Gaussian Process Modeling of Approximate Inference Errors for Variational Autoencoders

Minyoung Kim
Samsung AI Center Cambridge, UK
mikim21@gmail.com

Abstract

Variational autoencoder (VAE) is a very successful generative model whose key element is the so-called amortized inference network, which can perform test time inference using a single feed forward pass. Unfortunately, this comes at the cost of degraded accuracy in posterior approximation, often underperforming the instance-wise variational optimization. Although the latest semi-amortized approaches mitigate the issue by performing a few variational optimization updates starting from the VAE's amortized inference output, they inherently suffer from computational overhead for inference at test time. In this paper, we address the problem in a completely different way by considering a random inference model, where we model the mean and variance functions of the variational posterior as random Gaussian processes (GP). The motivation is that the deviation of the VAE's amortized posterior distribution from the true posterior can be regarded as random noise, which allows us to view the approximation error as uncertainty in posterior approximation that can be dealt with in a principled GP manner. In particular, our model can quantify the difficulty in posterior approximation by a Gaussian variational density. Inference in our GP model is done by a single feed forward pass through the network, significantly faster than semi-amortized methods. We show that our approach attains higher test data likelihood than the state-of-the-arts on several benchmark datasets.

1. Introduction

Variational Autoencoder (VAE) [14, 31] is a very successful generative model where a highly complex deep non-linear generative process can be easily incorporated. A key element of the VAE, the deep inference (a.k.a. encoder) network, can perform the test time inference using a single feed forward pass through the network, bringing significant computational speed-up. This feature, known as *amortized inference*, allows the VAE to circumvent other-wise time-consuming steps of solving the variational optimization problem for each individual instance at test time, required in the standard variational inference techniques, such as the stochastic variational inference (SVI) [9].

As suggested by the recent study [2], however, the amortized inference can also be a drawback of the VAE, specifically the accuracy of posterior approximation by the amortized inference network is often lower than the accuracy of the SVI's full variational optimization. There are two general approaches to reduce this amortization error. The first is to increase the network capacity of the inference model (e.g., flow-based models [13, 37]). The other direction is the so-called *semi-amortized* approach [12, 15, 24, 27], where the key idea is to use the VAE's amortized inference network to produce a good initial distribution, from which a few SVI steps are performed at test time to further reduce the amortization error, quite similar in nature to the test time model adaptation of the MAML [6] in multi-task (meta) learning. Although these models often lead to improved posterior approximation, they raise several issues: Training the models for the former family of approaches is usually difficult because of the increased model complexity; the latter approaches inadvertently suffer from computational overhead of additional SVI gradient steps at test time.

In this paper, we propose a novel approach to address these drawbacks. We retain the amortized inference framework similar to the standard VAE for its computational benefits, but consider a *random inference model*. Specifically, the mean and the variance functions of the variational posterior distribution are a priori assumed to be Gaussian process (GP) distributed. There are two main motivations for this idea. The first one stems from the suboptimality of the VAE, where the estimated amortized inference network suffers from deviation from the true posteriors. This inaccuracy can be viewed and modeled as uncertainty in the posterior approximation of the deterministic amortized inference network, suggesting the need for a principled Bayesian uncertainty treatment. The second intuition is that the deviation of the VAE's variational posterior distributions from the true posteriors can be naturally regarded as random noise.

Whereas the semi-amortized approaches perform extra SVI gradient updates at test time to account for this noise, we model the discrepancy using a Bayesian neural network (GP), resulting in a faster and more accurate amortized model via principled uncertainty marginalization. Another benefit of the Bayesian treatment is that we can quantify the discrepancy in approximation, which can serve as indicators for goodness of posterior approximations. The inference in our model is significantly faster than that of semi-amortized methods, accomplished by a single feed forward pass through the GP posterior marginalized inference network. We show that our approach attains higher test data likelihood scores than the state-of-the-art semi-amortized approaches and even the high-capacity flow-based encoder models on several benchmark datasets.

2. Background

Let $\mathbf{x} \in \mathcal{X}$ be an input data point and $\mathbf{z} \in \mathbb{R}^d$ be the latent vector. We consider the VAE model[1]:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}), \quad p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}), \sigma_{\mathbf{x}}^2 \mathbf{I}), \quad (1)$$

where $\mathbf{g}_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathcal{X}$ is a (deep) neural network with the weight parameters denoted by $\boldsymbol{\theta}$, and $\sigma_{\mathbf{x}}^2$ is the variance[2] of the white noise. For the given data $\mathcal{D} = \{\mathbf{x}^i\}_{i=1}^N$, we maximize the data log-likelihood, $\sum_{i=1}^N \log p_{\boldsymbol{\theta}}(\mathbf{x}^i)$, with respect to $\boldsymbol{\theta}$ where $p_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{p(\mathbf{z})}[p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]$. Due to the infeasibility of the marginal log-likelihood, the variational inference exploits the following inequality,

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\big[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\big], \quad (2)$$

which holds for *any* density $q(\mathbf{z}|\mathbf{x})$. The inequality becomes tighter as $q(\mathbf{z}|\mathbf{x})$ becomes closer to the true posterior, as the gap equals $\mathrm{KL}(q(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))$. Then we adopt a tractable density family (e.g., Gaussian) $q_{\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{x})$ parametrized by $\boldsymbol{\lambda}$, and maximize the lower bound in (2) with respect to $\boldsymbol{\lambda}$. Since our goal is maximizing the log-marginal, $\log p_{\boldsymbol{\theta}}(\mathbf{x})$, we also need to optimize the lower bound with respect to $\boldsymbol{\theta}$ together with $\boldsymbol{\lambda}$, either concurrently or in an alternating fashion.

Note that at current $\boldsymbol{\theta}$, the lower bound optimization with respect to $\boldsymbol{\lambda}$ needs to be specific to each input $\mathbf{x}$, and hence the optimal solution is dependent on $\mathbf{x}$. Formally, we can denote the optimum by $\boldsymbol{\lambda}^*(\mathbf{x})$. The *stochastic variational inference* (SVI) [9] faithfully implements this idea, and the approximate posterior inference for a new input point $\mathbf{x}$ in SVI amounts to solving the ELBO optimization on the fly by gradient ascent. Although this can yield very accurate

---

[1]Although there can be possible variations (e.g., heteroscedastic variance for $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$), we assume a homoscedastic model for simplicity, and our approach is easily extendable to the variants.

[2]For simplicity we regard $\sigma_{\mathbf{x}}^2$ as a fixed constant although it can be subsumed in $\boldsymbol{\theta}$ and trained.

posterior approximation, it incurs computational overhead since we have to perform full variational optimization for each and every input $\mathbf{x}$. The VAE [14] addresses this problem by introducing a deep neural network $\boldsymbol{\lambda}(\mathbf{x}; \boldsymbol{\phi})$ with the weight parameters $\boldsymbol{\phi}$ as a universal function approximator of the optimum $\boldsymbol{\lambda}^*(\mathbf{x})$, and optimize the lower bound with respect to $\boldsymbol{\phi}$. This approach is called the *amortized variational inference* (AVI). Thus the main benefit of the VAE is computational speed-up as one can simply do feed forward pass through the inference network $\boldsymbol{\lambda}(\mathbf{x}; \boldsymbol{\phi})$ to perform posterior inference for each $\mathbf{x}$.

The recent study in [2] raised the issue of the amortized inference in the VAE, where the quality of data fitting is degraded due to the approximation error between $\boldsymbol{\lambda}^*(\mathbf{x})$ and $\boldsymbol{\lambda}(\mathbf{x}; \boldsymbol{\phi})$, dubbed the *amortization error*. To retain the AVI's computational advantage and reduce the amortization error, there were attempts to take the benefits of SVI and AVI, which are referred to as *semi-amortized variational inference* (SAVI) [12, 15, 24]. The key idea is to learn the amortized inference network to produce a reasonably good initial iterate for the follow-up SVI optimization, perhaps just a few steps. This warm-start SVI gradient ascent would be faster than full SVI optimization, and could reduce the approximation error of the AVI. Although the inference in the SAVI is faster than SVI, it still requires gradient ascent optimization at test time, which might be the main drawback. In the next section we propose a novel approach that is much faster than the SAVI, avoiding gradient updates at test time and requiring only feed forward pass through a single network, and at the same time can yield more accurate posterior approximation.

3. Gaussian Process Inference Network

We start from the variational density of the VAE, but with slightly different notation, as follows:

$$q(\mathbf{z}|\mathbf{x}, \mathbf{f}, \mathbf{h}) = \mathcal{N}\big(\mathbf{z}; \mathbf{f}(\mathbf{x}), \mathrm{Diag}(\mathbf{h}(\mathbf{x}))^2\big), \quad (3)$$

where $\mathbf{f}, \mathbf{h} : \mathcal{X} \to \mathbb{R}^d$ are the mean and standard deviation functions of the variational posterior distribution. Note that if we model $\mathbf{f}$ and $\mathbf{h}$ as *deterministic* functions (neural networks) and optimize their weight parameters (i.e., point estimation), then it reduces to the standard VAE for which $\mathbf{f}$ and $\mathbf{h}$ constitute $\boldsymbol{\lambda}(\mathbf{x}; \boldsymbol{\phi})$. However, such point estimates may be inaccurate, and we consider random functions following the Bayesian treatment. Specifically we let $\mathbf{f} = [f_1(\cdot), ..., f_d(\cdot)]^\top$ and $\mathbf{h} = [h_1(\cdot), ..., h_d(\cdot)]^\top$ be independent random GP distributed functions a priori [30],

$$\mathbf{f}(\cdot) \sim \prod_{j=1}^d \mathcal{GP}(b_j(\cdot), k^m(\cdot, \cdot)), \ \mathbf{h}(\cdot) \sim \prod_{j=1}^d \mathcal{GP}(c_j(\cdot), k^s(\cdot, \cdot)).$$

$$(4)$$

Here $\mathbf{b}(\cdot) = [b_1(\cdot), ..., b_d(\cdot)]^\top$, $\mathbf{c}(\cdot) = [c_1(\cdot), ..., c_d(\cdot)]^\top$ are the GP mean functions which can be modeled by deep neural networks, and the GP covariance functions of $\mathbf{f}$ and $\mathbf{h}$ are denoted by k^m and k^s, respectively, where we share the same covariance function across dimensions for simplicity.

Relation to SAVI. Note that the GP-priored variational density model in (3–4) can be written as:

$$q(\mathbf{z}|\mathbf{x}, \mathbf{f}, \mathbf{h}) = \mathcal{N}\big(\mathbf{b}(\mathbf{x}) + \mathbf{f}(\mathbf{x}), \mathrm{Diag}(\mathbf{c}(\mathbf{x}) + \mathbf{h}(\mathbf{x}))^2\big), \quad (5)$$

where $\mathbf{f}$ and $\mathbf{h}$ now follow *zero-mean* Gaussian processes. If we view the VAE's point estimate inference model as: $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}\big(\mathbf{b}(\mathbf{x}), \mathrm{Diag}(\mathbf{c}(\mathbf{x}))^2\big)$, then (5) effectively models the *discrepancy* between the VAE's $q(\mathbf{z}|\mathbf{x})$ and the true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ via stochastic noise models. Recall that in order to reduce this discrepancy, the semi-amortized approaches perform extra SVI gradient updates starting from $\mathbf{b}(\mathbf{x})$ and $\mathbf{c}(\mathbf{x})$ at test time on the fly. Instead, we aim to learn the discrepancy using Bayesian neural networks $\mathbf{f}$ and $\mathbf{h}$ (GP as a special case; see Sec. 3.1), resulting in a faster and more accurate amortized inference model by taking into account uncertainty (stochastic noise) in a principled manner.

For instance, the GP posterior $p(\mathbf{f}, \mathbf{h}|\mathcal{D})$ can predict the above-mentioned discrepancy accurately, while their variances (e.g., $\mathbb{V}(f(\mathbf{x})|\mathcal{D})$) can serve as *gauge* that quantifies the degree of (instance-wise) uncertainty/difficulty in posterior approximation via the amortized inference network. To this end, we describe a reasonable likelihood model to establish a GP framework, and derive an efficient GP posterior inference algorithm in what follows.

3.1. Likelihood Model and GP Posterior Inference

To establish a valid Bayesian framework, we define a likelihood model, that is, the compatibility score of how each individual instance $\mathbf{x} \sim \mathcal{D}$ is likely to be generated under the given functions $\mathbf{f}$ and $\mathbf{h}$. A reasonable choice is the variational lower bound (2), which we denote as:

$$\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{f}, \mathbf{h}; \mathbf{x}) := \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \mathbf{f}, \mathbf{h})}\big[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}|\mathbf{x}, \mathbf{f}, \mathbf{h})\big]. \quad (6)$$

Clearly $\log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \mathcal{L}_{\boldsymbol{\theta}}(\mathbf{f}, \mathbf{h}; \mathbf{x})$, and (6) can serve as surrogate[3] for the log-likelihood function $\log p(\mathbf{x}|\mathbf{f}, \mathbf{h})$. Given the data $\mathcal{D} = \{\mathbf{x}^i\}_{i=1}^N$, combining the GP priors and the likelihood model leads to the GP posterior,

$$p(\mathbf{f}, \mathbf{h}|\mathcal{D}) \propto p(\mathbf{f})\, p(\mathbf{h}) \prod_{\mathbf{x} \in \mathcal{D}} \exp\big(\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{f}, \mathbf{h}; \mathbf{x})\big). \quad (7)$$

However, solving (7) requires time and memory cubic in the number of data points N, which is prohibitive for large-

---

[3]Technically, $e^{\mathcal{L}}$ may not be a valid density (integration not equal to 1), and one has to deal with the difficult normalizing partition function in principle. For simplicity, we do not consider it and regard $\mathcal{L}$ as *unnormalized* log-likelihood function.

scale data. Although there exist efficient scalable approximate inference techniques in the GP literature [5, 8, 28, 32, 35], here we adopt the *linear deep kernel* trick [10, 39], which we briefly summarize below.

Linear deep kernel trick for approximating GP. A random (scalar) function $f(\mathbf{x})$ that follows the 0-mean GP with covariance (kernel) k, namely $f(\cdot) \sim \mathcal{GP}(0, k(\cdot, \cdot))$, can be represented as a linear form with an explicit feature space mapping. Consider a feature mapping $\boldsymbol{\psi}$: $\mathcal{X} \to \mathbb{R}^p$ such that the covariance function is approximated as inner product in the feature space (of dimension p), namely $k(\mathbf{x}, \mathbf{x}') \approx \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\psi}(\mathbf{x}')$. Now, introducing the p-variate random vector $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, allows us to write the GP function as $f(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x})$. It is because $\mathrm{Cov}(f(\mathbf{x}), f(\mathbf{x}')) = \mathrm{Cov}(\mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}), \mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}')) = \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\psi}(\mathbf{x}') \approx k(\mathbf{x}, \mathbf{x}')$. A main advantage of this representation is that we can turn the non-parametric GP into a parametric Bayesian model, where the posterior inference can be done on the finite dimensional random vector $\mathbf{w}$ instead. The feature mapping $\boldsymbol{\psi}(\cdot)$ can be modeled as a deep neural network, and its weight parameters constitute the covariance (kernel) parameters of the GP. This way, we can (approximately) view GP as a special case of Bayesian neural networks where we treat the final fully connected layer $\mathbf{w}$ as random [4, 7, 20, 25]. Note that although this is rather a simplified form of the deep kernel [39] by applying the linear kernel on the outputs of $\boldsymbol{\psi}(\cdot)$, it has been widely used with great success [10, 36].

Returning to our GP posterior inference (7), the two GP-priored functions can be written as: $f_j(\mathbf{x}) = \mathbf{w}_j^\top \boldsymbol{\psi}^m(\mathbf{x})$ and $h_j(\mathbf{x}) = \mathbf{u}_j^\top \boldsymbol{\psi}^s(\mathbf{x})$ for $j = 1, \ldots, d$, where $\mathbf{w}_j$'s and $\mathbf{u}_j$'s are mutually independent p-variate random vectors from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The feature functions $\boldsymbol{\psi}^m, \boldsymbol{\psi}^s : \mathcal{X} \to \mathbb{R}^p$ are deep neural networks that define the covariance functions: $k^m(\mathbf{x}, \mathbf{x}') = \boldsymbol{\psi}^m(\mathbf{x})^\top \boldsymbol{\psi}^m(\mathbf{x}')$, $k^s(\mathbf{x}, \mathbf{x}') = \boldsymbol{\psi}^s(\mathbf{x})^\top \boldsymbol{\psi}^s(\mathbf{x}')$. By letting $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_d]^\top$ and $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_d]^\top$ be the $(d \times p)$ matrices with the random vectors in the rows, we have $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), ..., f_d(\mathbf{x})]^\top = \mathbf{W}\boldsymbol{\psi}^m(\mathbf{x})$, $\mathbf{h}(\mathbf{x}) = [h_1(\cdot), ..., h_d(\cdot)]^\top = \mathbf{U}\boldsymbol{\psi}^s(\mathbf{x})$. The inference in (5) can be written as $q(\mathbf{z}|\mathbf{x}, \mathbf{W}, \mathbf{U})$ where

$$q(\mathbf{z}|\mathbf{x}, \mathbf{W}, \mathbf{U}) = \mathcal{N}\big(\mathbf{b}(\mathbf{x}) + \mathbf{W}\boldsymbol{\psi}^m(\mathbf{x}), \mathrm{Diag}(\mathbf{c}(\mathbf{x}) + \mathbf{U}\boldsymbol{\psi}^s(\mathbf{x}))^2\big) \quad (8)$$

while (7) becomes:

$$p(\mathbf{W}, \mathbf{U}|\mathcal{D}) \propto \mathcal{N}(\mathbf{W}; \mathbf{0}, \mathbf{I})\mathcal{N}(\mathbf{U}; \mathbf{0}, \mathbf{I}) \prod_{\mathbf{x} \in \mathcal{D}} e^{\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{W}, \mathbf{U}; \mathbf{x})},$$

$$\text{where } \mathcal{L}_{\boldsymbol{\theta}}(\mathbf{W}, \mathbf{U}; \mathbf{x}) := \mathbb{E}_q\bigg[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x}, \mathbf{W}, \mathbf{U})}\bigg]. \quad (9)$$

We approximate (9) by $q(\mathbf{W}, \mathbf{U})$ defined as:

$$q(\mathbf{W}, \mathbf{U}; \boldsymbol{\Lambda}) = \prod_{j=1}^d \mathcal{N}(\mathbf{w}_j; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\, \mathcal{N}(\mathbf{u}_j; \boldsymbol{\eta}_j, \boldsymbol{\Gamma}_j) \quad (10)$$

where $\boldsymbol{\Lambda} := \{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, \boldsymbol{\eta}_j, \boldsymbol{\Gamma}_j\}_{j=1}^d$ constitutes the GP variational parameters.

3.2. GP Posterior Marginalized Encoder

Before we proceed to GP inference and learning (Sec. 3.3), we derive the posterior averaged encoder,

$$q(\mathbf{z}|\mathbf{x}) = \iint q(\mathbf{W}, \mathbf{U})q(\mathbf{z}|\mathbf{x}, \mathbf{W}, \mathbf{U})d\mathbf{W}d\mathbf{U}. \qquad (11)$$

Note that (11) can be seen as the final latent inference model of our GP VAE model, where the uncertainty captured in the GP posterior $q(\mathbf{W}, \mathbf{U})$ is all marginalized out. For instance, the test log-likelihood score $\log p(\mathbf{x})$ under our model can be estimated by the importance weighted sampling method [1] as the proposal distribution. As it also appears in the GP learning in the next section, we provide the derivation for $q(\mathbf{z}|\mathbf{x})$ here.

Although the two terms in the integrand of (11) are both Gaussians, it is infeasible to have close-form formula due to the dependency of the covariance of (8) on $\mathbf{U}$. Instead, we view (11) as (a limit of) a mixture of Gaussians, $\sum_i \alpha_i \mathcal{N}(\mathbf{z}; \mathbf{m}_i, \mathbf{V}_i)$ where α_i corresponds to $q(\mathbf{W}, \mathbf{U})$ with index i mapped to $(\mathbf{W}, \mathbf{U})$, and $\mathbf{m}_i, \mathbf{V}_i$ denote the mean and covariance of $q(\mathbf{z}|\mathbf{x}, \mathbf{W}, \mathbf{U})$ in (8). Since a Gaussian mixture can be approximated by a single Gaussian by the second-order moment matching[4], namely $\sum_i \alpha_i \mathcal{N}(\mathbf{m}_i, \mathbf{V}_i) \approx \mathcal{N}(\mathbf{m}, \mathbf{V})$ where $\mathbf{m} = \sum_i \alpha_i \mathbf{m}_i$ and $\mathbf{V} = \sum_i \alpha_i(\mathbf{m}_i \mathbf{m}_i^\top + \mathbf{V}_i) - \mathbf{m}\mathbf{m}^\top$, applying it to (11) yields: $q(\mathbf{z}|\mathbf{x}) \approx \mathcal{N}(\mathbf{z}; \mathbf{m}(\mathbf{x}), \mathrm{Diag}(\mathbf{v}(\mathbf{x})))$, where

$$m_j(\mathbf{x}) = b_j(\mathbf{x}) + \boldsymbol{\mu}_j^\top \boldsymbol{\psi}^m(\mathbf{x}),$$
$$v_j(\mathbf{x}) = c_j(\mathbf{x})^2 + 2\boldsymbol{\eta}_j^\top \boldsymbol{\psi}^s(\mathbf{x})c_j(\mathbf{x}) + \boldsymbol{\psi}^m(\mathbf{x})^\top \boldsymbol{\Sigma}_j \boldsymbol{\psi}^m(\mathbf{x})$$
$$\qquad + \boldsymbol{\psi}^s(\mathbf{x})^\top (\boldsymbol{\eta}_j \boldsymbol{\eta}_j^\top + \boldsymbol{\Gamma}_j)\boldsymbol{\psi}^s(\mathbf{x}), \qquad (12)$$

for $j = 1, \ldots, d$.

Note from (12) that as a special case, $q(\mathbf{z}|\mathbf{x})$ becomes the standard VAE's encoder distribution with means $b_j(\mathbf{x})$ and variances $c_j(\mathbf{x})^2$ if the GP posterior is ignored (i.e., $\boldsymbol{\Lambda} = \{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, \boldsymbol{\eta}_j, \boldsymbol{\Gamma}_j\} = \mathbf{0}$ to lead to the deterministic zero noise model). And our learned GP posterior (non-zero $\boldsymbol{\Lambda}$) informs us how the deviation from the true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ can be determined and compensated by (12).

3.3. GP Inference and Learning

Now we describe how the variational GP inference (i.e., optimizing $\boldsymbol{\Lambda}$ in $q(\mathbf{W}, \mathbf{U}; \boldsymbol{\Lambda})$) can be done. Similar to other GP variational learning, the objective function that we will derive establishes a lower bound of the model's data likelihood, and hence we can learn the *model parameters* as well by maximizing the lower bound (empirical Bayes). The model parameters consist of the parameters in the GP mean

---

[4]Equivalent to $\min_{\{\mathbf{m}, \mathbf{V}\}} \mathrm{KL}\big(\sum_i \alpha_i \mathcal{N}(\mathbf{m}_i, \mathbf{V}_i) \,||\, \mathcal{N}(\mathbf{m}, \mathbf{V})\big)$.

and covariance functions (i.e., the weight parameters of the deep networks $\mathbf{b}(\mathbf{x})$, $\mathbf{c}(\mathbf{x})$, $\boldsymbol{\psi}^m(\mathbf{x})$, and $\boldsymbol{\psi}^s(\mathbf{x})$), and those in the likelihood model (i.e., $\boldsymbol{\theta}$ in the decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$).

To approximate $q(\mathbf{W}, \mathbf{U}) \approx p(\mathbf{W}, \mathbf{U}|\mathcal{D})$, we aim to minimize $\mathrm{KL}\big(q(\mathbf{W}, \mathbf{U})||p(\mathbf{W}, \mathbf{U}|\mathcal{D})\big)$, and it can be shown that the KL can be written as follows (Supplement for details):

$$\mathrm{KL}(q||p) = \log \hat{p}_{\boldsymbol{\theta}}(\mathcal{D}) - \sum_{\mathbf{x} \in \mathcal{D}} \mathrm{ELBO}(\boldsymbol{\theta}, \boldsymbol{\Lambda}; \mathbf{x}), \qquad (13)$$

where $\hat{p}_{\boldsymbol{\theta}}(\mathcal{D}) = \mathbb{E}_{\mathbf{W}, \mathbf{U} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}\big[\prod_{\mathbf{x}} e^{\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{W}, \mathbf{U}; \mathbf{x})}\big]$ is the marginal data likelihood using our surrogate $\mathcal{L}$ in (9), and

$$\mathrm{ELBO} := -\mathbb{E}_{q(\mathbf{W}, \mathbf{U})}[\mathrm{KL}(q(\mathbf{z}|\mathbf{x}, \mathbf{W}, \mathbf{U})||p(\mathbf{z}))] + \quad (14)$$
$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\big[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\big] - \frac{1}{N}\mathrm{KL}(q(\mathbf{W}, \mathbf{U})||\mathcal{N}(\mathbf{0}, \mathbf{I})).$$

We now discuss how individual terms in the ELBO (14) can be derived. The last term of (14) is the KL divergence between Gaussian densities, and admits a close form. The second term is the expected log-likelihood with respect to the GP posterior marginalized encoder $q(\mathbf{z}|\mathbf{x})$, (12), and we can do this by Monte Carlo estimation with the well-known reparametrization trick [14]. Finally, the first term in (14) is the Gaussian averaged KL divergence between Gaussians, and thus it admits a closed form:

$$\frac{1}{2}\sum_{j=1}^d \bigg(v_j(\mathbf{x}) + \big(b_j(\mathbf{x}) + \boldsymbol{\mu}_j^\top \boldsymbol{\psi}^m(\mathbf{x})\big)^2 - 1 - \quad (15)$$
$$\mathbb{E}_{\mathcal{N}(\mathbf{u}_j; \boldsymbol{\eta}_j, \boldsymbol{\Gamma}_j)}\Big[\log\big(c_j(\mathbf{x}) + \mathbf{u}_j^\top \boldsymbol{\psi}^s(\mathbf{x})\big)^2\Big]\bigg).$$

The last term in (15) is essentially a Gaussian expected squared $\log$ function, which can be written as a closed form, albeit complicated, using the confluent hyper-geometric function [22]. However, for simplicity we estimate it using the reparametrized Monte-Carlo method.

Summary. The overall learning steps are as follows:
1. Initialize the variational parameters $\boldsymbol{\Lambda}$ and the model parameters $\boldsymbol{\theta}$, $\mathbf{b}(\mathbf{x})$, $\mathbf{c}(\mathbf{x})$, $\boldsymbol{\psi}^m(\mathbf{x})$, and $\boldsymbol{\psi}^s(\mathbf{x})$.
2. Repeat until convergence:
   (a) Estimate the marginalized $q(\mathbf{z}|\mathbf{x})$ using (12).
   (b) Optimize the ELBO (14) wrt all parameters.
3. (At test time) The GP marginalized encoder $q(\mathbf{z}|\mathbf{x})$ can be used to perform reconstruction, and evaluate the test likelihood $p_{\boldsymbol{\theta}}(\mathbf{x})$, e.g., using the importance weighted sampling method (IWAE) [1][5]. The uncertainty (variance) of the posterior noise $\mathrm{Tr}\mathbb{V}(\mathbf{f}(\mathbf{x})|\mathcal{D})$ (similarly for $\mathbf{h}$) can be approximately estimated as $\boldsymbol{\psi}^m(\mathbf{x})^\top \big(\sum_{j=1}^d \boldsymbol{\Sigma}_j\big)\boldsymbol{\psi}^m(\mathbf{x})$.

---

[5]IWAE is not used during training since it usually takes too many steps/samples (e.g., K=100) for training.

4. Related Work

As enumerating all related literature in this section can be infeasible, we briefly review some of the recent works that are highly related with ours. The issue of amortization error in VAE was raised in [2], after which several semi-amortized methods were attempted [12, 15, 24] that essentially follow a few SVI gradient steps at test time. An alternative line of research approaches the problem by enlarging the representational capacity of the encoder network, including the flow-based models that apply nonlinear invertible transformations to VAE's variational posterior [13, 37]. Recently [11] proposed a greedy recursive mixture estimation method for the encoder in VAE, where the idea is to iteratively augment the current mixture with new components to maximally reduce the divergence between the variational and the true posteriors. There were also previous attempts to incorporate additional random variables into inference networks [23]. Although they introduced latent random variables in both encoder and decoder models, their motivation is quite different from our random noise view in the posterior approximation, and they mainly aim at endowing a richer density family and functional capacity.

In parallel, there have been previous attempts to apply the Bayesian approach to the VAE modeling. However, they are in nature different from our random function modeling of the encoder uncertainty. The Bayesian Variational VAE [3] rather focused on modeling uncertainty in the *decoder* model, and their main focus is how to deal with out-of-distribution samples in the test set, hence more aligned with transfer learning. The Compound VAE [33] also tackled the similar problem of reducing the amortization gap of the VAE, however, their variational density modeling is less intuitive, inferring the latent vector $\mathbf{z}$ and the encoder weights $\mathbf{W}$ from each data instance. Note that we have more intuitive Bayesian inference for the encoder parameters, $q(\mathbf{W}|\mathcal{D})$ given the *entire* training data $\mathcal{D}$. Their treatment is deemed to augment the latent $\mathbf{z}$ with the weights $\mathbf{W}$ in the conventional VAE. The Variational GP [38], although looking similar to ours, is not specifically aimed for VAE or amortized inference, but for general Bayesian inference. In turn, they built the posterior model using a GP function defined on the Gaussian distributed *latent input* space, instead of defining GP on the input data as we did.

5. Evaluations

We evaluate our Gaussian process VAE model on several benchmarks to show its improved performance over the existing state-of-the-arts. Our focus is two-fold: i) improved test likelihood scores, and ii) faster test time inference than semi-amortized methods. We also contrast with the flow-based models that employ high capacity encoder networks. The competing approaches are as follows. 1) **VAE**: The standard VAE model with amortized inference [14, 31]. 2) **SA**: The semi-amortized VAE [12]. We fix the SVI gradient step size as 10^{-3}, but vary the number of SVI steps from $\{1, 2, 4, 8\}$. 3) **IAF**: The autoregressive-based flow model for the encoder $q(\mathbf{z}|\mathbf{x})$ [13], which has richer expressive capability than the VAE's post-Gaussian encoder. The number of flows is chosen from $\{1, 2, 4, 8\}$. 4) **HF**: The Householder flow encoder model that represents the full covariance using the Householder transformation [37]. The number of flows is chosen from $\{1, 2, 4, 8\}$. 5) **ME**: To enlarge the representational capacity of the encoder network, another possible baseline is a mixture model. More specifically, the inference model is defined as: $q(\mathbf{z}|\mathbf{x}) = \sum_{m=1}^{M} \alpha(m|\mathbf{x}) q_m(\mathbf{z}|\mathbf{x})$, where $q_m(\mathbf{z}|\mathbf{x})$ are amortized inference models (e.g., having the same network architectures as the VAE's encoder network), and $\alpha(m|\mathbf{x})$ are mixing proportions, dependent on the input $\mathbf{x}$, which can be modeled by a single neural network. The mixture encoder (ME) model is trained by gradient ascent to maximize the lower bound of $\log p(\mathbf{x})$ similarly as the VAE. The number of mixture components M is chosen from $\{1, 2, 4, 8\}$. 6) **RME**: The recursive mixture estimation method for the encoder in VAE [11], which showed superiority to ME's blind mixture estimation. 7) **GPVAE**: Our proposed GP encoder model. The GP means and feature functions have the same network architectures as the VAE's encoder.

Datasets. We use the following five benchmark datasets: **MNIST** [19], **OMNIGLOT** [18], **CIFAR10**[6] [16], **SVHN** [26], and **CelebA** [21]. For CelebA, we use tightly cropped face images of size $(64 \times 64 \times 3)$, and randomly split the data into $80\%/10\%/10\%$ train/validation/test sets. For the others, we follow the splits provided in the data, with 10% of the training sets held out for validation.

Network architectures. We adopt the convolutional neural networks for both encoder and decoder models for all competing approaches. The main reason is that the convolutional networks are believed to outperform fully connected networks for many tasks on the image domain[7] [17, 29, 34]. For the encoder architecture, we first apply L convolutional layers with (4×4)-pixels kernels, followed by two fully-connected layers with hidden layers dimension h. For the decoder, the input images first go through two fully connected layers, followed by L transposed convolutional layers with (4×4)-pixels filters. Here, $L = 3$ for all datasets except CelebA ($L = 4$), and $h = 256$ for the MNIST/OMNIGLOT and $h = 512$ for the others. The deep kernel feature functions $\boldsymbol{\psi}^{m,s}(\mathbf{x})$ in our GPVAE model have exactly the same architecture as the encoder network except that the last fully connected layer is removed. This ensures that the GP functions $\mathbf{f}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$ have equal functional capacity to the base encoder network since they are defined to be

---

[6]Results on CIFAR10 can be found in the Supplement.

[7]We empirically compared the two networks in the Supplement.

Table 1. (MNIST & OMNIGLOT) Test log-likelihood scores (unit in nat). The figures in the parentheses next to model names indicate: the number of SVI steps in SA, the number of flows in IAF and HF, and the number of mixture components in ME and RME. The superscripts are the standard deviations. The best (on average) results are boldfaced in red. In each column, we perform the two-sided t-test to measure the statistical significance of the difference between the best model (red) and each competing method. We depict those with p-values greater than 0.01 as boldfaced blue (little evidence of difference). So, anything plain non-colored indicates $p \leq 0.01$ (significantly different).

	MNIST			OMNIGLOT		
	$\mathrm{DIM}(\mathbf{z})=10$	$\mathrm{DIM}(\mathbf{z})=20$	$\mathrm{DIM}(\mathbf{z})=50$	$\mathrm{DIM}(\mathbf{z})=10$	$\mathrm{DIM}(\mathbf{z})=20$	$\mathrm{DIM}(\mathbf{z})=50$
VAE	$685.1^{1.8}$	$930.7^{3.9}$	$1185.7^{3.9}$	$347.0^{1.7}$	$501.6^{1.6}$	$801.6^{4.0}$
SA$^{(1)}$	$688.1^{2.7}$	$921.2^{2.3}$	$1172.1^{1.8}$	$344.1^{1.4}$	$499.3^{2.5}$	$792.7^{7.9}$
SA$^{(2)}$	$682.2^{1.5}$	$932.0^{2.4}$	$1176.3^{3.4}$	$349.5^{1.4}$	$501.0^{2.7}$	$793.1^{4.8}$
SA$^{(4)}$	$683.5^{1.5}$	$925.5^{2.6}$	$1171.3^{3.5}$	$342.1^{1.0}$	$488.2^{1.8}$	$794.4^{1.9}$
SA$^{(8)}$	$684.6^{1.5}$	$928.1^{3.9}$	$1183.2^{3.4}$	$344.8^{1.1}$	$490.3^{2.8}$	$799.4^{2.7}$
IAF$^{(1)}$	$687.3^{1.1}$	$934.0^{3.3}$	$1180.6^{2.7}$	$347.8^{1.6}$	$489.9^{1.9}$	$788.8^{4.1}$
IAF$^{(2)}$	$677.7^{1.6}$	$931.4^{3.7}$	$1190.1^{1.9}$	$344.2^{1.6}$	$494.9^{1.4}$	$795.7^{2.7}$
IAF$^{(4)}$	$685.0^{1.5}$	$926.3^{2.6}$	$1178.1^{1.6}$	$347.9^{1.9}$	$496.0^{2.0}$	$775.1^{2.2}$
IAF$^{(8)}$	$689.7^{1.4}$	$934.1^{2.4}$	$1150.0^{2.2}$	$343.9^{1.4}$	$498.8^{2.3}$	$774.7^{2.9}$
HF$^{(1)}$	$682.5^{1.4}$	$917.2^{2.6}$	$1204.3^{4.0}$	$335.5^{1.2}$	$488.6^{2.0}$	$795.9^{3.3}$
HF$^{(2)}$	$677.6^{2.2}$	$923.9^{3.1}$	$1191.5^{10.8}$	$340.6^{1.3}$	$495.9^{1.8}$	$784.5^{4.8}$
HF$^{(4)}$	$683.3^{2.6}$	$927.3^{2.8}$	$1197.2^{1.5}$	$343.3^{1.2}$	$487.0^{2.7}$	$799.7^{3.2}$
HF$^{(8)}$	$679.6^{1.5}$	$928.5^{3.1}$	$1184.1^{1.8}$	$343.3^{1.3}$	$488.3^{2.4}$	$794.6^{4.0}$
ME$^{(2)}$	$685.7^{1.2}$	$926.7^{3.0}$	$1152.8^{1.7}$	$344.2^{1.5}$	$491.7^{1.4}$	$793.4^{3.8}$
ME$^{(3)}$	$678.5^{2.5}$	$933.1^{4.1}$	$1162.8^{4.7}$	$350.3^{1.8}$	$491.2^{2.1}$	$807.5^{4.9}$
ME$^{(4)}$	$680.0^{0.9}$	$914.7^{2.3}$	$1205.1^{2.3}$	$337.7^{1.1}$	$491.3^{1.8}$	$732.0^{3.1}$
ME$^{(5)}$	$682.0^{1.7}$	$920.6^{1.9}$	$1198.5^{3.5}$	$343.0^{1.4}$	$478.0^{2.8}$	$805.7^{3.8}$
RME$^{(2)}$	$\mathbf{697.2}^{1.1}$	$\mathbf{943.9}^{1.6}$	$1201.7^{0.9}$	$349.3^{1.5}$	$508.2^{1.2}$	$\mathbf{821.0}^{3.1}$
RME$^{(3)}$	$\mathbf{698.2}^{1.1}$	$\mathbf{945.1}^{1.6}$	$1202.4^{1.0}$	$349.9^{1.6}$	$507.5^{1.1}$	$\mathbf{820.4}^{0.9}$
RME$^{(4)}$	$\mathbf{699.0}^{1.0}$	$\mathbf{\color{red}945.2}^{1.6}$	$1203.1^{1.0}$	$350.7^{1.7}$	$509.0^{1.2}$	$\mathbf{819.9}^{0.9}$
RME$^{(5)}$	$\mathbf{\color{red}699.4}^{2.1}$	$\mathbf{945.0}^{1.7}$	$1203.7^{1.0}$	$351.1^{1.7}$	$509.1^{1.4}$	$\mathbf{819.9}^{0.9}$
GPVAE	$\mathbf{696.5}^{1.5}$	$\mathbf{944.3}^{2.8}$	$\mathbf{\color{red}1212.9}^{3.2}$	$\mathbf{\color{red}354.8}^{1.5}$	$\mathbf{\color{red}516.3}^{2.1}$	$\mathbf{\color{red}821.8}^{4.4}$

products of the features and the Gaussian random weights $\mathbf{W}$ and $\mathbf{U}$. And, accordingly the feature dimension p is set equal to h. The full covariance matrices of the variational density $q(\mathbf{W}, \mathbf{U})$ are represented by Cholesky parametrization to ensure positive definiteness (e.g., $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^\top$ where $\mathbf{L}$ is a lower triangle matrix with strictly positive diagonals).

Experimental setup. The latent dimension is chosen from $\{10, 20, 50\}$. To report the test log-likelihood scores, we use the importance weighted estimation (IWAE)[8] [1] with 100 samples. For each model/dataset, we perform 10 runs with different random train/validation splits, where each run consists of three trainings by starting with different random model parameters, among which only one model with the best validation result is chosen.

5.1. Results

The test log-likelihood scores are summarized in Table 1 (MNIST and OMNIGLOT), Table 2 (SVHN and CelebA). Our GPVAE overall outperforms the competing approaches consistently for all datasets. The performance of the semi-amortized approach (SA) is mixed, sometimes achieving improvement over VAE, but not consistently. SA's performance is very sensitive to the number of SVI gradient up-date steps, another drawback of the SA where the gradient-based adaption has to be performed at test time. Although one could adjust the gradient step size (currently we are using a fixed gradient step size) to improve the performance, as far as we know, there is little principled way to tune the step size at test time that can attain optimal accuracy and inference time trade off. The flow-based models (IAF and HF) adopt nonlinear invertible transformations to enrich the representational capacity of the variational posterior. In principle, they are capable of representing highly nonlinear non-Gaussian conditional densities, perhaps subsuming the true posteriors, via autoregressive flows (IAF) and the Householder transformed full covariance matrices (HF). However, their improvement in accuracy over the VAE trails that of our GPVAE; they often perform only as well as the VAE. The failure of the flow-based models might be due to the difficulty in optimizing complex encoder models where similar observations were made in related previous work [11, 27]. This result suggests that sophisticated and incremental learning criteria are critical, beyond just enlarging the structural capacity of the neural networks. Our GPVAE's explicit modeling of the deviation of the base encoder density from the true posterior via GP noise processes accomplishes this goal.

Similarly, despite its increased functional capacity, the

---

[8]The details can be also found in the Supplement.

Table 2. (SVHN & CelebA) Test log-likelihood scores. The same interpretation as Table 1.

	SVHN			CELEBA		
	$\mathrm{DIM}(\mathbf{z})=10$	$\mathrm{DIM}(\mathbf{z})=20$	$\mathrm{DIM}(\mathbf{z})=50$	$\mathrm{DIM}(\mathbf{z})=10$	$\mathrm{DIM}(\mathbf{z})=20$	$\mathrm{DIM}(\mathbf{z})=50$
VAE	$3360.2^{9.1}$	$4054.5^{14.3}$	$5363.7^{21.4}$	$9767.7^{36.0}$	$12116.4^{25.3}$	$15251.9^{39.7}$
SA$^{(1)}$	$3358.7^{8.9}$	$4031.5^{19.0}$	$5362.1^{35.7}$	$9735.5^{21.4}$	$12091.1^{21.6}$	$15285.8^{29.4}$
SA$^{(2)}$	$3356.0^{8.8}$	$4041.5^{15.5}$	$5377.0^{23.2}$	$9754.2^{20.4}$	$12087.1^{21.5}$	$15252.7^{29.0}$
SA$^{(4)}$	$3327.8^{8.2}$	$4051.9^{22.2}$	$5391.7^{20.4}$	$9769.1^{20.6}$	$12116.3^{20.5}$	$15187.3^{27.9}$
SA$^{(8)}$	$3352.8^{11.5}$	$4041.6^{9.5}$	$5370.8^{18.5}$	$9744.8^{19.4}$	$12100.6^{22.8}$	$15096.5^{27.2}$
IAF$^{(1)}$	$3377.1^{8.4}$	$4050.0^{9.4}$	$5368.3^{11.5}$	$9750.3^{27.4}$	$12098.0^{20.6}$	$15271.2^{28.6}$
IAF$^{(2)}$	$3362.3^{8.9}$	$4054.6^{10.5}$	$5360.0^{10.0}$	$9794.4^{23.3}$	$12104.5^{21.8}$	$15262.2^{27.8}$
IAF$^{(4)}$	$3346.1^{8.7}$	$4048.6^{8.7}$	$5338.1^{10.2}$	$9764.7^{29.5}$	$12094.6^{22.6}$	$15261.0^{28.1}$
IAF$^{(8)}$	$3372.6^{8.3}$	$4042.0^{9.6}$	$5341.8^{10.1}$	$9764.0^{21.6}$	$12109.3^{22.0}$	$15241.5^{27.9}$
HF$^{(1)}$	$3381.4^{8.9}$	$4028.8^{9.7}$	$5372.0^{10.1}$	$9748.3^{29.5}$	$12077.2^{31.4}$	$15240.5^{27.6}$
HF$^{(2)}$	$3342.4^{8.3}$	$4030.7^{9.9}$	$5376.6^{10.2}$	$9765.8^{25.6}$	$12093.0^{25.6}$	$15258.2^{30.3}$
HF$^{(4)}$	$3370.0^{8.2}$	$4038.4^{9.7}$	$5371.8^{9.8}$	$9754.3^{23.8}$	$12082.0^{27.0}$	$15266.5^{29.5}$
HF$^{(8)}$	$3343.8^{8.2}$	$4035.9^{8.9}$	$5351.1^{11.1}$	$9737.5^{24.5}$	$12087.3^{25.5}$	$15248.7^{29.7}$
ME$^{(2)}$	$3352.3^{9.9}$	$4037.2^{11.0}$	$5343.2^{13.1}$	$9825.3^{20.7}$	$12072.7^{23.3}$	$15290.5^{29.3}$
ME$^{(3)}$	$3335.2^{10.9}$	$4053.8^{16.1}$	$5367.7^{15.8}$	$9797.6^{22.3}$	$12100.3^{21.7}$	$15294.6^{28.3}$
ME$^{(4)}$	$3358.2^{14.9}$	$4061.3^{12.0}$	$5191.9^{18.5}$	$9834.9^{25.4}$	$12092.2^{22.6}$	$15270.7^{20.6}$
ME$^{(5)}$	$3360.6^{7.8}$	$4057.5^{12.2}$	$5209.2^{12.8}$	$9717.0^{23.2}$	$12095.3^{25.1}$	$15268.8^{27.5}$
RME$^{(2)}$	$3390.0^{8.1}$	$4085.3^{9.7}$	$5403.2^{10.2}$	$9837.9^{24.6}$	$12193.1^{23.5}$	$15363.0^{31.7}$
RME$^{(3)}$	$3392.0^{12.6}$	$4085.9^{9.8}$	$5405.1^{10.4}$	$9838.5^{25.0}$	$12192.3^{23.5}$	$15365.6^{31.4}$
RME$^{(4)}$	$3388.6^{8.3}$	$4080.7^{9.9}$	$5403.8^{10.2}$	$9849.5^{12.1}$	$12192.6^{23.4}$	$15364.3^{31.5}$
RME$^{(5)}$	$3391.9^{8.2}$	$4086.9^{10.9}$	$5405.5^{8.5}$	$9843.5^{25.0}$	$12194.2^{11.5}$	$15366.2^{12.7}$
GPVAE	$3417.6^{11.8}$	$4133.2^{13.3}$	$5410.0^{19.5}$	$9843.4^{23.8}$	$12184.0^{22.7}$	$15370.9^{27.6}$

mixture encoder (ME) also has difficulty in learning a good model, being quite sensitive to the initial parameters. Except for one case on CelebA with $\mathrm{dim}(\mathbf{z})=10$, it consistently underperforms our GPVAE. The blind mixture estimation can potentially suffer from collapsed mixture components and dominant single component issues. The fact that even the baseline VAE often performs comparably to the ME with different mixture orders supports this observation. This, again, signifies the importance of employing more incremental learning criteria, as done by our GPVAE's explicit modeling of the posterior deviation. The RME, by following this direction of adopting an incremental learning objective, performs equally well with our GPVAE on many cases, but slightly underperforms ours on the others.

Inference time. Compared to semi-amortized methods, our GP encoder is much faster as inference is accomplished by a single feed forward pass through the encoder network. Unlike the semi-amortized approaches where one has to perform the SVI gradient adaptation at test time, in our GP-VAE model, after the training stage, the posterior model is fixed, with no further adaptation required. To verify computational speed-up over the semi-amortized approaches and others, we measure the inference. The per-batch inference times (batch size 128) on all benchmark datasets are shown in Table 3. To report the results, for each method and each dataset, we run inference over the entire test set batches, measure the running time, then take the per-batch average. We repeat the procedure five times and report the average. All models are run on the same machine with a single GPU

Table 3. (Per-batch) Test inference time (unit in milliseconds) with batch size 128. The latent dimension $\mathrm{dim}(\mathbf{z})=50$.

	MNIST	OMNIG.	CIFAR10	SVHN	CELEBA
VAE	3.6	4.8	3.7	2.2	2.7
SA$^{(1)}$	9.7	11.6	9.8	7.0	8.4
SA$^{(2)}$	18.1	19.2	16.8	15.5	13.8
SA$^{(4)}$	32.2	34.4	27.9	30.1	27.1
SA$^{(8)}$	60.8	65.7	60.5	60.3	53.8
IAF$^{(1)}$	4.8	5.7	5.1	3.4	4.4
IAF$^{(2)}$	5.9	6.4	5.6	3.7	5.1
IAF$^{(4)}$	6.2	7.0	6.3	4.7	5.7
IAF$^{(8)}$	7.7	8.2	7.6	5.7	7.7
GPVAE	9.9	10.2	9.3	8.0	9.2

(RTX 2080 Ti), Core i7 3.50GHz CPU, and 128 GB RAM.

Note that we only report test times for the latent dimension $\mathrm{dim}(\mathbf{z})=50$ since the impact of the latent dimension appears to be less significant for all models except for our GPVAE. In the GPVAE, the latent dimension can considerably affect the inference time because of the matrix operations performed per latent dimension (c.f., (12)). Hence, we consider the most complex (worst) case for our GPVAE model, $\mathrm{dim}(\mathbf{z})=50$, the highest dimension in our experimental setup. Most notably yet as expected, the semi-amortized approach (SA) suffers from the computational overhead of test time gradient updates, with the inference time significantly growing as a function of the number of increasing updates. Our GPVAE is significantly faster than

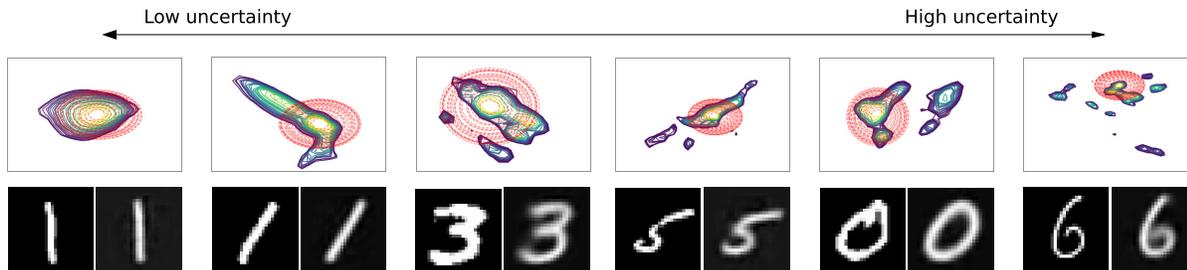Low uncertainty ⟵ ⟶ High uncertainty

Figure 1. Uncertainty vs. posterior approximation difficulty. After the GPVAE model is trained on MNIST with 2D latent space, we evaluate the uncertainty $\mathrm{Tr}\mathbb{V}(\mathbf{f}(\mathbf{x})|\mathcal{D}) + \mathrm{Tr}\mathbb{V}(\mathbf{h}(\mathbf{x})|\mathcal{D})$, and depict six different instances $\mathbf{x}$ in the order of increasing uncertainty values. Top panel shows the true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ (contour plots) and the base encoder $\mathcal{N}(\mathbf{b}(\mathbf{x}), \mathbf{c}(\mathbf{x})^2)$ (red dots) superimposed (in log scale). Bottom panel contains the original inputs $\mathbf{x}$ (left) and reconstructed images (right). For the cases with lower uncertainty, the true posteriors are more Gaussian-like. On the other hand, the higher uncertainty cases have highly non-Gaussian true posteriors with multiple modes.

Table 4. Test data log-likelihood scores on **Binary MNIST** with "CNN" and "FC" networks.

	VAE	SA$^{(1)}$	SA$^{(4)}$	IAF$^{(1)}$	IAF$^{(4)}$	HF$^{(1)}$	HF$^{(4)}$	ME$^{(2)}$	ME$^{(4)}$	VLAE$^{(2)}$	VLAE$^{(4)}$	RME$^{(2)}$	RME$^{(4)}$	GPVAE
CNN	-84.49	-83.64	-83.85	-83.37	-83.08	-83.82	-83.87	-83.77	-83.83	-	-	-83.14	-83.09	-83.18
FC	-85.38	-85.20	-85.43	-84.26	-84.03	-85.27	-85.22	-	-	-83.72	-83.73	-	-	-

the SA with more than one SVI step, albeit on par or slower than the flow-based IAF. We believe that the inference time of the GPVAE can be further improved by more effective implementations of the dimension-wise matrix operations[9].

Uncertainty vs. posterior approximation difficulty. Another important benefit of our Bayesian treatment is that we can quantify the *uncertainty* in posterior approximation. Recall that our GP posterior $p(\mathbf{f}, \mathbf{h}|\mathcal{D})$ captures the discrepancy between the base encoder $\mathcal{N}(\mathbf{b}(\mathbf{x}), \mathbf{c}(\mathbf{x})^2)$ and the true posterior $p_{\boldsymbol{\theta}}(\mathbf{x})$ via the GP noise processes $\mathbf{f}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$. In particular, the variance $\mathbb{V}(f(\mathbf{x})|\mathcal{D})$ (similarly for $\mathbf{h}$) at given input $\mathbf{x}$, can serve as an indicator that gauges the goodness of posterior approximation via a single Gaussian. For instance, the large posterior variance (*high uncertainty*) implies that the posterior approximation is *difficult*, suggesting the true posterior is distinct from a Gaussian (e.g., having multiple modes). On the other hand, if the variance is small (*low uncertainty*), one can anticipate that the true posterior might be close to a Gaussian. Table 1 illustrates this intuition on the MNIST with 2D latent space, where the uncertainty measured by $\mathrm{Tr}\mathbb{V}(\mathbf{f}(\mathbf{x})|\mathcal{D}) + \mathrm{Tr}\mathbb{V}(\mathbf{h}(\mathbf{x})|\mathcal{D})$ accurately aligns with non-Gaussianity of the true posterior, closely related to the quality of reconstruction. For verification, in Fig. 2 we visually
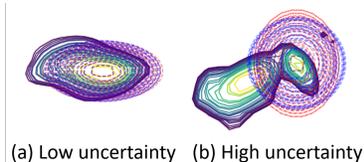

(a) Low uncertainty   (b) High uncertainty

Figure 2. True $p(z|x)$ (contour) and two approximates (blue/red).

[9]Our current implementation uses `for` loop to iterate matrix operations over latent dimensions, but can be potentially converted to block operations without a loop, possibly with parallelization.

contrast the marginalized $q(z|x)$ (blue) and the base encoder $\mathcal{N}(b(x), c(x)^2)$ (red) on the 2D latent MNIST case. Although they look similar, the marginalized one (blue) has a better corrected covariance toward the true posterior in case (b), which helps reducing the approximation error further.

Results on Binary MNIST. Performance on binarized input images is shown in Table 4. Our GPVAE performs well compared to IAF and RME although the differences among the competing approaches are not very pronounced compared to real-valued image cases.

6. Conclusions

We proposed a novel Gaussian process encoder model to significantly reduce the posterior approximation error of the amortized inference in VAE, while being computationally efficient. Our Bayesian treatment that regards the discrepancy in posterior approximation as a random noise process, leads to improvements in the accuracy of inference within the fast amortized inference framework. It also offers the ability to quantify the uncertainty (variance of stochastic noise) in variational inference, intuitively interpreted as inherent difficulty in posterior approximation. There could be more extensive experimental setups to be carried out, e.g., deeper network architectures, hierarchical models, or larger/structured latent spaces. In this work we aimed to focus on proving our concept mainly, and this was done with popular network architectures on benchmark datasets that we used. We leave this extensive empirical study as our future work. Another important future study not included in this paper is the rigorous analysis of the impact of the series of approximations that we used in our model.

References

[1] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders, 2016. In Proceedings of the Second International Conference on Learning Representations, ICLR. 4, 6

[2] Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, 2018. 1, 2, 5

[3] Erik Daxberger and José Miguel Hernández-Lobato. Bayesian variational autoencoders for unsupervised out-of-distribution detection. In *arXiv preprint*, 2019. 5

[4] Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks, 2018. In International Conference on Learning Representations. 3

[5] A. Dezfouli and E. V. Bonilla. Scalable inference for Gaussian process models with black-box likelihoods, 2015. In Advances in Neural Information Processing Systems. 3

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017. 1

[7] Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow Gaussian processes, 2019. In International Conference on Learning Representations. 3

[8] James Hensman, Nicolas Durrande, and Arno Solin. Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research*, 18:5537–5588, 2017. 3

[9] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 13:1303–1347, 2013. 1, 2

[10] Wenbing Huang, Deli Zhao, Fuchun Sun, Huaping Liu, and Edward Chang. Scalable Gaussian process regression using deep neural networks, 2015. Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI). 3

[11] Minyoung Kim and Vladimir Pavlovic. Recursive inference for variational autoencoders. *Advances in Neural Information Processing Systems*, 2020. 5, 6

[12] Y. Kim, S. Wiseman, A. C. Millter, D. Sontag, and A. M. Rush. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, 2018. 1, 2, 5

[13] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational inference with inverse autoregressive flow, 2016. In Advances in Neural Information Processing Systems. 1, 5

[14] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes, 2014. In Proceedings of the Second International Conference on Learning Representations, ICLR. 1, 2, 4, 5

[15] R. G. Krishnan, D. Liang, and M. D. Hoffman. On the challenges of learning with inference networks on sparse high-dimensional data. In *Artificial Intelligence and Statistics*, 2018. 1, 2, 5

[16] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images, 2009. Technical report, Computer Science Department, University of Toronto. 5

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks, 2012. In Advances in Neural Information Processing Systems. 5

[18] B. M. Lake, R. R. Salakhutdinov, and J. Tenenbaum. One-shot learning by inverting a compositional causal process, 2013. In Advances in Neural Information Processing Systems. 5

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 5

[20] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes, 2018. In International Conference on Learning Representations. 3

[21] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015. 5

[22] C. Lloyd, T. Gunter, M. A. Osborne, and S. J. Roberts. Variational inference for Gaussian process modulated Poisson processes. In *International Conference on Machine Learning*, 2015. 4

[23] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *International Conference on Machine Learning*, 2016. 5

[24] J. Marino, Y. Yisong, and S. Mandt. Iterative amortized inference. In *International Conference on Machine Learning*, 2018. 1, 2, 5

[25] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996. 3

[26] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 5

[27] Yookoon Park, Chris Kim, and Gunhee Kim. Variational Laplace autoencoders. In *International Conference on Machine Learning*, 2019. 1, 6

[28] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005. 3

[29] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *arXiv preprint*, 2015. 5

[30] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. 2

[31] D.J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models, 2014. International Conference on Machine Learning. 1, 5

[32] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs, 2006. In Advances in Neural Information Processing Systems. 3

[33] Shang-Yu Su, Shan-Wei Lin, and Yun-Nung Chen. Compound variational auto-encoder, 2019. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 5

[34] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *arXiv preprint*, 2013. 5

[35] M. K. Titsias. Variational learning of inducing variables in sparse Gaussian processes, 2009. In Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics. 3

[36] Michalis K. Titsias, Jonathan Schwarz, Alexander G. de G. Matthews, Razvan Pascanu, and Yee Whye Teh. Functional regularisation for continual learning with gaussian processes, 2020. International Conference on Learning Representations. 3

[37] J. M. Tomczak and M. Welling. Improving variational autoencoders using Householder flow, 2016. In Advances in Neural Information Processing Systems, Workshop on Bayesian Deep Learning. 1, 5

[38] Dustin Tran, Rajesh Ranganath, and David M. Blei. The Variational Gaussian Process, 2016. International Conference on Learning Representations. 5

[39] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning, 2016. AI and Statistics (AISTATS). 3