Bidirectional Bayesian and Non-Bayesian Out of Distribution Detection Evaluation

Authors
Harry Li
Jiajing Hu
Eric Zaks

Abstract

Purpose: Out of distribution (OOD) detection for image classification tasks is important for practical machine learning applications, in which models often encounter real-world inputs different than the training data. Previous work has shown that deep generative Bayesian models may not be able to detect OOD data and make highly confident but wrong predictions. **Hypothesis:** To evaluate the performance of OOD image detection methods, in this paper we compared a recent state-of-the-art Bayesian Variational Auto Encoder (VAE) OOD method against a non-Bayesian convolutional neural network (CNN) based OOD method. We hypothesized that the Bayesian-based method specifically designed for OOD detection would out perform the non-Bayesian method. **Evaluation Plan:** Because the train vs test dataset directionality can manifest asymmetric performance, we conducted "bidirectional" OOD image classification experiments with MNIST vs Fashion MNIST and CIFAR-10 vs SVHN to avoid coincidentally good results. We then used various metrics to evaluate the models' OOD detection performance, sensitivity-specificity tradeoff, and precision-recall tradeoff. **Conclusions:** We found that both methods had varying performance depending on the dataset pair and direction of the experiment, and that the Bayesian method in particular sometimes performed poorly. In reflection, we caution against using deep learning models in contexts where OOD detection performance has not been thoroughly evaluated, and propose a rigorous OOD testing framework to evaluate methods against multiple test datasets in multiple directions.

1 Section 1: Goals

1.1 Motivation

OOD detection is an exciting research area with many real-world applications across industries like autonomous vehicles, finance, and medicine. While supervised-learning models are trained on one set of data and can achieve high test accuracy, in production, models often encounter data that is unlike the training data, ie OOD. When faced with OOD input data, models can often make highly confident, yet dangerously wrong predictions that, for example, can lead to automotive fatalities [5]. Ideally, models should be able to detect whether the input data they receive look like the data they encountered in training, in order to allow for better uncertainty-aware decision making.

1.2 Task

We evaluated OOD image detection performance between a Bayesian (LMPBT [6]) and the non-Bayesian (ODIN [10]) method. Each method trains a model that takes an image as input and outputs a real-valued score representing whether the model believes that the input is in or out of distribution. For each method, we ran experiments in which we trained the method on one data set, and then

evaluated its OOD detection performance on an in-distribution (ID) and OOD data set. For example, in one experiment we trained a model on the CIFAR-10 training set and then evaluated its OOD detection performance on the CIFAR-10 and SVHN test sets. In this example, a well-trained model should recognize that the CIFAR-10 test images are ID, while the SVHN test images are OOD.

1.3 Hypothesis

Our hypothesis was that the Bayesian LMPBT VAE model would outperform a non-Bayesian ODIN CNN model in OOD image detection, by having better performance across the following metrics: area under receiver operating characteristic (AUROC) curve, area under precision recall (AUPR) curve, and false positive rate (FPR) at a 95% true positive rate (TPR). While Nalisnick et al [11] discovered that VAEs may not perform well in OOD detection tasks, we thought that because LMPBT is specifically designed for OOD detection and incorporates Bayesian hypothesis testing, it would perform better than ODIN.

2 Section 2: Methods

2.1 Baseline Method: ODIN

Our baseline models were feed-forward CNN Densenets [4] trained using mini-batch gradient descent to classify images and modified to use ODIN [10] for OOD detection. ODIN takes an existing pre-trained classification neural network and uses input perturbations as well as temperature scaling on the final softmax layer to improve OOD detection. Given a temperature value T, the input logits $f_i(x)$ of the softmax equation are all divided by the temperature, i.e. $f_i(x)/T$ to better calibrate the output probabilities:

$$S_i(x;T) = \frac{exp(f_i(x)/T)}{\sum_{j=1}^{N} exp(f_j(x)/T)} \tag{1}$$

Using this technique, output probabilities that are extremely high or low are brought closer together. If all the output probabilities fall below a selected threshold δ, then the input is considered to be OOD.

ODIN also adds input perturbations that are designed to improve resiliency to adversarial attacks and improve in-distribution data detection, where ϵ is the perturbation magnitude and $\tilde{x}$ is the new perturbed input:

$$\tilde{x} = x - \epsilon sign(-\nabla_x log S_{\hat{y}}(x;T)) \tag{2}$$

The goal of the input perturbations is to increase the softmax score of any given input without the need for a class label, and the authors observed that the perturbations can help a model separate between ID and OOD images.

2.2 Focus Method: LMPBT VAE

2.2.1 Variational Autoencoders (VAEs)

VAEs are a kind of Bayesian generative model consisting of two sub-models: an encoder network that maps input points x into an latent space z and a generative decoder network that takes in an encoding z and maps it back to output points $\hat{x}$. Through training, the encoder and generator learn to encode x to a z such that the decoded $\hat{x}$ closely resembles the original input x. After training, the generator can be given different samples from the prior to generate new examples $\hat{x}$ that are similar to the original input data.

Given a tractable prior on the embedding $p(z_i)$, for example a Gaussian prior with mean μ and standard deviation, σ would be $p(z_i) = \mathcal{N}(z_i; \mu, \sigma)$, and the likelihood $p(x_i|z_i) = Bern(x_i|decode(z_i, \theta))$. The posterior of the embedding given an image is

$$p(z_i|x_i) = \frac{p(z_i)p(x_i|z_i)}{\int p(z)p(x_i|z)dz} \tag{3}$$

which contains an intractable denominator.

VAEs instead approximate the posterior inference by defining a simpler model q with parameters ϕ, where

$$q(z_i|x_i) = \mathcal{N}(encode(x_i; \phi), \sigma^2) \tag{4}$$

We assume that there is an independent Normal q distribution for each example i. z, where the mean is set by the encoder's output given an input x_i, and the covariance matrix is the identity matrix scaled by factor σ. ϕ represents all the parameters of the model q, which includes all the neural network weights and biases as well as the parameter σ.

The encoder is defined as $q(z) = \Pi_i q(z_i|x_i)$ and the generator is $p(x, z) = \Pi_i p(z_i)p(x_i|z_i)$. Then training via optimization by maximizing the ELBO function

$$logp_\theta(x) \geq ELBO = E_{q_\phi}[logp_\theta(x|z) + logp(z) - logq_\phi(z|x)] \tag{5}$$

to achieve parameters that ideally closely match the true posterior. The loss function can be defined as minimizing the negative log probability

$$-logL(\theta|x) = -logp_\theta(x) \tag{6}$$

With the VAE model established, it can be optimized with gradient descent similar to other neural networks architectures. LMPBT, which we describe in more detail below, trains both the encoder and decoder neural networks using mini-batch gradient descent with Adam as the optimizer.

2.2.2 Locally Most Powerful Bayesian Test (LMPBT)

In this section, we switch to the LMPBT notation where θ refers to a set of VAE parameters from the entire possible parameter space, θ_0 refers to VAE parameters for in-distribution (ID) data, and θ_1 refers to VAE parameters for OOD data.

LMPBT trains a VAE and develops an OOD score based on a Bayesian hypothesis test. The null hypothesis $H_0 : \theta = \theta_0$ is that the test sample x_t is ID, ie can be explained by the trained VAE parameters θ_0. The alternative hypothesis $H_1 : \theta = \theta_1$ is the test sample x_t is OOD, ie the input can be better explained by a different set of parameters θ_1.

The Bayesian hypothesis test checks the ratio of the posterior odds that the alternative hypothesis is true given the observed data:

Posterior odds:

$$\frac{P(H_1|x_t)}{P(H_0|x_t)} = \frac{L(\theta_1|x_t)}{L(\theta_0|x_t)} \times \frac{P(H_1)}{P(H_0)} \tag{7}$$

$P(H_0)$ and $P(H_1)$ are the prior probabilities for their respective hypothesis H_0 and H_1. $L(\theta_0|x_t)$ and $L(\theta_1|x_t)$ are the likelihoods of x_t for their respective hypothesis H_0 and H_1, and $\frac{L(\theta_1|x_t)}{L(\theta_0|x_t)}$ is the Bayes factor in favor of the alternative hypothesis. Evaluating the hypothesis test requires specifying θ_0 and θ_1, and the prior probabilities for both hypotheses.

The authors estimate θ_0 via maximum likelihood estimation (MLE), denoted as

$$\hat{\theta} : \hat{\theta} = argmax_{\theta} \sum_{i=1}^{N} logL(\theta|x_i) \tag{8}$$

The authors define θ_1 as the MLE of the expanded training set with the test sample x_t, denoted as

$$\hat{\theta_t} : \hat{\theta_t} = argmax_{\theta}[logL(\theta|x_t) + \sum_{i=1}^{N} logL(\theta|x_i)] \tag{9}$$

Based on this definition, the optimization problem needs to be solved for each test sample x_t, which would require computationally intensive retraining for each test sample. The authors instead use an upweighting [7] method to approximate $\hat{\theta_t}$, by calculating the influence of a particular training data sample on the parameter change by upweighting the loss of that sample. The upweighting influence loss of a training data sample x by a scale ϵ is equal to

$$\frac{d\hat{\theta}_t}{d\epsilon}|\epsilon = 0 = -H_{\hat{\theta}}^{-1} \nabla_{\theta}(-logL(\hat{\theta}|x_t)) \tag{10}$$

where $H_{\hat{\theta}} = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\theta}^2(-logL(\hat{\theta}|x_t))$ is the Hessian for $\hat{\theta}$ and $\nabla_{\theta}(-logL(\hat{\theta}|x_t))$ is the loss gradient for sample x with respect to θ. Since the training set does not contain the test sample x_t, the weight of x_t in the training loss is zero. When the loss of x_t is aded to the training loss for the whole training set, it is equivalent to upweighting the loss of x_t by $1/n$, allowing us to approximate $\hat{\theta_t}$ as

$$\hat{\theta_t} \approx \tilde{\theta_t} = \hat{\theta} - \frac{1}{n} H_{\hat{\theta}}^{-1} \nabla_{\theta}(-logL(\hat{\theta}|x_t)) \tag{11}$$

With $\theta_0 = \hat{\theta}$ and $\theta_1 = \tilde{\theta}$, the authors assume equal prior probabilities for H_0 and H_1, so that $P(H_0) = P(H_1) = 0.5$, then propose the log of the posterior odds as the OOD score for a test input sample x_t as

The OOD score is denoted as:

$$S(x_t) = -logL(\hat{\theta}|x_t) + logL(\tilde{\theta_t}|x_t) \tag{12}$$

$S(x_t)$ maximizes the probability that the alternative hypothesis is accepted when it is true.

3 Section 3: Experiments

3.1 Data Sources

The data sources we used were MNIST, Fashion MNIST, CIFAR-10, and SVHN, shown in Figure 1.

MNIST [9] is a dataset of 70k (28×28) grayscale images of 10 classes of handwritten digits. Fashion MNIST [13] has the same input size as and number of examples as MNIST but consists of images of clothing items from 10 categories. CIFAR-10 [8] is a dataset of 60k (32×32) full color images in 10 categories that come from real-world objects. One version of SVHN [12] is a dataset of about 100k (32×32) full color images of 10 classes of real world house number digits.

(a) MNIST (b) Fashion MNIST (c) CIFAR-10 (d) SVHN

Figure 1: Example images from the datasets we will use for training and test

All of these datasets are publicly accessible. We chose these 4 datasets because they were used in some of the experiments by Nalisnick et al [11].

3.2 Bidirectional Experiments

Nalisnick et al observed asymmetric results when the train and test datasets directionality was swapped [11]. A model could perform well when trained on MNIST and tested on Fashion MNIST for OOD detection, but perform poorly when trained on Fashion MNIST but tested against MNIST. They hypothesized that this is possibility due to nested distributions of low-level pixel statistics of the datasets. In order to more exhaustively evaluate LMPBT's and ODIN's performance and avoid focusing on coincidentally good results, we ran 4 bidirectional experiments. We trained LMPBT and ODIN on

- MNIST training set, then tested on both MNIST (ID) and Fashion MNIST (OOD) test sets
- Fashion MNIST training set, then tested on both Fashion MNIST (ID) and MNIST (OOD) test sets
- CIFAR-10 training set, then tested on both CIFAR-10 (ID) and SVHN (OOD) test sets
- SVHN training set, then tested on both SVHN (ID) and CIFAR-10 (OOD) test sets

3.2.1 Hyperparameter Selection

We repurposed LMPBT and ODIN to run our bidirectional OOD experiments by varying the training and OOD test datasets. We also trained both methods using the hyperparameters presented in their respective papers, for simplicity and to compare each method as the authors presented them. These fixed parameters include the number of training epochs, data transforms, model architectures, learning rates, optimizers, loss functions, input perturbations, encoding space dimensionality ($|z| = 100$) for LMPBT, and temperature scaling factor ($T = 1000$) and input perturbation magnitude ($\epsilon = 0.0014$) for ODIN.

The models we used for evaluation were be selected by the best reconstruction error for LMPBT and the best validation accuracy for ODIN.

3.2.2 Metrics

After each experiment, we generated receiver operating characteristic (ROC) and precision-recall (PR) curves for LMPBT and ODIN and evaluated their AUROC, AUPR, and FPR at 95% TPR values. These metrics are important for evaluating a method's ability to determine whether an input is in or out of distribution, sensitivity-specificity tradeoff, and precision-recall tradeoff. Higher AUROC and AUPR values are better, and lower FPR values are better.

3.3 Code

For both LMPBT and ODIN, we adatped their public PyTorch implementations [2] [1], and used sklearn [3] to generate ROC and PR curves to compare the two methods.

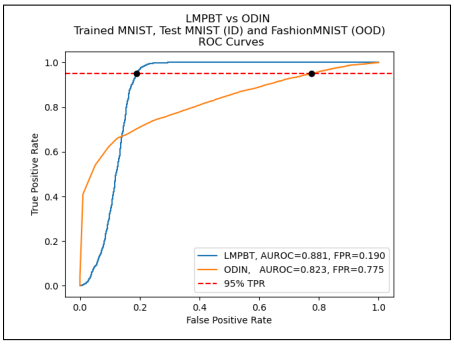
3.4 Results

Below in Table 1 are the results from our experiments, showing each method's performance using ROC and PR curves. Higher AUROC and AUPR values are better, and lower FPR values are better. The Bayesian LMPBT method is colored in blue and the non-Bayesian ODIN method is colored in orange.

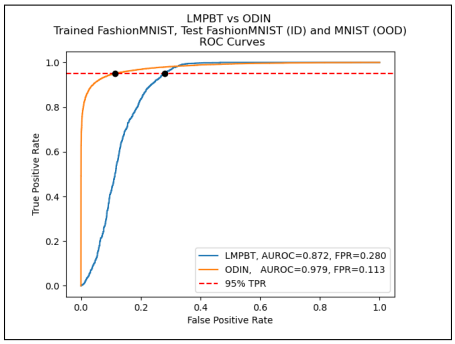
		LMPBT			ODIN		
ID Dataset	**OOD Dataset**	**AUROC**	**FPR**	**AUPR**	**AUROC**	**FPR**	**AUPR**
MNIST	Fashion MNIST	**0.881**	**0.190**	**0.928**	0.823	0.914	0.858
Fashion MNIST	MNIST	0.872	0.280	0.775	**0.979**	**0.113**	**0.983**
CIFAR-10	SVHN	0.874	**0.451**	0.889	**0.897**	0.512	**0.908**
SVHN	CIFAR-10	0.416	0.965	0.444	**0.981**	**0.098**	**0.984**

Table 1: Bidirectional experiments results

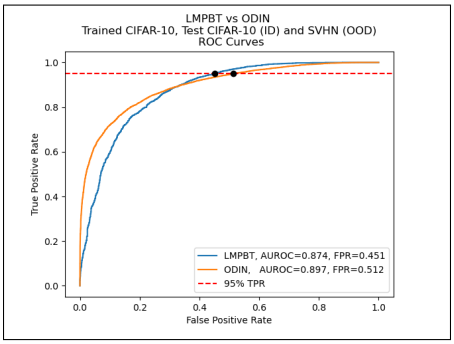
Our original hypothesis that LMPBT would always outperform ODIN was incorrect. LMPBT sometimes outperformed ODIN, and sometimes performed much worse. When MNIST was ID and Fashion MNIST was OOD, LMPBT had better AUROC and AUPR values, and a much better FPR. However, when we flipped the experiment direction, ODIN convincingly outperformed LMPBT in all three metrics. Both methods performed similarly but mediocrely when CIFAR-10 was ID and SVHN was OOD. However, when we flipped the experiment direction, ODIN performed well while LMPBT actually performed worse than random.



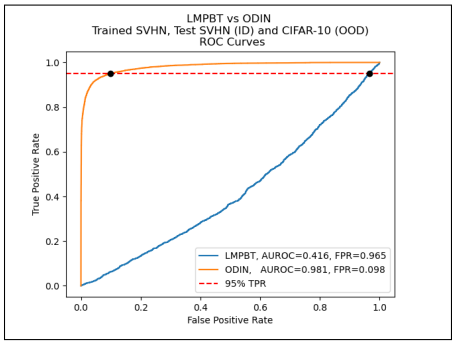
(a) MNIST (ID), Fashion MNIST (OOD)



(b) Fashion MNIST (ID), MNIST (OOD)

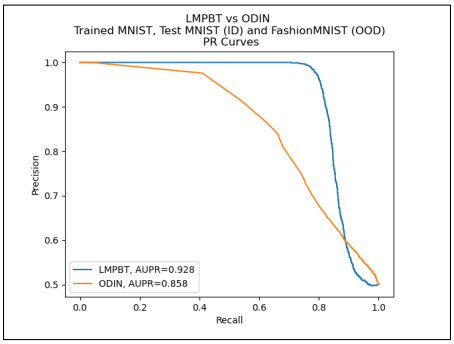


(c) CIFAR-10 (ID), SVHN (OOD)

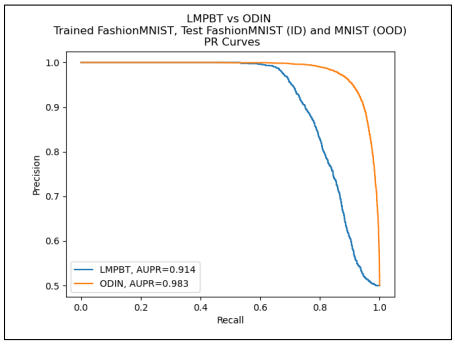


(d) SVHN (ID), CIFAR-10 (OOD)

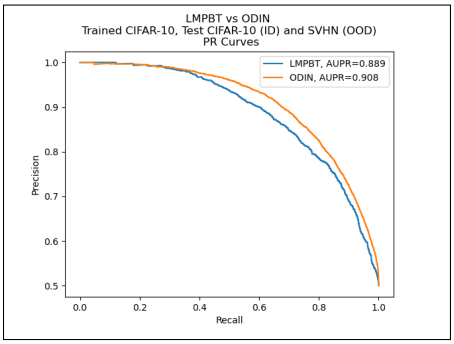
Figure 2: ROC curves comparing LMPBT against ODIN



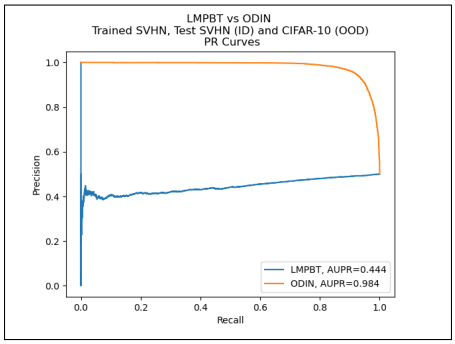
(a) MNIST (ID), Fashion MNIST (OOD)



(b) Fashion MNIST (ID), MNIST (OOD)



(c) CIFAR-10 (ID), SVHN (OOD)



(d) SVHN (ID), CIFAR-10 (OOD)

Figure 3: PR curves comparing LMPBT against ODIN

ID Dataset	OOD Dataset	Nalisnick et al	LMPBT	ODIN
MNIST	Fashion MNIST	better	better	worse
Fashion MNIST	MNIST	worse	worse	better
CIFAR-10	SVHN	worse	better	worse
SVHN	CIFAR-10	better	worse	better

Table 2: Each method's better/worse performance direction compared with what Nalisnick et al observed.

These results are partially consistent with Nalisnick et al's observations, in that an OOD detection method's performance may vary depending on the datasets direction [11], but the methods also varied in the direction in which they performed better. LMPBT and ODIN performed better in opposite directions for all four experiments, seen in Table 2. It appears then that not only can one particular method perform inconsistently depending on the test direction, but that two given methods may also perform better in opposite directions.

Nalisnick et al hypothesized that nested pixel statistics distributions leads to the OOD directionality confusion, ie that OOD detection is easier when the narrower distribution is ID (ie MNIST and SVHN), while the wider distribution is OOD (ie Fashion MNIST and CIFAR-10). This hypothesis could explain LMPBT's behavior for MNIST / Fashion MNIST and ODIN's behavior for CIFAR-10 / SVHN. However, LMBPT's behavior with CIFAR-10 / SVHN and ODIN's behavior with MNIST / Fashion MNIST run counter to this hypothesis; both models performed better when the dataset with the wider pixel statistics distribution was ID, while the narrower dataset was OOD. Future work can focus on investigating why the models had varying performances depending on the dataset direction.

4 Section 4: Reflection and Outlook

We were hoping that we would see generalized, robust, and competitive OOD detection, especially from the Bayesian LMPBT method dedicated to OOD detection. Instead we found that each method's performance, particularly LMPBT's, varied drastically depending on the dataset pair and direction. Based on these results, we caution against using deep learning models in contexts where the model's OOD detection performance has not been thoroughly evaluated.

In order to systematically evaluate a model's OOD detection performance, we propose a rigorous multi-directional image classification OOD testing framework, so that evaluations do not focus on a dataset pair and direction that coincidentally demonstrates good performance. In our experiments, we only tested four dataset pairs and directions, but future work can be done to implement a multi-directional testing framework shown in Figure 4, in which all the possible dataset pairs and directions are tested.

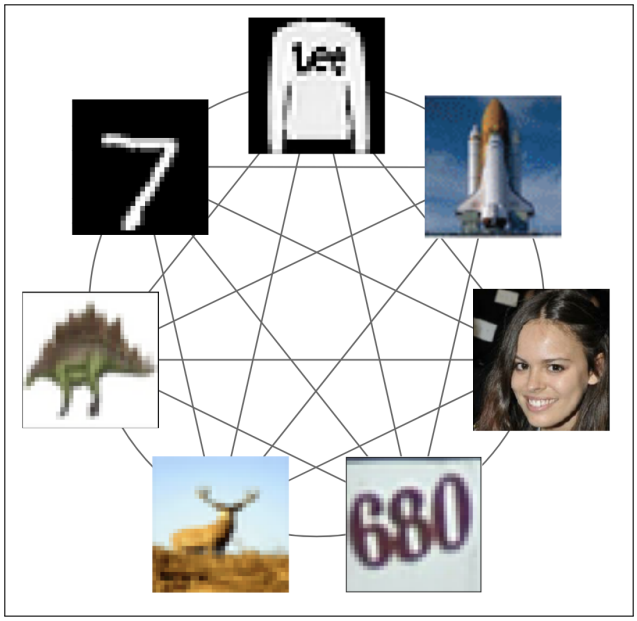


Figure 4: Datasets pictured are (from the top, then clockwise) Fashion MNIST, ImageNet, CelebA, SVHN, CIFAR-10, CIFAR-100, and MNIST

In this testing framework, a model would be trained on one training dataset, ex MNIST, then tested against the test sets from all the datasets, ex MNIST (ID), Fashion MNIST (OOD), CIFAR-10 (OOD), etc. The same training and testing process would then be repeated for all the available training datasets, ex Fashion MNIST, CIFAR-10, etc. Such a thorough and exhaustive testing framework would be better at evaluating a method's ability to generalize to many training and testing scenarios.

4.1 Future work

In the short term future, we could attempt to improve the performance of each method by tuning the hyperparameters, especially for LMPBT training on SVHN. In the long term, we would like to investigate why each method performed better in different dataset scenarios and directions. We would also like to implement our proposed multi-directional OOD testing framework using LMPBT and ODIN for evaluation.

References

[1] Facebookresearch/odin: A simple and effective method for detecting out-of-distribution images in neural networks. `https://github.com/facebookresearch/odin`.

[2] Keunseokim91/lmpbt: Locally most powerful bayesian test for out-of-distribution detection using deep generative models `https://github.com/keunseokim91/LMPBT`.

[3] Sklearn precision-recall `https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html`.

[4] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[5] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *CoRR*, abs/1703.04977, 2017.

[6] Keunseo Kim, JunCheol Shin, and Heeyoung Kim. Locally most powerful bayesian test for out-of-distribution detection using deep generative models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 14913–14924. Curran Associates, Inc., 2021.

[7] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

[8] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[9] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[10] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks, 2017.

[11] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know?, 2018.

[12] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[13] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.