Supplementary Material
Gaussian Process Modeling of Approximate Inference Errors for Variational Autoencoders

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

1. Detailed Derivations for GP Inference and Learning

We provide detailed derivations for the variational inference for GP (Sec. 3.3 in the main paper). Specifically, we show that

$$\mathrm{KL}\big(q(\mathbf{W},\mathbf{U})||p(\mathbf{W},\mathbf{U}|\mathcal{D})\big) = \log \hat{p}_{\boldsymbol{\theta}}(\mathcal{D}) - \sum_{\mathbf{x}\in\mathcal{D}} \mathrm{ELBO}(\boldsymbol{\theta},\boldsymbol{\Lambda};\mathbf{x}), \tag{1}$$

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 the surrogate likelihood function $\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{W},\mathbf{U};\mathbf{x}) := \mathbb{E}_q\Big[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q(\mathbf{z}|\mathbf{x},\mathbf{W},\mathbf{U})}\Big]$, and

$$\mathrm{ELBO}(\boldsymbol{\theta},\boldsymbol{\Lambda};\mathbf{x}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\big[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\big] - \mathbb{E}_{q(\mathbf{W},\mathbf{U})}[\mathrm{KL}(q(\mathbf{z}|\mathbf{x},\mathbf{W},\mathbf{U})||p(\mathbf{z}))] - \frac{1}{N}\mathrm{KL}(q(\mathbf{W},\mathbf{U})||\mathcal{N}(\mathbf{0},\mathbf{I})). \tag{2}$$

Proof. Starting from the left hand side of (1),

$$\mathrm{KL}\big(q(\mathbf{W},\mathbf{U})||p(\mathbf{W},\mathbf{U}|\mathcal{D})\big) = \mathbb{E}_{q(\mathbf{W},\mathbf{U})}\Big[\log \frac{q(\mathbf{W},\mathbf{U})}{p(\mathbf{W},\mathbf{U}|\mathcal{D})}\Big] \tag{3}$$

$$= \mathbb{E}_{q(\mathbf{W},\mathbf{U})}\left[\log \frac{q(\mathbf{W},\mathbf{U})\,\hat{p}_{\boldsymbol{\theta}}(\mathcal{D})}{\mathcal{N}(\mathbf{W},\mathbf{U};\mathbf{0},\mathbf{I})\prod_{\mathbf{x}\in\mathcal{D}} e^{\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{W},\mathbf{U};\mathbf{x})}}\right] \tag{4}$$

$$= \log \hat{p}_{\boldsymbol{\theta}}(\mathcal{D}) + \mathrm{KL}(q(\mathbf{W},\mathbf{U})||\mathcal{N}(\mathbf{0},\mathbf{I})) - \sum_{\mathbf{x}\in\mathcal{D}} \mathbb{E}_{q(\mathbf{W},\mathbf{U})}\big[\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{W},\mathbf{U};\mathbf{x})\big] \tag{5}$$

$$= \log \hat{p}_{\boldsymbol{\theta}}(\mathcal{D}) + \mathrm{KL}(q(\mathbf{W},\mathbf{U})||\mathcal{N}(\mathbf{0},\mathbf{I})) - \sum_{\mathbf{x}\in\mathcal{D}} \Big(\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\big[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\big] - \mathbb{E}_{q(\mathbf{W},\mathbf{U})}\big[\mathrm{KL}(q(\mathbf{z}|\mathbf{x},\mathbf{W},\mathbf{U})||p(\mathbf{z}))\big]\Big). \tag{6}$$

For the last equality we use the GP posterior marginalized encoder distribution,

$$q(\mathbf{z}|\mathbf{x}) = \iint q(\mathbf{W},\mathbf{U})q(\mathbf{z}|\mathbf{x},\mathbf{W},\mathbf{U})d\mathbf{W}d\mathbf{U}. \tag{7}$$

Arranging the last equation completes the proof.

2. Experimental Setups and Network Architectures

For all optimization, we used the Adam optimizer with batch size 128 and learning rate 0.0005. We run the optimization until 2000 epochs. We use the same encoder/decoder architectures for competing methods including our GPVAE, VAE, SA (semi-amortized approach), and also the base density in the flow-based models IAF and HF. The network architectures are slightly different across the datasets due to different input image dimensions. We summarize the full network architectures in Table 1 (MNIST and OMNIGLOT), Table 2 (CIFAR10 and SVHN), and Table 3 (CelebA).

3. Additional Experiments on CIFAR10

The results on the CIFAR10 dataset are reported in Table 4.

Table 1. Encoder and decoder network architectures for MNIST and OMNIGLOT datasets. In the convolutional and transposed convolutional layers, the paddings are properly adjusted to match the input/output dimensions.

ENCODER	DECODER
INPUT: $(28 \times 28 \times 1)$	INPUT: $\mathbf{z} \in \mathbb{R}^p$ $(p \in \{10, 20, 50\})$
32 (4×4) CONV.; STRIDE 2; LEAKYRELU (0.01)	FC. 256; RELU
32 (4×4) CONV.; STRIDE 2; LEAKYRELU (0.01)	FC. $3 \cdot 3 \cdot 64$; RELU
64 (4×4) CONV.; STRIDE 2; LEAKYRELU (0.01)	32 (4×4) TRANSPOSED CONV.; STRIDE 2; RELU
FC. 256; LEAKYRELU (0.01)	32 (4×4) TRANSPOSED CONV.; STRIDE 2; RELU
FC. $2 \times p$ $(p = \text{DIM}(\mathbf{z}) \in \{10, 20, 50\})$	1 (4×4) TRANSPOSED CONV.; STRIDE 2

Table 2. Encoder and decoder network architectures for CIFAR10 and SVHN datasets.

ENCODER	DECODER
INPUT: $(32 \times 32 \times 3)$	INPUT: $\mathbf{z} \in \mathbb{R}^p$ $(p \in \{10, 20, 50\})$
32 (4×4) CONV.; STRIDE 2; LEAKYRELU (0.01)	FC. 512; RELU
32 (4×4) CONV.; STRIDE 2; LEAKYRELU (0.01)	FC. $4 \cdot 4 \cdot 64$; RELU
64 (4×4) CONV.; STRIDE 2; LEAKYRELU (0.01)	32 (4×4) TRANSPOSED CONV.; STRIDE 2; RELU
FC. 512; LEAKYRELU (0.01)	32 (4×4) TRANSPOSED CONV.; STRIDE 2; RELU
FC. $2 \times p$ $(p = \text{DIM}(\mathbf{z}) \in \{10, 20, 50\})$	3 (4×4) TRANSPOSED CONV.; STRIDE 2

Table 3. Encoder and decoder network architectures for CelebA dataset.

ENCODER	DECODER
INPUT: $(64 \times 64 \times 3)$	INPUT: $\mathbf{z} \in \mathbb{R}^p$ $(p \in \{10, 20, 50\})$
32 (4×4) CONV.; STRIDE 2; LEAKYRELU (0.01)	FC. 512; RELU
32 (4×4) CONV.; STRIDE 2; LEAKYRELU (0.01)	FC. $4 \cdot 4 \cdot 64$; RELU
64 (4×4) CONV.; STRIDE 2; LEAKYRELU (0.01)	64 (4×4) TRANSPOSED CONV.; STRIDE 2; RELU
64 (4×4) CONV.; STRIDE 2; LEAKYRELU (0.01)	32 (4×4) TRANSPOSED CONV.; STRIDE 2; RELU
FC. 512; LEAKYRELU (0.01)	32 (4×4) TRANSPOSED CONV.; STRIDE 2; RELU
FC. $2 \times p$ $(p = \text{DIM}(\mathbf{z}) \in \{10, 20, 50\})$	3 (4×4) TRANSPOSED CONV.; STRIDE 2

4. Importance Weighted Sampling Estimation (IWAE)

To report the test log-likelihood scores, $\log p(\mathbf{x})$, we used the importance weighted sampling estimation (IWAE) [1]. More specifically, IWAE $= \log \frac{1}{K} \sum_{i=1}^{K} \frac{p(\mathbf{x}, \mathbf{z}_i)}{q(\mathbf{z}_i|\mathbf{x})}$, where $\mathbf{z}_1, \ldots, \mathbf{z}_K$ are i.i.d. samples from $q(\mathbf{z}|\mathbf{x})$. It can be shown that IWAE lower bounds $\log p(\mathbf{x})$ and can be arbitrarily close to the target as the number of samples K goes large. We used $K = 100$ throughout the experiments.

5. Fully-Connected Decoder Networks

In the main paper we used the convolutional networks for both encoder and decoder models. This is a reasonable architectural choice considering that all the datasets are images. It is also widely believed that convolutional networks outperform fully connected networks for many tasks in the image domain [2, 4, 5]. One can alternatively consider fully connected networks for either the encoder or the decoder, or both. Nevertheless, with the equal number of model parameters, having both convolutional encoder and decoder networks always outperformed the fully connected counterparts. In this section

Table 4. (CIFAR10) Test log-likelihood scores (unit in nat) estimated by the importance weighted sampling [1] with 100 samples. 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). Best viewed in color.

	$\text{DIM}(\mathbf{z}) = 10$	$\text{DIM}(\mathbf{z}) = 20$	$\text{DIM}(\mathbf{z}) = 50$
VAE	$1645.7^{4.9}$	$2089.7^{5.8}$	$2769.9^{7.1}$
$\text{SA}^{(1)}$	$1645.0^{5.6}$	$2086.0^{6.2}$	$2765.0^{7.1}$
$\text{SA}^{(2)}$	$1648.6^{4.8}$	$2088.2^{6.6}$	$2764.1^{7.7}$
$\text{SA}^{(4)}$	$1648.5^{5.2}$	$2083.9^{8.4}$	$2766.7^{6.6}$
$\text{SA}^{(8)}$	$1642.1^{5.4}$	$2086.0^{6.1}$	$2766.6^{7.5}$
$\text{IAF}^{(1)}$	$1646.0^{4.9}$	$2081.1^{5.4}$	$2762.6^{7.2}$
$\text{IAF}^{(2)}$	$1642.0^{4.9}$	$2084.6^{5.6}$	$2763.0^{4.3}$
$\text{IAF}^{(4)}$	$1646.0^{5.1}$	$2083.2^{6.1}$	$2760.6^{7.0}$
$\text{IAF}^{(8)}$	$1643.6^{4.6}$	$2087.1^{4.6}$	$2761.8^{6.9}$
$\text{HF}^{(1)}$	$1644.5^{4.4}$	$2079.1^{5.5}$	$2757.9^{4.4}$
$\text{HF}^{(2)}$	$1636.7^{4.9}$	$2086.0^{5.9}$	$2764.7^{4.4}$
$\text{HF}^{(4)}$	$1642.1^{4.9}$	$2082.3^{7.3}$	$2763.4^{4.4}$
$\text{HF}^{(8)}$	$1639.9^{5.4}$	$2084.7^{6.1}$	$2765.5^{7.2}$
$\text{ME}^{(2)}$	$1643.6^{5.1}$	$2086.6^{6.8}$	$2767.9^{9.4}$
$\text{ME}^{(3)}$	$1638.6^{5.8}$	$2079.8^{5.9}$	$2770.2^{7.8}$
$\text{ME}^{(4)}$	$1641.8^{5.4}$	$2084.7^{6.9}$	$2763.5^{9.3}$
$\text{ME}^{(5)}$	$1641.7^{5.6}$	$2080.2^{5.9}$	$2766.1^{6.3}$
$\text{RME}^{(2)}$	$1652.3^{5.0}$	$2095.7^{5.8}$	$2779.6^{6.6}$
$\text{RME}^{(3)}$	$1654.2^{4.9}$	$2099.1^{7.2}$	$\mathbf{2783.0}^{6.1}$
$\text{RME}^{(4)}$	$1655.0^{6.4}$	$2096.6^{5.9}$	$\mathbf{2781.1}^{6.6}$
$\text{RME}^{(5)}$	$1654.5^{4.6}$	$2098.4^{5.8}$	$\mathbf{2782.9}^{6.4}$
GPVAE	$\mathbf{1668.6}^{5.6}$	$\mathbf{2113.0}^{6.7}$	$\mathbf{2786.5}^{8.2}$

we empirically verify this by comparing the test likelihood performance between the two architectures. We particularly focus on comparing two architectures (convolutional vs. fully connected) for the *decoder* model alone, while retaining the convolutional network structure of encoders for both cases.

Using the fully connected decoder network allows us to test the recent Laplacian approximation approach [3] (denoted by **VLAE**), which we excluded from the main paper. They use a first-order approximation solver to find the mode of the true posterior (i.e., linearizing the decoder function), and compute the Hessian of the log-posterior at the mode to define the (full) covariance matrix. This procedure is computationally feasible only for a fully connected decoder model. We conduct experiments on MNIST and OMNIGLOT datasets where the fully connected decoder network consists of two hidden layers and the hidden layer dimensions are chosen in a manner where the total number of weight parameters is roughly equal to the convolutional decoder network that we used in the main paper.

Table 5 summarizes the results. Among the fully connected networks, the VLAE achieves the highest performance. Instead of doing SVI gradient updates as in the SAVI method (SA), the VLAE aims to directly solve the mode of the true posterior by decoder linearization, leading to more accurate posterior refinement without suffering from the step size issue. Our GPVAE, with the fully connected decoder networks still improves the VAE's scores, but the improvement is often less than that of the VLAE. However, when compared to the convnet decoder cases, even the conventional VAE significantly outperforms the VLAE in the ability to represent the data (markedly higher log-likelihood scores). The best VLAE's scores are lower than even VAE's using convolutional decoder network. Restricted network architecture in the VLAE is its main drawback.

We also compare the test inference times of our GPVAE model and the VLAE using the fully connected decoder networks. Note that VLAE is a semi-amortized approach that needs to perform the Laplace approximation at test time. Thus, another drawback is the computational overhead for inference, which can be demanding as the number of linearization steps increases. The per-batch inference times (batch size 128) are shown in Table 6. For the moderate or large linearization steps (e.g., 4 or

Table 5. (Fully connected vs. convolutional decoder networks) Test log-likelihood scores (unit in nat). The figures without parentheses are the scores using the fully connected networks, whereas figures in the parentheses are the scores using the convolutional decoder networks. Both architectures have roughly equal number of weight parameters. The number of linearization steps in the VLAE is chosen among $\{1, 2, 4, 8\}$.

	MNIST		OMNIGLOT	
	$\text{DIM}(\mathbf{z}) = 10$	$\text{DIM}(\mathbf{z}) = 50$	$\text{DIM}(\mathbf{z}) = 10$	$\text{DIM}(\mathbf{z}) = 50$
VAE	563.6 (685.1)	872.6 (1185.7)	296.8 (347.0)	519.4 (801.6)
SA (1)	565.1 (688.1)	865.8 (1172.1)	297.6 (344.1)	489.0 (792.7)
SA (2)	565.3 (682.2)	868.2 (1176.3)	295.3 (349.5)	534.1 (793.1)
SA (4)	565.9 (683.5)	852.9 (1171.3)	294.8 (342.1)	497.8 (794.4)
SA (8)	564.9 (684.6)	870.9 (1183.2)	299.0 (344.8)	500.0 (799.4)
VLAE (1)	590.0	922.2	307.4	644.0
VLAE (2)	595.1	908.8	307.6	621.4
VLAE (4)	605.2	841.4	318.0	597.7
VLAE (8)	605.7	779.9	316.6	553.1
GPVAE	568.7 (693.2)	882.3 (1213.0)	302.5 (352.1)	612.5 (813.0)

Table 6. (Fully connected networks as decoders) Per-batch inference time (unit in milliseconds) with batch size 128. The figures without parentheses are the times using the fully connected networks, whereas figures in the parentheses are the times using the convolutional decoder networks.

	MNIST		OMNIGLOT	
	$\text{DIM}(\mathbf{z}) = 10$	$\text{DIM}(\mathbf{z}) = 50$	$\text{DIM}(\mathbf{z}) = 10$	$\text{DIM}(\mathbf{z}) = 50$
VLAE (1)	10.1	12.9	11.2	12.1
VLAE (2)	11.2	13.4	13.2	16.9
VLAE (4)	14.8	17.8	15.4	18.7
VLAE (8)	20.7	30.8	22.1	26.4
GPVAE	5.7 (5.9)	9.8 (9.9)	6.9 (6.9)	10.2 (10.2)

8), the inference takes significantly (about $2\times$ to over $3\times$) longer than that of our GPVAE (amortized method).

References

[1] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders, 2016. In Proceedings of the Second International Conference on Learning Representations, ICLR. 2, 3

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks, 2012. In Advances in Neural Information Processing Systems. 2

[3] Yookoon Park, Chris Kim, and Gunhee Kim. Variational Laplace autoencoders. In *International Conference on Machine Learning*, 2019. 3

[4] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *arXiv preprint*, 2015. 2

[5] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *arXiv preprint*, 2013. 2