supervised loss term in (8), the closed-form natural gradient updates of Section 2.4.3 are no longer possible. When $\Delta > 0$, the M-step can employ first-order gradients in the mean parameterization of those factors.

As a final thought, we are free to set $\Delta = 0$ once we are satisfied that $q_\phi(\mathbf{z}|\mathbf{x})$ primarily represents a latent male and female signal in \mathbf{z}_1 , for instance. In that case we are left with a probabilistic graphical model in which some $\mathbf{r}_i^{(n)}$ are observed, hence still encouraging a separation of representation of named properties.

2.6. Sampling

We wish to sample from the marginal distribution of \mathbf{x} , conditional on all observed data \mathbf{X} . It is an intractable average over the posterior $p(\boldsymbol{\xi}|\mathbf{X})$,

$$p_\theta(\mathbf{x}|\mathbf{X}) = \sum_{\mathbf{r}} \int p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{r}, \boldsymbol{\xi}|\mathbf{X}) d\mathbf{z} d\boldsymbol{\xi} \\ = \sum_{\mathbf{r}} \int p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}|\mathbf{r}, \boldsymbol{\xi}) p(\mathbf{r}|\boldsymbol{\xi}) p(\boldsymbol{\xi}|\mathbf{X}) d\mathbf{z} d\boldsymbol{\xi}, \quad (9)$$

which we approximate by substituting $p(\boldsymbol{\xi}|\mathbf{X})$ with $q(\boldsymbol{\xi})$. Call the distribution resulting from this substitution $p_Q(\mathbf{x}) \approx p_\theta(\mathbf{x}|\mathbf{X})$. To sample from $p_Q(\mathbf{x})$, one-hot cluster indices are first sampled with $\boldsymbol{\pi}_i \sim q_\lambda(\boldsymbol{\pi}_i)$ and $\mathbf{r}_i \sim \mathcal{C}(\boldsymbol{\pi}_i)$, a categorical distribution with $\boldsymbol{\pi}_i$ as its event probabilities. Then, let k_i be the non-zero index of \mathbf{r}_i , and sample $\mathbf{z}_i \sim q_\lambda(\mathbf{z}_i|k_i) = \int p(\mathbf{z}_i|\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik}) q_\lambda(\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik}) d\boldsymbol{\mu}_{ik} d\boldsymbol{\alpha}_{ik}$. Each $z_{d'}$ is drawn from a heavy-tailed Student-t distribution with $2a_{ikd}$ degrees of freedom

$$q_\lambda(z_{d'}|k_i) = \mathcal{T}\left(z_{d'}; m_{ikd}, \frac{s_{ikd} + 1}{s_{ikd}} \frac{b_{ikd}}{a_{ikd}}, 2a_{ikd}\right), \quad (10)$$

where $d' = d + (i-1)D$ indexes block i ’s entry d . Finally, sample $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$.

2.7. Predictive Log Likelihood

The predictive likelihood of a new data point \mathbf{x} is estimated with a lower bound to $p_Q(\mathbf{x})$, which decomposes with two terms containing KL divergences,

$$\begin{aligned} \log p_Q(\mathbf{x}) &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})] \\ &- \underbrace{\mathbb{E}_{q_\lambda(\boldsymbol{\mu}, \boldsymbol{\alpha})} \left[\sum_{i,k} \mathbb{E}_q[r_{ik}] \text{KL}\left(q_\phi(\mathbf{z}_i|\mathbf{x}) \parallel p(\mathbf{z}_i|\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik})\right) \right]}_{\text{KL}[\mathbf{z}]} \\ &- \underbrace{\mathbb{E}_{q_\lambda(\boldsymbol{\pi})} \left[\sum_i \text{KL}\left(q_\gamma(\mathbf{r}_i) \parallel p(\mathbf{r}_i|\boldsymbol{\pi}_i)\right) \right]}_{\text{KL}[\mathbf{r}]} . \end{aligned} \quad (11)$$

The bound in (11) is first maximized over $q(\mathbf{r}_i)$, using (5), before being evaluated. In our results, we consider the tradeoff between encoding in \mathbf{z} and in \mathbf{r} , by evaluating the two terms $\text{KL}[\mathbf{z}]$ and $\text{KL}[\mathbf{r}]$.

3. Related Work

The factorial representation in this paper is a collection of ‘‘Gaussian mixture model structured VAEs (SVAEs)’’, and our optimization scheme in Section 2.4 mirrors that of Algorithm 1 in (Johnson et al., 2016). In a larger historical context, the model has roots in variational approximations for mixtures of factor analyzers (Ghahramani & Beal, 2000). In our work, the prior $p(\mathbf{z}|\mathbf{r}, \boldsymbol{\xi})$ is a Gaussian on \mathbf{z} , and so is $q_\phi(\mathbf{z}|\mathbf{x})$. As the product of two Gaussians yields an unnormalized Gaussian (with closed form normalizer), the prior structure can further be incorporated in the inference network of a VAE (Lin et al., 2018).

When the conjugate hyperprior is ignored and $\boldsymbol{\xi}$ treated as parameters, and a single mixture set at $I = 1$, we recover the model of (Jiang et al., 2017). Alternatively, the ‘‘stick-breaking VAE’’ uses a stick-breaking prior on $\boldsymbol{\pi}$ for $I = 1$,

and treats all other ξ as parameters (Nalisnick & Smyth, 2017; Nalisnick et al., 2016). The Gaussian mixture VAE (Dilokthanakul et al., 2016) uses deep networks to transform a sample from a unit-variance isotropic Gaussian into a the mean and variance for each mixture component of a Gaussian mixture model prior on \mathbf{z} . Instead of inferring separate means and variances of the mixture components, they are learned non-linear transformations of the same random variable. Another variation on the mixture-theme, the “variational mixture of posteriors” prior writes the mixture prior as a mixture of inference networks, conditioned on learnable pseudo-inputs (Tomczak & Welling, 2018). In (Graves et al., 2018), the notion of a mixture prior is taken to the extreme, storing the approximate posterior distributions of every element in the training set as mixture components, and using hard k -nearest neighbour lookups to determine the cluster assignment for a new \mathbf{x} .

The vector-quantized VAE (VQ-VAE) model (van den Oord et al., 2017) is similar to the work presented here, with these differences: here, there are three types random variables, $\mathbf{r}^{(n)}$ (discrete) together with $\mathbf{z}^{(n)}$ and ξ , while VQ-VAE consists only of discrete random variable (a form of $\mathbf{r}^{(n)}$). VQ-VAE introduces a “stop-gradient”, which on the surface resembles an E-step in an EM algorithm (see Section 2.4).

A growing body of literature exists around obtaining “untangled” representations from unsupervised models. These range from bootstrapping partial supervised data (Narayanaswamy et al., 2017), changing the relative contribution of the KL term(s) in the ELBO (Higgins et al., 2017) to total correlation penalties (Chen et al., 2018b; Kim & Mnih, 2018) to specialized domain-adverserial training on the latent space (Lample et al., 2017). In this work, the only additional penalties that are added to the ELBO appear in the form of the semi-supervised loss term in (8). The subdivision of data into a discrete and somewhat interpretable Cartesian product of clusters is purely the result of a hierarchical mixture model. The model presented in this paper can be viewed as a VAE with latent code vector \mathbf{z} with a “KD encoding” prior (Chen et al., 2018a), learned in a Bayesian probabilistic way.

Instead of letting \mathbf{y} be parent random variables of \mathbf{z} , they can be treated as “side information” whose prediction helps recover an interpretable common latent variable \mathbf{z}^* . Adel et al. (2018)’s “interpretable lens variable model” constructs an invertible mapping (normalizing flow) between \mathbf{z} and \mathbf{z}^* , and uses it to add an interpretable layer to, for example, pre-trained $p_\theta(\mathbf{x}|\mathbf{z})$ ’s and $q_\phi(\mathbf{z}|\mathbf{x})$ ’s. Like this work, $|\Omega|$ might be much smaller than N .

4. Experimental results

We expand the example of Figure 1, by illustrate a factorization of CelebA faces into a product of cluster indices (k_1, \dots, k_5) , the first three of which are interpretable. Using the binary MNIST and Omniglot data sets, we empirically show the trade-off between costs for representing \mathbf{x} as discrete \mathbf{r} and continuous \mathbf{z} latent variables in the predictive log likelihood (11), as K varies.

4.1. Binary MNIST and Omniglot

For both the binary MNIST (Salakhutdinov & Murray, 2008) and Omniglot (Lake et al., 2015) data sets, we use a conventional convolutional architecture for the encoder and decoder, taken from Gulrajani et al. (2017). The encoder is a convolutional neural network (CNN) with 8 layers of kernel size 3, alternating strides of 1 and 2, and ReLU activations between the layers. The number of channels is 32 in the first 3 layers, then 64 in the last five layers. Lastly, a linear layer transforms the CNN output to a vector of length $2DI$, to produce the means $\mu_\phi(\mathbf{x})$ and log variances $\log \sigma_\phi^2(\mathbf{x})$ of the posterior approximation of \mathbf{z} . The decoder emits a Bernoulli likelihood for every pixel independently. Training details are provided in Appendix E.

To demonstrate the benefits of using a structured prior, we ran experiments on MNIST and Omniglot using $I = 1$, with an increasing number of mixture components K . (The data sets did not exhibit enough variability for additional factors $I > 1$ not to be pruned away (MacKay, 2001)). Figure 3 shows the resulting test ELBO, log-likelihood (negative reconstruction error) and KL-divergences from (11).

We decompose the ELBO in (11) into a log likelihood and two expected KL-divergences, $\text{KL}[\mathbf{z}]$ and $\text{KL}[\mathbf{r}]$. In Figure 3 we see the ELBO increase, and then seemingly converge, as model complexity increases. Following (11), the figure shows that there is a trade-off in the penalty that the encoder pays, between encoding into \mathbf{z} , and encoding into \mathbf{r} . As K increases, $\text{KL}[\mathbf{r}]$ increases and $\text{KL}[\mathbf{z}]$ decreases, pushing information about the encoding “higher up” in the representation. Beyond a certain K , proportionally less mixture components are used, reflected in a diminishing $\text{KL}[\mathbf{r}]$. We would expect Figure 3 to be smooth, and its variability can be ascribed to random seeds, sensible initialization of $q_\lambda(\xi)$ and local minima in the (θ, ϕ, λ) landscape of \mathcal{L} .

An ELBO of -85.77 at $K = 512$ on binarized MNIST represents a typical state-of-the-art result (≥ -87.4) for unconditional generative models using simple convolutional neural networks (Gulrajani et al., 2017). On Omniglot, our best test ELBO of -90.89 at $K = 256$ is comparable to some of the leading models of the day (-89.8 for the variational lossy auto-encoder in (Chen et al., 2017); ≥ -95.5 in (Rezende et al., 2016); ≥ -103.4 in (Burda et al., 2016)).

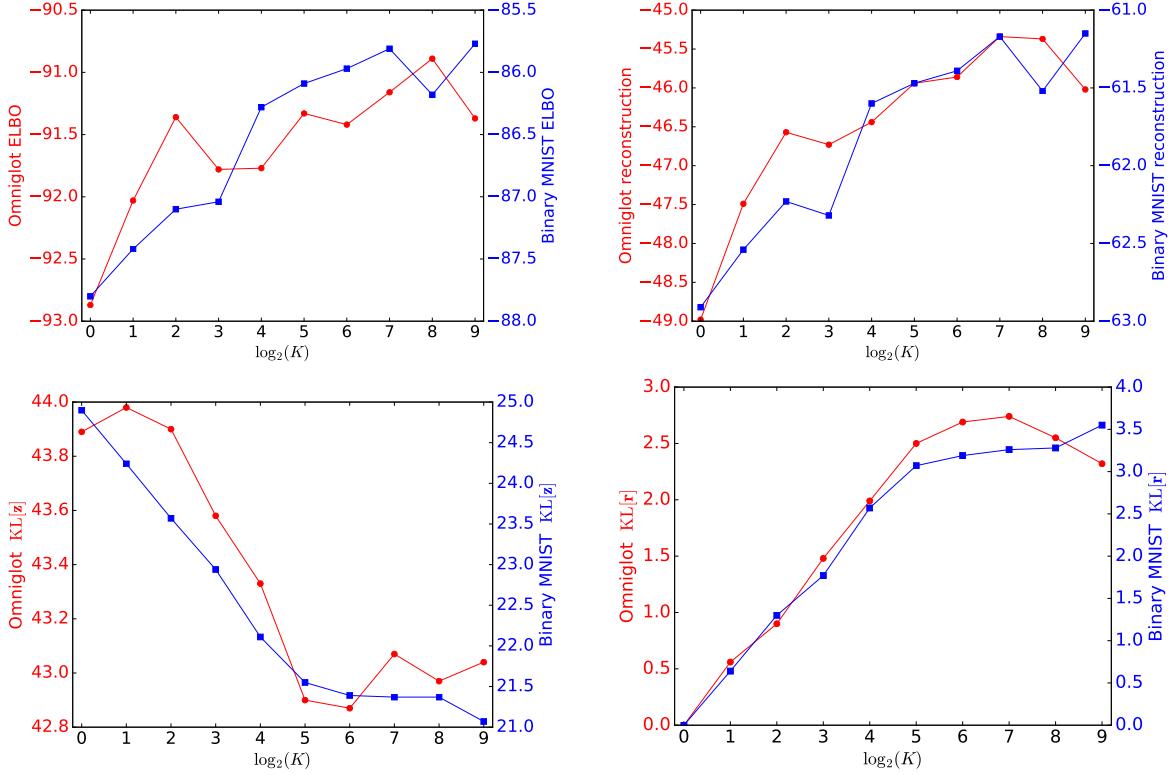


Figure 3. The ELBO of (11), averaged over respective test sets, as a function of $\log_2 K$. Generally, the cost $\text{KL}[\mathbf{z}]$ of encoding \mathbf{x} decreases with K , at the expense of a higher relative entropy $\text{KL}[\mathbf{r}]$. More mixture components aid the “reconstruction term” $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$, also shown as averaged over the respective test sets.



Figure 4. Omniglot cluster means, variances, and samples (for 6 out of $K = 32$ clusters).

In Figure 4 we inspect how the latent space clustering of Omniglot is reflected in input space. The figure shows the mean and variance of samples of six mixture components, obtained by repeatedly drawing latent space samples, decoding those, and computing the mean and variance of the resulting images. Additionally, five samples from each cluster are shown.



Figure 5. The mean, standard deviation, and a $p_\theta(\mathbf{x}|\mathbf{z})$ sample.

4.2. CelebA

The images $\mathbf{x}^{(n)}$ in the CelebA dataset are 64-by-64 pixels with 3 channels (RGB), with each element scaled to $[-1, 1]$. For semi-supervised training, we use the annotations of 40% of images, so that $|\Omega| = 0.4N$. Hence 40% of data points $\mathbf{x}^{(n)}$ have labels $\mathbf{y}^{(n)}$, for which we use the subset of gender, glasses and smiling labels as a semi-supervised signal. Our primary aim is to illustrate the factorial mixture prior capturing and encouraging compositional latent representations.

For CelebA, we use a 4-layer convolutional neural network with (128, 256, 512, 1024) channels, kernel size 3, and stride 2. The output of each layer is clipped with rectified linear units (ReLU), before being input to the next. As with MNIST and Omniglot, the final layer output is transformed with a linear layer to a vector of length $2DI$, to produce

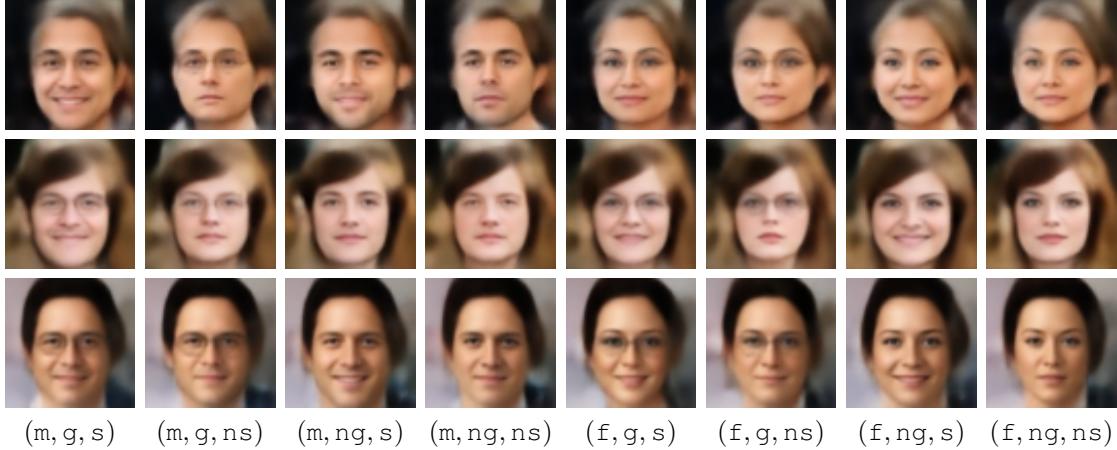


Figure 6. The means of $p_\theta(\mathbf{x}|\mathbf{z})$, with $\mathbf{z} \sim p_\lambda(\mathbf{z}|k_1, k_2)$ sampled from (10) using factorial mixture prior with $I = 5$ properties (samples from $p_\theta(\mathbf{x}|\mathbf{z})$ add position-dependent jitter to the mean; see Figure 5). Semi-supervised labels $k_1 \in \{\text{male, female}\}$, $k_2 \in \{\text{glasses, no glasses}\}$, and $k_3 \in \{\text{smiling, not smiling}\}$ are incorporated in learning, and the figure shows randomly generated faces using the same set-up as that of Figure 1.

the means $\mu_\phi(\mathbf{x})$ and log variances $\log \sigma_\phi^2(\mathbf{x})$ of \mathbf{z} . The decoder is a deconvolutional neural network whose architecture is the transpose of that of the encoder. The decoder is different for this data set, as it needs to model 3 color channels (RGB). The last layer of the deconvolutional network has 6 channels, which represent the mean and standard deviation of $p_\theta(\mathbf{x}|\mathbf{z})$ for each color channel for each pixel; see Figure 5. The standard deviations of p_θ are parameterized via a scaled sigmoid function, to be in $[0.001, 0.4]$.

Training started with a prior initialization phase for λ of 2×10^4 iterations, followed by a joint optimization phase of 3×10^5 iterations. The Adam optimizer (Kingma & Ba, 2015) with 10^{-4} learning rate was used for θ and ϕ . A forgetting rate $\kappa = -0.52$ and $\tau_0 = 2 \times 10^3$ was used for λ 's gradient updates. In training, we used $\Delta = 1000$ in (8), as a typical unscaled cross entropy loss in (8) is a few orders of magnitude smaller than the reconstruction term in (4a), which scales with the number of pixels and color channels. Empirically, we found that initially up-weighting the KL term in (4b) by a factor of 20 encouraged a crisper clustered representation to be learned (Higgins et al., 2017).

Figure 6 visually illustrates a model with $I = 5$ factors with $D = 64$, of which the first three are semi-supervised with two mixture components each, while the remainder are unsupervised with $K_4 = K_5 = 64$. For each row, we sample (k_4, k_5) once from $q_\lambda(\xi)$ and sample $[\mathbf{z}_4, \mathbf{z}_5]$ once from (10). By iterating over (k_1, k_2, k_3) settings and generating random faces from the model, the change in properties is perceptible as the latent code changes. Further results and examples are given in Appendix F.

To formally test the interpretability of samples like that in Figure 6, we uniformly sampled (k_1, k_2, k_3) from $\{1, 2\}^3$

Table 1. Human judgements of properties of generated \mathbf{x} 's, sampled according to Section 2.6 but with latent variables k_1 to k_3 clamped to intended, semi-supervised ground truth properties.

	Latent property	Human judgement
k_1	male / female	94%
k_2	glasses / no glasses	85%
k_3	smiling / not smiling	81%

as the model's “intention” to test all properties equally. It was appended with a (k_4, k_5) sample from $q(\pi)$, $\mathbf{z}|k_i$ was sampled according to (10) and the mean $p_\theta(\mathbf{x}|\mathbf{z})$ rendered. For 54 such random images, Table 1 evaluates shows the accuracy of human evaluations compared to their latent k_i -labels. Ten people of varying race and gender evaluated the generated images. The evaluation is subjective—the criteria for smiling need not be consistent with that of the creators of CelebA (Liu et al., 2015)—but gives evidence that the k_i 's capture interpretable attributes.

On the CelebA test set, the estimated cluster assignments $q_{\phi\lambda}(\mathbf{r}_i|\mathbf{x})$ were 97% for gender, 99% for the presence or absence of glasses, and 91% for whether the subject is smiling or not. Note that unlike the data evaluated for Table 1, labeled properties are not uniformly distributed over the test set.

Acknowledgements

We are grateful to three anonymous Neural Information Processing Systems reviewers, who provided invaluable suggestions for improving this paper.

References

- Adel, T., Ghahramani, Z., and Weller, A. Discovering interpretable representations for both deep generative and discriminative models. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 50–59, 2018.
- Attias, H. Inferring parameters and structure of latent variable models by variational Bayes. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 21–30, 1999.
- Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. In *International Conference on Learning Representations*, 2016.
- Chen, T., Min, M. R., and Sun, Y. Learning k-way d-dimensional discrete codes for compact embedding representations. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 854–863, 2018a.
- Chen, T. Q., Li, X., Grosse, R., and Duvenaud, D. Isolating sources of disentanglement in variational autoencoders. *arXiv:1802.04942*, 2018b.
- Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational lossy autoencoder. In *International Conference on Learning Representations*, 2017.
- Dilokthanakul, N., Mediano, P. A. M., Garnelo, M., Lee, M. C. H., Salimbeni, H., Arulkumaran, K., and Shanahan, M. Deep unsupervised clustering with Gaussian mixture variational autoencoders. *arXiv:1611.02648*, 2016.
- Fraccaro, M., Kamronn, S., Paquet, U., and Winther, O. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems 30*, pp. 3604–3613, 2017.
- Ghahramani, Z. and Beal, M. J. Variational inference for Bayesian mixtures of factor analysers. In *Advances in Neural Information Processing Systems 12*, pp. 449–455, 2000.
- Graves, A., Menick, J., and van den Oord, A. Associative compression networks. *arXiv:1804.02476*, 2018.
- Gulrajani, I., Kumar, K., Ahmed, F., Taiga, A. A., Visin, F., Vazquez, D., and Courville, A. Pixelvae: A latent variable model for natural images. In *International Conference on Learning Representations*, 2017.
- Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. Beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.
- Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347, 2013.
- Jegou, H., Douze, M., and Schmid, C. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.
- Jiang, Z., Zheng, Y., Tan, H., Tang, B., and Zhou, H. Variational deep embedding: An unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 1965–1972, 2017.
- Johnson, M., Duvenaud, D. K., Wiltschko, A., Adams, R. P., and Datta, S. R. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems 29*, pp. 2946–2954, 2016.
- Kim, H. and Mnih, A. Disentangling by factorising. *arXiv:1802.05983*, 2018.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.
- Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.
- Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.
- Lample, G., Zeghidour, N., Usunier, N., Bordes, A., Denoyer, L., and Ranzato, M. Fader networks: Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems 30*, pp. 5967–5976. 2017.
- Lin, W., Hubacher, N., and Khan, M. E. Variational message passing with structured inference networks. In *International Conference on Learning Representations*, 2018.
- Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.
- MacKay, D. J. C. Local minima, symmetry-breaking, and model pruning in variational free energy minimization. Technical report, University of Cambridge Cavendish Laboratory, 2001.
- Nalisnick, E. and Smyth, P. Stick-breaking variational autoencoders. In *International Conference on Learning Representations (ICLR)*, 2017.

- Nalisnick, E., Hertel, L., and Smyth, P. Approximate inference for deep latent Gaussian mixtures. In *NIPS Workshop on Bayesian Deep Learning*, volume 2, 2016.
- Narayanaswamy, S., Paige, T. B., van de Meent, J.-W., Desmaison, A., Goodman, N., Kohli, P., Wood, F., and Torr, P. Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems 30*, pp. 5925–5935. 2017.
- Paquet, U. On the convergence of stochastic variational inference in Bayesian networks. *arXiv:1507.04505*, 2015.
- Paquet, U. and Koenigstein, N. One-class collaborative filtering with random graphs. In *Proceedings of the 22nd International Conference on World Wide Web (WWW)*, pp. 999–1008. 2013.
- Rezende, D. J., Mohamed, S., Danihelka, I., Gregor, K., and Wierstra, D. One-shot generalization in deep generative models. *arXiv:1603.05106*, 2016.
- Salakhutdinov, R. and Murray, I. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 872–879, 2008.
- Tomczak, J. M. and Welling, M. VAE with a VampPrior. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.
- van den Oord, A., Vinyals, O., and Kavukcuoglu, K. Neural discrete representation learning. In *Advances in Neural Information Processing Systems 30*, pp. 6306–6315. 2017.
- Wainwright, M. J. and Jordan, M. I. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.
- Waterhouse, S. R., MacKay, D. J. C., and Robinson, A. J. Bayesian methods for mixtures of experts. In *Advances in Neural Information Processing Systems*, pp. 351–357, 1996.

A. Evidence Lower Bound

We bound the marginal likelihood of the entire data set—assuming $\mathbf{x}^{(n)}$'s are generated i. i. d.—with Jensen's inequality,

$$\begin{aligned}\log p(\mathbf{X}) &= \sum_{\mathbf{R}} \int p_{\theta}(\mathbf{X}|\mathbf{Z}, \mathbf{R}) p(\mathbf{R}|\xi) p(\xi) d\mathbf{Z} d\xi \\ &\geq \sum_{\mathbf{R}} \int q_{\phi}(\mathbf{Z}|\mathbf{X}) q(\mathbf{R}) q(\xi) \cdots \\ &\quad \cdots \log [p_{\theta}(\mathbf{X}|\mathbf{Z}, \mathbf{R}) p(\mathbf{R}|\xi) p(\xi)] d\mathbf{Z} d\xi \\ &\quad - \sum_{\mathbf{R}} \int q_{\phi}(\mathbf{Z}|\mathbf{X}) q(\mathbf{R}) q(\xi) \cdots \\ &\quad \cdots \log [q_{\phi}(\mathbf{Z}|\mathbf{X}) q(\mathbf{R}) q(\xi)] d\mathbf{Z} d\xi \\ &\stackrel{\text{def}}{=} \mathcal{L}(\theta, \phi, \lambda, \gamma).\end{aligned}$$

Using the joint distribution in (2), and the approximation in (3), we rewrite \mathcal{L} as

$$\begin{aligned}\mathcal{L} = \sum_n \left\{ \mathbb{E}_{q_{\phi}(\mathbf{z}^{(n)}|\mathbf{x}^{(n)})} [\log p_{\theta}(\mathbf{x}^{(n)}|\mathbf{z}^{(n)})] \right. \\ + \sum_i \mathbb{E}_{q(\mathbf{r}_i^{(n)})} \left[\sum_k r_{ik}^{(n)} \mathbb{E}_{q_{\phi}(\mathbf{z}_i^{(n)}|\mathbf{x}^{(n)})} \cdots \right. \\ \cdots \mathbb{E}_{q(\mu_{ik}, \alpha_{ik}) q(\pi_i)} [\log p(\mathbf{z}_i^{(n)}|\mu_{ik}, \alpha_{ik}) + \log \pi_{ik}] \\ - \sum_i \mathbb{E}_{q_{\phi}(\mathbf{z}_i^{(n)}|\mathbf{x}^{(n)})} [\log q_{\phi}(\mathbf{z}_i^{(n)}|\mathbf{x}^{(n)})] \\ - \sum_i \mathbb{E}_{q(\mathbf{r}_i^{(n)})} [\log q(\mathbf{r}_i^{(n)})] \\ + \frac{1}{N} \sum_{i,k} \mathbb{E}_{q(\mu_{ik}, \alpha_{ik})} [\log p(\mu_{ik}, \alpha_{ik})] \\ + \frac{1}{N} \sum_i \mathbb{E}_{q(\pi_i)} [\log p(\pi_i)] \\ - \frac{1}{N} \sum_{i,k} \mathbb{E}_{q(\mu_{ik}, \alpha_{ik})} [\log q(\mu_{ik}, \alpha_{ik})] \\ \left. - \frac{1}{N} \sum_i \mathbb{E}_{q(\pi_i)} [\log q(\pi_i)] \right\}.\end{aligned}$$

Notice that we split the contribution of the priors equally over all data points through $\sum_n \frac{1}{N}$: this is simply so that we can run a stochastic gradient descent algorithm and appropriately weigh gradients with respect to $q(\mu_{ik}, \alpha_{ik})$ in mini-batch steps. Rearranging \mathcal{L} gives the ELBO in (4a) to (4d).

The difference between a vanilla VAE and the model in this paper, is illustrated in Figure 7.

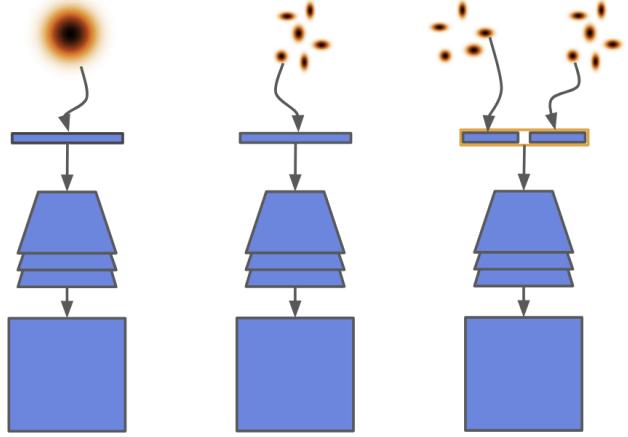


Figure 7. Left, standard VAE generation. Center, generation of \mathbf{x} using a Gaussian mixture model prior. Right, generation using a factorized structured prior.

B. Mean and Natural Parameters

Approximation (3) contains factors

$$\begin{aligned}q(\mu_{ikd}, \alpha_{ikd}) \\ = \mathcal{N}(\mu_{ikd}; m_{ikd}, (s_{ikd}\alpha_{ikd})^{-1}) \mathcal{G}(\alpha_{ikd}; a_{ikd}, b_{ikd})\end{aligned}$$

which are stated in terms of their mean parameters $\{m_{ikd}, s_{ikd}, a_{ikd}, b_{ikd}\}$. Instead of doing gradient descent in the mean parameterization, we will view $q_{\lambda}(\xi)$ through the lens of its *natural parameters* λ . As an example of this lens, the Normal-Gamma distribution comprises of an inner product between natural parameter vector λ_{ikd} and sufficient statistics $\mathbf{t}(\mu_{ikd}, \alpha_{ikd})$,

$$\begin{aligned}\log q(\mu, \alpha) &= [a - \frac{1}{2}, -(b + \frac{1}{2}sm^2), sm, -\frac{1}{2}s]^T \cdots \\ &\quad \cdots [\log \alpha, \alpha, \alpha\mu, \alpha\mu^2] + \text{const} \\ &= \lambda_{ikd}^T \mathbf{t}(\mu_{ikd}, \alpha_{ikd}) + \text{const}\end{aligned}\tag{12}$$

(dropping subscripts ikd for brevity, and grouping constants). The mapping from mean to natural parameters is therefore:

$$\begin{aligned}\lambda_{ikd,1} &= a_{ikd} - \frac{1}{2} \\ \lambda_{ikd,2} &= -(b_{ikd} + \frac{1}{2}s_{ikd}m_{ikd}^2) \\ \lambda_{ikd,3} &= s_{ikd}m_{ikd} \\ \lambda_{ikd,4} &= -\frac{1}{2}s_{ikd}.\end{aligned}$$

C. Kullback-Leibler Divergences

C.1. KL Divergence Between Normal-Gamma Distributions

In (4d), we are required to compute the KL divergence between two Normal-Gamma distributions. We first state

the prior Normal-Gamma distribution fully, for clarity:

$$\begin{aligned}
& p(\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik}) \\
&= \prod_d \mathcal{N}(\mu_{ikd}; m_0, (s_0 \alpha_{ikd})^{-1}) \mathcal{G}(\alpha_{ikd}; a_0, b_0) \\
&= \prod_d \frac{b_0^{a_0}}{\Gamma(a_0)} \sqrt{\frac{s_0}{2\pi}} \exp \left\{ \left(a_0 - \frac{1}{2} \right) \log \alpha_{ikd} \right. \\
&\quad \left. - b_0 \alpha_{ikd} - \frac{1}{2} s_0 \alpha_{ikd} (\mu_{ikd} - m_0)^2 \right\} \\
&= \prod_d \frac{b_0^{a_0}}{\Gamma(a_0)} \sqrt{\frac{s_0}{2\pi}} \exp \left\{ \left(a_0 - \frac{1}{2} \right) \underbrace{\log \alpha_{ikd}}_* \right. \\
&\quad \left. - \left(b_0 + \frac{1}{2} s_0 m_0^2 \right) \underbrace{\alpha_{ikd}}_* \right. \\
&\quad \left. + s_0 m_0 \underbrace{\alpha_{ikd} \mu_{ikd}}_* - \frac{1}{2} s_0 \underbrace{\alpha_{ikd} \mu_{ikd}^2}_* \right\}.
\end{aligned}$$

The sufficient statistics are indicated with a *, and their moments are

$$\begin{aligned}
\mathbb{E}[\log \alpha_{ikd}] &= \psi(a_0) - \log b_0 \\
\mathbb{E}[\alpha_{ikd}] &= \frac{a_0}{b_0} \\
\mathbb{E}[\alpha_{ikd} \mu_{ikd}] &= m_0 \frac{a_0}{b_0} \\
\mathbb{E}[\alpha_{ikd} \mu_{ikd}^2] &= \frac{1}{s_0} + m_0^2 \frac{a_0}{b_0},
\end{aligned}$$

where $\psi(\cdot)$ indicates the digamma function.

The KL divergence, for one dimension d —where subscript q denotes $q(\mu, \alpha)$'s parameters and subscript 0 denotes the prior parameters—is

$$\begin{aligned}
& \text{KL}(q(\mu, \alpha) \| p(\mu, \alpha)) \\
&= \frac{1}{2} \left(s_0 \frac{a_q}{b_q} (m_q - m_0)^2 + \frac{s_0}{s_q} - (\log(s_0) - \log(s_q)) - 1 \right) \\
&\quad + a_0 \log \left(\frac{b_q}{b_0} \right) - (\psi(a_q) - \psi(a_0)) \\
&\quad + (a_q - a_0) \psi(a_q) - (b_q - b_0) \frac{a_q}{b_q}.
\end{aligned}$$

C.2. Expected KL Divergence $\text{KL}[\mathbf{z}_i]$

In (4b) and (11) we require

$$\begin{aligned}
\text{KL}[\mathbf{z}_i] &= \mathbb{E}_{q_{\lambda}(\boldsymbol{\mu}, \boldsymbol{\alpha})} \left[\sum_{k=1}^{K_i} \mathbb{E}_q[r_{ik}] \cdots \right. \\
&\quad \left. \cdots \text{KL} \left(q_{\phi}(\mathbf{z}_i | \mathbf{x}) \| p(\mathbf{z}_i | \boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik}) \right) \right]
\end{aligned}$$

$$\begin{aligned}
&= \mathbb{E}_{q_{\lambda}(\boldsymbol{\mu}, \boldsymbol{\alpha})} \left[-\frac{1}{2} \sum_{k=1}^{K_i} \gamma_{ik} \sum_{d=1}^D \left(\cdots \right. \right. \\
&\quad \left. \log \boldsymbol{\sigma}_{\phi}^2(\mathbf{x})_{d'} + \log \alpha_{ikd'} + 1 \right. \\
&\quad \left. - \alpha_{ikd'} \left(\boldsymbol{\sigma}_{\phi}^2(\mathbf{x})_{d'} + (\boldsymbol{\mu}_{\phi}(\mathbf{x})_{d'} - \mu_{ikd'})^2 \right) \right) \right] \\
&= -\frac{1}{2} \sum_k \gamma_{ik} \sum_d \left(\cdots \right. \\
&\quad \left. \log \boldsymbol{\sigma}_{\phi}^2(\mathbf{x})_{d'} + \psi(a_{ikd'}) - \log b_{ikd'} \right. \\
&\quad \left. - \frac{a_{ikd'}}{b_{ikd'}} \left(\boldsymbol{\sigma}_{\phi}^2(\mathbf{x})_{d'} + (\boldsymbol{\mu}_{\phi}(\mathbf{x})_{d'} - m_{ikd'})^2 \right) \right. \\
&\quad \left. - \frac{1}{s_{ikd'}} + 1 \right).
\end{aligned}$$

Indices $d' = d + (i-1)D$ index the correct elements of $\boldsymbol{\mu}_{\phi}(\mathbf{x})$ and $\boldsymbol{\sigma}_{\phi}^2(\mathbf{x})$.

C.3. Expected KL Divergence $\text{KL}[\mathbf{r}_i]$

In (4c) and (11) we require

$$\begin{aligned}
\text{KL}[\mathbf{r}_i] &= \mathbb{E}_{q_{\lambda}(\boldsymbol{\pi}_i)} \left[\text{KL} \left(q_{\gamma}(\mathbf{r}_i) \| p(\mathbf{r}_i | \boldsymbol{\pi}_i) \right) \right] \\
&= \mathbb{E}_{q_{\lambda}(\boldsymbol{\pi}_i)} \left[\sum_k \gamma_{ik} (\log \gamma_{ik} - \log \pi_{ik}) \right] \\
&= \sum_k \gamma_{ik} \left(\log \gamma_{ik} + \psi \left(\sum_{k'} c_{ik'} \right) - \psi(c_{ik}) \right).
\end{aligned}$$

D. Natural Gradients

We wish to update the mixture component parameters \mathbf{m}_{ik} , \mathbf{s}_{ik} , \mathbf{a}_{ik} and \mathbf{b}_{ik} , and other *mean* parameters. We will write the gradient step in terms of *natural* parameters. In particular, the stochastic ELBO for minibatch \mathcal{B} with respect to $\boldsymbol{\lambda}_{ik}$ is

$$\begin{aligned}
\mathcal{L}_{\mathcal{B}}(\boldsymbol{\lambda}_{ik}) &= \\
&\mathbb{E}_{q(\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik})} \left[\frac{N}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \gamma_{ik}^{(n)} \mathbb{E}_{q_{\phi_i}(\mathbf{z}_i^{(n)} | \mathbf{x}_i^{(n)})} \left[\log p(\mathbf{z}_i^{(n)} | \boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik}) \right. \right. \\
&\quad \left. \left. + \log p(\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik}) - \log q(\boldsymbol{\mu}_{ik}, \boldsymbol{\alpha}_{ik}) \right] \right].
\end{aligned}$$

Expanding the inner terms yields

$$\begin{aligned}
\mathcal{L}_{\mathcal{B}}(\boldsymbol{\lambda}_{ik}) &= \\
&\sum_d \mathbb{E}_{q(\mu_{ikd}, \alpha_{ikd})} \left[\frac{N}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \gamma_{ik}^{(n)} \left(\right. \right. \\
&\quad \left. \left. \frac{1}{2} \underbrace{\log \alpha_{ikd}}_* - \frac{1}{2} \log(2\pi) \right) \right]
\end{aligned}$$

$$\begin{aligned}
& -\frac{1}{2} \left(\boldsymbol{\mu}_\phi(\mathbf{x}^{(n)})_{d'}^2 + \boldsymbol{\sigma}_\phi^2(\mathbf{x}^{(n)})_{d'} \right) \underbrace{\alpha_{ikd}}_* \\
& + \boldsymbol{\mu}_\phi(\mathbf{x}^{(n)})_{d'} \underbrace{\alpha_{ikd}\mu_{ikd}}_* - \frac{1}{2} \underbrace{\alpha_{ikd}\mu_{ikd}^2}_* \\
& + \log \left(\frac{b_0^{a_0}}{\Gamma(a_0)} \sqrt{\frac{s_0}{2\pi}} \right) + \left(a_0 - \frac{1}{2} \right) \underbrace{\log \alpha_{ikd}}_* \\
& - \left(b_0 + \frac{1}{2}s_0m_0^2 \right) \underbrace{\alpha_{ikd}}_* \\
& + s_0m_0 \underbrace{\alpha_{ikd}\mu_{ikd}}_* - \frac{1}{2}s_0 \underbrace{\alpha_{ikd}\mu_{ikd}^2}_* \\
& - \sum_d \mathbb{E}_{q(\mu_{ikd}, \alpha_{ikd})} \left[\log q(\mu_{ikd}, \alpha_{ikd}) \right],
\end{aligned}$$

where we have arranged the terms matching the natural parameters and sufficient statistics, which we indicate with a *, in the same order as in (12). Indices $d' = d + (i-1)D$ index the correct elements of $\boldsymbol{\mu}_\phi(\mathbf{x}^{(n)})$ and $\boldsymbol{\sigma}_\phi^2(\mathbf{x}^{(n)})$, which we conveniently wrote over terms matching the natural parameters and sufficient statistics in (12). We regroup terms:

$$\begin{aligned}
\mathcal{L}_B(\boldsymbol{\lambda}_{ik}) = & \\
& \sum_d \mathbb{E}_{q(\mu_{ikd}, \alpha_{ikd})} \left[\left(\left(a_0 + \frac{1}{2} \frac{N}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \gamma_{ik}^{(n)} \right) - \frac{1}{2} \right) \underbrace{\log \alpha_{ikd}}_* \right. \\
& - \left(b_0 + \frac{1}{2}s_0m_0^2 + \frac{1}{2} \frac{N}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \gamma_{ik}^{(n)} \dots \right. \\
& \quad \dots \left. \left(\boldsymbol{\mu}_\phi(\mathbf{x}^{(n)})_{d'}^2 + \boldsymbol{\sigma}_\phi^2(\mathbf{x}^{(n)})_{d'} \right) \right) \underbrace{\alpha_{ikd}}_* \\
& + \left(s_0m_0 + \frac{N}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \gamma_{ik}^{(n)} \boldsymbol{\mu}_\phi(\mathbf{x}^{(n)})_{d'} \right) \underbrace{\alpha_{ikd}\mu_{ikd}}_* \\
& \quad \left. - \frac{1}{2} \left(s_0 + \frac{N}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \gamma_{ik}^{(n)} \right) \underbrace{\alpha_{ikd}\mu_{ikd}^2}_* \right] \\
& - \sum_d \mathbb{E}_{q(\mu_{ikd}, \alpha_{ikd})} \left[\log q(\mu_{ikd}, \alpha_{ikd}) \right] + \text{const.}
\end{aligned}$$

For mini-batch \mathcal{B} of data points, $\mathcal{L}_B(\boldsymbol{\lambda}_{ik})$ is minimized with respect at $\boldsymbol{\lambda}_{ikd}$, as natural parameters in the form of (12):

$$\begin{aligned}
\lambda_{ikd,1}^* &= a_0 + \frac{1}{2} \frac{N}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \gamma_{ik}^{(n)} - \frac{1}{2} \\
\lambda_{ikd,2}^* &= - \left(b_0 + \frac{1}{2}s_0m_0^2 \right. \\
& \quad \left. + \frac{1}{2} \frac{N}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \gamma_{ik}^{(n)} \left(\boldsymbol{\mu}_\phi(\mathbf{x}^{(n)})_{d'}^2 + \boldsymbol{\sigma}_\phi^2(\mathbf{x}^{(n)})_{d'} \right) \right)
\end{aligned}$$

$$\begin{aligned}
\lambda_{ikd,3}^* &= \left(s_0m_0 + \frac{N}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \gamma_{ik}^{(n)} \boldsymbol{\mu}_\phi(\mathbf{x}^{(n)})_{d'} \right) \\
\lambda_{ikd,4}^* &= - \frac{1}{2} \left(s_0 + \frac{N}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \gamma_{ik}^{(n)} \right).
\end{aligned} \tag{13}$$

The values of $\boldsymbol{\lambda}_{ikd}^*$ in (13) constitute the natural gradients.

At iteration τ , we determine $\boldsymbol{\lambda}^{(\tau)}$ from iteration τ 's mean parameters of $q_\lambda(\xi)$, using the form in (12). Using $\boldsymbol{\lambda}^*$ in (13), we update $\boldsymbol{\lambda}$ using the convex combination

$$\boldsymbol{\lambda}^{(\tau+1)} \leftarrow (1 - \rho_\tau) \boldsymbol{\lambda}^{(\tau)} + \rho_\tau \boldsymbol{\lambda}^*. \tag{14}$$

This parameter update is a rewritten from of updating $\boldsymbol{\lambda}^{(\tau)}$ with the natural gradient $\hat{\nabla}_{\boldsymbol{\lambda}^{(\tau)}} \mathcal{L}(\boldsymbol{\lambda})$ that is rescaled by ρ_τ (Hoffman et al., 2013; Paquet, 2015). The step sizes should satisfy $\sum_{\tau=1}^\infty \rho_\tau = \infty$ and $\sum_{\tau=1}^\infty \rho_\tau^2 < \infty$. In our results $\rho_\tau = (\tau_0 + \tau)^{-\kappa}$ was used, with forgetting rate $\kappa \in (\frac{1}{2}, 1]$ and delay $\tau_0 \geq 0$. The updated *mean parameters* of $q(\xi)$ are obtained from the updated natural parameters $\boldsymbol{\lambda}^{(\tau+1)}$. The Dirichlet pseudo-counts of $q(\boldsymbol{\pi}_i)$ are similarly updated with $c_{ik}^{(\tau+1)} \leftarrow (1 - \rho_\tau)c_{ik}^{(\tau)} + \rho_\tau(c_0 + \frac{N}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \gamma_{ik}^{(n)})$.

E. Training details

All models were trained on a P100 GPU with 16 GB of memory. Only the CelebA models required that amount of memory; the MNIST and Omniglot models needed significantly less on account of their lower numbers of parameters and smaller activation vectors.

Pre-training (see Section 2.4), when used, lasted for 10^5 iterations in all cases, followed by 2×10^4 prior initialization iterations. The batch size was constant at 64 across runs, phases and models.

E.1. MNIST and Omniglot

The MNIST and Omniglot models all reached convergence after 2×10^5 iterations of joint optimization; the complete runs took approximately 4 hours.

The MNIST and Omniglot models had 567,775 parameters in θ and ϕ taken together, and for the prior another (latent dimension $\times 3 + 1$) \times number of components. For a model with 64 latent dimensions and 64 prior components, that adds to 12,352 extra parameters, leading to a total model size of 580,127.

E.2. CelebA

The CelebA runs took approximately 3×10^5 iterations to converge, for a total training time of about 14 hours. The CelebA iterations took longer because they required more computation, and more data fetching: data points are approximately 16 times larger than for MNIST and Omniglot.

The CelebA model had 27,857,670 parameters between them θ and ϕ , and the prior approximation for the best run an additional 49,280 parameters in λ , for a total of 27,906,050.

E.3. Optimization

All models were trained using the Adam optimizer (Kingma & Ba, 2015), with a learning rate of 10^{-4} for the embedding and generation parameters θ and ϕ , while a forgetting rate $\kappa = -0.7$ and $\tau_0 = 2 \times 10^3$ was used for λ 's gradient updates. The initial Robbins-Monro fraction was 0.01 in all cases.

F. CelebA

In Figure 8 we show the average of 100 conditional samples for \mathbf{x} from $p_\theta(\mathbf{x}|\mathbf{z})p_\lambda(\mathbf{z}|k_1, k_2)$ where $p_\lambda(\mathbf{z}|k_1, k_2)$ is sampled according to (10). The model has $I = 2$, $D = 128$, $K_1 = K_2 = 64$, and is trained in a completely unsupervised way. There were no KL weights (like what would appear in models based on the β -VAE set-up), and the clustering structure is a consequence of the Bayesian hierarchical prior. We iterate over all (k_1, k_2) pairs to create the grid of images, and the figure shows a 10×10 subset of the 64×64 grid. Specific visual properties of the clusters can be detected in the rows and columns.

In the eight sub-figures in Figures 9 and 10 we illustrate the eight settings of (k_1, k_2, k_3) for the model in Section 4.2. Each sub-figure shows a 9×9 sub-grids of the 64×64 grid, iterating settings $k_4 = 1, \dots, 64$ and $k_2 = 1, \dots, 64$ across the rows and columns. Each face is an average of 100 conditional samples for \mathbf{x} from $p_\theta(\mathbf{x}|\mathbf{z})p_\lambda(\mathbf{z}|k_1, k_2, k_3, k_4, k_5)$ for the respective k_i -settings.



Figure 8. The means of samples for \mathbf{x} , generated from $p_\theta(\mathbf{x}|\mathbf{z})p_\lambda(\mathbf{z}|k_1, k_2)$ as k_1 and k_2 varies along the rows and columns. The figure shows a 10×10 subset of a 64×64 grid. The model was trained wholly unsupervised, with no KL weights (like what would appear in models based on the β -VAE set-up), and the clustering structure is a consequence of the Bayesian hierarchical prior. We may discern image properties that some settings of k_1 and k_2 are responsible for modeling; the bottom row's k_1 captures a “glasses” cluster, even though no supervised labels were ever presented. Some of the mixture component approximations $q_\lambda(\pi_i)$ had pseudo-counts very close to that of the prior c_0 for some index values of k_i and were effectively pruned; this explains the very “washed” column (third left).



Figure 9. The means of samples for \mathbf{x} , generated from $p_{\theta}(\mathbf{x}|\mathbf{z}) p_{\lambda}(\mathbf{z}|k_1, k_2, k_3, k_4, k_5)$ as k_4 and k_5 varies along the rows and columns. The figure shows a 9×9 subset of a 64×64 grid. Mixture component indices (k_1, k_2) are set to their four settings, to which we associated semi-supervised properties. Mixture component index $k_3 = 1$ is set to its associated property “smiling”. Sub-figure titles show their semi-supervised (k_1, k_2, k_3) settings.



Figure 10. The means of samples for \mathbf{x} , generated from $p_{\theta}(\mathbf{x}|\mathbf{z}) p_{\lambda}(\mathbf{z}|k_1, k_2, k_3, k_4, k_5)$ as k_4 and k_5 varies along the rows and columns. The figure shows a 9×9 subset of a 64×64 grid. Mixture component indices (k_1, k_2) are set to their four settings, to which we associated semi-supervised properties. Mixture component index $k_3 = 2$ is set to its associated property “not smiling”. Sub-figure titles show their semi-supervised (k_1, k_2, k_3) settings. Compare Figure 9.