Stein Variational Autoencoder

Yunchen Pu [1] Zhe Gan [1] Ricardo Henao [1] Chunyuan Li [1] Shaobo Han [1] Lawrence Carin [1]

Abstract

A new method for learning variational autoencoders is developed, based on an application of Stein's operator. The framework represents the encoder as a deep nonlinear function through which samples from a simple distribution are fed. One need not make parametric assumptions about the form of the encoder distribution, and performance is further enhanced by integrating the proposed encoder with importance sampling. Example results are demonstrated across multiple unsupervised and semi-supervised problems, including semi-supervised analysis of the ImageNet data, demonstrating the scalability of the model to large datasets.

1. Introduction

The autoencoder (Vincent et al., 2010) is a widely employed unsupervised framework to learn (typically) low-dimensional features from complex data. There has been significant recent interests in the *variational* autoencoder (VAE) (Kingma & Welling, 2014), which generalizes the original autoencoder in several ways. The VAE encodes input data to a *distribution* of codes (latent features). Further, the VAE decoder is a generative model, specifying a probabilistic representation of the data via a *likelihood function*. Another advantage of the VAE is that it yields efficient estimation of the often intractable latent-feature posterior via an approximate model, *i.e.*, the *recognition network* (Kingma & Welling, 2014; Mnih & Gregor, 2014). As a result, the encoder yields efficient inference of the latent features of the generative model (decoder), which is critical for fast computation at test time. The VAE may also be scaled to handle massive amounts of data, such as ImageNet (Pu et al., 2016).

The VAE is a powerful framework for unsupervised learning. Additionally, when given labels on a subset of data, a classifier may be associated with the latent features, allowing for *semi-supervised* learning (Kingma et al., 2014; Pu et al., 2016). Further, the latent features (codes) may be associated with state-of-the-art natural-language-processing models, such as the Long Short-Term Memory (LSTM) network, for semi-supervised learning of text captions from an image (Pu et al., 2016).

VAEs are typically trained by maximizing a variational lower bound of the data log-likelihood (Kingma & Welling, 2014; Mnih & Gregor, 2014; Rezende et al., 2014; Kingma et al., 2014; Ranganath et al., 2016; Pu et al., 2016; Kingma et al., 2016). This lower bound is maximized by alternating between optimizing the parameters of the recognition model (encoder) and the parameters of the generative model (decoder). For evaluation of the variational expression, being able to sample efficiently from the encoder is not sufficient; one must be able to explicitly evaluate the associated distribution of latent features. This requirement has motivated design of encoders in which a neural network maps input data to the parameters of a distribution in the exponential family (Kingma & Welling, 2014; Rezende et al., 2014), which serves as the latent-features distribution. For example, Gaussian distributions have been widely utilized (Kingma & Welling, 2014; Rezende et al., 2014; Ranganath et al., 2016; Burda et al., 2016).

The Gaussian assumption may be too restrictive in some cases (Rezende & Mohamed, 2015). Consequently, recent work has considered normalizing flows (Rezende & Mohamed, 2015), in which random variables from (for example) a Gaussian distribution are fed through a series of nonlinear functions to increase the complexity and representational power of the distribution over latent features. However, because of the need to explicitly evaluate the distribution within the variational expression, these nonlinear functions must be relatively simple, *e.g.*, planar flows. In this framework, a sequence of relatively simple nonlinear functions are "stacked" atop a probabilistic (Gaussian) encoder. However, because of the simple nature of these functions, one may require many layers to achieve the desired representational power.

Researchers have recently considered the idea of viewing the encoder as a *proposal* distribution for the latent features, thus using importance weighting when sampling from this distribution (Burda et al., 2016). This idea has been demonstrated to significantly improve the variational lower bound. Although importance sampling is useful to

[1]Duke University. Correspondence to: Yunchen Pu <yunchen.pu@duke.edu>.

improve performance, it does not address the fundamental limitation associated with the simple form of the encoder distribution (required for explicit evaluation of the variational bound).

From a different perspective, recent work extended ideas from Stein's identity and operator to machine learning (Chwialkowski et al., 2016; Liu et al., 2016). Motivated by these ideas, we present a new approach for learning the distribution of latent features within a VAE framework. We recognize that the need for an explicit form for the encoder distribution is only a consequence of the fact that learning is performed based on the variational lower bound. For inference (*e.g.*, at test time), we *do not* need an explicit form for the distribution of latent features, we only require fast sampling from the encoder. Consequently, in the proposed approach, we no longer explicitly use the variational lower bound to train the encoder. Rather, we seek an encoder that minimizes the Kullback-Leibler (KL) distance between the distribution of codes and the true (posterior) distribution on the latent features. To minimize this KL distance, we generalize use of Stein's identity, and employ ideas associated with a reproducing kernel Hilbert space (RKHS). Analogous work was introduced by Liu & Wang (2016); Wang & Liu (2016); however, they considered sampling from a general unnormalized distribution. They did not consider an autoencoder, and they didn't use Stein's identity to design a nonparametric recognition model (encoder).

A key contribution of this paper concerns integrating Stein-based sampling with importance sampling, for learning VAE parameters. An advantage of using the Stein formulation with importance sampling is that we need not assume a form (*e.g.*, Gaussian) for the distribution of the encoder, and hence we yield an improved proposal distribution.

The concepts developed here are demonstrated on a wide range of unsupervised and semi-supervised learning problems, including a large-scale semi-supervised analysis of the ImageNet dataset. These experimental results illustrate the advantage of a Stein VAE with nonparametric recognition model, relative to the traditional VAE that assumes a Gaussian form of encoder. Moreover, the results demonstrate further improvements realized by integrating the Stein VAE with importance sampling.

2. Stein Variational Autoencoder (Stein VAE)

Consider data $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$, where $\boldsymbol{x}_n$ are modeled as $\boldsymbol{x}_n|\boldsymbol{z}_n \sim p(\boldsymbol{x}|\boldsymbol{z}_n;\boldsymbol{\theta})$. Distribution $p(\boldsymbol{x}|\boldsymbol{z}_n;\boldsymbol{\theta})$ may be viewed as a probabilistic *decoder* of latent code $\boldsymbol{z}_n$, and $\boldsymbol{\theta}$ represents the decoder parameters. The set of codes associated with all $\boldsymbol{x}_n \in \mathcal{D}$ is represented $\mathcal{Z} = \{\boldsymbol{z}_n\}_{n=1}^N$. In Bayesian statistics, one sets a prior on $\{\boldsymbol{\theta}, \mathcal{Z}\}$, here represented $p(\boldsymbol{\theta}, \mathcal{Z}) = p(\boldsymbol{\theta}) \prod_{n=1}^N p(\boldsymbol{z}_n)$. We desire the posterior $p(\boldsymbol{\theta}, \mathcal{Z}|\mathcal{D})$, which we approximate here via samples, without imposing an explicit form for the posterior distribution (as is common in existing VAE work (Kingma & Welling, 2014)). We generalize concepts in Liu & Wang (2016) to manifest the approximate posterior samples.

Algorithm 1 Stein Variational Autoencoder.

```
Require: Input data $\mathcal{D}$ and number of samples $M$.
 1: Initialize samples $\{\boldsymbol{\theta}_j^0\}_{j=1}^M$ from $p(\boldsymbol{\theta})$ and $\boldsymbol{\eta}^0$.
 2: for $t = 1$ to Maximum Iterations do
 3:     Sample $\{\boldsymbol{\xi}_j\}_{j=1}^M$ from $q_0(\boldsymbol{\xi})$.
 4:     Draw minibatch from $\mathcal{D}$.
 5:     Evaluate samples $\boldsymbol{z}_{jn}^{(t)} = \boldsymbol{f}_{\boldsymbol{\eta}}(\boldsymbol{x}_n, \boldsymbol{\xi}_j)$.
 6:     Update samples $\boldsymbol{\theta}_j^{(t+1)} \leftarrow \boldsymbol{\theta}_j^{(t)}$ according to (3) and
        $\hat{\boldsymbol{z}}_{jn}^{(t)} \leftarrow \boldsymbol{z}_{jn}^{(t)}$ according to (4).
 7:     for $k = 1$ to $K$ do
 8:         Update $\boldsymbol{\eta}^{(t,k)} \leftarrow \boldsymbol{\eta}^{(t,k-1)}$ according to (7).
 9:     end for
10: end for
```

2.1. Stein Variational Gradient Descent (SVGD)

Assume we have samples $\{\boldsymbol{\theta}_j\}_{j=1}^M$ drawn from distribution $q(\boldsymbol{\theta})$, and samples $\{\boldsymbol{z}_{jn}\}_{j=1}^M$ drawn from distribution $q(\mathcal{Z})$. The distribution $q(\boldsymbol{\theta})q(\mathcal{Z})$ is some KL distance from the true posterior $p(\boldsymbol{\theta}, \mathcal{Z}|\mathcal{D})$. We wish to transform $\{\boldsymbol{\theta}_j\}_{j=1}^M$ by pushing them through a function, and the corresponding transformed distribution from which they are drawn is denoted $q_T(\boldsymbol{\theta})$. It is desired that, in a KL sense, $q_T(\boldsymbol{\theta})q(\mathcal{Z})$ is closer to $p(\boldsymbol{\theta}, \mathcal{Z}|\mathcal{D})$ than was $q(\boldsymbol{\theta})q(\mathcal{Z})$. The following theorem is useful for defining how best to update $\{\boldsymbol{\theta}_j\}_{j=1}^M$.

Theorem 1 *Assume $\boldsymbol{\theta}$ and $\mathcal{Z}$ are Random Variables (RVs) drawn from distributions $q(\boldsymbol{\theta})$ and $q(\mathcal{Z})$, respectively. Consider the transformation $T(\boldsymbol{\theta}) = \boldsymbol{\theta} + \epsilon\psi(\boldsymbol{\theta};\mathcal{D})$ and let $q_T(\boldsymbol{\theta})$ represent the distribution of $\boldsymbol{\theta}' = T(\boldsymbol{\theta})$. We have*

$$\nabla_\epsilon \Big(KL(q_T \| p)\Big)|_{\epsilon=0} = -\mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta})}\big(\mathit{trace}(\mathcal{A}_p(\boldsymbol{\theta};\mathcal{D}))\big) , \quad (1)$$

where $q_T = q_T(\boldsymbol{\theta})q(\mathcal{Z})$, $p = p(\boldsymbol{\theta}, \mathcal{Z}|\mathcal{D})$, and

$$\mathcal{A}_p(\boldsymbol{\theta};\mathcal{D}) = \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{\theta};\mathcal{D})\psi(\boldsymbol{\theta};\mathcal{D})^T + \nabla_{\boldsymbol{\theta}}\psi(\boldsymbol{\theta};\mathcal{D})$$
$$\log \tilde{p}(\boldsymbol{\theta};\mathcal{D}) = \mathbb{E}_{\mathcal{Z}\sim q(\mathcal{Z})}[\log p(\mathcal{D}, \mathcal{Z}, \boldsymbol{\theta})] .$$

The proof is provided in Appendix A. Following Liu & Wang (2016), we assume $\psi(\boldsymbol{\theta};\mathcal{D})$ lives in a reproducing kernel Hilbert space (RKHS) with kernel $k(\cdot,\cdot)$. Under this assumption, the solution for $\psi(\boldsymbol{\theta};\mathcal{D})$ that maximizes the decrease in the KL distance (1) is

$$\psi^*(\cdot;\mathcal{D}) = \mathbb{E}_{q(\boldsymbol{\theta})}[k(\boldsymbol{\theta},\cdot)\nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{\theta};\mathcal{D}) + \nabla_{\boldsymbol{\theta}} k(\boldsymbol{\theta},\cdot)] . \quad (2)$$

Theorem 1 concerns updating samples from $q(\boldsymbol{\theta})$ assuming fixed $q(\mathcal{Z})$. To similarly update $q(\mathcal{Z})$ with $q(\boldsymbol{\theta})$ fixed, we employ a complementary form of Theorem 1 (omitted for brevity); in that case we consider transformation $T(\mathcal{Z}) = \mathcal{Z} + \epsilon\psi(\mathcal{Z};\mathcal{D})$, with $\mathcal{Z} \sim q(\mathcal{Z})$. The function $\psi(\mathcal{Z};\mathcal{D})$ is also assumed to be in a RKHS.

The expectations in (1) and (2) are approximated by samples, and we have

$$\boldsymbol{\theta}_j^{(t+1)} = \boldsymbol{\theta}_j^{(t)} + \epsilon\Delta\boldsymbol{\theta}_j^{(t)}\,, \tag{3}$$

with

$$\Delta\boldsymbol{\theta}_j^{(t)} \approx \frac{1}{M}\sum_{j'=1}^{M}[k_{\boldsymbol{\theta}}(\boldsymbol{\theta}_{j'}^{(t)},\boldsymbol{\theta}_j^{(t)})\nabla_{\boldsymbol{\theta}_{j'}^{(t)}}\log\tilde{p}(\boldsymbol{\theta}_{j'}^{(t)};\mathcal{D}) + \nabla_{\boldsymbol{\theta}_{j'}^{(t)}}k_{\boldsymbol{\theta}}(\boldsymbol{\theta}_{j'}^{(t)},\boldsymbol{\theta}_j^{(t)}))].$$

$$\nabla_{\boldsymbol{\theta}}\log\tilde{p}(\boldsymbol{\theta};\mathcal{D}) \approx \frac{1}{M}\sum_{n=1}^{N}\sum_{j=1}^{M}\nabla_{\boldsymbol{\theta}}\log p(\boldsymbol{x}_n|\boldsymbol{z}_{jn},\boldsymbol{\theta})p(\boldsymbol{\theta})\,,$$

Similarly, when updating samples of the latent variables, we have

$$\boldsymbol{z}_{jn}^{(t+1)} = \boldsymbol{z}_{jn}^{(t)} + \epsilon\Delta\boldsymbol{z}_{jn}^{(t)}\,, \tag{4}$$

with

$$\Delta\boldsymbol{z}_{jn}^{(t)} = \frac{1}{M}\sum_{j'=1}^{M}[k_{\boldsymbol{z}}(\boldsymbol{z}_{j'n}^{(t)},\boldsymbol{z}_{jn}^{(t)})\nabla_{\boldsymbol{z}_{j'n}^{(t)}}\log\tilde{p}(\boldsymbol{z}_{j'n}^{(t)};\mathcal{D}) + \nabla_{\boldsymbol{z}_{j'n}^{(t)}}k_{\boldsymbol{z}}(\boldsymbol{z}_{j'n}^{(t)},\boldsymbol{z}_{jn}^{(t)})] \tag{5}$$

$$\nabla_{\boldsymbol{z}_n}\log\tilde{p}(\boldsymbol{z}_n;\mathcal{D}) \approx \tfrac{1}{M}\sum_{j=1}^{M}\nabla_{\boldsymbol{z}_n}\log p(\boldsymbol{x}_n|\boldsymbol{z}_n,\boldsymbol{\theta}'_j)p(\boldsymbol{z}_n)\,,$$

The kernels used to update samples of $\boldsymbol{\theta}$ and $\boldsymbol{z}_n$ are in general different, denoted respectively $k_{\boldsymbol{\theta}}(\cdot,\cdot)$ and $k_{\boldsymbol{z}}(\cdot,\cdot)$. ϵ is a small step size.

2.2. Stein Recognition Model

At iteration t of the above learning procedure, we realize a set of latent-variable (code) samples $\{\boldsymbol{z}_{jn}^{(t)}\}_{j=1}^{M}$ for each $\boldsymbol{x}_n \in \mathcal{D}$ under analysis. For large N this training may be computationally expensive. Further, the need to evolve (learn) samples $\{\boldsymbol{z}_{j*}\}_{j=1}^{M}$ for each new test sample $\boldsymbol{x}_*$ is undesirable. We therefore develop a *recognition model* that efficiently computes samples of codes for a data sample of interest. The recognition model draws samples via $\boldsymbol{z}_{jn} = \boldsymbol{f}_{\boldsymbol{\eta}}(\boldsymbol{x}_n,\boldsymbol{\xi}_{jn})$ with $\boldsymbol{\xi}_{jn} \sim q_0(\boldsymbol{\xi})$. Distribution $q_0(\boldsymbol{\xi})$ is selected such that it may be sampled easily, *e.g.*, isotropic Gaussian.

After each iteration of Algorithm 1, we refine recognition model $\boldsymbol{f}_{\boldsymbol{\eta}}(\boldsymbol{x},\boldsymbol{\xi})$ to mimic the Stein sample dynamics. Assume recognition-model parameters $\boldsymbol{\eta}^{(t)}$ have been learned thus far. Using $\boldsymbol{\eta}^{(t)}$, latent codes for iteration t are constituted as $\boldsymbol{z}_{jn}^{(t)} = \boldsymbol{f}_{\boldsymbol{\eta}^{(t)}}(\boldsymbol{x}_n,\boldsymbol{\xi}_{jn})$, with $\boldsymbol{\xi}_{jn} \sim q_0(\boldsymbol{\xi})$. These codes are computed for all data $\boldsymbol{x}_n \in \mathcal{B}_t$, where $\mathcal{B}_t \subset \mathcal{D}$ is the minibatch of data at iteration t. The change in the codes is $\Delta\boldsymbol{z}_{jn}^{(t)}$, as defined (5). We then update $\boldsymbol{\eta}$ to match the refined codes, as

$$\boldsymbol{\eta}^{(t+1)} = \arg\min_{\boldsymbol{\eta}}\sum_{\boldsymbol{x}_n\in\mathcal{B}_t}\sum_{j=1}^{M}\|\boldsymbol{f}_{\boldsymbol{\eta}}(\boldsymbol{x}_n,\boldsymbol{\xi}_{jn}) - \boldsymbol{z}_{jn}^{(t+1)}\|^2\,. \tag{6}$$

The analytic solution of (6) is intractable. We update $\boldsymbol{\eta}$ with K steps of gradient descent as

$$\boldsymbol{\eta}^{(t,k)} = \boldsymbol{\eta}^{(t,k-1)} - \delta\textstyle\sum_{\boldsymbol{x}_n\in\mathcal{B}_t}\sum_{j=1}^{M}\Delta\boldsymbol{\eta}_{jn}^{(t,k-1)} \tag{7}$$

$$\Delta\boldsymbol{\eta}_{jn}^{(t,k-1)} = \partial_{\boldsymbol{\eta}}\boldsymbol{f}_{\boldsymbol{\eta}}(\boldsymbol{x}_n,\boldsymbol{\xi}_{jn})(\boldsymbol{f}_{\boldsymbol{\eta}}(\boldsymbol{x}_n,\boldsymbol{\xi}_{jn}) - \boldsymbol{z}_{jn}^{(t+1)})|_{\boldsymbol{\eta}=\boldsymbol{\eta}^{(t,k-1)}}$$

where δ is a small step size, $\boldsymbol{\eta}^{(t)} = \boldsymbol{\eta}^{(t,0)}$ and $\boldsymbol{\eta}^{(t+1)} = \boldsymbol{\eta}^{(t,K)}$, and $\partial_{\boldsymbol{\eta}}\boldsymbol{f}_{\boldsymbol{\eta}}(\boldsymbol{x}_n,\boldsymbol{\xi}_{jn})$ is the transpose of the Jacobian of $\boldsymbol{f}_{\boldsymbol{\eta}}(\boldsymbol{x}_n,\boldsymbol{\xi}_{jn})$ w.r.t. $\boldsymbol{\eta}$. Note that the use of minibatches mitigates challenges of training with large training sets, $\mathcal{D}$.

A similar concept to the above was developed in Wang & Liu (2016), although in that work neither the VAE nor recognition model were considered.

3. Stein Variational Importance Weighted Autoencoder (Stein VIWAE)

3.1. Multi-sample importance-weighted KL divergence

Consider a *known* joint distribution $p(\boldsymbol{x},\boldsymbol{z})$, where $\boldsymbol{x}$ is observed and $\boldsymbol{z}$ is latent (*e.g.*, corresponding to the code discussed above). The marginal log-likelihood $\log p(\boldsymbol{x}) = \log\int p(\boldsymbol{x},\boldsymbol{z})d\boldsymbol{z}$ is typically intractable. One often seeks to bound $\log p(\boldsymbol{x})$ via an approximation $q(\boldsymbol{z}|\boldsymbol{x})$ to the posterior $p(\boldsymbol{z}|\boldsymbol{x})$:

$$\mathcal{L}(\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\{\log[p(\boldsymbol{x},\boldsymbol{z})/q(\boldsymbol{z}|\boldsymbol{x})]\} \le \log p(\boldsymbol{x})\,.$$

This lower bound plays a pivotal in the VAE (Kingma & Welling, 2014), and $q(\boldsymbol{z}|\boldsymbol{x})$ is analogous to the encoder discussed above. Recently, Burda et al. (2016); Mnih & Rezende (2016) showed that the multi-sample (k samples) importance-weighted estimator

$$\mathcal{L}_k(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}^1,\ldots,\boldsymbol{z}^k\sim q(\boldsymbol{z}|\boldsymbol{x})}\left[\log\tfrac{1}{k}\textstyle\sum_{i=1}^{k}\frac{p(\boldsymbol{x},\boldsymbol{z}^i)}{q(\boldsymbol{z}^i|\boldsymbol{x})}\right]\,, \tag{8}$$

provides a tighter lower bound and a better proxy for the log-likelihood, where $\boldsymbol{z}^1,\ldots,\boldsymbol{z}^k$ are random variables sampled independently from $q(\boldsymbol{z}|\boldsymbol{x})$.

Recall from (1) that the KL divergence played a key role in the Stein-based learning of Sec. 2. Equation (8) motivates replacement of the KL objective function with the multi-sample importance-weighted KL divergence

$$\mathrm{KL}_{q,p}^{k}(\boldsymbol{\Theta};\mathcal{D}) \triangleq -\mathbb{E}_{\boldsymbol{\Theta}^{1:k}\sim q(\boldsymbol{\Theta})}\left[\log\tfrac{1}{k}\textstyle\sum_{i=1}^{k}\frac{p(\boldsymbol{\Theta}^i|\mathcal{D})}{q(\boldsymbol{\Theta}^i)}\right]\,, \tag{9}$$

where $\mathbf{\Theta} = (\boldsymbol{\theta}, \mathcal{Z})$ and $\mathbf{\Theta}^{1:k} = \mathbf{\Theta}^1, \ldots, \mathbf{\Theta}^k$ are independent samples from $q(\boldsymbol{\theta}, \mathcal{Z})$. Note that the special case of $k = 1$ recovers the standard KL divergence.

Inspired by Burda et al. (2016), the following theorem (proved in Appendix A) shows that increasing the number of samples k is guaranteed to reduce the KL divergence and provide a better approximation of target distribution.

Theorem 2 *For any natural number k, we have*

$$KL^k_{q,p}(\mathbf{\Theta}; \mathcal{D}) \geq KL^{k+1}_{q,p}(\mathbf{\Theta}; \mathcal{D}) \geq 0 \,,$$

and if $q(\mathbf{\Theta})/p(\mathbf{\Theta}|\mathcal{D})$ *is bounded*

$$\lim_{k\to\infty} KL^k_{q,p}(\mathbf{\Theta}; \mathcal{D}) = 0 \,.$$

Simliar to the Stein autoencoder, we minimize (9) with a sample transformation based on Stein's operator (detailed in Sec. 3.2) and the recognition model is trained in the same way as Sec. 2.2. Specifically, we first draw samples $\{\boldsymbol{\theta}_j^{1:k}\}_{j=1}^M$ and $\{\boldsymbol{z}_{jn}^{1:k}\}_{j=1}^M$ from a simple distribution $q_0(\cdot)$, and convert these to approximate draws from $p(\boldsymbol{\theta}^{1:k}, \mathcal{Z}^{1:k}|\mathcal{D})$ by minimizing the multi-sample importance weighted KL divergence via nonlinear functional transformation.

3.2. Importance-weighted learning procedure

Theorem 3 *Let* $\mathbf{\Theta}^{1:k}$ *be RVs drawn independently from distribution* $q(\mathbf{\Theta})$ *and* $KL^k_{q,p}(\mathbf{\Theta}, \mathcal{D})$ *is the multi-sample importance weighted KL divergence in (9). Let* $T(\mathbf{\Theta}) = \mathbf{\Theta} + \epsilon\psi(\mathbf{\Theta}; \mathcal{D})$ *and* $q_T(\mathbf{\Theta})$ *represent the distribution of* $\mathbf{\Theta}' = T(\mathbf{\Theta})$. *We have*

$$\nabla_\epsilon \Big(KL^k_{q,p}(\mathbf{\Theta}'; \mathcal{D})\Big)|_{\epsilon=0} = -\mathbb{E}_{\mathbf{\Theta}^{1:k}\sim q(\mathbf{\Theta})}(\mathcal{A}^k_p(\mathbf{\Theta}^{1:k}; \mathcal{D})) \,,$$

where

$$\mathcal{A}^k_p(\mathbf{\Theta}^{1:k}; \mathcal{D}) = \tfrac{1}{\tilde{\omega}} \textstyle\sum_{i=1}^k \omega_i \Big(trace\big(\mathcal{A}_p(\mathbf{\Theta}^i; \mathcal{D})\big)\Big)$$
$$\omega_i = p(\mathbf{\Theta}^i; \mathcal{D})/q(\mathbf{\Theta}^i), \quad \tilde{\omega} = \textstyle\sum_{i=1}^k \omega_i$$
$$\mathcal{A}_p(\mathbf{\Theta}; \mathcal{D}) = \nabla_{\mathbf{\Theta}} \log \tilde{p}(\mathbf{\Theta}; \mathcal{D}) \psi(\mathbf{\Theta}; \mathcal{D})^T + \nabla_{\mathbf{\Theta}} \psi(\mathbf{\Theta}; \mathcal{D}) \,.$$

The proof is provided in Appendix A.

The following corollary generalizes Theorem 1 via use of importance sampling.

Corollary 3.1 $\boldsymbol{\theta}^{1:k}$ *and* $\mathcal{Z}^{1:k}$ *are RVs drawn independently from distributions* $q(\boldsymbol{\theta})$ *and* $q(\mathcal{Z})$, *respectively. Let* $T(\boldsymbol{\theta}) = \boldsymbol{\theta} + \epsilon\psi(\boldsymbol{\theta}; \mathcal{D})$, $q_T(\boldsymbol{\theta})$ *represent the distribution of* $\boldsymbol{\theta}' = T(\boldsymbol{\theta})$, *and* $\mathbf{\Theta}' = (\boldsymbol{\theta}', \mathcal{Z})$. *We have*

$$\nabla_\epsilon \Big(KL^k_{q_T,p}(\mathbf{\Theta}'; \mathcal{D})\Big)|_{\epsilon=0} = -\mathbb{E}_{\boldsymbol{\theta}^{1:k}\sim q(\boldsymbol{\theta})}(\mathcal{A}^k_p(\boldsymbol{\theta}^{1:k}; \mathcal{D})) \quad (10)$$

where

$$q_T = q_T(\boldsymbol{\theta})q(\mathcal{Z}), \quad p = p(\boldsymbol{\theta}, \mathcal{Z}|\mathcal{D})$$
$$\mathcal{A}^k_p(\boldsymbol{\theta}^{1:k}; \mathcal{D}) = \tfrac{1}{\tilde{\omega}} \textstyle\sum_{i=1}^k \omega_i \mathcal{A}_p(\boldsymbol{\theta}^i; \mathcal{D})$$
$$\omega_i = \mathbb{E}_{\mathcal{Z}^i\sim q(\mathcal{Z})}\left[\tfrac{p(\boldsymbol{\theta}^i, \mathcal{Z}^i, \mathcal{D})}{q(\boldsymbol{\theta}^i)q(\mathcal{Z}^i)}\right], \quad \tilde{\omega} = \textstyle\sum_{i=1}^k \omega_i$$
$$\mathcal{A}_p(\boldsymbol{\theta}; \mathcal{D}) = \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{\theta}; \mathcal{D}) \psi(\boldsymbol{\theta}; \mathcal{D})^T + \nabla_{\boldsymbol{\theta}} \psi(\boldsymbol{\theta}; \mathcal{D})$$
$$\log \tilde{p}(\boldsymbol{\theta}; \mathcal{D}) = \mathbb{E}_{\mathcal{Z}\sim q(\mathcal{Z})}[\log p(\mathcal{D}, \mathcal{Z}, \boldsymbol{\theta})] \,.$$

The following corollary generalizes (2).

Corollary 3.2 *Assume* $\psi(\boldsymbol{\theta}; \mathcal{D})$ *lives in a reproducing kernel Hilbert space (RKHS) with kernel* $k_{\boldsymbol{\theta}}(\cdot, \cdot)$. *The solution for* $\psi(\boldsymbol{\theta}; \mathcal{D})$ *that maximizes the decrease in the KL distance (10) is*

$$\psi^*(\cdot; \mathcal{D}) = \mathbb{E}_{\boldsymbol{\theta}^{1:k}\sim q(\boldsymbol{\theta})}\Big[\tfrac{1}{\tilde{\omega}} \textstyle\sum_{i=1}^k \omega_i \big(\nabla_{\boldsymbol{\theta}^i} k_{\boldsymbol{\theta}}(\boldsymbol{\theta}^i, \cdot)$$
$$+ k_{\boldsymbol{\theta}}(\boldsymbol{\theta}^i, \cdot) \nabla_{\boldsymbol{\theta}^i} \log \tilde{p}(\boldsymbol{\theta}^i; \mathcal{D})\big)\Big] \,. \quad (11)$$

Corollary 3.1 and Corollary 3.2 provide a means of updating multiple samples $\{\boldsymbol{\theta}_j^{1:k}\}_{j=1}^M$ from $q(\boldsymbol{\theta})$ via $T(\boldsymbol{\theta}^i) = \boldsymbol{\theta}^i + \epsilon\psi(\boldsymbol{\theta}^i; \mathcal{D})$. The expectation wrt $q(\mathcal{Z})$ is approximated via samples drawn from $q(\mathcal{Z})$. Similarly, we can employ a complementary form of Corollary 3.1 and Corollary 3.2 to update multiple samples $\{\mathcal{Z}_j^{1:k}\}_{j=1}^M$ from $q(\mathcal{Z})$. This suggests an importance-weighted learning procedure that alternates between update of particles $\{\boldsymbol{\theta}_j^{1:k}\}_{j=1}^M$ and $\{\mathcal{Z}_j^{1:k}\}_{j=1}^M$, which is similiar as Sec. 2.1.

Specifically, let $\{\boldsymbol{\theta}_j^{1:k,t}\}_{j=1}^M$ and $\{\boldsymbol{z}_{jn}^{1:k,t}\}_{j=1}^M$ denote the samples acquired at iteration t of the learning procedure. To update samples of $\boldsymbol{\theta}^{1:k}$, we apply the transformation $\boldsymbol{\theta}_j^{(i,t+1)} = T(\boldsymbol{\theta}_j^{(i,t)}; \mathcal{D}) = \boldsymbol{\theta}_j^{(i,t)} + \epsilon\psi(\boldsymbol{\theta}_j^{(i,t)}; \mathcal{D})$, for $i = 1, \ldots, k$, by approximating the expectation by samples $\{\boldsymbol{z}_{jn}^{1:k}\}_{j=1}^M$, and we have

$$\boldsymbol{\theta}_j^{(i,t+1)} = \boldsymbol{\theta}_j^{(i,t)} + \epsilon\Delta\boldsymbol{\theta}_j^{(i,t)}, \text{ for } i = 1, \ldots, k, \quad (12)$$

with

$$\Delta\boldsymbol{\theta}_j^{(i,t)} \approx \tfrac{1}{M} \textstyle\sum_{j'=1}^M \Big[\tfrac{1}{\tilde{\omega}} \sum_{i'=1}^k \omega_i \big(\nabla_{\boldsymbol{\theta}_{j'}^{(i',t)}} k_{\boldsymbol{\theta}}(\boldsymbol{\theta}_{j'}^{(i',t)}, \boldsymbol{\theta}_j^{(i,t)})\big)$$
$$+ k(\boldsymbol{\theta}_{j'}^{(i',t)}, \boldsymbol{\theta}_j^{(i,t)}) \nabla_{\boldsymbol{\theta}_{j'}^{(i',t)}} \log \tilde{p}(\boldsymbol{\theta}_{j'}^{(i',t)}; \mathcal{D})\Big]$$
$$\omega_i \approx \tfrac{1}{M} \textstyle\sum_{n=1}^N \sum_{j=1}^M \tfrac{p(\boldsymbol{\theta}^i, \boldsymbol{z}_{jn}^i, \boldsymbol{x}_n)}{q(\boldsymbol{\theta}^i)q(\boldsymbol{z}_{jn}^i)}, \; \tilde{\omega} = \sum_{i=1}^k \omega_i$$
$$\nabla_{\boldsymbol{\theta}} \log \tilde{p}(\boldsymbol{\theta}; \mathcal{D}) \approx \tfrac{1}{M} \textstyle\sum_{n=1}^N \sum_{j=1}^M \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x}_n|\boldsymbol{z}_{jn}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \,.$$

Similarly, when updating samples of the latent variables, we have

$$\boldsymbol{z}_{jn}^{(i,t+1)} = \boldsymbol{z}_{jn}^{(i,t)} + \epsilon\Delta\boldsymbol{z}_{jn}^{(i,t)}, \text{ for } i = 1, \ldots, k, \quad (13)$$

with

$$\Delta \boldsymbol{z}_{jn}^{(i,t)} \approx \frac{1}{M}\sum_{j'=1}^{M}\Big[\frac{1}{\tilde{\omega}_n}\sum_{i'=1}^{k}\omega_{in}\big(\nabla_{\boldsymbol{z}_{j'n}^{(i',t)}} k_{\boldsymbol{z}}(\boldsymbol{z}_{j'n}^{(i',t)}, \boldsymbol{z}_{jn}^{(i,t)}))$$
$$+ k_{\boldsymbol{z}}(\boldsymbol{z}_{j'n}^{(i',t)}, \boldsymbol{z}_{jn}^{(i,t)})\nabla_{\boldsymbol{z}_{j'n}^{(i',t)}} \log \tilde{p}(\boldsymbol{z}_{j'n}^{(i',t)}; \mathcal{D})\Big]$$
$$\omega_{in} \approx \frac{1}{M}\sum_{j=1}^{M}\frac{p(\boldsymbol{\theta}^i, \boldsymbol{z}_{jn}^i, \boldsymbol{x}_n)}{q(\boldsymbol{\theta}^i)q(\boldsymbol{z}_{jn}^i)},\ \tilde{\omega}_n = \sum_{i=1}^{k}\omega_{in}$$
$$\nabla_{\boldsymbol{z}_n}\log\tilde{p}(\boldsymbol{z}_n; \mathcal{D}) \approx \frac{1}{M}\sum_{j=1}^{M}\nabla_{\boldsymbol{z}_n}\log p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta}'_j)p(\boldsymbol{z}_n)$$

One may worry about the variance of this gradient. However, Burda et al. (2016) has showed that the estimator based on *log* of importance weighted average will not lead to high variance.

4. Semi-supervised Learning with Stein VAE and Stein VIWAE

We extend our model to semi-supervised learning. Consider labeled data as pairs $\mathcal{D}_l = \{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^{N_l}$, where the label $\boldsymbol{y}_n \in \{1, \dots, C\}$ and the decoder is modeled as $(\boldsymbol{x}_n, \boldsymbol{y}_n|\boldsymbol{z}_n) \sim p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{z}_n; \boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) = p(\boldsymbol{x}|\boldsymbol{z}_n; \boldsymbol{\theta})p(\boldsymbol{y}|\boldsymbol{z}_n; \tilde{\boldsymbol{\theta}})$, where $\tilde{\boldsymbol{\theta}}$ represents the parameters of the decoder for labels. The set of codes associated with all labeled data is represented $\mathcal{Z}_l = \{\boldsymbol{z}_n\}_{n=1}^{N_l}$. We desire to approximate the posterior distribution on the entire dataset $p(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}, \mathcal{Z}, \mathcal{Z}_l|\mathcal{D}, \mathcal{D}_l)$ via samples, where $\mathcal{D}$ represents the unlabeled data, and $\mathcal{Z}$ is the set of codes associated with $\mathcal{D}$. In the following, we will only discuss how to update the samples of $\boldsymbol{\theta}$, $\tilde{\boldsymbol{\theta}}$ and $\mathcal{Z}_l$. Updating samples $\mathcal{Z}$ is the same as (4) and (13) for Stein VAE and Stein VIWAE, respectively.

To make the following discussion concrete, we describe learning within the context of Stein VAE, generalizing the models in Sec. 2. This setup is also applied to Stein VIWAE, generalizing the models in Sec. 3. Assume $\{\boldsymbol{\theta}_j\}_{j=1}^M$ drawn from distribution $q(\boldsymbol{\theta})$, $\{\tilde{\boldsymbol{\theta}}_j\}_{j=1}^M$ drawn from distribution $q(\tilde{\boldsymbol{\theta}})$, and samples $\{\boldsymbol{z}_{jn}\}_{j=1}^M$ drawn from (distinct) distribution $q(\mathcal{Z}_l)$. The following corollary generalizes Theorem 1 and (2), which is useful for defining how best to update $\{\boldsymbol{\theta}_j\}_{j=1}^M$.

Corollary 3.3 *Assume $\boldsymbol{\theta}$, $\tilde{\boldsymbol{\theta}}$, $\mathcal{Z}$ and $\mathcal{Z}_l$ are RVs drawn from distributions $q(\boldsymbol{\theta})$, $q(\tilde{\boldsymbol{\theta}})$, $q(\mathcal{Z})$ and $q(\mathcal{Z}_l)$, respectively. Consider the transformation $T(\boldsymbol{\theta}) = \boldsymbol{\theta} + \epsilon\psi(\boldsymbol{\theta}; \mathcal{D}, \mathcal{D}_l)$ where $\psi(\boldsymbol{\theta}; \mathcal{D}, \mathcal{D}_l)$ lives in a RKHS with kernel $k_{\boldsymbol{\theta}}(\cdot, \cdot)$. Let $q_T(\boldsymbol{\theta})$ represent the distribution of $\boldsymbol{\theta}' = T(\boldsymbol{\theta})$. We have*

$$\nabla_\epsilon\Big(KL(q_T\|p)\Big)|_{\epsilon=0} = -\mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta})}(\mathcal{A}_p(\boldsymbol{\theta}; \mathcal{D}, \mathcal{D}_l)), \quad (14)$$

and the solution for $\psi(\boldsymbol{\theta}; \mathcal{D}, \mathcal{D}_l)$ that maximizes the change in the KL distance (14) is

$$\psi^*(\cdot; \hat{\mathcal{D}}) = \mathbb{E}_{q(\boldsymbol{\theta})}[k(\boldsymbol{\theta}, \cdot)\nabla_{\boldsymbol{\theta}}\log\tilde{p}(\boldsymbol{\theta}; \hat{\mathcal{D}}) + \nabla_{\boldsymbol{\theta}}k(\boldsymbol{\theta}, \cdot)], \quad (15)$$

where $\hat{\mathcal{D}} = (\mathcal{D}, \mathcal{D}_l)$, and

$$q_T = q_T(\boldsymbol{\theta})q(\mathcal{Z})q(\tilde{\boldsymbol{\theta}}),\ \ p = p(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}, \mathcal{Z}|\mathcal{D}, \mathcal{D}_l)$$
$$\mathcal{A}_p(\boldsymbol{\theta}; \mathcal{D}, \mathcal{D}_l) = \nabla_{\boldsymbol{\theta}}\log\tilde{p}(\boldsymbol{\theta}; \mathcal{D}, \mathcal{D}_l)\psi(\boldsymbol{\theta}; \mathcal{D}, \mathcal{D}_l)^T + \nabla_{\boldsymbol{\theta}}\psi(\boldsymbol{\theta}; \mathcal{D}, \mathcal{D}_l)$$
$$\log\tilde{p}(\boldsymbol{\theta}; \mathcal{D}, \mathcal{D}_l) = \mathbb{E}_{\mathcal{Z}\sim q(\mathcal{Z})}[\log p(\mathcal{D}|\mathcal{Z}, \boldsymbol{\theta})] + \mathbb{E}_{\mathcal{Z}_l\sim q(\mathcal{Z}_l)}[\log p(\mathcal{D}_l|\mathcal{Z}_l, \boldsymbol{\theta})].$$

The expectations in (14) and (15) are approximated by samples. Updating samples $\boldsymbol{\theta}$ and $\tilde{\boldsymbol{\theta}}$ is similar to (3), with details provided in Appendix B. Samples of $\boldsymbol{z}_n \in \mathcal{Z}_l$ are updated

$$\boldsymbol{z}_{jn}^{(t+1)} = \boldsymbol{z}_{jn}^{(t)} + \epsilon\Delta\boldsymbol{z}_{jn}^{(t)},$$

with

$$\Delta\boldsymbol{z}_{jn}^{(t)} = \frac{1}{M}\sum_{j'=1}^{M}[k_{\boldsymbol{z}}(\boldsymbol{z}_{j'n}^{(t)}, \boldsymbol{z}_{jn}^{(t)})\nabla_{\boldsymbol{z}_{j'n}^{(t)}}\log\tilde{p}(\boldsymbol{z}_{j'n}^{(t)}; \mathcal{D}_l) + \nabla_{\boldsymbol{z}_{j'n}^{(t)}}k_{\boldsymbol{z}}(\boldsymbol{z}_{j'n}^{(t)}, \boldsymbol{z}_{jn}^{(t)}))]$$
$$\nabla_{\boldsymbol{z}_n}\log\tilde{p}(\boldsymbol{z}_n; \mathcal{D}_l) \approx \frac{1}{M}\sum_{j=1}^{M}\nabla_{\boldsymbol{z}_n}p(\boldsymbol{z}_n)\{\log p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta}'_j) + \zeta\log p(\boldsymbol{y}_n|\boldsymbol{z}_n, \tilde{\boldsymbol{\theta}}'_j)\},$$

where ζ is a tuning parameter that balances the two components. Motivated by assigning the same weight to every data point (Pu et al., 2016), we set $\zeta = N_X/(C\rho)$ in the experiments, where N_X is the dimension of $\boldsymbol{x}_n$, C is the number of categories for the corresponding label and ρ is the proportion of labeled data in the mini-batch.

5. Experiments

For all experiments, we use a radial basis-function (RBF) kernel as in Liu & Wang (2016), *i.e.*, $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\frac{1}{h}\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2)$, where the bandwidth, h, is the median of pairwise distances between current samples. $q_0(\boldsymbol{\theta})$ and $q_0(\boldsymbol{\xi})$ are set to isotropic Gaussian distributions. We share the samples of $\boldsymbol{\xi}$ across data points, *i.e.*, $\boldsymbol{\xi}_{jn} = \boldsymbol{\xi}_j$, for $n = 1, \dots, N$ (this is not necessary, but it saves computation). The samples of $\boldsymbol{\theta}$ and $\boldsymbol{z}$, and parameters of the recognition model, $\boldsymbol{\eta}$, are optimized via Adam (Kingma & Ba, 2015) with learning rate 0.0002, and with the gradient approximations in (3), (4), (7), (12) and (13). We do not perform any dataset-specific tuning or regularization other than dropout (Srivastava et al., 2014) and early stopping on validation sets. We set $M = 100$ and $k = 50$, and use minibatches of size 64 for all experiments, unless otherwise specified.

5.1. Expressive power of Stein recognition model

We first evaluate the expressive power of non-Gaussian posterior approximation based on the Stein recognition

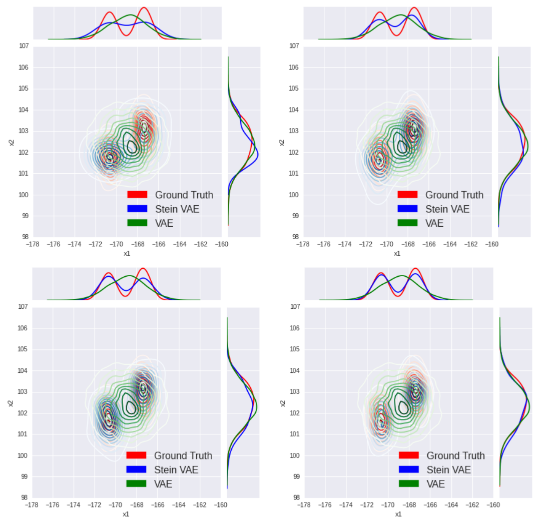


Figure 1. Approximation of posterior distribution: Stein VAE *vs.* VAE. The figures represent different samples of Stein VAE. (Top-left) 10 samples. (Top-right) 20 samples. (Bottom-left) 50 samples. (Bottom-right) 100 samples.

model developed in Section 2.2. This is done generating samples of $\boldsymbol{z}$ using standard VAE (Kingma & Welling, 2014) and via the proposed Stein VAE, with each compared to ground truth on testing data (considering a non-trivial example for which the exact posterior is available). We fix the decoder (generative model) parameters $\boldsymbol{\theta}$, and only generate samples of $\boldsymbol{z}$ and update encoder parameters $\boldsymbol{\eta}$ during training, for both standard VAE and Stein VAE.

Gaussian Mixture Model We synthesize data by (i) drawing $\boldsymbol{z}_n \sim \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_1, \mathbf{I}) + \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_2, \mathbf{I})$, where $\boldsymbol{\mu}_1 = [5,5]^T$, $\boldsymbol{\mu}_2 = [-5,-5]^T$; (ii) drawing $\boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{\theta}\boldsymbol{z}_n, \sigma^2\mathbf{I})$, where $\boldsymbol{\theta} = \left(\begin{smallmatrix} 2 & -1 \\ 1 & -2 \end{smallmatrix}\right)$ and $\sigma = 0.1$. The recognition model $f_{\boldsymbol{\eta}}(\boldsymbol{x}_n, \boldsymbol{\xi}_j)$ is specified as a multi-layer perceptron (MLP) with 100 hidden units, by first concatenating $\boldsymbol{\xi}_j$ and $\boldsymbol{x}_n$ into a long vector. The dimension of $\boldsymbol{\xi}_j$ is set to 2. The recognition model for standard VAE is also an MLP with 100 hidden units, and with the assumption of a Gaussian distribution for the latent codes (Kingma & Welling, 2014).

We generate $N = 10,000$ data points for training and 10 data points for testing. The *analytic* form of true posterior distribution is provided in Appendix C. Figure 1 shows the performance of Stein VAE approximations for the true posterior using $M = 10, 20, 50$ and 100 samples on one test data; other similar examples are provided in Appendix D. Our Stein recognition model is able to capture the multi-modal posterior and produce accurate density approximation. As the number of samples M is increased, we observe a substantial improvement in the approximation quality. By contrast, the Gaussian form of the standard VAE posterior is incapable of capturing the multimodal form of the true posterior.

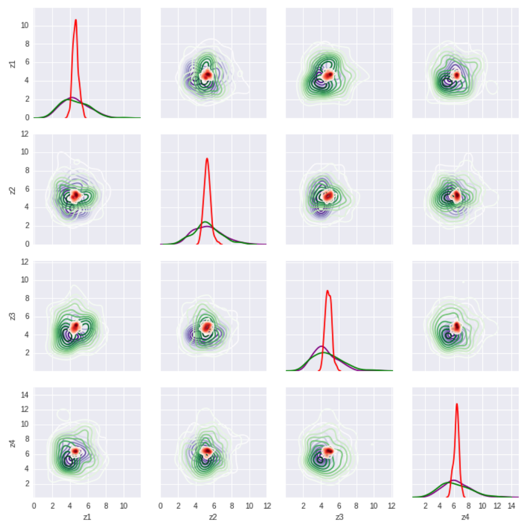


Figure 2. Univariate marginals and pairwise posteriors. Purple, red and green represent the distribution inferred from MCMC, standard VAE and Stein VAE, respectively.

Poisson Factor Analysis Given a discrete vector $\boldsymbol{x}_n \in \mathbb{Z}_+^P$, Poisson factor analysis (Zhou et al., 2012) assumes $\boldsymbol{x}_n$ is a weighted combination of V latent factors $\boldsymbol{x}_n \sim \text{Pois}(\boldsymbol{\theta}\boldsymbol{z}_n)$, where $\boldsymbol{\theta} \in \mathbb{R}_+^{P\times V}$ is the factor loadings matrix and $\boldsymbol{z}_n \in \mathbb{R}_+^V$ is the vector of factor scores. We consider topic modeling with Dirichlet priors on $\boldsymbol{\theta}_v$ (v-th column of $\boldsymbol{\theta}$) and gamma priors on each component of $\boldsymbol{z}_n$.

We evaluate our model on the *20 Newsgroups* dataset containing $N = 18,845$ documents with a vocabulary of $P = 2,000$. The data are partitioned into 10,314 training, 1,000 validation and 7,531 test documents. The number of factors (topics) is set to $V = 128$. $\boldsymbol{\theta}$ is first learned by Markov chain Monte Carlo (MCMC) (Gan et al., 2015). We then fix $\boldsymbol{\theta}$ at its MAP value, and only learn the recognition model $\boldsymbol{\eta}$ using standard VAE and Stein VAE; this is done, as in the previous example, to examine the accuracy of the recognition model to estimate the posterior of the latent codes (factor scores), isolated from estimation of $\boldsymbol{\theta}$. The recognition model is an MLP with 100 hidden units.

An analytic form of the true posterior distribution $p(\boldsymbol{z}_n|\boldsymbol{x}_n)$ is intractable for this problem. Consequently, we employ samples collected from MCMC as ground truth. Specifically, with $\boldsymbol{\theta}$ fixed, we sample $\boldsymbol{z}_n$ via Gibbs sampling, using 2,000 burn-in iterations followed by 2,500 collection draws, retaining every 10th collection sample. We show the marginal and pairwise posterior of one test data point in Fig. 2. Additional results are provided in Appendix D. As observed, Stein VAE leads to a more accurate approximation than standard VAE, compared to the MCMC samples.

Considering Fig. 2, note that VAE significantly underestimates the variance of the posterior (examining the marginals), a well-known problem of variational Bayesian

Table 1. Negative log-likelihood(NLL) on MNIST. †Trained with VAE and tested with IWAE. ‡Trained and tested with IWAE.

Method	NLL
DGLM (Rezende et al., 2014)	89.90
Normalizing flow (Rezende & Mohamed, 2015)	85.10
VAE + IWAE (Burda et al., 2016)†	86.76
IWAE + IWAE (Burda et al., 2016)‡	84.78
Stein VAE + ELBO	85.21
Stein VAE + S-ELBO	84.98
Stein VIWAE + ELBO	83.01
Stein VIWAE + S-ELBO	**82.88**

analysis (Han et al., 2016). This is manifested here despite the fact that the marginals are not significantly different in form from the Gaussian assumption of the standard VAE. By contrast, the proposed Stein VAE does not assume a parametric form for the posterior, and therefore it does not seek to directly estimate the mean or variance of the posterior. We observe that Stein VAE yields highly accurate approximations to the true posterior, with this performed efficiently at test time via the recognition model.

5.2. Density estimation

Data We consider five benchmark datasets: MNIST and four text corpora: *20 Newsgroups* (20News), *New York Times* (NYT), *Science* and *RCV1-v2* (RCV2). For MNIST, we used the standard split of 50K training, 10K validation and 10K test examples. The latter three text corpora consist of 133K, 166K and 794K documents. These three datasets are split into 1K validation, 10K testing and the rest for training.

Evaluation Given new data $\boldsymbol{x}_*$ (testing data), the marginal log-likelihood/perplexity values are estimated by the variational lower bound while integrating the decoder parameters $\boldsymbol{\theta}$ out

$$\log p(\boldsymbol{x}_*) \geq \mathbb{E}_{q(\boldsymbol{z}_*)}[\log p(\boldsymbol{x}_*, \boldsymbol{z}_*)] + \mathcal{H}(q(\boldsymbol{z}_*)) , \quad (16)$$

where $p(\boldsymbol{x}_*, \boldsymbol{z}_*) = \mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\boldsymbol{x}_*, \boldsymbol{\theta}, \boldsymbol{z}_*)]$ and $\mathcal{H}(q(\cdot)) = -\mathbb{E}_q(\log q(\cdot))$ is the entropy. The expectation is approximated with samples $\{\boldsymbol{\theta}_j\}_{j=1}^M$ and $\{\boldsymbol{z}_{*j}\}_{j=1}^M$ with $\boldsymbol{z}_{*j} = \boldsymbol{f}_{\boldsymbol{\eta}}(\boldsymbol{x}_*, \boldsymbol{\xi}_j)$, $\boldsymbol{\xi}_j \sim q_0(\boldsymbol{\xi})$. Directly evaluating $q(\boldsymbol{z}_*)$ is intractable. We alternatively estimate it via density transformation $q(\boldsymbol{z}) = q_0(\boldsymbol{\xi}) \left| \det \frac{\partial \boldsymbol{f}_{\boldsymbol{\eta}}(\boldsymbol{x}, \boldsymbol{\xi})}{\partial \boldsymbol{\xi}} \right|^{-1}$. Note that we only need to compute the Jacobian determinant one time during testing, which will not lead to very high computational complexity.

We further estimate the marginal log-likelihood/perplexity values via the stochastic variational lower bound, as the mean of 5K-sample importance weighting estimate (Burda et al., 2016). Therefore, for each dataset, we report four results: (i) *Stein VAE + ELBO*, (ii) *Stein VAE + S-ELBO*, (iii) *Stein VIWAE + ELBO* and (iv) *Stein VIWAE + S-ELBO*; the first term denotes the training procedure is employed as Stein VAE in Sec. 2 or Stein VIWAE in Section 3; the second term denotes the testing log-likelihood/perplexity is estimated by the ELBO in (16) or the stochastic variational lower bound, S-ELBO (Burda et al., 2016).

Table 2. Test perplexities on four text corpora. §(Larochelle & Laulyi, 2012); †(Ranganath et al., 2015); ‡(Miao et al., 2016).

Method	20News	NYT	Science	RCV2
DocNADE§	896	2496	1725	742
DEF†	—	2416	1576	—
NVDM‡	852	—	—	550
Stein VAE + ELBO	849	2402	1499	549
Stein VAE + S-ELBO	845	2401	1497	544
Stein VIWAE + ELBO	837	2315	1453	523
Stein VIWAE + S-ELBO	**829**	**2277**	**1421**	**518**

Model For MNIST, we train the model with one stochastic layer, $\boldsymbol{z}_n$, with 50 hidden units and two deterministic layers, each with 200 units. The nonlinearity is set as $\tanh$. The visible layer, $\boldsymbol{x}_n$, follows a Bernoulli distribution. For the text corpora, we build a three-layer deep Poisson network (Ranganath et al., 2015). The sizes of hidden units are 200, 200 and 50 for the first, second and third layer, respectively (see Ranganath et al. (2015) for detailed architectures).

Results The log-likelihood/perplexity results are summarized in Tables 1 and 2. On MNIST, our Stein VAE achieves a variational lower bound of -85.21 nats, which outperforms standard VAE with the same model architecture. Our Stein VIWAE achieves a log-likelihood of -82.88 nats, exceeding normalizing flow (-85.1 nats) and importance weighted autoencoder (-84.78 nats), which is the best prior result obtained by feedforward neural network(FNN). Gregor et al. (2015) and Oord et al. (2016), which exploit spatial structure, achieved log-likelihoods of around -80 nats. Our model can also be applied on these models, but this is left as interesting future work. For the text corpora, we observe our Stein VAEs outperform other models and Stein VIWAE further improves the performance on all datasets. These results demonstrate that the proposed models are able to provide a better approximation of the posterior distribution.

5.3. Semi-supervised Classification

We consider semi-supervised classification on MNIST and ImageNet (Russakovsky et al., 2014) data. For each dataset, we report the results obtained by (i) VAE, (ii) Stein VAE, and (iii) Stein VIWAE.

Table 3. Semi-supervised classification error (%) on MNIST. N_ρ is the number of labeled images per class. [§](Kingma et al., 2014); [†]our implementation.

N_ρ	FNN			CNN		
	VAE[§]	Stein VAE	Stein VIWAE	VAE[†]	Stein VAE	Stein VIWAE
10	3.33 ± 0.14	2.78 ± 0.24	2.67 ± 0.09	2.44 ± 0.17	1.94 ± 0.24	**1.90 ± 0.05**
60	2.59 ±0.05	2.13 ± 0.08	2.09 ± 0.03	1.88 ±0.05	1.44 ± 0.04	**1.41 ± 0.02**
100	2.40 ±0.02	1.92 ± 0.05	1.88 ± 0.01	1.47 ±0.02	1.01 ± 0.03	**0.99 ± 0.02**
300	2.18 ±0.04	1.77 ± 0.03	1.75 ± 0.01	0.98 ±0.02	0.89 ± 0.03	**0.86 ± 0.01**

MNIST We randomly split the training set into a labeled and unlabeled set, and the number of labeled samples in each category varies from 10 to 300. We perform testing on the standard test set with 20 different training-set splits.

The decoder for labels is implemented as $p(\boldsymbol{y}_n|\boldsymbol{z}_n, \tilde{\boldsymbol{\theta}}) = \text{softmax}(\tilde{\boldsymbol{\theta}}\boldsymbol{z}_n)$. We consider two types of decoders for images $p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\theta})$ and encoder $\boldsymbol{f}_{\boldsymbol{\eta}}(\boldsymbol{x}, \boldsymbol{\xi})$: *(i) FNN*: Following Kingma et al. (2014), we use a 50-dimensional latent variables $\boldsymbol{z}_n$ and two hidden layers, each with 600 hidden units, for both encoder and decoder; softplus $\log(1 + e^x)$ is employed as the nonlinear activation function. *(ii) All convolutional nets (CNN)*: Inspired by Springenberg et al. (2015), we replace the two hidden layers with 32 and 64 kernels of size 5×5 and a stride of 2. A fully connected layer is stacked on the CNN to produce a 50-dimensional latent variables $\boldsymbol{z}_n$. We use the *leaky* rectified activation (Maas et al., 2013). This architecture is employed for both the encoder and decoder. The input of the encoder is formed by spatially aligning and stacking $\boldsymbol{x}_n$ and $\boldsymbol{\xi}$, while the output of decoder is the image itself.

Table 3 shows the classification results. Our Stein VAE and Stein VIWAE consistently achieve better performance than the VAE, demonstrating the effectiveness of our model in providing good representations of images. We further observe that the variance of Stein VIWAE results is much lower than that of Stein VAE results on small labeled data, indicating the former produces more robust parameter estimates. State-of-the-art results (Rasmus et al., 2015) are achieved by the Ladder network, which can be employed with our Stein-based approach, however, we will consider this extension as future work.

Table 4. Semi-supervised classification accuracy (%) on ImageNet. [†](Pu et al., 2016)

	VAE	Stein VAE	Stein VIWAE	DGDN[†]
1 %	35.92± 1.91	36.44 ± 1.66	36.91 ± 0.98	**43.98± 1.15**
2 %	40.15± 1.52	41.71 ± 1.14	42.57 ± 0.84	**46.92± 1.11**
5 %	44.27± 1.47	46.14 ± 1.02	46.20 ± 0.52	**47.36± 0.91**
10 %	46.92± 1.02	47.83 ± 0.88	**48.67 ± 0.31**	48.41± 0.76
20 %	50.43± 0.41	51.62 ± 0.24	**51.77 ± 0.12**	51.51± 0.28
30 %	53.24± 0.33	55.02 ± 0.22	**55.45 ± 0.11**	54.14± 0.12
40 %	56.89± 0.11	58.17 ± 0.16	**58.21 ± 0.12**	57.34± 0.18

ImageNet 2012 ImageNet 2012 is used to assess the scalability of our model to large datasets. We split the 1.3M training images into an unlabeled and labeled set, and vary the proportion of labeled images from 1% to 40%. The classes are balanced to ensure that no particular class is over-represented, *i.e.*, the ratio of labeled and unlabeled images is the same for each class. We repeat the training process 10 times for the training setting with labeled images ranging from 1% to 10% , and 5 times for the the training setting with labeled images ranging from 20% to 40%. Each time we utilize different sets of images as the unlabeled ones.

We employ all convolutional net (Springenberg et al., 2015) for both the encoder and decoder, which replaces deterministic pooling (*e.g.*, max-pooling) with stridden convolutions. Such a model has been shown to be effective for training higher resolution and deeper generative models (Radford et al., 2016). The decoder for labels is employed as global average pooling with softmax, which has been utilized in state-of-the-art image classification models (He et al., 2016). Batch normalization (Ioffe & Szegedy, 2015) is used to stabilize learning by normalizing the activations throughout the network and preventing relatively small parameter changes from being amplified into larger but suboptimal activation changes in other layers. We use the leaky rectified activation (Maas et al., 2013). Residual connections (He et al., 2016) are incorporated to encourage gradient flow. The model architecture is detailed in Appendix E. Following Krizhevsky et al. (2012), images are resized to 256×256. A 224×224 crop is randomly sampled from the images or its horizontal flip with the mean subtracted (Krizhevsky et al., 2012). We set the $M = 20$ and $k = 10$.

Table 4 shows classification results indicating that Stein VAE and Stein IVWAE outperform VAE in all the experiments, demonstrating the effectiveness of our approach for semi-supervised classification. When the proportion of labeled examples is too small ($< 10\%$), DGDN outperforms all the VAE-based models, which is not surprising provided that our models are deeper, thus have considerably more parameters than DGDN (Pu et al., 2016).

6. Conclusion

We have employed the Stein operator to develop a new method for designing the encoder in a variational autoencoder, and for learning an approximate distribution (samples) for the parameters of the decoder. The distributions for the codes and for the decoder parameters are represented non-parametrically in terms of samples, inferred by minimizing a KL distance and via use of a RKHS. Fast inference is manifested by learning a recognition model that mimics the manner in which the inferred code samples are manifested. The method is further generalized and improved by performing importance sampling. An extensive set of results, for unsupervised and semi-supervised learning, demonstrate excellent performance and scaling to large datasets.

References

Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. In *ICLR*, 2016.

Chwialkowski, K., Strathmann, H., and Gretton, A. A kernel test of goodness of fit. In *ICML*, 2016.

Gan, Z., Chen, C., Henao, R., Carlson, D., and Carin, L. Scalable deep poisson factor analysis for topic modeling. In *ICML*, 2015.

Gregor, K., Danihelka, I., Graves, A., and Wierstra, D. Draw: A recurrent neural network for image generation. In *ICML*, 2015.

Han, S., Liao, X., Dunson, D.B., and Carin, L. Variational gaussian copula inference. In *AISTATS*, 2016.

He, K., Zhang, X., Ren, S., and J, Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *ICLR*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X.i, Sutskever, I., and Welling, M. Improving variational inference with inverse autoregressive flow. In *NIPS*, 2016.

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

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Larochelle, H. and Laulyi, S. A neural autoregressive topic model. In *NIPS*, 2012.

Liu, Q. and Wang, D. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *NIPS*, 2016.

Liu, Q., Lee, J.D., and Jordan, M. A kernelized stein discrepancy for goodness-of-fit tests. In *ICML*, 2016.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013.

Miao, Y., Yu, L., and Blunsomi, Phil. Neural variational inference for text processing. In *ICML*, 2016.

Mnih, A. and Gregor, K. Neural variational inference and learning in belief networks. In *ICML*, 2014.

Mnih, A. and Rezende, D. J. Variational inference for monte carlo objectives. In *ICML*, 2016.

Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural network. In *ICML*, 2016.

Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., and Carin, L. Variational autoencoder for deep learning of images, labels and captions. In *NIPS*, 2016.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

Ranganath, R., Tang, L., Charlin, L., and M.Blei, D. Deep exponential families. In *AISTATS*, 2015.

Ranganath, R., Tran, D., and Blei, D. M. Hierarchical variational models. In *ICML*, 2016.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. Semi-supervised learning with ladder networks. In *NIPS*, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

Rezende, D.J. and Mohamed, S. Variational inference with normalizing flows. In *ICML*, 2015.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-fei, L. Imagenet large scale visual recognition challenge. *IJCV*, 2014.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. In *ICLR workshop*, 2015.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 2010.

Wang, D. and Liu, Q. Learning to samples: With application to amoritized mle for generalized adversarial learning. In *arXiv:1611.01722v2*, 2016.

Zhou, M., Hannah, L., Dunson, D., and Carin, L. Beta-negative binomial process and Poisson factor analysis. In *AISTATS*, 2012.

A Proof

Proof of Theorem 1 Recall the definition of KL divergence:

$$\mathrm{KL}(q_T\|p) = \mathrm{KL}(q_T(\boldsymbol{\theta})q(\mathcal{Z})||p(\boldsymbol{\theta},\mathcal{Z}|\mathcal{D})) = \int\int q_T(\boldsymbol{\theta})q(\mathcal{Z})\log\frac{p(\boldsymbol{\theta},\mathcal{Z}|\mathcal{D})}{q_T(\boldsymbol{\theta})q(\mathcal{Z})}d\boldsymbol{\theta}d\mathcal{Z} \tag{1}$$

$$= \int q_T(\boldsymbol{\theta})\Big\{\int q(\mathcal{Z})\log p(\boldsymbol{\theta},\mathcal{Z},\mathcal{D})d\mathcal{Z}\Big\}d\boldsymbol{\theta} - \int q_T(\boldsymbol{\theta})\log q_T(\boldsymbol{\theta})d\boldsymbol{\theta} - \int q(\mathcal{Z})\log q(\mathcal{Z})d\mathcal{Z} - \log p(\mathcal{D}) \tag{2}$$

$$= \int q_T(\boldsymbol{\theta})\log\tilde{p}(\boldsymbol{\theta};\mathcal{D})d\boldsymbol{\theta} - \int q_T(\boldsymbol{\theta})\log q_T(\boldsymbol{\theta})d\boldsymbol{\theta} - \int q(\mathcal{Z})\log q(\mathcal{Z})d\mathcal{Z} - \log p(\mathcal{D}) \tag{3}$$

$$= \mathrm{KL}\Big(q_T(\boldsymbol{\theta})||\tilde{p}(\boldsymbol{\theta};\mathcal{D})\Big) - \int q(\mathcal{Z})\log q(\mathcal{Z})d\mathcal{Z} - \log p(\mathcal{D})\,, \tag{4}$$

where $\log\tilde{p}(\boldsymbol{\theta};\mathcal{D}) = \int q(\mathcal{Z})\log p(\boldsymbol{\theta},\mathcal{Z},\mathcal{D})d\mathcal{Z}$. Since $\nabla_\epsilon \int q(\mathcal{Z})\log q(\mathcal{Z})d\mathcal{Z} = \nabla_{\epsilon_1}\log p(\mathcal{D}) = 0$, we have

$$\nabla_\epsilon \mathrm{KL}(q_T(\boldsymbol{\theta})q(\mathcal{Z})||p(\boldsymbol{\theta},\mathcal{Z}|\mathcal{D})) = \nabla_\epsilon \mathrm{KL}(q_T(\boldsymbol{\theta})||\tilde{p}(\boldsymbol{\theta};\mathcal{D}))\,. \tag{5}$$

We have

$$\nabla_\epsilon(\mathrm{KL}(q_T(\boldsymbol{\theta})q(\mathcal{Z})||p(\boldsymbol{\theta},\mathcal{Z}|\mathcal{D})|_{\epsilon_1=0} = -\mathbb{E}_{\boldsymbol{\theta}\sim q(\theta)}[\nabla_{\boldsymbol{\theta}}\log\tilde{p}(\boldsymbol{\theta};\mathcal{D})^T\psi(\boldsymbol{\theta};\mathcal{D}) + \mathrm{trace}(\nabla_{\boldsymbol{\theta}}\psi(\boldsymbol{\theta};\mathcal{D}))] \tag{6}$$

$$= -\mathbb{E}_{\boldsymbol{\theta}\sim q(\theta)}\Big[\mathrm{trace}\Big(\nabla_{\boldsymbol{\theta}}\log\tilde{p}(\boldsymbol{\theta};\mathcal{D})\psi(\boldsymbol{\theta};\mathcal{D})^T + \psi(\boldsymbol{\theta};\mathcal{D})\Big)\Big]\,. \tag{7}$$

Proof of Theorem 2 We have $\mathbb{E}_{I=\{i_1,\ldots,i_m\}}\Big[\frac{1}{m}\sum_{i=j}^m a_{i_j}\Big] = \frac{a_1+\cdots+a_k}{k}$, where $I \subset \{1,\ldots,k\}$ with $|I| = m < k$, is a uniformly distributed subset of $\{1,\ldots,k\}$. Using Jensen's inequality, we have

$$\mathrm{KL}^k_{q,p}(\boldsymbol{\Theta};\mathcal{D}) = -\mathbb{E}_{\boldsymbol{\Theta}^{1:k}\sim q(\boldsymbol{\Theta})}\Big[\log\frac{1}{k}\sum_{i=1}^k\frac{p(\boldsymbol{\Theta}^i|\mathcal{D})}{q(\boldsymbol{\Theta}^i)}\Big] \tag{8}$$

$$= -\mathbb{E}_{\boldsymbol{\Theta}^{1:k}\sim q(\boldsymbol{\Theta})}\Big[\log\mathbb{E}_{I=\{i_1,\ldots,i_m\}}\Big[\frac{1}{m}\sum_{i=1}^m\frac{p(\boldsymbol{\Theta}^i|\mathcal{D})}{q(\boldsymbol{\Theta}^i)}\Big]\Big] \tag{9}$$

$$\le -\mathbb{E}_{\boldsymbol{\Theta}^{1:k}\sim q(\boldsymbol{\Theta})}\Big[\mathbb{E}_{I=\{i_1,\ldots,i_m\}}\Big[\log\frac{1}{m}\sum_{i=1}^m\frac{p(\boldsymbol{\Theta}^i|\mathcal{D})}{q(\boldsymbol{\Theta}^i)}\Big]\Big] \tag{10}$$

$$= -\mathbb{E}_{\boldsymbol{\Theta}^{1:m}\sim q(\boldsymbol{\Theta})}\Big[\log\frac{1}{m}\sum_{i=1}^m\frac{p(\boldsymbol{\Theta}^i|\mathcal{D})}{q(\boldsymbol{\Theta}^i)}\Big] \tag{11}$$

$$= \mathrm{KL}^m_{q,p}(\boldsymbol{\Theta};\mathcal{D})\,, \tag{12}$$

if $q(\boldsymbol{\Theta})/p(\boldsymbol{\Theta}|\mathcal{D})$ is bounded, we have

$$\lim_{k\to\infty} \frac{1}{k}\sum_{i=1}^{k} \frac{p(\boldsymbol{\Theta}^i|\mathcal{D})}{q(\boldsymbol{\Theta}^i)} = \mathbb{E}_{q(\boldsymbol{\Theta})}\Big[\frac{p(\boldsymbol{\Theta}|\mathcal{D})}{q(\boldsymbol{\Theta})}\Big] = \int p(\boldsymbol{\Theta}|\mathcal{D})d\boldsymbol{\Theta} = 1\,. \tag{13}$$

Therefore

$$\mathrm{KL}^k_{q,p}(\boldsymbol{\Theta};\mathcal{D}) = -\lim_{k\to\infty}\mathbb{E}_{\boldsymbol{\Theta}^{1:k}\sim q(\boldsymbol{\Theta})}\Big[\log 1\Big] = 0\,.$$

Proof of Theorem 3 Assume $p_{[T^{-1}]}(\boldsymbol{\Theta})$ denote the density of $\hat{\boldsymbol{\Theta}} = T^{-1}(\boldsymbol{\Theta})$. We have

$$\begin{aligned}
\nabla_\epsilon\Big(\mathrm{KL}^k_{q,p}(\boldsymbol{\Theta}';\mathcal{D})\Big) &= -\nabla_\epsilon\left\{\mathbb{E}_{\boldsymbol{\Theta}^{1:k}\sim q(\boldsymbol{\Theta})}\Big[\log\frac{1}{k}\sum_{i=1}^{k}\frac{p_{[T^{-1}]}(\boldsymbol{\Theta}^i|\mathcal{D})}{q(\boldsymbol{\Theta}^i)}\Big]\right\} &(14)\\
&= -\mathbb{E}_{\boldsymbol{\Theta}^{1:k}\sim q(\boldsymbol{\Theta})}\left\{\nabla_\epsilon\Big[\log\frac{1}{k}\sum_{i=1}^{k}\frac{p_{[T^{-1}]}(\boldsymbol{\Theta}^i|\mathcal{D})}{q(\boldsymbol{\Theta}^i)}\Big]\right\} &(15)\\
&= -\mathbb{E}_{\boldsymbol{\Theta}^{1:k}\sim q(\boldsymbol{\Theta})}\left\{\Big[\frac{1}{k}\sum_{i=1}^{k}\frac{p_{[T^{-1}]}(\boldsymbol{\Theta}^i|\mathcal{D})}{q(\boldsymbol{\Theta}^i)}\Big]^{-1}\Big[\frac{1}{k}\sum_{i=1}^{k}\frac{\nabla_\epsilon p_{[T^{-1}]}(\boldsymbol{\Theta}^i|\mathcal{D})}{q(\boldsymbol{\Theta}^i)}\Big]\right\}. &(16)
\end{aligned}$$

Note that

$$\nabla_\epsilon p_{[T^{-1}]}(\boldsymbol{\Theta}^i|\mathcal{D}) = p_{[T^{-1}]}(\boldsymbol{\Theta}^i|\mathcal{D})\nabla_\epsilon\log p_{[T^{-1}]}(\boldsymbol{\Theta}^i|\mathcal{D})\,, \tag{17}$$

and when $\epsilon = 0$, we have

$$\begin{aligned}
p_{[T^{-1}]}(\boldsymbol{\Theta}^i|\mathcal{D}) &= p(\boldsymbol{\Theta}^i|\mathcal{D}), \quad \nabla_\epsilon T(\boldsymbol{\Theta}) = \psi(\boldsymbol{\Theta};\mathcal{D}), &(18)\\
\nabla_{\boldsymbol{\Theta}}T(\boldsymbol{\Theta}) &= \mathbf{I}, \nabla_\epsilon\nabla_{\boldsymbol{\Theta}}T(\boldsymbol{\Theta}) = \nabla_\epsilon\psi(\boldsymbol{\Theta};\mathcal{D}) &(19)\\
\nabla_\epsilon\log p_{[T^{-1}]}(\boldsymbol{\Theta}^i|\mathcal{D}) &= \nabla_\epsilon\log p(\boldsymbol{\Theta}^i|\mathcal{D})^T\nabla_\epsilon T(\boldsymbol{\Theta}^i) + \mathrm{trace}\Big(\big(\nabla_{\boldsymbol{\Theta}^i}T(\boldsymbol{\Theta}^i)\big)^{-1}\cdot\nabla_\epsilon\nabla_{\boldsymbol{\Theta}^i}T(\boldsymbol{\Theta}^i)\Big) &(20)\\
&= \nabla_\epsilon\log p(\boldsymbol{\Theta}^i|\mathcal{D})^T\psi(\boldsymbol{\Theta}^i;\mathcal{D}) + \mathrm{trace}\big(\nabla_\epsilon\psi(\boldsymbol{\Theta}^i;\mathcal{D})\big) &(21)\\
&= \mathrm{trace}\big(\nabla_\epsilon\log p(\boldsymbol{\Theta}^i|\mathcal{D})\psi(\boldsymbol{\Theta}^i;\mathcal{D})^T + \nabla_\epsilon\psi(\boldsymbol{\Theta}^i;\mathcal{D})\big) &(22)\\
&= \mathrm{trace}\big(\mathcal{A}_p(\boldsymbol{\Theta}^i;\mathcal{D})\big)\,. &(23)
\end{aligned}$$

Therefore, (16) can be rewritten as

$$\nabla_\epsilon\Big(\mathrm{KL}^k_{q,p}(\boldsymbol{\Theta}';\mathcal{D})\Big) = -\mathbb{E}_{\boldsymbol{\Theta}^{1:k}\sim q(\boldsymbol{\Theta})}\left\{\Big[\frac{1}{k}\sum_{i=1}^{k}\frac{p_{[T^{-1}]}(\boldsymbol{\Theta}^i|\mathcal{D})}{q(\boldsymbol{\Theta}^i)}\Big]^{-1}\Big[\frac{1}{k}\sum_{i=1}^{k}\frac{\nabla_\epsilon p_{[T^{-1}]}(\boldsymbol{\Theta}^i|\mathcal{D})}{q(\boldsymbol{\Theta}^i)}\Big]\right\} \tag{24}$$

$$= -\mathbb{E}_{\boldsymbol{\Theta}^{1:k}\sim q(\boldsymbol{\Theta})}\left\{\Big[\sum_{i=1}^{k}\frac{p(\boldsymbol{\Theta}^i|\mathcal{D})}{q(\boldsymbol{\Theta}^i)}\Big]^{-1}\Big[\sum_{i=1}^{k}\frac{p(\boldsymbol{\Theta}^i|\mathcal{D})}{q(\boldsymbol{\Theta}^i)}\nabla_\epsilon \log p_{[T^{-1}]}(\boldsymbol{\Theta}^i|\mathcal{D})\Big]\right\} \tag{25}$$

$$= -\mathbb{E}_{\boldsymbol{\Theta}^{1:k}\sim q(\boldsymbol{\Theta})}\left\{\frac{1}{\tilde{\omega}}\sum_{i=1}^{k}\omega_i\Big[\mathrm{trace}\big(\mathcal{A}_p(\boldsymbol{\Theta}^i;\mathcal{D})\big)\Big]\right\}, \tag{26}$$

where $\omega_k = p(\boldsymbol{\Theta}^i;\mathcal{D})/q(\boldsymbol{\Theta}^i)$ and $\tilde{\omega} = \sum_{i=1}^{k}\omega_i$.

B Samples Updating for Semi-supervised Learning

For updating samples of $\boldsymbol{\theta}$, we have

$$\boldsymbol{\theta}_j^{(t+1)} = \boldsymbol{\theta}_j^{(t)} + \epsilon_1\Delta\boldsymbol{\theta}_j^{(t)}, \tag{27}$$

with

$$\Delta\boldsymbol{\theta}_j^{(t)} \approx \frac{1}{M}\sum_{j'=1}^{M}[k_{\boldsymbol{\theta}}(\boldsymbol{\theta}_{j'}^{(t)},\boldsymbol{\theta}_j^{(t)})\nabla_{\boldsymbol{\theta}_{j'}^{(t)}}\log\tilde{p}(\boldsymbol{\theta}_{j'}^{(t)};\mathcal{D},\mathcal{D}_l)e + \nabla_{\boldsymbol{\theta}_{j'}^{(t)}}k_{\boldsymbol{\theta}}(\boldsymbol{\theta}_{j'}^{(t)},\boldsymbol{\theta}_j^{(t)}))] \tag{28}$$

$$\nabla_{\boldsymbol{\theta}}\log\tilde{p}(\boldsymbol{\theta};\mathcal{D},\mathcal{D}_l) \approx \frac{1}{M}\sum_{j=1}^{M}\Big\{\sum_{\boldsymbol{x}_n\in\mathcal{D}}\nabla_{\boldsymbol{\theta}}\log p(\boldsymbol{x}_n|\boldsymbol{z}_{jn},\boldsymbol{\theta}) + \sum_{\boldsymbol{x}_n\in\mathcal{D}_l}\nabla_{\boldsymbol{\theta}}\log p(\boldsymbol{x}_n|\boldsymbol{z}_{jn},\boldsymbol{\theta})\Big\}p(\boldsymbol{\theta}). \tag{29}$$

Similarly, when updating samples of $\tilde{\boldsymbol{\theta}}$, we have

$$\tilde{\boldsymbol{\theta}}_j^{(t+1)} = \tilde{\boldsymbol{\theta}}_j^{(t)} + \epsilon_2\Delta\tilde{\boldsymbol{\theta}}_j^{(t)}, \tag{30}$$

with

$$\Delta\tilde{\boldsymbol{\theta}}_j^{(t)} \approx \frac{1}{M}\sum_{j'=1}^{M}[k_{\tilde{\boldsymbol{\theta}}}(\tilde{\boldsymbol{\theta}}_{j'}^{(t)},\tilde{\boldsymbol{\theta}}_j^{(t)})\nabla_{\tilde{\boldsymbol{\theta}}_{j'}^{(t)}}\log\tilde{p}(\tilde{\boldsymbol{\theta}}_{j'}^{(t)};\mathcal{D}_l) + \nabla_{\tilde{\boldsymbol{\theta}}_{j'}^{(t)}}k_{\tilde{\boldsymbol{\theta}}}(\tilde{\boldsymbol{\theta}}_{j'}^{(t)},\tilde{\boldsymbol{\theta}}_j^{(t)}))]$$

$$\nabla_{\tilde{\boldsymbol{\theta}}}\log\tilde{p}(\tilde{\boldsymbol{\theta}};\mathcal{D}_l) \approx \frac{1}{M}\sum_{j=1}^{M}\sum_{\boldsymbol{y}_n\in\mathcal{D}_l}\nabla_{\tilde{\boldsymbol{\theta}}}\log p(\boldsymbol{y}_n|\boldsymbol{z}_{jn},\tilde{\boldsymbol{\theta}})p(\tilde{\boldsymbol{\theta}}). \tag{31}$$

C Posterior of Gaussian Mixture Model

Consider $\boldsymbol{z} \sim \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_1, \mathbf{I}) + \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_2, \mathbf{I})$ and $\boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{\theta}\boldsymbol{z}, \sigma^2\mathbf{I})$, where $\boldsymbol{z} \in \mathbb{R}^K$, $\boldsymbol{x} \in R^P$ and $\boldsymbol{\theta} \in \mathbb{R}^{P\times K}$. We have

$$\begin{aligned}
p(\boldsymbol{z}|\boldsymbol{x}) \propto p(\boldsymbol{x})p(\boldsymbol{z}) &\propto \exp\left\{-\frac{(\boldsymbol{x}-\boldsymbol{\theta}\boldsymbol{z})^T(\boldsymbol{x}-\boldsymbol{\theta}\boldsymbol{z})}{2\sigma^2}\right\} \\
&\times \left\{\exp\left\{-\frac{(\boldsymbol{z}-\boldsymbol{\mu}_1)^T(\boldsymbol{z}-\boldsymbol{\mu}_1)}{2}\right\} + \exp\left\{-\frac{(\boldsymbol{z}-\boldsymbol{\mu}_2)^T(\boldsymbol{z}-\boldsymbol{\mu}_2)}{2}\right\}\right\} \\
&= \exp\left\{-\frac{1}{2}\Big[\boldsymbol{z}^T\big(\frac{\boldsymbol{\theta}^T\boldsymbol{\theta}}{\sigma^2}+\mathbf{I}\big)\boldsymbol{z} - 2\big(\frac{\boldsymbol{y}^T\boldsymbol{\theta}}{\sigma^2}+\boldsymbol{\mu}_1\big)\boldsymbol{z} + \frac{\boldsymbol{x}^T\boldsymbol{x}}{\sigma^2} + \boldsymbol{\mu}_1^T\boldsymbol{\mu}_1\Big]\right\} \\
&+ \exp\left\{-\frac{1}{2}\Big[\boldsymbol{z}^T\big(\frac{\boldsymbol{\theta}^T\boldsymbol{\theta}}{\sigma^2}+\mathbf{I}\big)\boldsymbol{z} - 2\big(\frac{\boldsymbol{y}^T\boldsymbol{\theta}}{\sigma^2}+\boldsymbol{\mu}_2\big)\boldsymbol{z} + \frac{\boldsymbol{x}^T\boldsymbol{x}}{\sigma^2} + \boldsymbol{\mu}_2^T\boldsymbol{\mu}_2\Big]\right\}.
\end{aligned} \tag{32}$$

Let

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{\theta}^T\boldsymbol{\theta}}{\sigma^2} + \mathbf{I}, \quad \hat{\boldsymbol{\mu}}_1 = \boldsymbol{\Sigma}^{-1}\big(\frac{\boldsymbol{y}^T\boldsymbol{\theta}}{\sigma^2} - \boldsymbol{\mu}_1\big), \qquad p_1 = \frac{\boldsymbol{x}^T\boldsymbol{x}}{\sigma^2} + \boldsymbol{\mu}_1^T\boldsymbol{\mu}_1 - \hat{\boldsymbol{\mu}}_1^T\boldsymbol{\Sigma}\hat{\boldsymbol{\mu}}_1, \tag{33}$$

$$\hat{\boldsymbol{\mu}}_2 = \boldsymbol{\Sigma}^{-1}\big(\frac{\boldsymbol{y}^T\boldsymbol{\theta}}{\sigma^2} - \boldsymbol{\mu}_2\big), \qquad p_2 = \frac{\boldsymbol{x}^T\boldsymbol{x}}{\sigma^2} + \boldsymbol{\mu}_2^T\boldsymbol{\mu}_2 - \hat{\boldsymbol{\mu}}_2^T\boldsymbol{\Sigma}\hat{\boldsymbol{\mu}}_2\,, \tag{34}$$

The density in (32) can be rewritten as

$$p(\boldsymbol{z}|\boldsymbol{x}) \propto \exp\{p_1\}\exp\left\{-\frac{1}{2}(\boldsymbol{z}-\hat{\boldsymbol{\mu}}_1)^T\boldsymbol{\Sigma}(\boldsymbol{z}-\hat{\boldsymbol{\mu}}_1)\right\} + \exp\{p_2\}\exp\left\{-\frac{1}{2}(\boldsymbol{z}-\hat{\boldsymbol{\mu}}_2)^T\boldsymbol{\Sigma}(\boldsymbol{z}-\hat{\boldsymbol{\mu}}_2)\right\}. \tag{35}$$

Therefore, we have $\boldsymbol{z}|\boldsymbol{x} \sim p(\boldsymbol{z}|\boldsymbol{x}) = \hat{p}\mathcal{N}(\hat{\boldsymbol{\mu}}_1, \boldsymbol{\Sigma}) + (1-\hat{p})\mathcal{N}(\hat{\boldsymbol{\mu}}_2, \boldsymbol{\Sigma})$, where

$$\hat{p} = \frac{1}{1+\exp(p_2 - p_1)}\,. \tag{36}$$

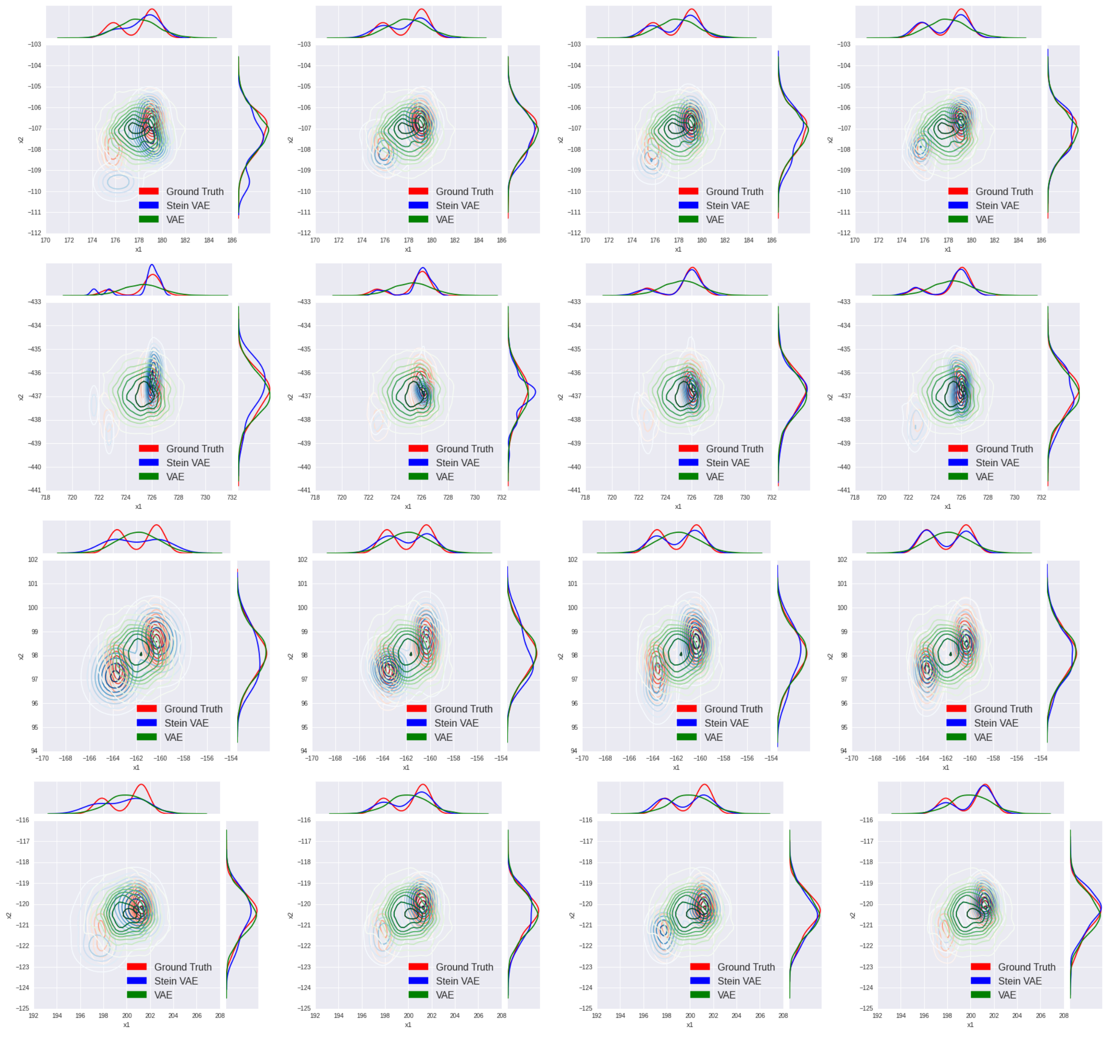
D Model Architecture

Table 1: Architecture of the models for semi-supervised classification on ImageNet. BN denotes batch normalization. The layer in bracket indicates the number of layers stacked.

Output Size	Encoder	Decoder
$224 \times 224 \times 4$ for encoder $224 \times 224 \times 3$ for decoder	RGB image $\boldsymbol{x}_n$ stacked by $\boldsymbol{\xi}$	RGB image $\boldsymbol{x}_n$
$56 \times 56 \times 64$	7×7 conv, 64 kernels, LeakyRelu, stride 4, BN	
	$\left[3 \times 3 \text{ conv, 64 kernels, LeakyRelu, stride 1, BN} \right] \times 3$	
$28 \times 28 \times 128$	3×3 conv, 128 kernels, LeakyRelu, stride 2, BN	
	$\left[3 \times 3 \text{ conv, 128 kernels, LeakyRelu, stride 1, BN} \right] \times 3$	
$14 \times 14 \times 256$	3×3 conv, 256 kernels, LeakyRelu, stride 2, BN	
	$\left[3 \times 3 \text{ conv, 256 kernels, LeakyRelu, stride 1, BN} \right] \times 3$	
$7 \times 7 \times 512$	3×3 conv, 512 kernels, LeakyRelu, stride 2, BN	
	$\left[3 \times 3 \text{ conv, 512 kernels, LeakyRelu, stride 1, BN} \right] \times 3$	
latent code $\boldsymbol{z}_n$		
1×1 conv, 2048 kernels, LeakyRelu		
average pooling, 1000-dimentional fully connected layer		
softmax, label $\boldsymbol{y}_n$		

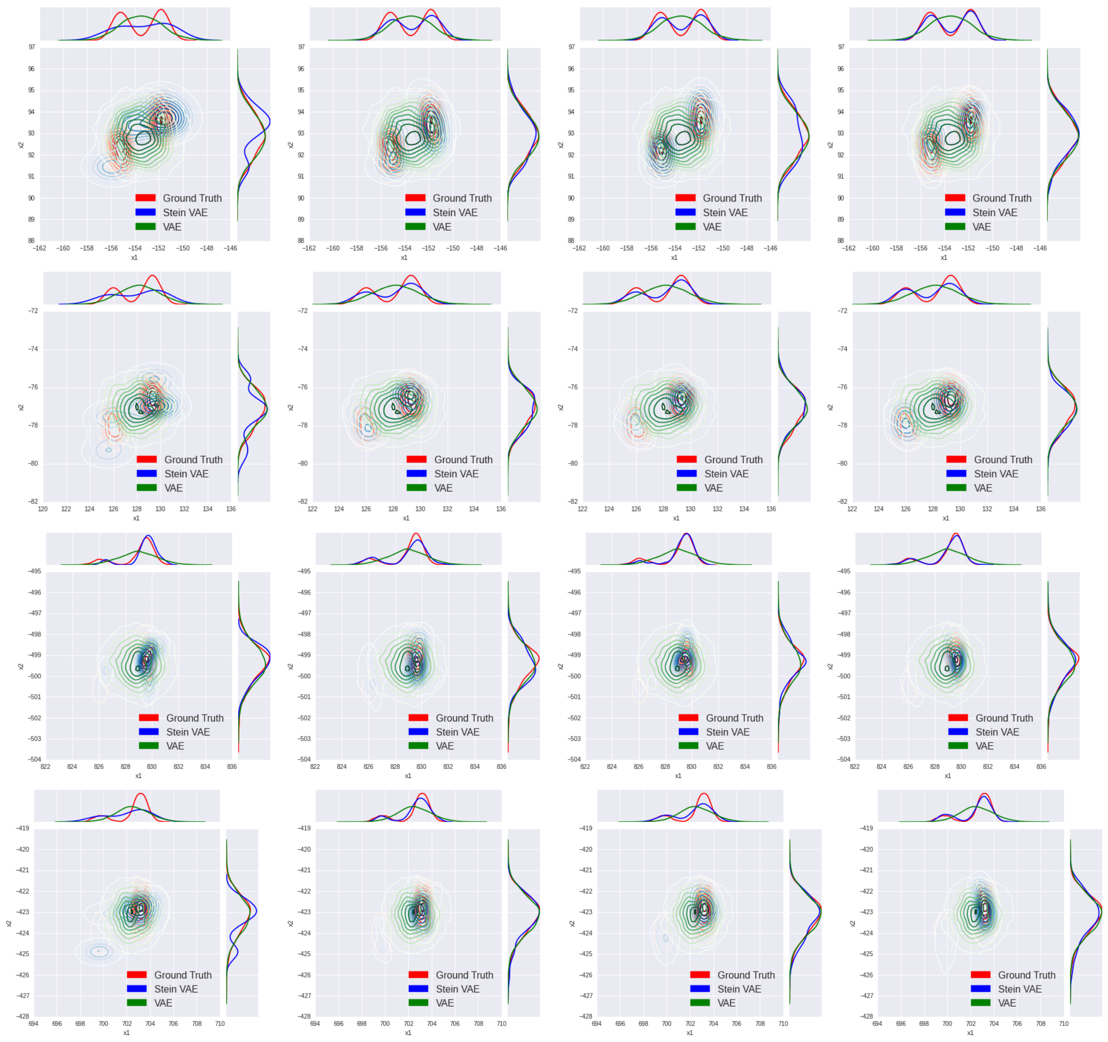
E Additional Results

Gaussian Mixture Model Figure 1 and 2 show the performance of Stein VAE approximations for the true posterior using $M = 10$, $M = 20$, $M = 50$ and $M = 100$ samples on test data.



(a) $M = 10$ (b) $M = 20$ (c) $M = 50$ (d) $M = 100$

Figure 1: Approximation of posterior distribution: Stein VAE *vs.* VAE. The figures represent different samples of Stein VAE. Each row corresponds to the same test data, and each column corresponds to the same number of samples with (a) 10 samples; (b) 20 samples; (c) 50 samples; (d) 100 samples.



(a) $M = 10$ (b) $M = 20$ (c) $M = 50$ (d) $M = 100$

Figure 2: Approximation of posterior distribution: Stein VAE *vs.* VAE. The figures represent different samples of Stein VAE. Each row corresponds to the same test data, and each column corresponds to the same number of samples with (a) 10 samples; (b) 20 samples; (c) 50 samples; (d) 100 samples.

Poisson Factor Analysis We show the marginal and pairwise posteriors of test data in Figure 3.

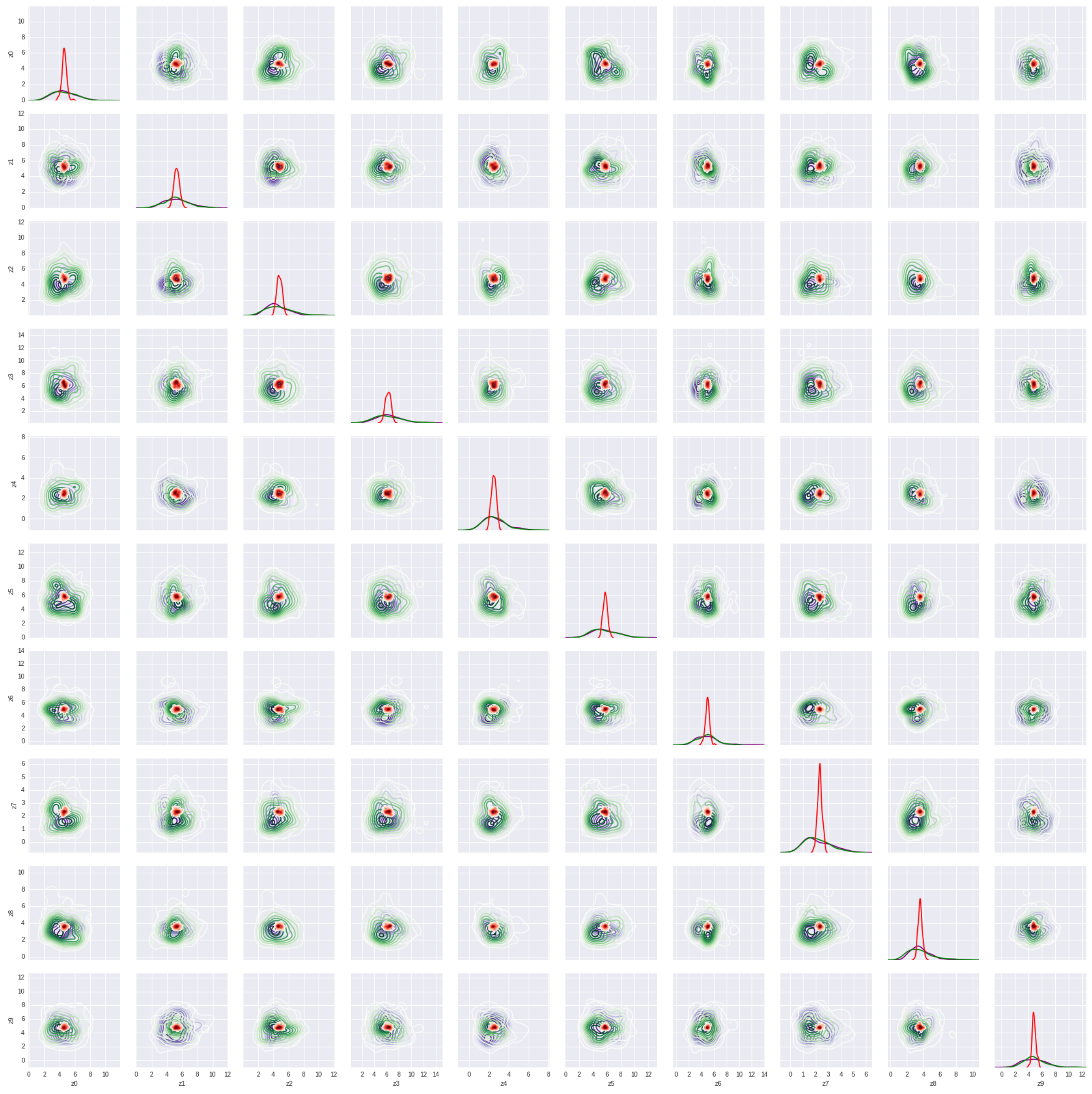


Figure 3: Univariate marginals and pairwise posteriors. Purple, red and green represent the distribution inferred from MCMC, standard VAE and Stein VAE, respectively.