Locally Most Powerful Bayesian Test for Out-of-Distribution Detection using Deep Generative Models

Keunseo Kim **Juncheol Shin** **Heeyoung Kim**
Department of Industrial and Systems Engineering, KAIST
{kkseo91, jczy0402, heeyoungkim}@kaist.ac.kr

Abstract

Several out-of-distribution (OOD) detection scores have been recently proposed for deep generative models, as the direct use of the likelihood threshold for OOD detection has shown to be problematic. In this paper, we propose a new OOD score based on a Bayesian hypothesis test called the locally most powerful Bayesian test (LMPBT). The LMPBT is locally most powerful in the sense that the alternative hypothesis (representative parameter for OOD sample) is specified so as to maximize the probability that the Bayes factor exceeds the evidence threshold in favor of the alternative hypothesis as long as the parameter specified under the alternative hypothesis is in a neighborhood of the parameter specified under the null hypothesis. That is, under this neighborhood parameter condition, the test with the proposed alternative hypothesis maximizes the probability of correct detection of OOD samples. We also propose numerical strategies for more efficient and reliable computation of the LMPBT for practical application to deep generative models. We evaluate the OOD detection performance of the LMPBT on various benchmark datasets and demonstrate its superior performance over existing OOD detection methods.

1 Introduction

In several real-world applications of deep learning models, detecting anomalous samples that significantly deviate from the distribution of the training data, that is, out-of-distribution (OOD) samples, is crucial for reliable decision making, and various OOD detection methods have been studied in this regard. Deep generative models [9, 8, 17] have also been investigated for OOD detection as an intuitive strategy owing to their ability to evaluate the likelihood of a test sample. However, recent studies have shown that deep generative models can assign higher likelihoods to OOD samples than in-distribution samples [12], causing the direct use of raw likelihoods for OOD detection to be problematic.

To overcome this difficulty, several OOD scores have been recently proposed for deep generative models and have shown effective OOD detection performance [15, 16, 18]. Ren et al. [15] proposed the use of a ratio of the likelihood obtained from a model trained using pure input data and that obtained from a background model trained using noise-perturbed input data as an OOD score. Xiao et al. [18] proposed a likelihood regret score, which can be calculated as the difference between the likelihood obtained with the optimized parameters for a test sample and that approximated by the VAE. Serrà et al. [16] proposed a penalized log-likelihood with an input complexity as an OOD score, where the input complexity can be computed as the normalized size of the compressed input image.

In this paper, we propose a new OOD detection method that is optimal in some sense in the framework of a Bayesian hypothesis test, inspired by a uniformly most powerful Bayesian test [7] and a locally

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

most powerful test [14]. The Bayesian hypothesis test is based on the ratio of the posterior odds that the alternative hypothesis is true, given the observed data [3]. We formulate the OOD detection problem as a Bayesian test with the null hypothesis that a test sample is an in-distribution sample and the alternative hypothesis that the test sample is an OOD sample. Specifically, the null hypothesis is specified so as to represent the model trained on in-distribution training set by the maximum likelihood estimation, as typically done in previous studies. Against this specific null hypothesis, we propose to specify the alternative hypothesis so as to represent the model trained on the expanded training set with the test sample. Then, we show that the test with our proposed alternative hypothesis is locally most powerful, in the sense that the alternative hypothesis is specified so as to maximize the probability that the Bayes factor exceeds the evidence threshold in favor of the alternative hypothesis among all alternative hypotheses that have the model parameters in a neighborhood of the model parameter specified under the null hypothesis. That is, under this neighborhood parameter condition, the test with the proposed alternative hypothesis maximizes the probability of correct detection of OOD samples. The proposed test is called a locally most powerful Bayesian test (LMPBT).

As we specify the alternative hypothesis of the LMPBT for a test sample so as to represent the model trained on the expanded training set with that test sample, we need to re-train the model for each test sample. However, the computational cost for this re-training is significant. To address this issue, we adopt the upweighting method [10]. Using this method, we analyze the influence of upweighting a test sample on the parameter change and estimate the new parameters without retraining.

However, there are computational issues that are encountered when the upweighting method is directly applied in practice. First, the upweighting method involves the Hessian matrix, which is assumed to have all positive eigenvalues. However, the loss function of deep generative models is known to be nonconvex, and it is infeasible to find the global optimum. Consequently, the Hessian can have negative eigenvalues. Second, the upweighting method requires the computation of the inverse of the Hessian matrix. However, deep generative models, which have a large number of parameters, require huge costs for the calculation of the Hessian and its inverse.

To address the aforementioned issues, we use a low-rank approximation [2, 20, 21, 13] of the Hessian. Then, we can ensure that the approximated Hessian has all positive eigenvalues and can significantly reduce the computational cost for calculating the inverse of the Hessian. We evaluate the performance of the LMPBT using deep generative models on benchmark datasets, and demonstrate more effective OOD detection performance than competing methods. In summary, our contributions are as follows:

- We propose a new OOD detection method, a locally most powerful Bayesian test (LMPBT), which maximizes the probability of correct detection of OOD samples under some conditions.
- We address the computational issues that are encountered when practically implementing the LMPBT for deep generative models.
- We evaluate the LMPBT using variational autoencoders (VAEs) [9] and Glows [8] on benchmark datasets and demonstrate state-of-the-art performance for OOD detection. The code for the LMPBT is available at https://github.com/keunseokim91/LMPBT.

2 Background: Deep generative models

In general, deep probabilistic generative models comprise two sub-models: a generative model and an inference model. By introducing a latent variable $\mathbf{z}$, the generative model defines the generative process of $\mathbf{x}$ from $\mathbf{z}$, and the inference model infers the distribution of $\mathbf{x}$ given $\mathbf{z}$. For example, in flow-based generative models, the generative and inference models are defined using an invertible function f, which is composed of a sequence of invertible functions, $f = f_1 \circ f_2 \circ ... \circ f_k$. The generative and inference processes between $\mathbf{x}$ and $\mathbf{z}$ can be expressed as

$$\mathbf{x} \overset{f_1}{\longleftrightarrow} \mathbf{h}_1 \overset{f_2}{\longleftrightarrow} \mathbf{h}_2 \cdots \overset{f_k}{\longleftrightarrow} \mathbf{z},$$

where the latent variable $\mathbf{z}$ has a tractable density $p_\theta(\mathbf{z})$. Then, the log probability density function of the model given a sample $\mathbf{x}$ is given by

$$\log p_\theta(\mathbf{x}) = \log p_\theta(\mathbf{z}) + \log \left| \det \left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = \log p_\theta(\mathbf{z}) + \sum_{i=1}^{k} \log \left| \det \left(\frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right) \right|, \tag{1}$$

where we define $\mathbf{h}_0 = \mathbf{x}$ and $\mathbf{h}_k = \mathbf{z}$.

In VAEs, the generative and inference processes are modeled using the encoder-decoder structure. We assume that $\mathbf{x}|\mathbf{z} \sim p_\theta(\mathbf{x}|\mathbf{z})$ and $\mathbf{z} \sim p(\mathbf{z})$, where $p_\theta(\mathbf{x}|\mathbf{z})$ is parameterized using deep neural networks with parameters θ and $p(\mathbf{z})$ is a tractable prior distribution of $\mathbf{z}$. The maximum likelihood estimate (MLE) of θ requires maximizing the marginal likelihood $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$. However, this computation is intractable, and instead, variational inference is used to derive the evidence lower bound (ELBO) as follows:

$$\log p_\theta(\mathbf{x}) \geq E_{q_\phi}(\mathbf{z}|\mathbf{x}) \left[\log p_\theta(\mathbf{x}|\mathbf{z}) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))\right] = \text{ELBO}, \tag{2}$$

where the variational posterior $g_\phi(\mathbf{z}|\mathbf{x})$ approximates the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ and is parameterized by deep neural networks with variational parameters ϕ, and D_{KL} is the Kullback-Liebler (KL) divergence. By Eq. (2), ELBO may be used as an approximation of the log-likelihood. Using the exact likelihood in Eq.(1) and the approximated likelihood in Eq.(2), respectively, flow-based generative models and VAEs are typically trained by minimizing the negative log-likelihood of the training dataset $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ as a training loss:

$$-\sum_{i=1}^{n} \log L(\theta|\mathbf{x}_i) = -\sum_{i=1}^{n} \log p_\theta(\mathbf{x}_i). \tag{3}$$

3 Locally Most Powerful Bayesian Test for OOD Detection

We formulate the OOD detection problem as a Bayesian hypothesis test. We compare the null hypothesis H_0 that a test sample $\mathbf{x}_t$ is an in-distribution sample and the alternative hypothesis H_1 that the test sample $\mathbf{x}_t$ is an OOD sample. Specifically, we consider simple null and alternative hypotheses as follows:

$$\begin{aligned} H_0 &: \theta = \theta_0, \\ H_1 &: \theta = \theta_1, \end{aligned} \tag{4}$$

where θ_0 and θ_1 are representative parameters of a deep generative model for in-distribution and OOD samples, respectively, where θ is a set of parameters from the parameter space Θ.

The Bayesian hypothesis test is based on the ratio of the posterior odds that the alternative hypothesis is true, given the observed data [3]. The posterior odds in favor of the alternative hypothesis equals the Bayes factor multiplied by the prior odds in favor of the alternative hypothesis. When the null and alternative hypotheses are both simple, the Bayes factor represents the ratio of the likelihoods of the data evaluated under the two hypotheses. The posterior odds for our hypothesis test in Eq.(4) for the test sample $\mathbf{x}_t$ can be computed as

$$\frac{P(H_1|\mathbf{x}_t)}{P(H_0|\mathbf{x}_t)} = \frac{L(\theta_1|\mathbf{x}_t)}{L(\theta_0|\mathbf{x}_t)} \times \frac{P(H_1)}{P(H_0)}, \tag{5}$$

where $P(H_i)$ is a prior probability for hypothesis H_i, and $L(\theta_1|\mathbf{x}_t)/L(\theta_0|\mathbf{x}_t)$ is the Bayes factor in favor of the alternative hypothesis, where $L(\theta_1|\mathbf{x}_t)$ and $L(\theta_0|\mathbf{x}_t)$ are the likelihoods of $\mathbf{x}_t$ under H_1 and H_0, respectively.

To complete the test, we need to specify θ_0 and θ_1, and the prior probabilities for H_0 and H_1. Typically, θ_0 is specified as the MLE, denoted by $\hat{\theta}$, that maximizes the log-likelihood function of the training dataset $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ as follows:

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log L(\theta|\mathbf{x}_i). \tag{6}$$

However, the specification of θ_1 is difficult under the lack of information on OOD samples. In fact, the previous likelihood-ratio based OOD detection methods [15, 16, 18] can be expressed using the hypothesis test in Eq.(4) with different specifications of θ_1. The OOD detection performance would be affected by the specification of θ_1, and thus it is important to choose θ_1 optimally.

3.1 Proposed OOD score

In this study, we propose a theoretically grounded way to specify θ_1 optimally in some sense. Specifically, we propose to specify θ_1 as the MLE of the expanded training set with the test sample

$\mathbf{x}_t$, denoted as $\hat{\theta}_t$, as follows:

$$\hat{\theta}_t = \underset{\theta}{\operatorname{argmax}} \left[\log L(\theta|\mathbf{x}_t) + \sum_{i=1}^{n} \log L(\theta|\mathbf{x}_i) \right]. \tag{7}$$

Then, for each test sample $\mathbf{x}_t$, we need to re-solve the optimization problem in Eq.(7) to obtain $\hat{\theta}_t$. However, re-training the model for each test sample requires excessive computational cost. Instead, we use a computational approximation for the solution of Eq.(7), which allows to obtain $\hat{\theta}_t$ easily using the already obtained $\hat{\theta}$. Specifically, we adopt the upweighting method [10] for the approximation. This method analyzes the influence of a certain training data point on the parameter change by upweighting the loss of that point. It is known that the influence of upweighing the loss of a training data point $\mathbf{x}$ by scale ϵ is given by

$$\frac{d\hat{\theta}_t}{d\epsilon}|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_\theta(-\log L(\hat{\theta}|\mathbf{x})), \tag{8}$$

where $H_{\hat{\theta}} = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta^2(-\log L(\hat{\theta}|\mathbf{x}_i))$ is the Hessian at $\hat{\theta}$ and $\nabla_\theta(-\log L(\hat{\theta}|\mathbf{x}))$ is the loss gradient at $\mathbf{x}$ with respect to θ. In our case, the training dataset X_n does not contain the test sample $\mathbf{x}_t$, and thus the weight of $\mathbf{x}_t$ in the training loss in Eq.(3) can be regarded as 0. Then, adding the loss of $\mathbf{x}_t$ into the training loss for X_n is equivalent to upweighting the loss of $\mathbf{x}_t$ by $1/n$. Finally, $\hat{\theta}_t$ can be approximated as

$$\hat{\theta}_t \approx \tilde{\theta}_t = \hat{\theta} - \frac{1}{n} H_{\hat{\theta}}^{-1} \nabla_\theta(-\log L(\hat{\theta}|\mathbf{x}_t)). \tag{9}$$

Finally, with the specification of $\theta_0 = \hat{\theta}$ and $\theta_1 = \tilde{\theta}_t$ with the additional assumption of equal prior probabilities for H_0 and H_1, i.e., $P(H_1) = P(H_2) = 0.5$, we propose the log of the posterior odds in Eq.(5) as an OOD score for test sample $\mathbf{x}_t$, denoted as $S(\mathbf{x}_t)$, as follows:

$$S(\mathbf{x}_t) = -\log L(\hat{\theta}|\mathbf{x}_t) + \log L(\tilde{\theta}_t|\mathbf{x}_t). \tag{10}$$

In the next section, we show that the test with the proposed OOD score in Eq.(10) maximizes the probability that the alternative hypothesis is accepted when it is true (i.e., the probability of correct detection of OOD sample) as long as the parameter specified under the alternative hypothesis is in a neighborhood of the parameter specified under the null hypothesis.

3.2 Locally most powerful Bayesian test for OOD detection

Inspired by a uniformly most powerful Bayesian test [7] and a locally most powerful test [14], we first define a locally most powerful Bayesian test (LMPBT) for OOD detection.

Definition 3.1 (Locally most powerful Bayesian test (LMPBT) for OOD detection)**.** Consider the OOD detection test in Eq. (4) for test sample $\mathbf{x}_t$. Let θ_* be the true parameter associated with $\mathbf{x}_t$. A locally most powerful Bayesian test for evidence threshold $\gamma > 0$, denoted by LMPBT(γ), in favor of the alternative hypothesis $H_1 : \theta = \theta_1$ against a fixed null hypothesis $H_0 : \theta = \theta_0$, is a Bayesian hypothesis test in which the Bayes factor for the test satisfies the following inequality for any $\theta_* \in \Theta$ and all alternative hypotheses $H_1' : \theta = \theta_1'$, under the condition $0 < d(\theta_1', \theta_0) < \delta$, where δ is a sufficiently small constant and d is a measure of the distance:

$$P_{\theta_*}\left[\frac{L(\theta_1|\mathbf{x}_t)}{L(\theta_0|\mathbf{x}_t)} > \gamma\right] \geq P_{\theta_*}\left[\frac{L(\theta_1'|\mathbf{x}_t)}{L(\theta_0|\mathbf{x}_t)} > \gamma\right]. \tag{11}$$

That is, with a fixed the null hypothesis $H_0 : \theta = \theta_0$, LMPBT(γ) is a Bayesian test in which the alternative hypothesis $H_1 : \theta = \theta_1$ is specified so as to maximize the probability that the Bayes factor $L(\theta_1|\mathbf{x}_t)/L(\theta_0|\mathbf{x}_t)$ exceeds the evidence threshold γ for all possible values of the true parameter θ_* associated with test sample $\mathbf{x}_t$, under the condition that θ_1 is in a neighborhood of θ_0.

Proposition 1. *The test using the OOD score in Eq.*(10) *is LMPBT(γ), given a sufficiently large training dataset.*

Proof. We prove that in our OOD detection test with the specification of $\theta_0 = \hat{\theta}$ and $\theta_1 = \tilde{\theta}_t$ for test sample $\mathbf{x}_t$, the Bayes factor $L(\tilde{\theta}_t|\mathbf{x}_t)/L(\hat{\theta}|\mathbf{x}_t)$ satisfies the inequality in Eq.(11) by showing that

the Bayes factor in favor of the alternative $H_1 : \theta = \theta_1$, i.e., $L(\theta_1|\mathbf{x}_t)/L(\hat{\theta}|\mathbf{x}_t)$, is maximized when $\theta_1 = \tilde{\theta}_t$. Under the condition that θ_1 is in a neighborhood of $\theta_0 = \hat{\theta}$, we can use the Taylor expansion of $\log L(\theta_1|\mathbf{x}_t)$ at $\hat{\theta}$ as follows:

$$\log L(\theta_1|\mathbf{x}_t) \approx \log L(\hat{\theta}|\mathbf{x}_t) + (\theta_1 - \hat{\theta})^T \nabla_\theta \log L(\hat{\theta}|\mathbf{x}_t) + \frac{1}{2}(\theta_1 - \hat{\theta})^T(-H_{\hat{\theta}})(\theta_1 - \hat{\theta}) + R(\theta_1),$$

where $R(\theta_1)$ is the remainder term, which has an upper bound $\frac{M}{6}||\theta_1 - \hat{\theta}||^3$ (Lagrange error bound [6]), where M is an upper bound of the third derivative of the log-likelihood for $||\theta_1 - \hat{\theta}|| < \delta$. With sufficiently small δ, the first- and second-terms will dominate the remainder term because higher order terms vanish much more rapidly, and the remainder term will be negligible. Then, the log-Bayes factor can be approximated as

$$\log L(\theta_1|\mathbf{x}_t) - \log L(\hat{\theta}|\mathbf{x}_t) \approx (\theta_1 - \hat{\theta})^T \nabla_\theta \log L(\hat{\theta}|\mathbf{x}_t) + \frac{1}{2}(\theta_1 - \hat{\theta})^T(-H_{\hat{\theta}})(\theta_1 - \hat{\theta}). \quad (12)$$

To obtain θ_1 that maximizes the log-Bayes factor, we find θ_1 that makes the gradient of Eq. (12) equal to 0. That is,

$$\nabla_\theta[\log L(\theta_1|\mathbf{x}_t) - \log L(\hat{\theta}|\mathbf{x}_t)] \approx \nabla_\theta \log L(\hat{\theta}|\mathbf{x}_t) + (\theta_1 - \hat{\theta})^T(-H_{\hat{\theta}}) = 0. \quad (13)$$

Then, the solution of Eq. (13) is obtained as $(\theta_1 - \hat{\theta}) = -H_{\hat{\theta}}^{-1}\nabla_\theta(-\log L(\hat{\theta}|\mathbf{x}_t))$. To ensure the condition $0 < d(\theta_1, \hat{\theta}) < \delta$ for sufficiently small δ, we re-scale $(\theta_1 - \hat{\theta})$ with a sufficiently small constant ϵ as $(\theta_1 - \hat{\theta}) = -\epsilon H_{\hat{\theta}}^{-1}\nabla_\theta(-\log L(\hat{\theta}|\mathbf{x}_t))$. By setting $\epsilon = \frac{1}{n}$ with sufficiently large n, we can obtain the desired result:

$$\theta_1 = \hat{\theta} - \frac{1}{n}H_{\hat{\theta}}^{-1}\nabla_\theta(-\log L(\hat{\theta}|\mathbf{x}_t)) = \tilde{\theta}_t.$$

□

We can see in Definition 3.1 that the Bayesian hypothesis test requires a predefined evidence threshold γ. Practically, we can set the threshold as the specified (e.g., fifth) percentile in the distribution of the LMPBT score evaluated on the training set. Specifically, we can obtain the empirical distribution of the LMPBT score using "in-distribution" training samples. Then, we set a specified percentile in the distribution of the LMPBT score as the threshold, depending on the desired false positive rate. For example, we can set the fifth percentile in the distribution of the LMPBT score as the threshold, according to the fact that the false positive rate of 0.05 is typically used as a default value in practice.

4 Computational issues on LMPBT

4.1 Low-rank approximation of the Hessian

In using the upweighting method in Eq.(9), $H_{\hat{\theta}}$ is assumed to be positive definite, that is, $H_{\hat{\theta}}$ is assumed to have all positive eigenvalues. However, the loss function of deep neural networks is known to be non-convex, and it is infeasible to find the global optimum. In fact, gradient-descent (GD) methods that are generally used for deep generative models are only ensured to find first-order stationary points, including saddle points [19]. Theoretically, GD methods are always able to escape saddle points, but they can take an exponentially long time [4]. In practice, $\hat{\theta}$ obtained by running GD methods with early stopping differs from the global optimum and could be a saddle point. As a result, $H_{\hat{\theta}}$ could have negative eigenvalues.

To handle this issue, we use a low-rank approximation of the Hessian using eigenvalue decomposition [13]. Recent empirical studies show that the Hessians of neural networks are typically of a low rank and often have at least one negative eigenvalue [1]. We empirically demonstrate this phenomenon in deep generative models in Figure 1, which shows the distribution of eigenvalues of $H_{\hat{\theta}}$ calculated from a Glow trained on the CIFAR-10 dataset (left) and a VAE trained on the CIFAR-10 dataset (right). We can see that most eigenvalues are zeros, which indicates that the Hessian is of a low rank. We can also see that the Hessian has negative eigenvalues. Benefiting from the low-rank property of

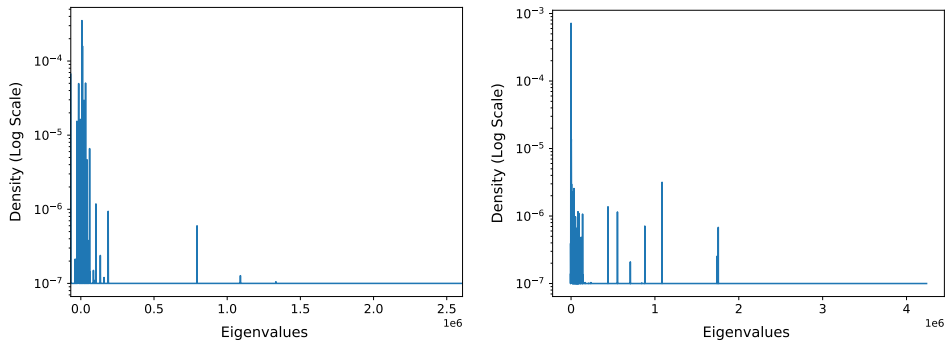

Figure 1: Distribution of eivenvalues of $H_{\hat{\theta}}$ calculated from a Glow trained on CIFAR-10 dataset (left) and a VAE trained on CIFAR-10 dataset (right). To compute the full empirical spectral density of the eigenvalues, we used the stochastic Lanczos quadrature algorithm [20]. It is notable that most eigenvalues are centered around zero, and negative eigenvalues exist.

the Hessian of deep generative models, we approximate the Hessian using a low-rank approximation with top r eigenvalues. Specifically, let the eigenvalue decomposition of the Hessian be given by

$$H = U\Gamma U^T = \sum_{i=1}^{p} \gamma_i u_i u_i^T, \tag{14}$$

where p is the number of parameters, eigenvalues γ_i are sorted with respect to their magnitudes, that is, $\gamma_i > \gamma_j$ for $i < j$, and u_i are the corresponding eigenvectors. A low-rank approximation of the Hessian of rank r is

$$\tilde{H} = U_r \Gamma_r U_r^T = \sum_{i=1}^{r} \gamma_i u_i u_i^T. \tag{15}$$

In practice, we need to find the appropriate value of the hyperparameter r for each experiment. In our experiments in Section 5.2, we investigate the change in the performance of the LMPBT with different specifications of r. Even the models with millions of parameters exhibited effective OOD performance using a low-rank approximation of the Hessian with only top 20$\sim$30 eigenvalues.

4.2 Scalability

The computation of the Hessian and its inverse requires costs proportional to the square and cube of the model parameter size, respectively. For deep generative models with a large number of parameters, the computation of Eq. (9), which involves the inverse of the Hessian, is not feasible. We can overcome this issue by computing the Hessian-vector product instead of forming the Hessian matrix explicitly. The Hessian-vector product can be computed as follows [20]:

$$\frac{\partial g_\theta^T z}{\partial \theta} = \frac{\partial g_\theta^T}{\partial \theta} z + g_\theta^T \frac{\partial z}{\partial \theta} = \frac{\partial g_\theta^T}{\partial \theta} z = \mathbf{Hz}, \tag{16}$$

where $g_\theta = \nabla_\theta(-\log L(\theta|\cdot))$ is the gradient with respect to θ, and $\mathbf{z}$ is a random vector. In Eq. (16), the first equality comes from the chain rule, the second equality comes from the independence of $\mathbf{z}$ and θ, and the third equality comes from the definition of the Hessian. It is notable that the Hessian-vector product only requires a cost that is the same as that of one gradient back-propagation. Then, with the Hessian-vector product results, the top r eigenvalues and the corresponding eigenvectors of the Hessian, and hence $\tilde{H}$ in Eq. (15), can be easily obtained using power iteration [20].

Another advantage of using the low-rank approximation of the Hessian in Eq. (15) is that we can easily compute Eq. (9) instead of directly computing the inverse of the Hessian, by utilizing the following:

$$\tilde{H}_{\hat{\theta}}^{-1} \nabla_\theta(-\log L(\hat{\theta}|\mathbf{x}_t)) = -\sum_{i=1}^{r} \frac{1}{\gamma_i} u_i u_i^T \nabla_\theta \log L(\hat{\theta}|\mathbf{x}_t). \tag{17}$$

Similar to the Hessian-vector product, Eq. (17) can be computed without computing the outer product $u_i u_i^T$, and it only requires the computational cost proportional to the model parameter size.

5 Experiments

In this section, we evaluate the performance of the LMPBT. We show that the LMPBT yields superior OOD detection performance than those of competing OOD detection methods on various benchmark datasets.

5.1 Experimental setup

Datasets We used six benchmark image datasets, namely MNIST, FASHION-MNIST (FMNIST), CelebA, CIFAR-10, CIFAR-100, and SVHN, and two synthetic datasets, namely Noise and Constant. To generate the Noise and Constant datasets, we adopted the sampling methods employed in [18]. We resized the images such that their width and height were both 32 pixels. In the experiments, we used three datasets, FMNIST, CIFAR-10, and CIFAR-100, as the training datasets. Specifically, we used the training partition of these datasets for training. In the experiments using FMNIST, three-channel images were converted to grayscale to match the number of channels with one-channel FMNIST images. In the experiments using CIFAR-10 and CIFAR-100, one-channel images were duplicated three times to match the number of channels with three-channel CIFAR-10 and CIFAR-100 images. After training deep generative models on FMNIST, CIFAR-10, and CIFAR-100 datasets, we used the test partition of other datasets for testing to evaluate the OOD detection performance.

Base deep generative models We selected two representatives of deep generative models, VAEs [9] and Glows [8], to evaluate the performance of the LMPBT. For training the VAE, we used the Adam optimizer for maximizing the ELBO in Eq. (2). For the Glow, we adopted the structure used in [8] and used the Adam optimizer to maximize the log-likelihood in Eq. (1). To approximate the Hessian, we used the top 20 eigenvalues for the Glow and top 30 eigenvalues for the VAE. We discuss the specification of the number of eigenvalues in more detail the following section. In the VAE experiments, we fixed the decoder and calculated the Hessian with respect to the encoder. The experimental settings and network structures are described in the supplementary material.

Metrics To quantitatively evaluate the OOD detection performance, we employed the Area Under the Receiver Operating Characteristic curve (AUROC), Area Under the Precision-Recall curve (AUPR) [5], and False Positive Rate at 80% True Positive Rate (FPR80) as evaluation metrics [15].

Baseline OOD detection methods for comparison We employed deep generative model-based OOD detection methods as baselines for comparison. These include the likelihood regret (LR) score [18], the likelihood ratio (LLR) score [15], and the input complexity (IC) method [16], discussed in Section 1. These OOD detection methods were evaluated on both Glows and VAEs. For VAEs, we re-implemented the experiments according to the experimental settings in [18]. For Glows, we also re-implemented the experiments according to the experimental settings in the original papers of LLR [15] and IC [16]. Specifically, we used the PNG compressor to measure the input complexity when computing the IC. When computing the LR score for each test sample, we started with the trained encoder of a VAE or the trained Glow, and optimized its parameters for 100 steps.

5.2 Sensitivity and computational cost analysis for LMPBT

Before the sensitivity analysis, we first visualize the OOD detection ability of the LMPBT, similar to [12]. Figure 2 (left) shows the histogram of the LMPBT scores when a VAE trained on CIFAR-10 was tested on CIFAR-10 as an in-distribution dataset and SVHN as an OOD dataset. We can see that if the LMPBT scores are higher, the corresponding test samples are more likely to be classified as OOD samples. As desired, SVHN test samples (i.e., OOD samples) tended to be assigned higher LMPBT scores than CIFAR-10 test samples.

Specification of low-rank approximation Figure 2 (b) shows the OOD detection performance (AUROC, AUPR, and FPR80) of the LMPBT according to the number of eigenvalues used for a low-rank approximation of the Hessian. The experiments were conducted using a VAE trained on CIFAR-10 and tested on CIFAR-10 and SVHN. Although the model has 1.4 million parameters, it achieved the best performance using only the top 30 eigenvalues. Even in the case with less than 10

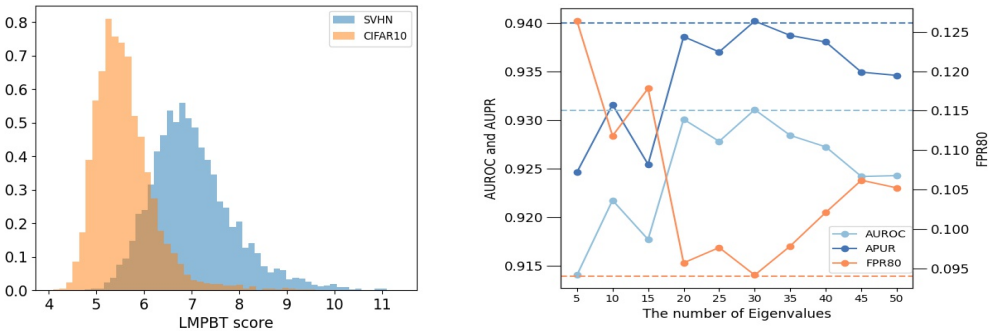

Figure 2: (left) LMPBT scores obtained with a VAE trained on CIFAR-10 and tested on both CIFAR-10 and SVHN. (right) The OOD detection performance (AUROC, AUPR, and FPR80) versus the number of eigenvalues.

eigenvalues, the LMPBT effectively detected OOD samples (AUROC : 0.913). This could be possible because the Hessian was sparse. Interestingly, the performance does not increase monotonically with the number of eigenvalues. This behavior may be explained by an identifiability issue of overparameterized models. For overparameterized models, there are often “degenerate" directions in the parameter space [11], which are corresponding to the smallest eigenvalues of the Hessian. The change in the parameters in degenerate directions leads to almost no difference in the outputs and causes an identifiability issue. Deep generative models also can suffer from this identifiability issue. However, we may overcome this issue by using a low-rank approximation of the Hessian with the largest eigenvalues. Because changes in the output values in the degenerate directions can be considered like random noise, the test accuracy of a compressed model can sometimes be better than that of the full model.

Computation cost The LMPBT requires the computation of the gradient and Hessian-vector product for each test sample, whereas the LLR and IC only require the likelihood computation. On the other hand, the LR requires several iterations for optimization for each test sample. To determine whether the computation cost of the LMPBT is acceptable, we compared the computation cost of the LMPBT with that of the LR. In our experiments using GeForce RTX 2080 GPU and a VAE trained on CIFAR-10, on average, the LMPBT and LR spent 0.11s and 0.30s to test a single sample, respectively. This demonstrates that the computation cost of the LMBPT is acceptable, although it is higher than those of the LLR and IC, which spent 0.0125 s and 0.0128 s on average to test a single sample, respectively.

Moreover, to evaluate the computational cost for computing top r Hessian eigenvectors, we performed two sets of additional experiments. First, we compared the computational cost required for the Hessian eigenvalue decomposition with that for network parameter estimation, which can be considered as a baseline because network parameter estimation is equally performed for other OOD detection methods. In our additional experiments using FMNIST ($1\times32\times32$) with VAEs, it took 2543 s to obtain the top 30 eigenvectors. This cost is relatively small (approximately 18$\%$) compared with the cost for network parameter estimation (13448 s).

Second, we evaluated the scalability of Hessian eigenvalue decomposition. Specifically, we measured the computational cost for the eigenvalue decomposition of the Hessian (top 30 eigenvectors) for datasets of various sizes with VAEs. We also calculated the ratio of time cost to the number of parameters to see the relationship between the dimension of the Hessian and computational cost. The results are in the Table 1. The ratio was obtained as 0.0028, 0.0023, and 0.0025 s for various data sizes. The ratio remained similar, which indicates that the computational cost tends to increase linearly with the number of parameters. These results show that our method is scalable and can be applicable to large-scale networks.

Table 1: Computational costs for Hessian eigenvalue decomposition

Data size	$1\times16\times16$	$1\times32\times32$	$3\times32\times32$
Number of parameters	504,904	1,074,760	1,337,928
Time cost	1,414 s	2,543 s	3,371 s
Time/Parameter	(0.0028 s/dim)	(0.0023 s/dim)	(0.0025 s/dim)

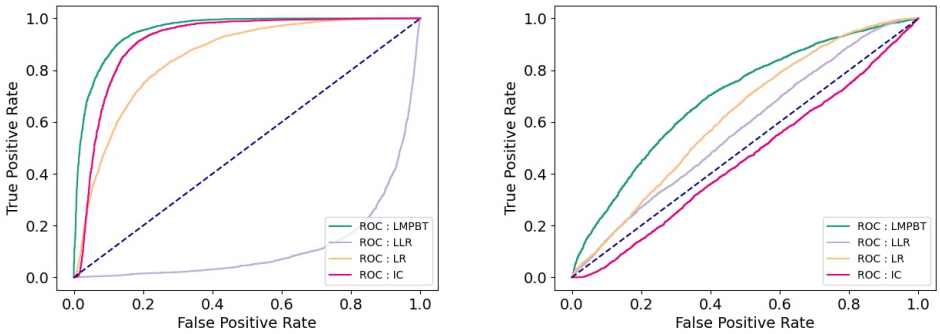

Figure 3: Comparison of ROC curves of using the LMPBT, LLR, IC, and LR for OOD detection using a VAE trained on CIFAR-100 training dataset and (a) tested on CIFAR-100 test dataset as in-distribution samples and SVHN test dataset as OOD samples; (b) tested on CIFAR-100 test dataset as in-distribution samples and CelebA test dataset as OOD samples.

5.3 Performance comparison with other OOD scores

To evaluate the LMPBT, we compared the ROC curve of using the LMPBT with those of using the LR, IC, and LLR in OOD detection using a VAE trained on CIFAR-100 and tested on CIFAR-100 and SVHN in Figure 3 (a) and tested on CIFAR-100 and CelebA in Figure 3 (b). In Figure 3 (a), the LBPBT (green line) showed higher true positive rates at all false positive rates than all other methods, including the IC (red purple line), which previously achieved state-of-the-art performance [18]. That is, the LMPBT universally outperformed the IC, regardless of the threshold for OOD detection. In contrast, the LLR (violet line) showed inferior performance than random guessing (blue line). This confirms that the LLR is not suitable for OOD detection when a VAE is used [18]. Similarly, in Figure 3 (b) using the CelebA dataset, the LMPBT achieved superior OOD detection performance than all other methods. All scores showed inferior performance compared to the case using the SVHN dataset. However, the LMPBT still outperformed the other methods even in this difficult OOD task.

For an in-depth evaluation of the LMPBT, we measured its performance using the AUROC and compared it with those of competing methods, LLR, IC, and LR. We report the average AUROC over five repeated experiments using different random seeds. In all experiments, the standard error of AUROC was less than 0.001. A higher AUROC value indicates better performance. Tables 2 (a) and 2 (b) present the results of VAEs trained on CIFAR-10 and CIFAR-100, respectively, and tested on various datasets including Noise, Constant, FMNIST, SVHN, MNIST and CelebA. We can see that the LMPBT achieved state-of-the-art performance for all the cases. In particular, in the case of the model trained on CIFAR-10 and tested on SVHN, the LMPBT outperformed (AUROC: 0.931) the IC (AUROC: 0.912), which previously showed state-of-the-art performance [18]. Even for the CelebA test dataset, a more complicated dataset, the LMPBT using the model trained on CIFAR-10 showed superior performance (AUROC: 0.783) than the LR (AUROC: 0.714), which previously showed state-of-the-art performance [18]. Similarly, the LMBPT showed significantly superior performance when the model was trained on CIFAR-100, a more complicated dataset. For example, the LMPBT using the model trained on CIFAR-100 showed superior performance (AUROC: 0.683) than the LR (AUROC: 0.598).

Table 2: Average AUROC of OOD detection methods on various test datasets over five repeated experiments using different random seeds.

(a) Trained on CIFAR-10

Dataset	LMPBT	LLR	IC	LR
Noise	**1.0**	1	0.032	0.994
Constant	**1.0**	0.258	1	0.974
FMNIST	**0.994**	0.074	0.992	0.991
SVHN	**0.931**	0.193	0.912	0.875
MNIST	**0.998**	0.008	0.994	0.998
CelebA	**0.783**	0.465	0.641	0.714

(b) Trained on CIFAR-100

Dataset	LMPBT	LLR	IC	LR
Noise	**1**	1	0.993	0.993
Constant	**1**	0.042	0.999	0.948
FMNIST	**0.997**	0.325	0.987	0.970
SVHN	**0.956**	0.123	0.912	0.820
MNIST	**0.998**	0.376	0.979	0.995
CelebA	**0.683**	0.575	0.464	0.598

We also present the results evaluated with AUPR and FPR80 in the supplementary material. Furthermore, we present the results using FMNIST as a training dataset with various test datasets considered in Table 2 in the supplementary material. Furthermore, we repeated the entire set of experiments using Glows, and present the results in the supplementary material. In summary, the results in the supplementary material show that the LMPBT consistently outperformed across different metrics and different datasets.

6 Conclusion

In this study, we proposed a new OOD score for deep generative models based on a Bayesian hypothesis test. We proposed the LMPBT, which maximizes the probability that the alternative hypothesis is accepted when it is true among all alternative hypotheses that have the model parameters in a neighborhood of the model parameter specified under the null hypothesis. The LMPBT effectively performed OOD detection on all the tasks that we evaluated. We also addressed practical computational issues in the implementation of the LMPBT for deep generative models. For future research, more efficient computation of the LMPBT could be studied; e.g., recent second-order optimization methods could be used for computing the inverse of the Hessian matrix or its approximation. The proposed method can support reliable decision making by automatically detecting anomalies in complex systems in a variety of real-world problems, e.g., manufacturing system monitoring, fraud detection, and disease surveillance. However, there is also a potential risk that the proposed method could be used as a tool to identify and discriminate against minorities. To prevent this, we need to ensure that the proposed method cannot be used for any purpose that could have negative social impacts.

Acknowledgements

This work was supported by Samsung Electronics Co., Ltd (IO201209-07871-01) and the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (2018R1C1B6004511).

References

[1] G. Alain, N. L. Roux, and P.-A. Manzagol. Negative eigenvalues of the hessian in deep neural networks. *arXiv preprint arXiv:1902.02366*, 2019.

[2] N. Andrei. Accelerated conjugate gradient algorithm with finite difference hessian/vector product approximation for unconstrained optimization. *Journal of Computational and Applied Mathematics*, 230(2):570–582, 2009.

[3] J. M. Bernardo and R. Rueda. Bayesian hypothesis testing: A reference approach. *International Statistical Review*, 70(3):351–372, 2002.

[4] S. S. Du, C. Jin, J. D. Lee, M. I. Jordan, A. Singh, and B. Poczos. Gradient descent can take exponential time to escape saddle points. In *Advances in neural information processing systems*, pages 1067–1077, 2017.

[5] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[6] D. Hughes-Hallett, A. M. Gleason, and W. G. McCallum. Calculus: Single and multivariable, 1998.

[7] V. E. Johnson. Uniformly most powerful bayesian tests. *Annals of statistics*, 41(4):1716, 2013.

[8] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224, 2018.

[9] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[10] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017.

[11] W. J. Maddox, G. Benton, and A. G. Wilson. Rethinking parameter counting in deep models: Effective dimensionality revisited. *arXiv preprint arXiv:2003.02139*, 2020.

[12] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. Do deep generative models know what they don't know? *arXiv preprint arXiv:1810.09136*, 2018.

[13] T. O'Leary-Roseberry, N. Alger, and O. Ghattas. Low rank saddle free newton: Algorithm and analysis. *arXiv preprint arXiv:2002.02881*, 2020.

[14] M. Omelka. The behavior of locally most powerful tests. *Kybernetika*, 41(6):699–712, 2005.

[15] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. Depristo, J. Dillon, and B. Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, pages 14707–14718, 2019.

[16] J. Serrà, D. Álvarez, V. Gómez, O. Slizovskaia, J. F. Núñez, and J. Luque. Input complexity and out-of-distribution detection with likelihood-based generative models. *arXiv preprint arXiv:1909.11480*, 2019.

[17] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29: 4790–4798, 2016.

[18] Z. Xiao, Q. Yan, and Y. Amit. Likelihood regret: An out-of-distribution detection score for variational auto-encoder. *arXiv preprint arXiv:2003.02977*, 2020.

[19] Z. Yao, P. Xu, F. Roosta-Khorasani, and M. W. Mahoney. Inexact non-convex newton-type methods. *arXiv preprint arXiv:1802.06925*, 2018.

[20] Z. Yao, A. Gholami, K. Keutzer, and M. Mahoney. Pyhessian: Neural networks through the lens of the hessian. *arXiv preprint arXiv:1912.07145*, 2019.

[21] Z. Yao, A. Gholami, S. Shen, K. Keutzer, and M. W. Mahoney. Adahessian: An adaptive second order optimizer for machine learning. *arXiv preprint arXiv:2006.00719*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 1.
   (b) Did you describe the limitations of your work? [Yes] See Section 5.2 and Section 6. We discussed the limitations about the computation cost.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3. We provided the full set of assumptions in detail.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section 3.2. We provided complete proof of the proposition.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Section 5.1 and the supplemental material. We explain the dataset in Section 5.1 and submit the code as the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5.1. We specify the data splits and choice of the hyperparameters in detail.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 5.2. We repeated five experiments and reported the standard error.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5.2. We repeated the computation time and specified the type of GPUs.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We used PyHessian library and cited the "Z. Yao, A. Gholami, K Keutzer, M. Mahoney. PyHessian: Neural Networks Through the Lens of the Hessian".
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]