An Exploration of Bayesian Methods for Auto-Encoders

Ruqi Zhang[*1], **Wesley J. Maddox**[*2], **Ben Athiwaratkun**[3], **Andrew Gordon Wilson**[2]
[1]Cornell University, [2]New York University, [3]Amazon

Abstract

We propose two new methods for fully Bayesian inference in auto-encoder models. These methods are developed as generalizations to the standard variational auto-encoder by viewing the VAE in its variational expectation maximization (EM) formulation. We then use a related Gibbs sampler to alternate sampling of the parameters of the decoder network and the latent variables themselves, producing the Bayesian auto-encoder (BAE). We then use stochastic gradient Hamiltonian Monte Carlo (SGHMC) to perform approximate inference over the posterior of this model. For both unsupervised and semi-supervised learning, we demonstrate the efficacy of these approaches on text and image modeling, achieving strong results without standard interventions such as KL cost annealing.

1 Introduction

Deep generative modelling attempts to model the entire data generating process, producing simulated data, while Bayesian methods allow the incorporation of prior beliefs about the generation of a given dataset into the inferential procedure. We propose to merge the two methods by introducing the Bayesian auto-encoder (BAE).

The variational autoencoder (VAE) [Rezende et al., 2014, Kingma and Welling, 2014] and the associated re-parameterization trick helped to bring new interest into the field of deep generative modeling. VAEs infer a variational distribution over the latent space of the model while optimizing the mapping of the latent space to the data using two separate neural networks: one to reconstruct the data and one to infer the latent distribution. VAEs thus incorporate observation-based uncertainty (i.e. modelling noise in the data), but not model-based uncertainty (i.e. modelling uncertainty in the class of functions used to model the data). However, modelling both types of uncertainty should improve model calibration, providing more reliable and interpretable confidence estimates. For example, Kendall and Gal [2017] show that modeling both observation-based and model-based uncertainty in deep Bayesian models is needed to achieve state-of-the-art results in multiple tasks such as image segmentation and depth regression.

Moreover, Bayesian models help automatically calibrate between model complexity and data misfitting [MacKay, 1992, MacKay and Mac Kay, 2003], allowing for a wider range of functions and generative processes modelled. In this setting, we hypothesize that each mode in the posterior of the decoder should correspond to a different generative process, providing a rich family of decoders that can potentially capture many generation styles. Saatchi and Wilson [2017] demonstrated this property for Bayesian inference in generative adversarial networks (GANs).

Bayesian inference on the parameters of a VAE is a compelling undertaking: since VAEs are so flexible, their likelihood functions are diffuse and multi-modal, which should yield different solutions when one marginalizes a posterior, instead of representing the posterior with a point mass as when optimizing the model parameters. Bayesian marginalization provides much needed robustness to over-fitting and prevents the variance of the latent variables from being pushed to zero, a key drawback of the standard VAE loss function [Dai et al., 2017]. KL cost annealing [Bowman et al., 2016] is typically used as a remedy, as it reduces over-fitting to the prior by encouraging dependence on the latent variables, but it has no obvious motivation.

In this paper, we propose two methods for fully Bayesian auto-encoders (BAEs) by viewing the latent

[*] Equal contribution.

variable estimation process as a Gibbs sampling mechanism; this connects VAEs to the proposed models when VAEs are viewed as variational expectation maximization (EM). BAEs thus introduce Bayesian deep generative modelling, with uncertainty accounted for in both model and latent spaces. In contrast to the standard VAE framework, variational assumptions about the posterior of the latent variables are removed, unlike in the method briefly mentioned (but not implemented) in Appendix F of Kingma and Welling [2014], which still uses a variational approximation to the latent posterior. Finally, we provide algorithms that perform scalable posterior inference on BAEs using stochastic gradient MCMC methods.

Our models often outperform their non-Bayesian counterparts on a wide variety of tasks, ranging from language modeling, density estimation, dimensionality reduction, and semi-supervised classification. PyTorch code for all models is available at `https://github.com/ruqizhang/bayes-autoencoder`.

2 Models

In this section, we propose Bayesian treatments of auto-encoders. We give a brief background on variational auto-encoder (VAE) in Section 2.1 and define our Bayesian VAE (BVAE) that incorporates additional uncertainty information on the model weights in Section 2.2. In Section 2.3, we introduce a model called the Bayesian Auto-encoder (BAE), a Bayesian approach for approximate inference that allows arbitrary posterior distributions on latent codes.

2.1 Background: Variational Auto-encoders

Variational auto-encoders (VAEs) Kingma and Welling [2014] combine a generative model (decoder) that operates on the data with an encoding model (encoder) over the posterior distribution of a latent variable. The generative model is given by the following process:

$$z^{(i)} \sim p(z^{(i)}) = \mathcal{N}(0, I); \quad x^{(i)} \sim p(x^{(i)}|f(z^{(i)};\theta)),$$

where x_i is a single data point, assumed to be part of the complete dataset, $\mathbf{X}$, $z^{(i)}$ is a single latent variable, with the entire set of latent variables, $\mathbf{Z}$ and θ are global model parameters. The encoder model is traditionally assumed to be a factorized Gaussian distribution, that is $q_\phi(\mathbf{Z}|\mathbf{X}) = \mathcal{N}(z|h_\mu(\mathbf{X},;\phi), h_\sigma(\mathbf{X};\phi))$. It is typically assumed that f and h are multi-layer perceptrons (MLPs) with outputs for each parameter in the distribution and with parameters θ and ϕ respectively. Training proceeds by optimizing θ and ϕ via minimizing the evidence lower bound (ELBO) [Kingma and Welling, 2014, Blei et al., 2017], given by

$$L(\theta, \phi; \mathbf{X}) = E_{q_\phi(\mathbf{Z}|\mathbf{X})}(\log \frac{p_\theta(\mathbf{X}, z)}{q_\phi(z|\mathbf{X})}) = \quad (1)$$
$$- KL(q_\phi(\mathbf{Z}|\mathbf{X})\|p_\theta(\mathbf{Z})) + E_{q_\phi(\mathbf{Z}|\mathbf{X})}[\log p_\theta(\mathbf{X}|\mathbf{Z})]$$

Other bounds on the log model evidence do exist; however, the ELBO bound has an interpretation as maximum likelihood estimation (MLE), so that VAE training corresponds to maximum likelihood estimation. One important bound to note is the importance weighted bound of Burda et al. [2016] that uses several samples from the variational distribution, q, when approximating the log-likelihood. The updated ELBO then becomes

$$L(\theta, \phi; \mathbf{X}) = E_{q_\phi(\mathbf{Z}|\mathbf{X})} \log \left(\frac{1}{K} \sum_{i=1}^{K} \frac{p_\theta(\mathbf{X}, z)}{q_\phi(z|\mathbf{X})} \right), \quad (2)$$

where K is the number of samples used in the approximation. By convexity, this bound has been shown to be a better bound on the model evidence than the VAE bound.

Assuming that some sort of regularization exists on θ (i.e. standard weight decay), then the update steps for these parameters consist of iteratively finding

$$\theta^* = \arg\max_\theta E_{q_\phi(z|\mathbf{X})} \log \left(p(\mathbf{X}|h(z;\theta))p(\theta) \right). \quad (3)$$

Eq. (3) is the standard M(aximization) step when viewing the VAE update as a variational EM algorithm, yielding a maximum likelihood (MLE) estimate of θ. That is, no uncertainty in the model parameters is modelled in this update [Turner and Sahani, 2011]. Similarly, the E(xpectation) step is performed by computing the parameters of the approximation distribution, $q_\phi(\mathbf{Z}|\mathbf{X})$, which is done by using the KL term in Equation 1. These two updates can be combined into a single loss function, Equation 1, which is used in Kingma and Welling [2014].

2.2 Bayesian VAEs

We place distributions over the decoder parameters θ and the encoder parameters ϕ and derive Bayesian VAEs (BVAEs) through Bayes' rule. With a prior $p(\theta, \phi)$, the joint posterior distribution over θ and ϕ is given by

$$p(\theta, \phi|\mathbf{X}) \propto p(\mathbf{X}|\theta, \phi)p(\theta, \phi). \quad (4)$$

Taking the logarithm of this posterior distribution and applying the ELBO gives

$$\log p(\theta, \phi|\mathbf{X}) = \log p(\mathbf{X}|\theta, \phi) + \log p(\theta, \phi) + \text{const.}$$
$$\propto KL(q_\phi(\mathbf{Z}|\mathbf{X})\|p_\theta(\mathbf{Z}|\mathbf{X})) + L(\theta, \phi; \mathbf{X}) + \log p(\theta, \phi)$$
$$\geq L(\theta, \phi; \mathbf{X}) + \log p(\theta, \phi). \quad (5)$$

Thus, the loss reduces to the standard ELBO plus the priors on the encoder and decoder parameters.

We perform approximate inference over the posteriors of θ and ϕ using an MCMC method which allows for arbitrary distributions and mini-batches. In particular, we use stochastic gradient Hamiltonian Monte Carlo (SGHMC) [Chen et al., 2014, Neal, 2011] due to its scalability to mini-batches, potential to explore multiple modes, and its connection to stochastic gradient descent with momentum. These enable easier training and tuning of the BVAE.

Algorithm 1 Bayesian VAE.

Initialize (θ, ϕ).
while not converged to posterior $p(\theta, \phi|\mathbf{Z}, X)$ **do**
MCMC sampling of θ and ϕ using Equation 5.

We summarize the inference procedure for θ and ϕ in Algorithm 1. Note that we use a variant of SGHMC with 1 leapfrog step described in Algorithm 2 as an MCMC step in Alg. 1. Alg. 2 can be used as an MCMC sampling step in many of our models as well.

Algorithm 2 Stochastic Gradient Hamiltonian Monte Carlo [Chen et al., 2014] with L = 1, α: friction, and η : stepsize.

while not converged to distribution $\pi(x)$ **do**
If re-sampled, $r_t \sim \mathcal{N}(0, I)$
$r_t = (1 - \alpha)r_{t-1} + \eta\nabla L(x) + \mathcal{N}(0, 2\eta\alpha I)$
$x_t = x_{t-1} + r_t$
$\pi(x)$ is a log-probability of x in which we desire to sample from.

2.3 Bayesian Auto-Encoders (BAEs)

Instead of using a variational approximate latent posterior distribution, we propose to approximately sample from this distribution using stochastic gradient MCMC, producing the Bayesian auto-encoder (BAE). This method does *not* rely on variational assumptions about the posterior of the latent variables, and is fully Bayesian in both inference of the latent space and model parameters. For simplicity of exposition in this section, we assume i.i.d data, but the methods will only slightly change for dependent data (i.e for recurrent neural networks (RNNs)).

We assume the response, $\mathbf{X}$, is defined and assumed to be conditional on latent (or noisy measurement) variables, $\mathbf{Z}$, and model parameters θ. Conditional on the global variables, θ, the joint likelihood of the response and latent variables can be written as

$$p(\mathbf{X}, \mathbf{Z}|\theta) = p(\mathbf{X}|\mathbf{Z}, \theta)p(\mathbf{Z}|\theta).$$

Including inference over θ gives the following joint distribution[1]:

$$p(\mathbf{X}, \mathbf{Z}, \theta) = \pi(\theta)\prod_{i=1}^{N} p(x_i|z_i, \theta)p(z_i)\,. \tag{6}$$

Conditioning on θ, the posterior distribution over z_i is given by

$$p(z_i|x_i, \theta) = \frac{p(x_i, |z_i, \theta)p(z_i)}{\int p(x_i, z|\theta)dz}, \tag{7}$$

while conditioning on $\mathbf{Z}$ gives a posterior over the model parameters θ,

$$p(\theta|\mathbf{X}, \mathbf{Z}) = \frac{p(\mathbf{X}|\mathbf{Z}, \theta)p(\theta)}{\int p(\mathbf{X}, \mathbf{Z}|\theta)p(\theta)d\theta}. \tag{8}$$

Up to a proportionality constant, Equation 8 is the same as the term inside the integral in Equation 3; however, now the conditioning is over the true posterior on $\mathbf{Z}$, rather than a variational approximation.

The log joint likelihood of both the latent space, $\mathbf{Z}$ and the parameters, θ, can be estimated using a mini-batch via re-weighting[2]:

$$\log p(\theta, \mathbf{Z}|\mathbf{X}) \approx \frac{N}{M}\left(\sum_{i=1}^{M} \log p(x^{(i)}|\theta, z^{(i)}) + \log p(z^{(i)})\right) + \log p(\theta). \tag{9}$$

We seek to estimate and sample from this posterior distribution. Under mild assumptions that the likelihood and prior in Equation 9 are differentiable with respect to θ and $z^{(i)}$, we will directly apply gradient-based sampling algorithms to efficiently sample from the posterior. After the burn-in phase, we will obtain a set of θ's and $z^{(i)}$'s that reflect the true posterior distribution. The choice of likelihood is arbitrary, and so for the experiments in this paper on binary data, we will assume that $p(x^{(i)}|z^{(i)}, \theta) = \text{Bernoulli}(h(z^{(i)}; \theta))$, where $h(.; \theta)$ is a multi-layer perceptron (MLP) with input $z^{(i)}$ and parameters θ; this model satisfies our differentiability assumptions.

Based on Equations 6 and 9, we now describe three algorithms that attempt to learn this posterior distribution. We had originally proposed an encoder-free Bayesian auto-encoder that learns both θ and $\mathbf{Z}$ jointly at every step, with another Markov chain offloading inference of new $\mathbf{Z}$ to test time; however, this model

[1] Derivatives of this distribution are given in Appendix **??**. We also assume i.i.d data for ease of exposition, but the model can be easily extended to dependent data.

[2] Note that like the VAE, we do not assume any observed data, $\mathbf{Y}$; however we could include dependence on observed data easily.

was too flexible to train properly, and so we were forced to include an encoder network to intialize the Markov chain for each batch. The second and third algorithms introduce an efficient encoder model to approximate each posterior sample $\mathbf{Z}$; in doing so, the second algorithm updates θ and $\mathbf{Z}$ jointly (collapsing the Gibbs updates into a single update like in the VAE), while the third algorithm alternates updates on θ and $\mathbf{Z}$ (directly applying the Gibbs framework of the previous sections).

2.3.1 Gibbs-Like BAE (BAE-G)

We propose alternating updates of $\mathbf{Z}$ using Equation 7 and θ using Equation 8 on each mini-batch of data. This produces a mini-batch Gibbs sampler, and is summarized in Algorithm 3 in full generality. We take $J + \#$burn-in steps on the same mini-batch to create an approximation to the posterior on that minibatch. The complexity is $\#\text{burnin}+J$ forwards passes through the network, a slightly increased computational cost, where J is the maximum number of forwards passes on a mini-batch. When using SGHMC, there is minimal computational overhead over standard stochastic gradient descent [Chen et al., 2014].

Due to the mini-batched Gibbs updates, we may no longer have a guarantee of reaching the true posterior distribution. However, this approach has been used successfully in the literature before: for example, the alternating generator and discriminator updates of Saatchi and Wilson [2017] and the alternating $\mathbf{Z}$ and θ optimization in Pu et al. [2017], and direct mini-batch updates for latent dirichlet allocation Griffiths and Steyvers [2004]. Other possible methods for global and local updating of parameters are explored in Neal [2011], where a method for performing MCMC on global and local variables is described, and Smolyakov et al. [2018], which performs adaptive Gibbs sampling on selected mini-batches.

To initialize the latent variables at each step, we propose using another network to estimate the latent code distribution. That is, instead of storing every $z^{(i)}$, we use an encoder model f_φ which maps $x^{(i)}$ to the posterior distribution of latent code $z^{(i)}$. This is distinct from the encoder in VAE since no assumptions on the posterior are made, and we just want a good initialization for our Markov chain. The outputs from this model can be used to initialize the latent variable in Equation 6, allowing inference to be performed over both this latent variable and the decoder parameters. For real-valued latent spaces, we train this encoder model with the following loss function:

$$\min_{\varphi} \sum_{i=1}^{M} \|z_j^{(i)} - f_\varphi(x^{(i)})\|_2, \tag{10}$$

Algorithm 3 BAE-G.

Initialize θ and φ
for each mini-batch of data, $\mathbf{X}^{(i)}$ **do**
  Initialize $\mathbf{Z}_0^{(i)} = f_\varphi(\mathbf{X}^{(i)})$.
  while not converged to $p(\theta, \mathbf{Z}^{(i)}|\mathbf{X}^{(i)})$ **do**
    MCMC update of $\mathbf{Z}_j^{(i)}$ with Eq. 7.
    MCMC update of θ using Eq. 8.
    if j > #burn-in **then**
      Update the encoder φ using Eq. 10 and $f_\varphi(\mathbf{X}^{(i)})$ and $\mathbf{Z}_j^{(i)}$.

Algorithm 4 BAE-S.

Initialize θ and φ
for each mini-batch of data, $\mathbf{X}^{(i)}$ **do**
  Initialize $\mathbf{Z}_0^{(i)} = f_\varphi(\mathbf{X}^{(i)})$.
  while not converged to $p(\theta, \mathbf{Z}^{(i)}|\mathbf{X}^{(i)})$ **do**
    MCMC update of $\mathbf{Z}_j^{(i)}$ and θ with Eq. 9.
    if j > #burn-in **then**
      Update the encoder, φ, using Eq. 10 with $f_\varphi(\mathbf{X}^{(i)})$ and $\mathbf{Z}_j^{(i)}$.

where $z_j^{(i)}$ is the jth posterior sample of the latent variable $z^{(i)}$. Here, we choose the 2-norm as it is closely tied to the likelihood of a fixed variance normal distribution, which is the prior on the latent variable on the model. This is akin to learning the mean of a normal distribution. Now, for every data point, we can produce $z^{(i)'} = f_\phi(x^{(i)})$ and use this as the starting point to sample from the stationary distribution $p(\theta, \mathbf{Z}|\mathbf{X})$ using Equation 6. This vastly reduces the number of forwards passes through the encoder network necessary to reach a reasonable state.

2.3.2 Simultaneous Update BAE (BAE-S)

To more closely align this model with the standard VAE model, we can instead collapse the two Gibbs steps into a single update, producing the simultaneous update Bayesian auto-encoder.

After samples $z^{(i)}$ have been drawn from the posterior distribution, these can be used to update the parameters of the encoder Equation 10. New data points are then evaluated using the encoder model. This method is described in Alg. 4. Note that this algorithm requires $J+\#$burnin forwards passes for each mini-batch, where J is the maximum number of forwards passes on a mini-batch, the same additional computational cost as before.

3 Semi-Supervised Learning

In this section, we describe our methodology to use the proposed Bayesian models for semi-supervised learning. We develop a semi-supervised algorithm for BVAE based on the M2-VAE model by Kingma et al. [2014]. We also develop a semi-supervised learning method for Bayesian auto-encoders in Section 3.2.

3.1 Semi-Supervised BVAE with M2

We will first describe the M2 model [Kingma et al., 2014] for VAEs. The M2 model is based on the process that the data $\mathbf{X}$ is generated by a class variable y and a latent code $\mathbf{Z}$ through a conditional $p_\theta(\mathbf{X}|y, \mathbf{Z})$. This is different from the standard VAE where $\mathbf{X}$ is generated only from $\mathbf{Z}$. The class variable y can then be inferred through $p(y|\mathbf{X})$. In order to estimate $p(y|\mathbf{X})$, M2-VAE additionally uses an approximate posterior $q(y|\mathbf{X}) = \text{Cat}(y|\pi_\phi(\mathbf{X}))$ which serves as a classifier. The evidence lower bound of this model can be written as:

$$\log p(\mathbf{X}, y) \geq E_{q(\mathbf{Z}|\mathbf{X})}\left(\log \frac{p(\mathbf{X}|y, z)p(z)}{q(z|\mathbf{X}, y)}\right) + \log p(y) \equiv L(\mathbf{X}, y) \quad (11)$$

When the y is known, this term can be directly used in the loss function and combined with the entropy of the classifier, giving[3]

$$J_l = L(\mathbf{X_l}, y_l) - \alpha E_{p(\mathbf{X_l}, y_l)}(\log q(y_l|\mathbf{X_l}), \quad (12)$$

where $\mathbf{X}_l$ denotes labeled data. The unsupervised data are trained via the marginalized loss over the label y.

$$J_u = \sum_y q_\phi(y|\mathbf{X_u})(L(\mathbf{X_u}, y) - \log q(y|\mathbf{X_u})), \quad (13)$$

where $\mathbf{X}_u$ denotes unlabeled data. Typically, both losses are optimized at once by combining mini-batches of both the labeled and unlabeled data, giving $J = J_l + J_u$.

To incorporate the BVAE into this framework, we can derive the ELBO on $\log p(\theta, \phi, X, y)$:

$$\log p(\theta, \phi, \mathbf{X}, y) \geq L(\theta, \phi, \mathbf{X}, y) \equiv L(\mathbf{X}, y) + \log p(\theta, \phi). \quad (14)$$

where $L(\mathbf{X}, y)$ is defined in Eq. 11. We use $L(\theta, \phi, \mathbf{X}, y)$ to replace $L(\mathbf{X}, y)$ in Eq. 12 and 13, yielding a loss $J(\theta, \phi) = J_l(\theta, \phi) + J_u(\theta, \phi)$ which we then use to perform posterior sampling.

[3]Note: $\alpha = 0.1 * N_l$, a hyperparameter estimated from cross-validation in Kingma et al. [2014].

3.2 BAEs for Semi-Supervised Learning

In the Bayesian auto-encoder models, we no longer have a likelihood for the decoder distribution, necessitating a change in the framework of semi-supervised learning. Similar to Pu et al. [2017], it is possible to train a classifier on the latent representations from the generative model. This is in fact a similar method to a method M1/M2-VAE of Kingma et al. [2014], where a second VAE is trained on the latent variables.

However, in our setting, this approach ignores both the decoder layer and the data itself. Instead, we assume that the class label, $y^{(i)}$, is dependent on both the features, $x^{(i)}$, and on the latent variable, $z^{(i)}$. We then train an MLP with parameters ϑ, estimating $q(y^{(i)}|x^{(i)}, z^{(i)}; \vartheta)$. For supervised data, this approach can be summarized for the simultaneous-update algorithm in Algorithm 5, and can be similarly applied to the Gibbs-like updating algorithm. This method does not require marginalizing $y^{(i)}$, and is more computationally efficient compared to BVAE. However, the drawback compared to BVAE is that the model is not trained on unlabeled data.

Algorithm 5 Simultaneous updates algorithm for semi-supervised training.

```
Initialize θ and φ
for each mini-batch of data, X^(i), Y^(i). do
  Initialize Z^(i) = f_φ(X_i).
  while not converged to p(θ, Z^(i)|X^(i)) do
    MCMC sampling of Z^(i) using Equation 9.
    if j > #burn-in then
      Update the encoder parameters φ by optimizing Eq. 10 using Z_j^(i) and f_φ(X_i).
      Update the classifier parameters ϑ.
```

4 Related Work

VAEs were originally introduced with a Gaussian distribution as the posterior approximation [Kingma and Welling, 2014] to utilize the re-parameterization trick, while the ELBO (Equation 1) was used to optimize the model parameters. Both frameworks for different bounds in variational methods and more expressive posterior approximations do exist. For example, Pu et al. [2017] and Feng et al. [2017] propose using Stein variational gradient descent, which places the latent variables in a RKHS, and allows for the possibility that the model parameters are themselves random variables. However, the model parameters are learned via a generalization of the re-parameterization trick in practice. Bowman et al. [2016] instead propose using KL cost annealing or deterministic warm-up to burn-in the KL divergence regularization, in an attempt to

learn a more expressive latent representation. Higgins et al. [2016] moves beyond this procedure, proposing a modification to standard KL divergence loss minimization to encourage sample diversity. Alternatively, the auxiliary deep generative model [Maaløe et al., 2016] introduces a second hidden variable to increase the posterior expressivity. One alternative that attempts to blend MCMC methods with variational inference is by Li et al. [2017], which learns MCMC updates for latent variables. However, all of these methods perform optimization over the space of the model parameters instead of sampling.

Many of these approaches ignore the rich history of Bayesian methods in uncertainty estimation and automatic regularization in the model space [MacKay, 1992, Neal, 2012, Kendall and Gal, 2017]. By representing a posterior distribution over parameters of models, Bayesian learning is often able to strike a balance between minimizing model complexity and data misfitting. In the context of text generation, Bayesian recurrent neural networks (RNNs) with stochastic gradient Markov Chain Monte Carlo (SG-MCMC) have been applied for language modeling in Gan et al. [2016]. Similarly, Saatchi and Wilson [2017] showed that using a Bayesian framework and SGHMC for training the generator and discriminator for generative adversarial networks (GANs) seems to both generate diverse image samples and successfully solve the mode collapse problem. Our proposed models, particularly the Bayesian auto-encoder, unify this framework with the recent literature on deep generative modeling.

5 Experiments

We have proposed Bayesian Variational Auto-encoders (BVAEs), a Bayesian formulation of VAEs that captures uncertainty of the model parameters in addition to performing variational approximation on the latent code. We have also proposed a second novel approach, Bayesian Auto-encoders (BAEs), a model with no variational approximation that allows for a flexible distribution of the latent space which can better reflect the true posterior. We formulate algorithms for BAEs, allowing us to tractably estimate the posterior and obtain improved performance on tasks such as density estimation on MNIST and semi-supervised classification. Our models are effective on a wide range of tasks including language modeling, text generation, image generation, and semi-supervised classification.

In Section 5.1, we illustrate our algorithms on a variety of unsupervised learning tasks such as density estimation and sample generation (Section **??** and A), as well as applications in dimensionality reduction 5.1.1. We also demonstrate our models' performance on semi-supervised sentiment classification and image classification in Section 5.2.

5.1 Unsupervised Learning

5.1.1 Dimensionality Reduction on MNIST

We run several experiments with a low dimensional latent space ($\dim z = 3$) to test the efficacy of the Bayesian auto-encoding methods as dimensionality reduction techniques. Figure 1 shows the results, where each color in the scatter plot represents a MNIST digit. It is possible to see that the Bayesian auto-encoder models (Figure 1b and 1c) produce latent representations that are both highly non-Gaussian and well-defined in terms of the class structure. Even more, the Bayesian auto-encoder trained using Algorithm 4 (Figure 1b) produces a more well-defined class structure in comparison to the standard VAE (Figure 1a). All of the methods provide clearer class structures compared to the first three components of PCA[4] (Figure 1d), which demonstrate the benefits of our methods for dimensionality reduction on data with non-linear class boundaries. Finally, we additionally note that the results from the Bayesian auto-encoders come from a significantly smaller training time, either 100 epochs of training or 50,000 steps, which demonstrates the computational efficiency of our methods.

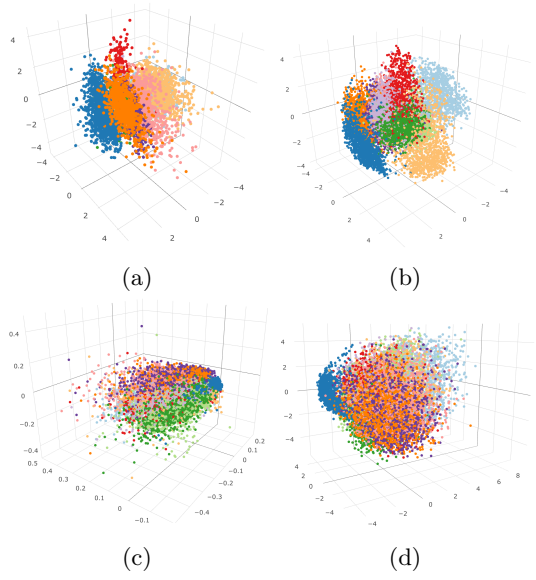

Figure 1: 3D latent spaces of VAE, BAEs and PCA on MNIST test set where each color represents a digit class. (a) VAE. (b) BAE-S (c) BAE-G. (d) The first three principal components of PCA. Interactive versions are available at `https://plot.ly/~uaisubmission2018/`.

[4]Using scikit-learn's implementation: `http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`.

Table 1: Semi-supervised error rates on MR, separated by number of labelled examples (either 2,000 or 3,000), and if model averaging with 3 models was used.

Model	2,000	3,000
LSTM	37.72	33.68
VAE	32.93	31.89
BVAE	31.71	30.86
BAE-S	**30.21**	30.58
BAE-G	30.58	**29.83**

5.2 Semi-supervised Learning

We next evaluate BVAE and BAEs on semi-supervised tasks. We compare out results with the model M2-VAE by Kingma et al. [2014]. Note that we adopt a more effective cycling of labeled data where we use several passes of labels given a single pass of unlabeled data Maaløe et al. [2016].

5.2.1 Sentiment Classification

We test our models on a semi-supervised sentiment classification task. In particular, we use MR [Pang and Lee, 2005], a standard benchmark movie review dataset. Since this dataset does not have a standard train and test split, we randomly divide 90% of all sentences for training and the other 10% for testing. We randomly select N_l sentences from the training set as labeled data and treat the rest as unlabeled data.

Table 1 shows the results of the Bayesian methods as well as a VAE benchmark. We observe that our models (BAE-S and BAE-G) outperform VAE for both $N_l = 2000$ and $N_l = 3000$ settings, where the method BAE-G outperforms all models in both cases. This might be due to the underlying latent space of sentences being more highly non-Gaussian. For BAEs, we save a model per epoch in the last 20 epochs and average their predictions.

To highlight the importance of VAE and Bayesian auto-encoders, we also show the results using an LSTM which does not use the unlabeled data for training. Note that this LSTM attains an error of 25.52% in the fully supervised setting, indicating the model's high capacity. However, the semi-supervised LSTM performs poorly, especially compared to BAEs. The improvement of BAEs over a generic LSTM and VAE demonstrates the strength of our methods which help learn high-quality latent representations.

6 Future Work and Conclusions

We have shown that our Bayesian models provide improvement over similar point-estimate models on many tasks including language modeling, density estimation, and semi-supervised learning.

While our results are near state-of-the-art, there are several other variance bounds that could be used to further improve our models. For example, the Renyi divergence bound of Li and Turner [2016] or the perturbative variational bound of Bamler et al. [2017] have shown promise in sample generation and test log-likelihood performance.

Additionally, our methods are orthogonal to many effective algorithms in the literatures such as virtual adversarial training [Miyato et al., 2017] or ladder networks [Rasmus et al., 2015] for semi-supervised learning. The Bayesian framework that has been developed in this paper can be applied to further improve the representation quality and consequently help with downstream tasks.

Finally, it is possible to create a better algorithm for training the BAE by adapting the Gibbs-like algorithm. Recent advances in Gibbs sampling for mini-batches from Smolyakov et al. [2018] and Dupuy and Bach [2017] can further help further stabilize our BAE update procedure.

References

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and variational inference in deep latent gaussian models. In *ICML*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NIPS*, 2017.

David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3): 448–472, 1992.

David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Yunus Saatchi and Andrew G Wilson. Bayesian gan. In *NIPS*, 2017.

Bin Dai, Yu Wang, John Aston, Gang Hua, and David Wipf. Hidden Talents of the Variational Autoencoder. *arXiv:1706.05148*, 2017.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *CNLL*, 2016.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *JASA*, 112 (518):859–877, 2017.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *ICLR*, 2016.

Richard E Turner and Maneesh Sahani. Two problems with variational expectation maximisation for time-series models. *Bayesian Time series models*, 1(3.1):3–1, 2011.

Tianqi Chen, Emily Fox, and Carlo Guestrin. Stochastic gradient hamiltonian monte carlo. In *ICML*, 2014.

Radford M Neal. Mcmc using ensembles of states for problems with fast and slow variables such as gaussian process regression. *arXiv preprint arXiv:1101.0387*, 2011.

Yuchen Pu, Zhe Gan, Ricardo Henao, Chunyuan Li, Shaobo Han, and Lawrence Carin. Vae learning via stein variational gradient descent. In *NIPS*, 2017.

Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.

Vadim Smolyakov, Qiang Liu, and John W Fisher III. Adaptive scan gibbs sampler for large scale inference problems. *arXiv preprint arXiv:1801.09144*, 2018.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014.

Yihao Feng, Dilin Wang, and Qiang Liu. Learning to draw samples with amortized stein variational gradient descent. In *UAI*, 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2016.

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *ICML*, volume 33, 2016.

Yingzhen Li, Richard E Turner, and Qiang Liu. Approximate inference with amortised mcmc. *arXiv preprint arXiv:1702.08343*, 2017.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Zhe Gan, Chunyuan Li, Changyou Chen, Yunchen Pu, Qinliang Su, and Lawrence Carin. Scalable bayesian learning of recurrent neural networks for language modeling. *arXiv preprint arXiv:1611.08034*, 2016.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, 2005.

Yingzhen Li and Richard E Turner. Rényi divergence variational inference. In *NIPS*, 2016.

Robert Bamler, Cheng Zhang, Manfred Opper, and Stephan Mandt. Perturbative black box variational inference. In *NIPS*, 2017.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *NIPS*, 2015.

Christophe Dupuy and Francis Bach. Online but accurate inference for latent variable models with local gibbs sampling. *Journal of Machine Learning Research*, 18(126): 1–45, 2017.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June 1993. ISSN 0891-2017. URL `http://dl.acm.org/citation.cfm?id=972470.972475`.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *NIPS*, 2016.

A Sample Generation

A.1 Sentence Generation

We test our BVAE and BAE models on the sentence generation task using the Penn Treebank dataset (PTB) [Marcus et al., 1993]. The PTB dataset has a vocabulary of 10000 unique words. We take the standard train/test split dataset from Mikolov et al. [2010].

We compare our BVAE and BAEs to VAEs. The encoder and decoder for all models have one LSTM layer. The batch size is 64 and all models are trained for 100 epochs. We utilize SGD with momentum to optimize VAE and SGHMC for MCMC sampling on the Bayesian models. We halve the learning rate at each epoch when the validation loss does not decrease, starting from 20 to a minimum of 1. We perform a hyperparameter search for $\alpha \in \{0.5, 0.8, 1\}$. The word embedding dimension and the LSTM state dimension are both 200 and we use the latent code dimension of 20. These hyper-parameters are applied to all models for fair comparison. To avoid an undesired stable equilibrium at early epochs that produce vanishing KL divergences, first reported in [Bowman et al., 2016], we applied the "free bits" technique on all models [Kingma et al., 2016]. The hyper-parameter λ used in the "free bits" technique is 2. After epoch 90, we collect the models at the end of each epoch.

Generated sentences from BVAE and BAE are also reported in Table 2. A latent code is sampled from the prior and the decoders generate sentences based on it. The length of generated sentence is set to be 15. This shows that multi-modal posterior can correspond to infinite decoders. Bayesian inference enables the models to generate diverse sentences.

Shakespeare + PTB

Table 2: PTB: Generated Sentences from three generators given the same latent code. Top: BVAE. Middle: BAE-S. Bottom: BAE-G

is the program trader with possible events in what committee executives have n't reached about
or n't committed to what a majority market got a better return to your life
he tries to <unk> off brokers and several titles when they are possibilities even because
from the steps that teddy z is n't a lot of this famous event <eos>
smart brand of more than <unk> a year of between the past century and if
to think of <unk> 's insurance case said it said i would begin the voters
who transfer for the entire farm this food lengthy taxation for a prohibition that could
a private democratic leadership of the senate that than <unk> that underscore set shape <eos>
when if the rule in each case have been built this week telling the N

to die , besides warwick , i may be better to your business . EOS
my acquires is not a problem EOS
apothecary see them give you ; i would you had my daughter . EOS
go , most even to draw these unk temporarily motive because of a nation 's unk aggressive as the insurer
the desires soak will be thrown a massive coping entertainment in ad plunge friday unk view . had n't unk
buy marcos new york unk that any unwholesome is having a joint quarterly cost at most recently filed as the
who would you have save on flower and specific fool ! EOS
look : the duke no . N fadings will become together of the country 's N major purchase differently EOS

Table 3: Shakespeare+PTB: Generated Sentences given four latent code from two generators sampling from the posterior of SGHMC-AE