IMPORTANCE WEIGHTED AUTOENCODERS

Yuri Burda, Roger Grosse & Ruslan Salakhutdinov
Department of Computer Science
University of Toronto
Toronto, ON, Canada
{yburda,rgrosse,rsalakhu}@cs.toronto.edu

ABSTRACT

The variational autoencoder (VAE; Kingma & Welling (2014)) is a recently proposed generative model pairing a top-down generative network with a bottom-up recognition network which approximates posterior inference. It typically makes strong assumptions about posterior inference, for instance that the posterior distribution is approximately factorial, and that its parameters can be approximated with nonlinear regression from the observations. As we show empirically, the VAE objective can lead to overly simplified representations which fail to use the network's entire modeling capacity. We present the importance weighted autoencoder (IWAE), a generative model with the same architecture as the VAE, but which uses a strictly tighter log-likelihood lower bound derived from importance weighting. In the IWAE, the recognition network uses multiple samples to approximate the posterior, giving it increased flexibility to model complex posteriors which do not fit the VAE modeling assumptions. We show empirically that IWAEs learn richer latent space representations than VAEs, leading to improved test log-likelihood on density estimation benchmarks.

1 INTRODUCITON

In recent years, there has been a renewed focus on learning deep generative models (Hinton et al., 2006; Salakhutdinov & E., 2009; Gregor et al., 2014; Kingma & Welling, 2014; Rezende et al., 2014). A common difficulty faced by most approaches is the need to perform posterior inference during training: the log-likelihood gradients for most latent variable models are defined in terms of posterior statistics (e.g. Salakhutdinov & E. (2009); Neal (1992); Gregor et al. (2014)). One approach for dealing with this problem is to train a recognition network alongside the generative model (Dayan et al., 1995). The recognition network aims to predict the posterior distribution over latent variables given the observations, and can often generate a rough approximation much more quickly than generic inference algorithms such as MCMC.

The variational autoencoder (VAE; Kingma & Welling (2014); Rezende et al. (2014)) is a recently proposed generative model which pairs a top-down generative network with a bottom-up recognition network. Both networks are jointly trained to maximize a variational lower bound on the data log-likelihood. VAEs have recently been successful at separating style and content (Kingma et al., 2014; Kulkarni et al., 2015) and at learning to "draw" images in a realistic manner (Gregor et al., 2015).

VAEs make strong assumptions about the posterior distribution. Typically VAE models assume that the posterior is approximately factorial, and that its parameters can be predicted from the observables through a nonlinear regression. Because they are trained to maximize a variational lower bound on the log-likelihood, they are encouraged to learn representations where these assumptions are satisfied, i.e. where the posterior is approximately factorial and predictable with a neural network. While this effect is beneficial, it comes at a cost: constraining the form of the posterior limits the expressive power of the model. This is especially true of the VAE objective, which harshly penalizes approximate posterior samples which are unlikely to explain the data, even if the recognition network puts much of its probability mass on good explanations.

In this paper, we introduce the importance weighted autoencoder (IWAE), a generative model which shares the VAE architecture, but which is trained with a tighter log-likelihood lower bound de-

rived from importance weighting. The recognition network generates multiple approximate posterior samples, and their weights are averaged. As the number of samples is increased, the lower bound approaches the true log-likelihood. The use of multiple samples gives the IWAE additional flexibility to learn generative models whose posterior distributions do not fit the VAE modeling assumptions. This approach is related to reweighted wake sleep (Bornschein & Bengio, 2015), but the IWAE is trained using a single unified objective. Compared with the VAE, our IWAE is able to learn richer representations with more latent dimensions, which translates into significantly higher log-likelihoods on density estimation benchmarks.

2 BACKGROUND

In this section, we review the variational autoencoder (VAE) model of Kingma & Welling (2014). In particular, we describe a generalization of the architecture to multiple stochastic hidden layers. We note, however, that Kingma & Welling (2014) used a single stochastic hidden layer, and there are other sensible generalizations to multiple layers, such as the one presented by Rezende et al. (2014).

The VAE defines a generative process in terms of ancestral sampling through a cascade of hidden layers:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{\mathbf{h}^1,\dots,\mathbf{h}^L} p(\mathbf{h}^L|\boldsymbol{\theta})p(\mathbf{h}^{L-1}|\mathbf{h}^L,\boldsymbol{\theta})\cdots p(\mathbf{x}|\mathbf{h}^1,\boldsymbol{\theta}). \tag{1}$$

Here, $\boldsymbol{\theta}$ is a vector of parameters of the variational autoencoder, and $\mathbf{h} = \{\mathbf{h}^1, \dots, \mathbf{h}^L\}$ denotes the stochastic hidden units, or latent variables. The dependence on $\boldsymbol{\theta}$ is often suppressed for clarity. For convenience, we define $\mathbf{h}^0 = \mathbf{x}$. Each of the terms $p(\mathbf{h}^\ell|\mathbf{h}^{\ell+1})$ may denote a complicated nonlinear relationship, for instance one computed by a multilayer neural network. However, it is assumed that sampling and probability evaluation are tractable for each $p(\mathbf{h}^\ell|\mathbf{h}^{\ell+1})$. Note that L denotes the number of *stochastic* hidden layers; the deterministic layers are not shown explicitly here. We assume the recognition model $q(\mathbf{h}|\mathbf{x})$ is defined in terms of an analogous factorization:

$$q(\mathbf{h}|\mathbf{x}) = q(\mathbf{h}^1|\mathbf{x})q(\mathbf{h}^2|\mathbf{h}^1)\cdots q(\mathbf{h}^L|\mathbf{h}^{L-1}), \tag{2}$$

where sampling and probability evaluation are tractable for each of the terms in the product.

In this work, we assume the same families of conditional probability distributions as Kingma & Welling (2014). In particular, the prior $p(\mathbf{h}^L)$ is fixed to be a zero-mean, unit-variance Gaussian. In general, each of the conditional distributions $p(\mathbf{h}^\ell|\mathbf{h}^{\ell+1})$ and $q(\mathbf{h}^\ell|\mathbf{h}^{\ell-1})$ is a Gaussian with diagonal covariance, where the mean and covariance parameters are computed by a deterministic feed-forward neural network. For real-valued observations, $p(\mathbf{x}|\mathbf{h}^1)$ is also defined to be such a Gaussian; for binary observations, it is defined to be a Bernoulli distribution whose mean parameters are computed by a neural network.

The VAE is trained to maximize a variational lower bound on the log-likelihood, as derived from Jensen's Inequality:

$$\log p(\mathbf{x}) = \log \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\left[\frac{p(\mathbf{x},\mathbf{h})}{q(\mathbf{h}|\mathbf{x})}\right] \geq \mathbb{E}_{q(\mathbf{h}|\mathbf{x})}\left[\log\frac{p(\mathbf{x},\mathbf{h})}{q(\mathbf{h}|\mathbf{x})}\right] = \mathcal{L}(\mathbf{x}). \tag{3}$$

Since $\mathcal{L}(\mathbf{x}) = \log p(\mathbf{x}) - \mathrm{D}_{\mathrm{KL}}(q(\mathbf{h}|\mathbf{x})||p(\mathbf{h}|\mathbf{x}))$, the training procedure is forced to trade off the data log-likelihood $\log p(\mathbf{x})$ and the KL divergence from the true posterior. This is beneficial, in that it encourages the model to learn a representation where posterior inference is easy to approximate.

If one computes the log-likelihood gradient for the recognition network directly from Eqn. 3, the result is a REINFORCE-like update rule which trains slowly because it does not use the log-likelihood gradients with respect to latent variables (Dayan et al., 1995; Mnih & Gregor, 2014). Instead, Kingma & Welling (2014) proposed a reparameterization of the recognition distribution in terms of auxiliary variables with fixed distributions, such that the samples from the recognition model are a deterministic function of the inputs and auxiliary variables. While they presented the reparameterization trick for a variety of distributions, for convenience we discuss the special case of Gaussians, since that is all we require in this work. (The general reparameterization trick can be used with our IWAE as well.)

In this paper, the recognition distribution $q(\mathbf{h}^\ell|\mathbf{h}^{\ell-1},\boldsymbol{\theta})$ always takes the form of a Gaussian $\mathcal{N}(\mathbf{h}^\ell|\boldsymbol{\mu}(\mathbf{h}^{\ell-1},\boldsymbol{\theta}),\boldsymbol{\Sigma}(\mathbf{h}^{\ell-1},\boldsymbol{\theta}))$, whose mean and covariance are computed from the the states of

the hidden units at the previous layer and the model parameters. This can be alternatively expressed by first sampling an auxiliary variable $\boldsymbol{\epsilon}^\ell \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and then applying the deterministic mapping

$$\mathbf{h}^\ell(\boldsymbol{\epsilon}^\ell, \mathbf{h}^{\ell-1}, \boldsymbol{\theta}) = \boldsymbol{\Sigma}(\mathbf{h}^{\ell-1}, \boldsymbol{\theta})^{1/2} \boldsymbol{\epsilon}^\ell + \boldsymbol{\mu}(\mathbf{h}^{\ell-1}, \boldsymbol{\theta}). \tag{4}$$

The joint recognition distribution $q(\mathbf{h}|\mathbf{x}, \boldsymbol{\theta})$ over all latent variables can be expressed in terms of a deterministic mapping $\mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \boldsymbol{\theta})$, with $\boldsymbol{\epsilon} = (\boldsymbol{\epsilon}^1, \ldots, \boldsymbol{\epsilon}^L)$, by applying Eqn. 4 for each layer in sequence. Since the distribution of $\boldsymbol{\epsilon}$ does not depend on $\boldsymbol{\theta}$, we can reformulate the gradient of the bound $\mathcal{L}(\mathbf{x})$ from Eqn. 3 by pushing the gradient operator inside the expectation:

$$\nabla_{\boldsymbol{\theta}} \log \mathbb{E}_{\mathbf{h} \sim q(\mathbf{h}|\mathbf{x}, \boldsymbol{\theta})} \left[\frac{p(\mathbf{x}, \mathbf{h}|\boldsymbol{\theta})}{q(\mathbf{h}|\mathbf{x}, \boldsymbol{\theta})} \right] = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\epsilon}^1, \ldots, \boldsymbol{\epsilon}^L \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\log \frac{p(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \boldsymbol{\theta})|\boldsymbol{\theta})}{q(\mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \boldsymbol{\theta})|\mathbf{x}, \boldsymbol{\theta})} \right] \tag{5}$$

$$= \mathbb{E}_{\boldsymbol{\epsilon}^1, \ldots, \boldsymbol{\epsilon}^L \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\nabla_{\boldsymbol{\theta}} \log \frac{p(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \boldsymbol{\theta})|\boldsymbol{\theta})}{q(\mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \boldsymbol{\theta})|\mathbf{x}, \boldsymbol{\theta})} \right]. \tag{6}$$

Assuming the mapping $\mathbf{h}$ is represented as a deterministic feed-forward neural network, for a fixed $\boldsymbol{\epsilon}$, the gradient inside the expectation can be computed using standard backpropagation. In practice, one approximates the expectation in Eqn. 6 by generating k samples of $\boldsymbol{\epsilon}$ and applying the Monte Carlo estimator

$$\frac{1}{k} \sum_{i=1}^{k} \nabla_{\boldsymbol{\theta}} \log w\left(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}_i, \mathbf{x}, \boldsymbol{\theta}), \boldsymbol{\theta}\right) \tag{7}$$

with $w(\mathbf{x}, \mathbf{h}, \boldsymbol{\theta}) = p(\mathbf{x}, \mathbf{h}|\boldsymbol{\theta})/q(\mathbf{h}|\mathbf{x}, \boldsymbol{\theta})$. This is an unbiased estimate of $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathbf{x})$. We note that the VAE update and the basic REINFORCE-like update are both unbiased estimators of the same gradient, but the VAE update tends to have lower variance in practice because it makes use of the log-likelihood gradients with respect to the latent variables.

3 IMPORTANCE WEIGHTED AUTOENCODER

The VAE objective of Eqn. 3 heavily penalizes approximate posterior samples which fail to explain the observations. This places a strong constraint on the model, since the variational assumptions must be approximately satisfied in order to achieve a good lower bound. In particular, the posterior distribution must be approximately factorial and predictable with a feed-forward neural network. This VAE criterion may be too strict; a recognition network which places only a small fraction (e.g. 20%) of its samples in the region of high posterior probability region may still be sufficient for performing accurate inference. If we lower our standards in this way, this may give us additional flexibility to train a generative network whose posterior distributions do not fit the VAE assumptions. This is the motivation behind our proposed algorithm, the Importance Weighted Autoencoder (IWAE).

Our IWAE uses the same architecture as the VAE, with both a generative network and a recognition network. The difference is that it is trained to maximize a different lower bound on $\log p(\mathbf{x})$. In particular, we use the following lower bound, corresponding to the k-sample importance weighting estimate of the log-likelihood:

$$\mathcal{L}_k(\mathbf{x}) = \mathbb{E}_{\mathbf{h}_1, \ldots, \mathbf{h}_k \sim q(\mathbf{h}|\mathbf{x})} \left[\log \frac{1}{k} \sum_{i=1}^{k} \frac{p(\mathbf{x}, \mathbf{h}_i)}{q(\mathbf{h}_i|\mathbf{x})} \right]. \tag{8}$$

Here, $\mathbf{h}_1, \ldots, \mathbf{h}_k$ are sampled independently from the recognition model. The term inside the sum corresponds to the unnormalized importance weights for the joint distribution, which we will denote as $w_i = p(\mathbf{x}, \mathbf{h}_i)/q(\mathbf{h}_i|\mathbf{x})$.

This is a lower bound on the marginal log-likelihood, as follows from Jensen's Inequality and the fact that the average importance weights are an unbiased estimator of $p(\mathbf{x})$:

$$\mathcal{L}_k = \mathbb{E} \left[\log \frac{1}{k} \sum_{i=1}^{k} w_i \right] \leq \log \mathbb{E} \left[\frac{1}{k} \sum_{i=1}^{k} w_i \right] = \log p(\mathbf{x}), \tag{9}$$

where the expectations are with respect to $q(\mathbf{h}|\mathbf{x})$.

It is perhaps unintuitive that importance weighting would be a reasonable estimator in high dimensions. Observe, however, that the special case of $k = 1$ is equivalent to the standard VAE objective shown in Eqn. 3. Using more samples can only improve the tightness of the bound:

Theorem 1. *For all k, the lower bounds satisfy*

$$\log p(\mathbf{x}) \geq \mathcal{L}_{k+1} \geq \mathcal{L}_k. \tag{10}$$

Moreover, if $p(\mathbf{h}, \mathbf{x})/q(\mathbf{h}|\mathbf{x})$ is bounded, then $\mathcal{L}_k$ approaches $\log p(\mathbf{x})$ as k goes to infinity.

Proof. See Appendix A. □

The bound $\mathcal{L}_k$ can be estimated using the straightforward Monte Carlo estimator, where we generate samples from the recognition network and average the importance weights. One might worry about the variance of this estimator, since importance weighting famously suffers from extremely high variance in cases where the proposal and target distributions are not a good match. However, as our estimator is based on the *log* of the average importance weights, it does not suffer from high variance. This argument is made more precise in Appendix B.

3.1 TRAINING PROCEDURE

To train an IWAE with a stochastic gradient based optimizer, we use an unbiased estimate of the gradient of $\mathcal{L}_k$, defined in Eqn. 8. As with the VAE, we use the reparameterization trick to derive a low-variance upate rule:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_k(\mathbf{x}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{h}_1, \ldots, \mathbf{h}_k} \left[\log \frac{1}{k} \sum_{i=1}^{k} w_i \right] = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_k} \left[\log \frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{h}(\mathbf{x}, \boldsymbol{\epsilon}_i, \boldsymbol{\theta}), \boldsymbol{\theta}) \right] \tag{11}$$

$$= \mathbb{E}_{\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_k} \left[\nabla_{\boldsymbol{\theta}} \log \frac{1}{k} \sum_{i=1}^{k} w(\mathbf{x}, \mathbf{h}(\mathbf{x}, \boldsymbol{\epsilon}_i, \boldsymbol{\theta}), \boldsymbol{\theta}) \right] \tag{12}$$

$$= \mathbb{E}_{\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_k} \left[\sum_{i=1}^{k} \widetilde{w_i} \nabla_{\boldsymbol{\theta}} \log w(\mathbf{x}, \mathbf{h}(\mathbf{x}, \boldsymbol{\epsilon}_i, \boldsymbol{\theta}), \boldsymbol{\theta}) \right], \tag{13}$$

where $\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_k$ are the same auxiliary variables as defined in Section 2 for the VAE, $w_i = w(\mathbf{x}, \mathbf{h}(\mathbf{x}, \boldsymbol{\epsilon}_i, \boldsymbol{\theta}), \boldsymbol{\theta})$ are the importance weights expressed as a deterministic function, and $\widetilde{w_i} = w_i / \sum_{i=1}^{k} w_i$ are the normalized importance weights.

In the context of a gradient-based learning algorithm, we draw k samples from the recognition network (or, equivalently, k sets of auxiliary variables), and use the Monte Carlo estimate of Eqn. 13:

$$\sum_{i=1}^{k} \widetilde{w_i} \nabla_{\boldsymbol{\theta}} \log w \left(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}_i, \mathbf{x}, \boldsymbol{\theta}), \boldsymbol{\theta} \right). \tag{14}$$

In the special case of $k = 1$, the single normalized weight $\widetilde{w_1}$ takes the value 1, and one obtains the VAE update rule.

We unpack this update because it does not quite parallel that of the standard VAE.[1] The gradient of the log weights decomposes as:

$$\nabla_{\boldsymbol{\theta}} \log w(\mathbf{x}, \mathbf{h}(\mathbf{x}, \boldsymbol{\epsilon}_i, \boldsymbol{\theta}), \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}, \mathbf{h}(\mathbf{x}, \boldsymbol{\epsilon}_i, \boldsymbol{\theta})|\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} \log q(\mathbf{h}(\mathbf{x}, \boldsymbol{\epsilon}_i, \boldsymbol{\theta})|\mathbf{x}, \boldsymbol{\theta}). \tag{15}$$

The first term encourages the generative model to assign high probability to each $\mathbf{h}^\ell$ given $\mathbf{h}^{\ell+1}$ (following the convention that $\mathbf{x} = \mathbf{h}^0$). It also encourages the recognition network to adjust the hidden representations so that the generative network makes better predictions. In the case of a single stochastic layer (i.e. $L = 1$), the combination of these two effects is equivalent to backpropagation in a stochastic autoencoder. The second term of this update encourages the recognition network to have a spread-out distribution over predictions. This update is averaged over the samples with weight proportional to the importance weights, motivating the name "importance weighted autoencoder."

---

[1] Kingma & Welling (2014) separated out the KL divergence in the bound of Eqn. 3 in order to achieve a simpler and lower-variance update. Unfortunately, no analogous trick applies for $k > 1$. In principle, the IWAE updates may be higher variance for this reason. However, in our experiments, we observed that the performance of the two update rules was indistinguishable in the case of $k = 1$.

The dominant computational cost in IWAE training is computing the activations and parameter gradients needed for $\nabla_{\boldsymbol{\theta}} \log w(\mathbf{x}, \mathbf{h}(\mathbf{x}, \boldsymbol{\epsilon}_i, \boldsymbol{\theta}), \boldsymbol{\theta})$. This corresponds to the forward and backward passes in backpropagation. In the basic IWAE implementation, both passes must be done independently for each of the k samples. Therefore, the number of operations scales linearly with k. In our GPU-based implementation, the samples are processed in parallel by replicating each training example k times within a mini-batch.

One can greatly reduce the computational cost by adding another form of stochasticity. Specifically, only the forward pass is needed to compute the importance weights. The sum in Eqn. 14 can be stochastically approximated by choosing a single sample $\boldsymbol{\epsilon}_i$ proprtional to its normalized weight $\widetilde{w_i}$ and then computing $\nabla_{\boldsymbol{\theta}} \log w(\mathbf{x}, \mathbf{h}(\mathbf{x}, \boldsymbol{\epsilon}_i, \boldsymbol{\theta}), \boldsymbol{\theta})$. This method requires k forward passes and one backward pass per training example. Since the backward pass requires roughly twice as many addmultiply operations as the forward pass, for large k, this trick reduces the number of add-multiply operations by roughly a factor of 3. This comes at the cost of increased variance in the updates, but empirically we have found the tradeoff to be favorable.

4   RELATED WORK

There are several broad families of approaches to training deep generative models. Some models are defined in terms of Boltzmann distributions (Smolensky, 1986; Salakhutdinov & E., 2009). This has the advantage that many of the conditional distributions are tractable, but the inability to sample from the model or compute the partition function has been a major roadblock (Salakhutdinov & Murray, 2008). Other models are defined in terms of belief networks (Neal, 1992; Gregor et al., 2014). These models are tractable to sample from, but the conditional distributions become tangled due to the explaining away effect.

One strategy for dealing with intractable posterior inference is to train a recognition network which approximates the posterior. A classic approach was the wake-sleep algorithm, used to train Helmholtz machines (Dayan et al., 1995). The generative model was trained to model the conditionals inferred by the recognition net, and the recognition net was trained to explain synthetic data generated by the generative net. Unfortunately, wake-sleep trained the two networks on different objective functions. Deep autoregressive networks (Gregor et al., 2014) consisted of deep generative and recognition networks trained using a single variational lower bound. Neural variational inference and learning (Mnih & Gregor, 2014) is another algorithm for training recognition networks which reduces stochasticity in the updates by training a third network to predict reward baselines in the context of the REINFORCE algorithm (Williams, 1992). Salakhutdinov & Larochelle (2010) used a recognition network to approximate the posterior distribution in deep Boltzmann machines.

Variational autoencoders (Kingma & Welling, 2014; Rezende et al., 2014), as described in detail in Section 2, are another combination of generative and recognition networks, trained with the same variational objective as DARN and NVIL. However, in place of REINFORCE, they reduce the variance of the updates through a clever reparameterization of the random choices. The reparameterization trick is also known as "backprop through a random number generator" (Williams, 1992).

One factor distinguishing VAEs from the other models described above is that the model is described in terms of a simple distribution followed by a deterministic mapping, rather than a sequence of stochastic choices. Similar architectures have been proposed which use different training objectives. Generative adversarial networks (Goodfellow et al., 2014) train a generative network and a recognition network which act in opposition: the recognition network attempts to distinguish between training examples and generated samples, and the generative model tries to generate samples which fool the recognition network. Maximum mean discrepancy (MMD) networks (Li et al., 2015; Dziugaite et al., 2015) attempt to generate samples which match a certain set of statistics of the training data. They can be viewed as a kind of adversarial net where the adversary simply looks at the set of pre-chosen statistics (Dziugaite et al., 2015). In contrast to VAEs, the training criteria for adversarial nets and MMD nets are not based on the data log-likelihood.

Other researchers have derived log-probability lower bounds by way of importance sampling. Tang & Salakhutdinov (2013) and Ba et al. (2015) avoided recognition networks entirely, instead performing inference using importance sampling from the prior. Gogate et al. (2007) presented a variety of graphical model inference algorithms based on importance weighting. Reweighted wake-sleep

(RWS) of Bornschein & Bengio (2015) is another recognition network approach which combines the original wake-sleep algorithm with updates to the generative network equivalent to gradient ascent on our bound $\mathcal{L}_k$. However, Bornschein & Bengio (2015) interpret this update as following a biased estimate of $\nabla_{\boldsymbol{\theta}} \log p(\mathbf{x})$, whereas we interpret it as following an unbiased estimate of $\nabla_{\boldsymbol{\theta}} \mathcal{L}_k$. The IWAE also differs from RWS in that the generative and recognition networks are trained to maximize a single objective, $\mathcal{L}_k$. By contrast, the q-wake and sleep steps of RWS do not appear to be related to $\mathcal{L}_k$. Finally, the IWAE differs from RWS in that it makes use of the reparameterization trick.

Apart from our approach of using multiple approximate posterior samples, another way to improve the flexibility of posterior inference is to use a more sophisticated algorithm than importance sampling. Examples of this approach include normalizing flows (Rezende & Mohamed, 2015) and the Hamiltonian variational approximation of Salimans et al. (2015).

After the publication of this paper the authors learned that the idea of using an importance weighted lower bound for training variational autoencoders has been independently explored by Laurent Dinh and Vincent Dumoulin, and preliminary results of their work were presented at the 2014 CIFAR NCAP Deep Learning summer school.

5 EXPERIMENTAL RESULTS

We have compared the generative performance of the VAE and IWAE in terms of their held-out log-likelihoods on two density estimation benchmark datasets. We have further investigated a particular issue we have observed with VAEs and IWAEs, namely that they learn latent spaces of significantly lower dimensionality than the modeling capacity they are allowed. We tested whether the IWAE training method ameliorates this effect.

5.1 EVALUATION ON DENSITY ESTIMATION

We evaluated the models on two benchmark datasets: MNIST, a dataset of images of handwritten digits (LeCun et al., 1998), and Omniglot, a dataset of handwritten characters in a variety of world alphabets (Lake et al., 2013). In both cases, the observations were binarized 28×28 images.[2] We used the standard splits of MNIST into 60,000 training and 10,000 test examples, and of Omniglot into 24,345 training and 8,070 test examples.

We trained models with two architectures:

1. An architecture with a single stochastic layer $\mathbf{h}^1$ with 50 units. In between the observations and the stochastic layer were two deterministic layers, each with 200 units.

2. An architecture with two stochastic layers $\mathbf{h}^1$ and $\mathbf{h}^2$, with 100 and 50 units, respectively. In between $\mathbf{x}$ and $\mathbf{h}^1$ were two deterministic layers with 200 units each. In between $\mathbf{h}^1$ and $\mathbf{h}^2$ were two deterministic layers with 100 units each.

All deterministic hidden units used the tanh nonlinearity. All stochastic layers used Gaussian distributions with diagonal covariance, with the exception of the visible layer, which used Bernoulli distributions. An exp nonlinearity was applied to the predicted variances of the Gaussian distributions. The network architectures are summarized in Appendix C.

All models were initialized with the heuristic of Glorot & Bengio (2010). For optimization, we used Adam (Kingma & Ba, 2015) with parameters $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-4}$ and minibaches of size 20. The training proceeded for 3^i passes over the data with learning rate of $0.001 \cdot 10^{-i/7}$ for $i = 0 \ldots 7$ (for a total of $\sum_{i=0}^{7} 3^i = 3280$ passes over the data). This learning rate schedule was chosen based on preliminary experiments training a VAE with one stochastic layer on MNIST.

---

[2]Unfortunately, the generative modeling literature is inconsistent about the method of binarization, and different choices can lead to considerably different log-likelihood values. We follow the procedure of Salakhutdinov & Murray (2008): the binary-valued observations are sampled with expectations equal to the real values in the training set. See Appendix D for an alternative binarization scheme.

# stoch. layers	k	MNIST				OMNIGLOT			
		VAE		IWAE		VAE		IWAE	
		NLL	active units	NLL	active units	NLL	active units	NLL	active units
1	1	86.76	19	86.76	19	108.11	28	108.11	28
	5	86.47	20	85.54	22	107.62	28	106.12	34
	50	86.35	20	84.78	25	107.80	28	104.67	41
2	1	85.33	16+5	85.33	16+5	107.58	28+4	107.56	30+5
	5	85.01	17+5	83.89	21+5	106.31	30+5	104.79	38+6
	50	84.78	17+5	82.90	26+7	106.30	30+5	103.38	44+7

Table 1: Results on density estimation and the number of active latent dimensions. For models with two latent layers, "$k_1 + k_2$" denotes k_1 active units in the first layer and k_2 in the second layer. The generative performance of IWAEs improved with increasing k, while that of VAEs benefitted only slightly. Two-layer models achieved better generative performance than one-layer models.

For each number of samples $k \in \{1, 5, 50\}$ we trained a VAE with the gradient of $\mathcal{L}(\mathbf{x})$ estimted as in Eqn. 7 and an IWAE with the gradient estimated as in Eqn. 14. For each k, the VAE and the IWAE were trained for approximately the same length of time.

All log-likelihood values were estimated as the mean of $\mathcal{L}_{5000}$ on the test set. Hence, the reported values are stochastic lower bounds on the true value, but are likely to be more accurate than the lower bounds used for training.

The log-likelihood results are reported in Table 1. Our VAE results are comparable to those previously reported in the literature. We observe that training a VAE with $k > 1$ helped only slightly. By contrast, using multiple samples improved the IWAE results considerably on both datasets. Note that the two algorithms are identical for $k = 1$, so the results ought to match up to random variability.

On MNIST, IWAE with two stochastic layers and $k = 50$ achieves a log-likelihood of -82.90 on the permutation-invariant model on this dataset. By comparison, deep belief networks achieved log-likelihood of approximately -84.55 nats (Murray & Salakhutdinov, 2009), and deep autoregressive networks achieved log-likelihood of -84.13 nats (Gregor et al., 2014). Gregor et al. (2015), who exploited spatial structure, achieved a log-likelihood of -80.97. We did not find overfitting to be a serious issue for either the VAE or the IWAE: in both cases, the training log-likelihood was 0.62 to 0.79 nats higher than the test log-likelihood. We present samples from our models in Appendix E.

For the OMNIGLOT dataset, the best performing IWAE has log-likelihood of -103.38 nats, which is slightly worse than the log-likelihood of -100.46 nats achieved by a Restricted Boltzmann Machine with 500 hidden units trained with persistent contrastive divergence (Burda et al., 2015). RBMs trained with centering or FANG methods achieve a similar performance of around -100 nats (Grosse & Salakhudinov, 2015). The training log-likelihood for the models we trained was 2.39 to 2.65 nats higher than the test log-likelihood.

5.2   LATENT SPACE REPRESENTATION

We have observed that both VAEs and IWAEs tend to learn latent representations with effective dimensions far below their capacity. Our next set of experiments aimed to quantify this effect and determine whether the IWAE objective ameliorates this effect.

If a latent dimension encodes useful information about the data, we would expect its distribution to change depending on the observations. Based on this intuition, we measured activity of a latent dimension u using the statistic $A_u = \text{Cov}_{\mathbf{x}}\left(\mathbb{E}_{u \sim q(u|\mathbf{x})}[u]\right)$. We defined the dimension u to be active if $A_u > 10^{-2}$. We have observed two pieces of evidence that this criterion is both well-defined and meaningful:

1. The distribution of A_u for a trained model consisted of two widely separated modes, as shown in Appendix C.

	First stage			Second stage		
	trained as	NLL	active units	trained as	NLL	active units
Experiment 1	VAE	86.76	19	IWAE, $k = 50$	84.88	22
Experiment 2	IWAE, $k = 50$	84.78	25	VAE	86.02	23

Table 2: Results of continuing to train a VAE model with the IWAE objective, and vice versa. Training the VAE with the IWAE objective increased the latent dimension and test log-likelihood, while training the IWAE with the VAE objective had the opposite effect.

2. To confirm that the inactive dimensions were indeed insignificant to the predictions, we evaluated all models with the inactive dimensions removed. In all cases, this changed the test log-likelihood by less than 0.06 nats.

In Table 1, we report the numbers of active units for all conditions. In all conditions, the number of active dimensions was far less than the total number of dimensions. Adding more latent dimensions did not increase the number of active dimensions. Interestingly, in the two-layer models, the second layer used very little of its modeling capacity: the number of active dimensions was always less than 10. In all cases with $k > 1$, the IWAE learned more latent dimensions than the VAE. Since this coincided with higher log-likelihood values, we speculate that a larger number of active dimensions reflects a richer latent representation.

Superficially, the phenomenon of inactive dimensions appears similar to the problem of "units dying out" in neural networks and latent variable models, an effect which is often ascribed to difficulties in optimization. For example, if a unit is inactive, it may never receive a meaningful gradient signal because of a plateau in the optimization landscape. In such cases, the problem may be avoided through a better initialization. To determine whether the inactive units resulted from an optimization issue or a modeling issue, we took the best-performing VAE and IWAE models from Table 1, and continued training the VAE model using the IWAE objective and vice versa. In both cases, the model was trained for an additional 3^7 passes over the data with a learning rate of 10^{-4}.

The results are shown in Table 2. We found that continuing to train the VAE with the IWAE objective increased the number of active dimensions and the test log-likelihood, while continuing to train the IWAE with the VAE objective did the opposite. The fact that training with the VAE objective actively reduces both the number of active dimensions and the log-likelihood strongly suggests that inactivation of the latent dimensions is driven by the objective functions rather than by optimization issues. On the other hand, optimization also appears to play a role, as the results in Table 2 are not quite identical to those in Table 1.

6 CONCLUSION

In this paper, we presented the importance weighted autoencoder, a variant on the VAE trained by maximizing a tighter log-likelihood lower bound derived from importance weighting. We showed empirically that IWAEs learn richer latent representations and achieve better generative performance than VAEs with equivalent architectures and training time. We believe this method may improve the flexibility of other generative models currently trained with the VAE objective.

7 ACKNOWLEDGEMENTS

REFERENCES

Ba, J. L., Mnih, V., and Kavukcuoglu, K. Multiple object recognition with visual attention. In *International Conference on Learning Representations*, 2015.

Bornschein, J. and Bengio, Y. Reweighted wake-sleep. *International Conference on Learning Representations*, 2015.

Burda, Y., Grosse, R. B., and Salakhutdinov, R. Accurate and conservative estimates of MRF log-likelihood using reverse annealing. *Artificial Intelligence and Statistics*, pp. 102–110, 2015.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. The Helmholtz machine. *Neural Computation*, 7: 889–904, 1995.

Dziugaite, K. G., Roy, D. M., and Ghahramani, Z. Training generative neural networks via maximum mean discrepancy optimization. In *Uncertainty in Artificial Intelligence*, 2015.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Artificial Intelligence and Statistics*, pp. 249–256, 2010.

Gogate, V., Bidyuk, B., and Dechter, R. Studies in lower bounding probability of evidence using the Markov inequality. In *Uncertainty in Artificial Intelligence*, 2007.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Neural Information Processing Systems*, 2014.

Gregor, K., Danihelka, I., Mnih, A., Blundell, C., and Wierstra, D. Deep autoregressive networks. *International Conference on Machine Learning*, 2014.

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., and Wierstra, D. DRAW: A recurrent neural network for image generation. In *International Conference on Machine Learning*, pp. 1462–1471, 2015.

Grosse, R. and Salakhudinov, R. Scaling up natural gradient by sparsely factorizing the inverse fisher matrix. In *International Conference on Machine Learning*, 2015.

Hinton, G. E., Osindero, S., and Teh, Y. A fast learning algorithm for deep belief nets. *Neural Computation*, 2006.

Kingma, D. and Ba, J. L. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. *International Conference on Learning Representations*, 2014.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. Semi-supervised learning with deep generative models. In *Neural Information Processing Systems*, 2014.

Kulkarni, T. D., Whitney, W., Kohli, P., and Tenenbaum, J. B. Deep convolutional inverse graphics network. arXiv:1503.03167, 2015.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. One-shot learning by inverting a compositional causal process. In *Neural Information Processing Systems*, 2013.

Larochelle, H., Murray I. The neural autoregressive distribution estimator. In *Artificial Intelligence and Statistics*, pp. 29–37, 2011.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, Y., Swersky, K., and Zemel, R. Generative moment matching networks. In *International Conference on Machine Learning*, pp. 1718–1727, 2015.

Mnih, A. and Gregor, K. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, pp. 1791–1799, 2014.

Murray, I. and Salakhutdinov, R. Evaluating probabilities under high-dimensional latent variable models. In *Neural Information Processing Systems*, pp. 1137–1144, 2009.

Neal, R. M. Connectionist learning of belief networks. *Artificial Intelligence*, 1992.

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

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *International Conference on Machine Learning*, pp. 1278–1286, 2014.

Salakhutdinov, R. and E., Hinton G. Deep Boltzmann machines. In *Neural Information Processing Systems*, 2009.

Salakhutdinov, R. and Larochelle, H. Efficient learning of deep Boltzmann machines. In *Artificial Intelligence and Statistics*, 2010.

Salakhutdinov, R. and Murray, I. On the quantitative analysis of deep belief networks. In *International Conference on Machine Learning*, 2008.

Salimans, T., Kingma, D. P., and Welling, M. Markov chain Monte Carlo and variational inference: bridging the gap. In *International Conference on Machine Learning*, pp. 1218–1226, 2015.

Smolensky, P. Information processing in dynamical systems: foundations of harmony theory. In Rumelhart, D. E. and McClelland, J. L. (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, 1986.

Tang, Y. and Salakhutdinov, R. Learning stochastic feedforward neural networks. In *Neural Information Processing Systems*, 2013.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

APPENDIX A

Proof of Theorem 1. We need to show the following facts about the log-likelihood lower bound $\mathcal{L}_k$:

1. $\log p(\mathbf{x}) \geq \mathcal{L}_k$,
2. $\mathcal{L}_k \geq \mathcal{L}_m$ for $k \geq m$,
3. $\log p(\mathbf{x}) = \lim_{k \to \infty} \mathcal{L}_k$, assuming $p(\mathbf{h}, \mathbf{x})/q(\mathbf{h}|\mathbf{x})$ is bounded.

We prove each in turn:

1. It follows from Jensen's inequality that

$$\mathcal{L}_k = \mathbb{E}\left[\log \frac{1}{k}\sum_{i=1}^{k}\frac{p(\mathbf{x}, \mathbf{h}_i)}{q(\mathbf{h}_i|\mathbf{x})}\right] \leq \log \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k}\frac{p(\mathbf{x}, \mathbf{h}_i)}{q(\mathbf{h}_i|\mathbf{x})}\right] = \log p(\mathbf{x}) \qquad (16)$$

2. Let $I \subset \{1, \ldots, k\}$ with $|I| = m$ be a uniformly distributed subset of distinct indices from $\{1, \ldots, k\}$. We will use the following simple observation: $\mathbb{E}_{I=\{i_1,\ldots,i_m\}}\left[\frac{a_{i_1}+\ldots+a_{i_m}}{m}\right] = \frac{a_1+\ldots+a_k}{k}$ for any sequence of numbers $a_1, \ldots, a_k$.
   Using this observation and Jensen's inequality, we get

$$\mathcal{L}_k = \mathbb{E}_{\mathbf{h}_1,\ldots,\mathbf{h}_k}\left[\log \frac{1}{k}\sum_{i=1}^{k}\frac{p(\mathbf{x}, \mathbf{h}_i)}{q(\mathbf{h}_i|\mathbf{x})}\right] \qquad (17)$$

$$= \mathbb{E}_{\mathbf{h}_1,\ldots,\mathbf{h}_k}\left[\log \mathbb{E}_{I=\{i_1,\ldots,i_m\}}\left[\frac{1}{m}\sum_{j=1}^{m}\frac{p(\mathbf{x}, \mathbf{h}_{i_j})}{q(\mathbf{h}_{i_j}|\mathbf{x})}\right]\right] \qquad (18)$$

$$\geq \mathbb{E}_{\mathbf{h}_1,\ldots,\mathbf{h}_k}\left[\mathbb{E}_{I=\{i_1,\ldots,i_m\}}\left[\log \frac{1}{m}\sum_{j=1}^{m}\frac{p(\mathbf{x}, \mathbf{h}_{i_j})}{q(\mathbf{h}_{i_j}|\mathbf{x})}\right]\right] \qquad (19)$$

$$= \mathbb{E}_{\mathbf{h}_1,\ldots,\mathbf{h}_m}\left[\log \frac{1}{m}\sum_{i=1}^{m}\frac{p(\mathbf{x}, \mathbf{h}_i)}{q(\mathbf{h}_i|\mathbf{x})}\right] = \mathcal{L}_m \qquad (20)$$

3. Consider the random variable $M_k = \frac{1}{k}\sum_{i=1}^{k}\frac{p(\mathbf{x},\mathbf{h}_i)}{q(\mathbf{h}_i|\mathbf{x})}$. If $p(\mathbf{h}, \mathbf{x})/q(\mathbf{h}|\mathbf{x})$ is bounded, then it follows from the strong law of large numbers that M_k converges to $\mathbb{E}_{q(\mathbf{h}_i|\mathbf{x})}\left[\frac{p(\mathbf{x},\mathbf{h}_i)}{q(\mathbf{h}_i|\mathbf{x})}\right] = p(\mathbf{x})$ almost surely. Hence $\mathcal{L}_k = \mathbb{E}\log[M_k]$ converges to $\log p(\mathbf{x})$ as $k \to \infty$.

APPENDIX B

It is well known that the variance of an unnormalized importance sampling based estimator can be extremely large, or even infinite, if the proposal distribution is not well matched to the target distribution. Here we argue that the Monte Carlo estimator of $\mathcal{L}_k$, described in Section 3, does not suffer from large variance. More precisely, we bound the mean absolute deviation (MAD). While this does not directly bound the variance, it would be surprising if an estimator had small MAD yet extremely large variance.

Suppose we have a strictly positive unbiased estimator $\hat{Z}$ of a positive quantity Z, and we wish to use $\log \hat{Z}$ as an estimator of $\log Z$. By Jensen's inequality, this is a biased estimator, i.e. $\mathbb{E}[\log \hat{Z}] \leq \log Z$. Denote the bias as $\delta = \log Z - \mathbb{E}[\log \hat{Z}]$. We start with the observation that $\log \hat{Z}$ is unlikely to overestimate $\log Z$ by very much, as can be shown with Markov's Inequality:

$$\Pr(\log \hat{Z} > \log Z + b) \leq e^{-b}. \tag{21}$$

Let $(X)_+$ denote $\max(X, 0)$. We now use the above facts to bound the MAD:

$$\mathbb{E}\left[\left|\log \hat{Z} - \mathbb{E}[\log \hat{Z}]\right|\right] = 2\mathbb{E}\left[\left(\log \hat{Z} - \mathbb{E}[\log \hat{Z}]\right)_+\right] \tag{22}$$

$$= 2\mathbb{E}\left[\left(\log \hat{Z} - \log Z + \log Z - \mathbb{E}[\log \hat{Z}]\right)_+\right] \tag{23}$$

$$\leq 2\mathbb{E}\left[\left(\log \hat{Z} - \log Z\right)_+ + \left(\log Z - \mathbb{E}[\log \hat{Z}]\right)_+\right] \tag{24}$$

$$= 2\mathbb{E}\left[\left(\log \hat{Z} - \log Z\right)_+\right] + 2\delta \tag{25}$$

$$= 2\int_0^\infty \Pr\left(\log \hat{Z} - \log Z > t\right) \mathrm{d}t + 2\delta \tag{26}$$
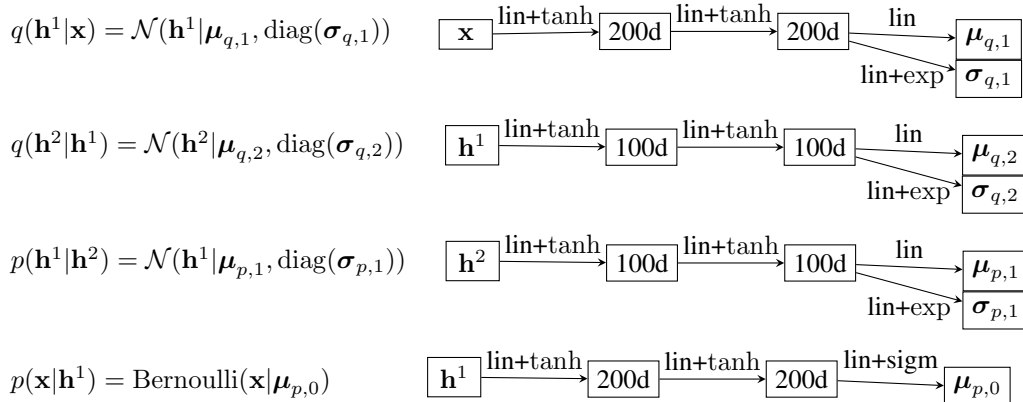
$$\leq 2\int_0^\infty e^{-t}\mathrm{d}t + 2\delta \tag{27}$$

$$= 2 + 2\delta \tag{28}$$

Here, (22) is a general formula for the MAD, (26) uses the formula $\mathbb{E}[Y] = \int_0^\infty \Pr(Y > t)\, \mathrm{d}t$ for a nonnegative random variable Y, and (27) applies the bound (21). Hence, the MAD is bounded by $2 + 2\delta$. In the context of IWAE, δ corresponds to the gap between $\mathcal{L}_k$ and $\log p(\mathbf{x})$.

APPENDIX C

NETWORK ARCHITECTURES

Here is a summary of the network architectures used in the experiments:

DISTRIBUTION OF ACTIVITY STATISTIC

In Section 5.2, we defined the activity statistic $A_u = \mathrm{Cov}_{\mathbf{x}} \left(\mathbb{E}_{u \sim q(u|\mathbf{x})}[u] \right)$, and chose a threshold of 10^{-2} for determining if a unit is active. One justification for this is that the distribution of this statistic consisted of two widely separated modes in every case we looked at. Here is the histogram of $\log A_u$ for a VAE with one stochastic layer:
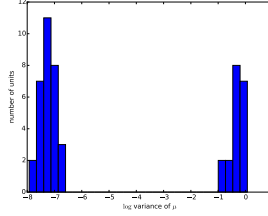
VISUALIZATION OF POSTERIOR DISTRIBUTIONS

We show some examples of true and approximate posteriors for VAE and IWAE models trained with two latent dimensions. Heat maps show true posterior distributions for 6 training examples, and the pictures in the bottom row show the examples and their reconstruction from samples from $q(\mathbf{h}|\mathbf{x})$. **Left:** VAE. **Middle:** IWAE, with $k = 5$. **Right:** IWAE, with $k = 50$. The IWAE prefers less regular posteriors and more spread out posterior predictions.
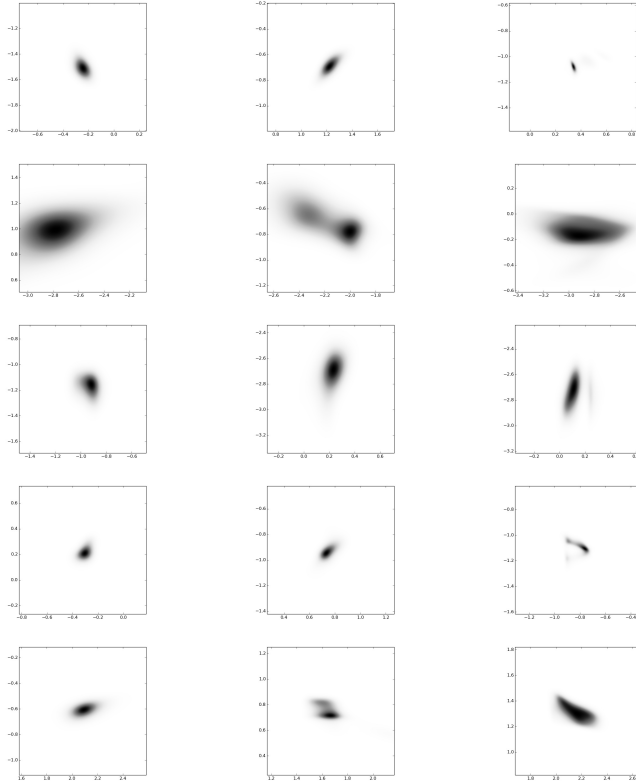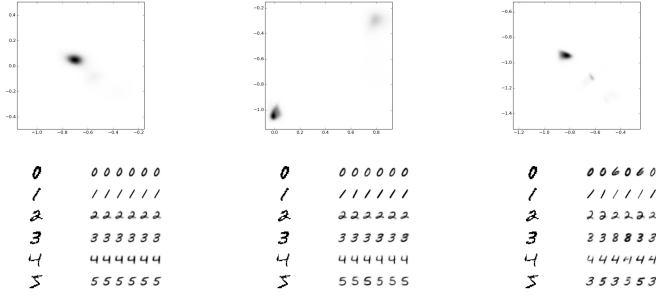
APPENDIX D

RESULTS FOR A FIXED MNIST BINARIZATION

Several previous works have used a fixed binarization of the MNIST dataset defined by Larochelle (2011). We repeated our experiments training the models on the 50000 examples from the training dataset, and evaluating them on the 10000 examples from the test dataset. Otherwise we used the same training procedure and hyperparameters as in the experiments in the main part of the paper. The results in table 3 indicate that the conclusions about the relative merits of VAEs and IWAEs are unchanged in the new experimental setup. In this setup we noticed significantly larger amounts of overfitting.

		VAE		IWAE	
# stoch. layers	k	NLL	active units	NLL	active units
1	1	88.71	19	88.71	19
	5	88.83	19	87.63	22
	50	89.05	20	87.10	24
2	1	88.08	16+5	88.08	16+5
	5	87.63	17+5	86.17	21+5
	50	87.86	17+6	85.32	24+7

Table 3: Results on density estimation and the number of active latent dimensions on the fixed binarization MNIST dataset. For models with two latent layers, "$k_1 + k_2$" denotes k_1 active units in the first layer and k_2 in the second layer. The generative performance of IWAEs improved with increasing k, while that of VAEs benefitted only slightly. Two-layer models achieved better generative performance than one-layer models.

APPENDIX E

SAMPLES

Table 4: Random samples from VAE (left column) and IWAE with $k = 50$ (right column) models. Row 1: models with one stochastic layer. Row 2: models with two stochastic layers. Samples are represented as the means of the corresponding Bernoulli distributions.