Likelihood Ratios for Out-of-Distribution Detection

Jie Ren[*,†] Google Research jjren@google.com

Peter J. Liu [‡] Google Research peterjliu@google.com

Emily Fertig[†] Google Research emilyaf@google.com

Jasper Snoek Google Research jsnoek@google.com

Ryan Poplin Google Research rpoplin@google.com

Mark A. DePristo Google Research mdepristo@google.com

Joshua V. Dillon [‡] Google Research jvdillon@google.com

Balaji Lakshminarayanan[*,‡] DeepMind balajiln@google.com

Abstract

Discriminative neural networks offer little or no performance guarantees when deployed on data not generated by the same process as the training distribution. On such out-of-distribution (OOD) inputs, the prediction may not only be erroneous, but confidently so, limiting the safe deployment of classifiers in real-world applications. One such challenging application is bacteria identification based on genomic sequences, which holds the promise of early detection of diseases, but requires a model that can output low confidence predictions on OOD genomic sequences from new bacteria that were not present in the training data. We introduce a genomics dataset for OOD detection that allows other researchers to benchmark progress on this important problem. We investigate deep generative model based approaches for OOD detection and observe that the likelihood score is heavily affected by population level background statistics. We propose a likelihood ratio method for deep generative models which effectively corrects for these confounding background statistics. We benchmark the OOD detection performance of the proposed method against existing approaches on the genomics dataset and show that our method achieves state-of-the-art performance. We demonstrate the generality of the proposed method by showing that it significantly improves OOD detection when applied to deep generative models of images.

1 Introduction

For many machine learning systems, being able to detect data that is anomalous or significantly different from that used in training can be critical to maintaining safe and reliable predictions. This is particularly important for deep neural network classifiers which have been shown to incorrectly classify such *out-of-distribution* (OOD) inputs into in-distribution classes with high confidence (Goodfellow et al., 2014; Nguyen et al., 2015). This behaviour can have serious consequences when the predictions inform real-world decisions such as medical diagnosis, e.g. falsely classifying a healthy sample as pathogenic or vice versa can have extremely high cost. The importance of dealing with OOD inputs, also referred to as distributional shift, has been recognized as an important problem for

[*]Corresponding authors
[†]Google AI Resident
[‡]Mentors

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

AI safety (Amodei et al., 2016). The majority of recent work on OOD detection for neural networks is evaluated on image datasets where the neural network is trained on one benchmark dataset (e.g. CIFAR-10) and tested on another (e.g. SVHN). While these benchmarks are important, there is a need for more realistic datasets which reflect the challenges of dealing with OOD inputs in practical applications.

Bacterial identification is one of the most important sub-problems of many types of medical diagnosis. For example, diagnosis and treatment of infectious diseases, such as sepsis, relies on the accurate detection of bacterial infections in blood (Blauwkamp et al., 2019). Several machine learning methods have been developed to perform bacteria identification by classifying existing known genomic sequences (Patil et al., 2011; Rosen et al., 2010), including deep learning methods (Busia et al., 2018) which are state-of-the-art. Even if neural network classifiers achieve high accuracy as measured through cross-validation, deploying them is challenging as real data is highly likely to contain genomes from unseen classes not present in the training data. Different bacterial classes continue to be discovered gradually over the years (see Figure S4 in Appendix C.1) and it is estimated that 60%-80% of genomic sequences belong to as yet unknown bacteria (Zhu et al., 2018; Eckburg et al., 2005; Nayfach et al., 2019). Training a classifier on existing bacterial classes and deploying it may result in OOD inputs being wrongly classified as one of the classes from the training data with high confidence. In addition, OOD inputs can also be the contaminations from the bacteria's host genomes such as human, plant, fungi, etc., which also need to be detected and excluded from predictions (Ponsero & Hurwitz, 2019). Thus having a method for accurately detecting OOD inputs is critical to enable the practical application of machine learning methods to this important problem.

A popular and intuitive strategy for detecting OOD inputs is to train a generative model (or a hybrid model cf. Nalisnick et al. (2019)) on training data and use that to detect OOD inputs at test time (Bishop, 1994). However, Nalisnick et al. (2018) and Choi et al. (2018) recently showed that deep generative models trained on image datasets can assign higher likelihood to OOD inputs. We report a similar failure mode for likelihood based OOD detection using deep generative models of genomic sequences. We investigate this phenomenon and find that the likelihood can be confounded by general population level background statistics. We propose a likelihood ratio method which uses a background model to correct for the background statistics and enhances the in-distribution specific features for OOD detection. While our investigation was motivated by the genomics problem, we found our methodology to be more general and it shows positive results on image datasets as well. In summary, our contributions are:

- We create a realistic benchmark for OOD detection, that is motivated by challenges faced in applying deep learning models on genomics data. The sequential nature of genetic sequences provides a new modality and hopefully encourages the OOD research community to contribute to "machine learning that matters" (Wagstaff, 2012).
- We show that likelihood from deep generative models can be confounded by background statistics.
- We propose a likelihood ratio method for OOD detection, which significantly outperforms the raw likelihood on OOD detection for deep generative models on image datasets.
- We evaluate existing OOD methods on the proposed genomics benchmark and demonstrate that our method achieves state-of-the-art (SOTA) performance on this challenging problem.

2 Background

Suppose we have an in-distribution dataset $\mathcal{D}$ of $(\boldsymbol{x}, y)$ pairs sampled from the distribution $p^*(\boldsymbol{x}, y)$, where $\boldsymbol{x}$ is the extracted feature vector or raw input and $y \in \mathcal{Y} := \{1, \ldots, k, \ldots, K\}$ is the label assigning membership to one of K in-distribution classes. For simplicity, we assume inputs to be discrete, i.e. $x_d \in \{A, C, G, T\}$ for genomic sequences and $x_d \in \{0, \ldots, 255\}$ for images. In general, OOD inputs are samples $(\boldsymbol{x}, y)$ generated from an underlying distribution other than $p^*(\boldsymbol{x}, y)$. In this paper, we consider an input $(\boldsymbol{x}, y)$ to be OOD if $y \notin \mathcal{Y}$: that is, the class y does not belong to one of the K in-distribution classes. Our goal is to accurately detect if an input $\boldsymbol{x}$ is OOD or not.

Many existing methods involve computing statistics using the predictions of (ensembles of) discriminative classifiers trained on in-distribution data, e.g. taking the confidence or entropy of the predictive distribution $p(y|\boldsymbol{x})$ (Hendrycks & Gimpel, 2016; Lakshminarayanan et al., 2017). An alternative is to use generative model-based methods, which are appealing as they do not require

labeled data and directly model the input distribution. These methods fit a generative model $p(\boldsymbol{x})$ to the input data, and then evaluate the likelihood of new inputs under that model. However, recent work has highlighted significant issues with this approach for OOD detection on images, showing that deep generative models such as Glow (Kingma & Dhariwal, 2018) and PixelCNN (Oord et al., 2016; Salimans et al., 2017) sometimes assign higher likelihoods to OOD than in-distribution inputs. For example, Nalisnick et al. (2018) and Choi et al. (2018) show that Glow models trained on the CIFAR-10 image dataset assign higher likelihood to OOD inputs from the SVHN dataset than they do to in-distribution CIFAR-10 inputs; Nalisnick et al. (2018), Shafaei et al. (2018) and Hendrycks et al. (2018) show failure modes of PixelCNN and PixelCNN++ for OOD detection.

Failure of density estimation for OOD detection We investigate whether density estimation-based methods work well for OOD detection in genomics. As a motivating observation, we train a deep generative model, more precisely LSTM (Hochreiter & Schmidhuber, 1997), on in-distribution genomic sequences (composed by {A, C, G, T}), and plot the log-likelihoods of both in-distribution and OOD inputs (See Section 5.2 for the dataset and the full experimental details). Figure 1a shows that the histogram of the log-likelihood for OOD sequences largely overlaps with that of in-distribution sequences with AUROC of 0.626, making it unsuitable for OOD detection. Our observations show a failure mode of deep generative models for OOD detection on genomic sequences and are complementary to earlier work which showed similar results for deep generative models on images (Nalisnick et al., 2018; Choi et al., 2018).

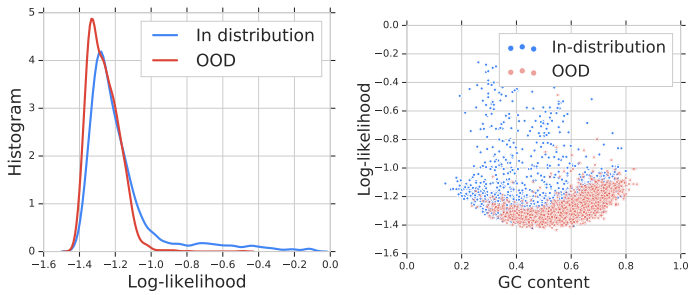


Figure 1: (a) Log-likelihood hardly separates in-distribution and OOD inputs with AUROC of 0.626. (b) The log-likelihood is heavily affected by the GC-content of a sequence.

When investigating this failure mode, we discovered that the log-likelihood under the model is heavily affected by a sequence's *GC-content*, see Figure 1b. GC-content is defined as the percentage of bases that are either G or C, and is used widely in genomic studies as a basic statistic for describing overall genomic composition (Sueoka, 1962), and studies have shown that bacteria have an astonishing diversity of genomic GC-content, from 16.5% to 75% (Hildebrand et al., 2010). Bacteria from similar groups tend to have similar GC-content at the population level, but they also have characteristic biological patterns that can distinguish them well from each other. The confounding effect of GC-content in Figure 1b makes the likelihood less reliable as a score for OOD detection, because an OOD input may result in a higher likelihood than an in-distribution input, because it has high GC-content (cf. the bottom right part of Figure 1b) and not necessarily because it contains characteristic patterns specific to the in-distribution bacterial classes.

3 Likelihood Ratio for OOD detection

We first describe the high level idea and then describe how to adapt it to deep generative models.

High level idea Assume that an input $\boldsymbol{x}$ is composed of two components, (1) a *background* component characterized by population level background statistics, and (2) a *semantic* component characterized by patterns specific to the in-distribution data. For example, images can be modeled as backgrounds plus objects; text can be considered as a combination of high frequency stop words plus semantic words (Luhn, 1960); genomes can be modeled as background sequences plus motifs (Bailey & Elkan, 1995; Reinert et al., 2009). More formally, for a D-dimensional input $\boldsymbol{x} = x_1, \ldots, x_D$, we assume that there exists an unobserved variable $\boldsymbol{z} = z_1, \ldots, z_D$, where $z_d \in \{B, S\}$ indicates if the dth dimension of the input x_d is generated from the *B*ackground model or the *S*emantic model.

Grouping the semantic and background parts, the input can be factored as $\boldsymbol{x} = \{\boldsymbol{x}_B, \boldsymbol{x}_S\}$ where $\boldsymbol{x}_B = \{x_d \mid z_d = B, d = 1, \ldots, D\}$. For simplicity, assume that the background and semantic components are generated independently. The likelihood can be then decomposed as follows,

$$p(\boldsymbol{x}) = p(\boldsymbol{x}_B)p(\boldsymbol{x}_S). \tag{1}$$

When training and evaluating deep generative models, we typically do not distinguish between these two terms in the likelihood. However, we may want to use just the semantic likelihood $p(\boldsymbol{x}_S)$ to avoid the likelihood term being dominated by the background term (e.g. OOD input with the same background but different semantic component). In practice, we only observe $\boldsymbol{x}$, and it is not always easy to split an input into background and semantic parts $\{\boldsymbol{x}_B, \boldsymbol{x}_S\}$. As a practical alternative, we propose training a background model by perturbing inputs. Adding the right amount of perturbations to inputs can corrupt the semantic structure in the data, and hence the model trained on perturbed inputs captures only the population level background statistics.

Assume that $p_{\boldsymbol{\theta}}(\cdot)$ is a model trained using in-distribution data, and $p_{\boldsymbol{\theta}_0}(\cdot)$ is a background model that captures general background statistics. We propose a likelihood ratio statistic that is defined as

$$\mathsf{LLR}(\boldsymbol{x}) = \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x})}{p_{\boldsymbol{\theta}_0}(\boldsymbol{x})} = \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_B)\, p_{\boldsymbol{\theta}}(\boldsymbol{x}_S)}{p_{\boldsymbol{\theta}_0}(\boldsymbol{x}_B)\, p_{\boldsymbol{\theta}_0}(\boldsymbol{x}_S)}, \tag{2}$$

where we use the factorization from Equation 1. Assume that (i) both models capture the background information equally well, that is $p_{\boldsymbol{\theta}}(\boldsymbol{x}_B) \approx p_{\boldsymbol{\theta}_0}(\boldsymbol{x}_B)$ and (ii) $p_{\boldsymbol{\theta}}(\boldsymbol{x}_S)$ is more peaky than $p_{\boldsymbol{\theta}_0}(\boldsymbol{x}_S)$ as the former is trained on data containing semantic information, while the latter model $\boldsymbol{\theta}_0$ is trained using data with noise perturbations. Then, the likelihood ratio can be approximated as

$$\mathsf{LLR}(\boldsymbol{x}) \approx \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_S) - \log p_{\boldsymbol{\theta}_0}(\boldsymbol{x}_S). \tag{3}$$

After taking the ratio, the likelihood for the background component $\boldsymbol{x}_B$ is cancelled out, and only the likelihood for the semantic component $\boldsymbol{x}_S$ remains. Our method produces a *background contrastive* score that captures the significance of the semantics compared with the background model.

Likelihood ratio for auto-regressive models Auto-regressive models are one of the popular choices for generating images (Oord et al., 2016; Van den Oord et al., 2016; Salimans et al., 2017) and sequence data such as genomics (Zou et al., 2018; Killoran et al., 2017) and drug molecules (Olivecrona et al., 2017; Gupta et al., 2018), and text (Jozefowicz et al., 2016). In auto-regressive models, the log-likelihood of an input can be expressed as $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{d=1}^{D} \log p_{\boldsymbol{\theta}}(x_d|\boldsymbol{x}_{<d})$, where $\boldsymbol{x}_{<d} = x_1 \ldots x_{d-1}$. Decomposing the log-likelihood into background and semantic parts, we have

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{d:x_d \in \boldsymbol{x}_B} \log p_{\boldsymbol{\theta}}(x_d|\boldsymbol{x}_{<d}) + \sum_{d:x_d \in \boldsymbol{x}_S} \log p_{\boldsymbol{\theta}}(x_d|\boldsymbol{x}_{<d}). \tag{4}$$

We can use a similar auto-regressive decomposition for the background model $p_{\boldsymbol{\theta}_0}(\boldsymbol{x})$ as well. Assuming that both the models capture the background information equally well, $\sum_{d:x_d \in \boldsymbol{x}_B} \log p_{\boldsymbol{\theta}}(x_d|\boldsymbol{x}_{<d}) \approx \sum_{d:x_d \in \boldsymbol{x}_B} \log p_{\boldsymbol{\theta}_0}(x_d|\boldsymbol{x}_{<d})$, the likelihood ratio is approximated as

$$\mathsf{LLR}(\boldsymbol{x}) \approx \sum_{d:x_d \in \boldsymbol{x}_S} \log p_{\boldsymbol{\theta}}(x_d|\boldsymbol{x}_{<d}) - \sum_{d:x_d \in \boldsymbol{x}_S} \log p_{\boldsymbol{\theta}_0}(x_d|\boldsymbol{x}_{<d}) = \sum_{d:x_d \in \boldsymbol{x}_S} \log \frac{p_{\boldsymbol{\theta}}(x_d|\boldsymbol{x}_{<d})}{p_{\boldsymbol{\theta}_0}(x_d|\boldsymbol{x}_{<d})}. \tag{5}$$

Training the Background Model In practice, we add perturbations to the input data by randomly selecting positions in $x_1 \ldots x_D$ following an independent and identical Bernoulli distribution with rate μ and substituting the original character with one of the other characters with equal probability. The procedure is inspired by genetic mutations. See Algorithm 1 in Appendix A for the pseudocode for generating input perturbations. The rate μ is a hyperparameter and can be easily tuned using a small amount of validation OOD dataset (different from the actual OOD dataset of interest). In the case where validation OOD dataset is not available, we show that μ can also be tuned using simulated OOD data. In practice, we observe that $\mu \in [0.1, 0.2]$ achieves good performance empirically for most of the experiments in our paper. Besides adding perturbations to the input data, we found other techniques that can improve model generalization and prevent model memorization, such as adding L_2 regularization with coefficient λ to model weights, can help to train a good background model. In fact, it has been shown that adding noise to the input is equivalent to adding L_2 regularization to the model weights under some conditions (Bishop, 1995a,b). Besides the methods above, we expect adding other types of noise or regularization methods would show a similar effect. The pseudocode for our proposed OOD detection algorithm can be found in Algorithm 2 in Appendix A.

4 Experimental setup

We design experiments on multiple data modalities (images, genomic sequences) to evaluate our method and compare with other baseline methods. For each of the datasets, we build an auto-regressive model for computing the log-likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{d=1}^{D} \log p_{\boldsymbol{\theta}}(x_d|\boldsymbol{x}_{<d})$. For training the background model $p_{\boldsymbol{\theta}_0}(\boldsymbol{x})$, we use the exact same architecture as $p_{\boldsymbol{\theta}}(\boldsymbol{x})$, and the only differences are that it is trained on perturbed inputs and (optionally) we apply L_2 regularization to model weights.

Baseline methods for comparison We compare our approach to several existing methods.

1. The maximum class probability, $p(\hat{y}|\boldsymbol{x}) = \max_k p(y = k|\boldsymbol{x})$. OOD inputs tend to have lower scores than in-distribution data (Hendrycks & Gimpel, 2016).
2. The entropy of the predicted class distribution, $-\sum_k p(y = k|\boldsymbol{x}) \log p(y = k|\boldsymbol{x})$. High entropy of the predicted class distribution, and therefore a high predictive uncertainty, which suggests that the input may be OOD.
3. The ODIN method proposed by Liang et al. (2017). ODIN uses temperature scaling (Guo et al., 2017), adds small perturbations to the input, and applies a threshold to the resulting predicted class to distinguish in- and out-of- distribution inputs. This method was designed for continuous inputs and cannot be directly applied to discrete genomic sequences. We propose instead to add perturbations to the input of the last layer that is closest to the output of the neural network.
4. The Mahalanobis distance of the input to the nearest class-conditional Gaussian distribution estimated from the in-distribution data. Lee et al. (2018) fit class-conditional Gaussian distributions to the activations from the last layer of the neural network.
5. The classifier-based ensemble method that uses the average of the predictions from multiple independently trained models with random initialization of network parameters and random shuffling of training inputs (Lakshminarayanan et al., 2017).
6. The log-odds of a binary classifier trained to distinguish between in-distribution inputs from all classes as one class and randomly perturbed in-distribution inputs as the other.
7. The maximum class probability over K in-distribution classes of a $(K + 1)$-class classifier where the additional class is perturbed in-distribution.
8. The maximum class probability of a K-class classifier for in-distribution classes but the predicted class distribution is explicitly trained to output uniform distribution on perturbed in-distribution inputs. This is similar to using simulated OOD inputs from GAN (Lee et al., 2017) or using auxiliary datasets of outliers (Hendrycks et al., 2018) for calibration purpose.
9. The generative model-based ensemble method that measures $\mathbb{E}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})] - \mathrm{Var}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})]$ of multiple likelihood estimations from independently trained model with random initialization and random shuffling of the inputs. (Choi et al., 2018).

Baseline methods 1-8 are classifier-based and method 9 is generative model-based. For classifier-based methods, we choose the commonly used model architecture, convolutional neural networks (CNNs). Methods 6-8 are based on perturbed inputs which aims to mimic OOD inputs. Perturbations are added to the input in the same way as that we use for training background models. Our method and methods 3, 6, 7, and 8 involve hyperparameter tuning; we follow the protocol of Hendrycks et al. (2018) where optimal hyperparameters are picked on a different OOD validation set than the final OOD dataset it is tested on. For Fashion-MNIST vs. MNIST experiment, we use the NotMNIST Bulatov (2011) dataset for hyperparameter tuning. For CIFAR-10 vs SVHN, we used gray-scaled CIFAR-10 for hyperparameter tuning. For genomics, we use the OOD bacteria classes discovered between 2011-2016, which are disjoint from the final OOD classes discovered after 2016. While this set of baselines is not exhaustive, it is broadly representative of the range of existing methods. Note that since our method does not rely on OOD inputs for training, we do not compare it with other methods that do utilize OOD inputs in training.

Evaluation metrics for OOD detection We trained the model using only in-distribution inputs, and we tuned the hyperparameters using validation datasets that include both in-distribution and OOD inputs. The test dataset is used for the final evaluation of the method. For the final evaluation, we randomly selected the same number of in-distribution and OOD inputs from the test dataset, and for each example $\boldsymbol{x}$ we computed the log likelihood-ratio statistic $\mathsf{LLR}(\boldsymbol{x})$ as the score. A small value of

the score suggests a high likelihood of being OOD. We use the area under the ROC curve (AUROC↑), the area under the precision-recall curve (AUPRC↑), and the false positive rate at 80% true positive rate (FPR80↓), as the metrics for evaluation. These three metrics are commonly used for evaluating OOD detection methods (Hendrycks & Gimpel, 2016; Hendrycks et al., 2018; Alemi et al., 2018).

5 Results

We first present results on image datasets as they are easier to visualize, and then present results on our proposed genomic dataset. For image experiments, our goal is not to achieve state-of-the-art performance but to show that our likelihood ratio effectively corrects for background statistics and significantly outperforms the likelihood. While previous work has shown the failure of PixelCNN for OOD detection, we believe ours is the first to provide an explanation for why this phenomenon happens for PixelCNN, through the lens of background statistics.

5.1 Likelihood ratio for detecting OOD images

Following existing literature (Nalisnick et al., 2018; Hendrycks et al., 2018), we evaluate our method using two experiments for detecting OOD images: (a) Fashion-MNIST as in-distribution and MNIST as OOD, (b) CIFAR-10 as in-distribution and SVHN as OOD. For each experiment, we train a PixelCNN++ (Salimans et al., 2017; Van den Oord et al., 2016) model using in-distribution data. We train a background model by adding perturbations to the training data. To compare with classifier-based baseline methods, we use CNN-based classifiers. See Appendix B.1 for model details. Based on the likelihood from the PixelCNN++ model, we confirm that the model assigns a higher likelihood to MNIST than Fashion-MNIST, as previously reported by Nalisnick et al. (2018), and the AUROC for OOD detection is only 0.091, even worse than random (Figure 2a). We discover that the proportion of zeros i.e. *number of pixels belonging to the background in an image is a confounding factor to the likelihood score* (Pearson Correlation Coefficient 0.85, see Figure 2b, Figure S1). Taking the likelihood ratio between the original and the background models, we see that the AUROC improves significantly from 0.091 to 0.996 (Figure 2d). The log likelihood-ratio for OOD images are highly concentrated around value 0, while that for in-distribution images are mostly positive (Figure 2c).

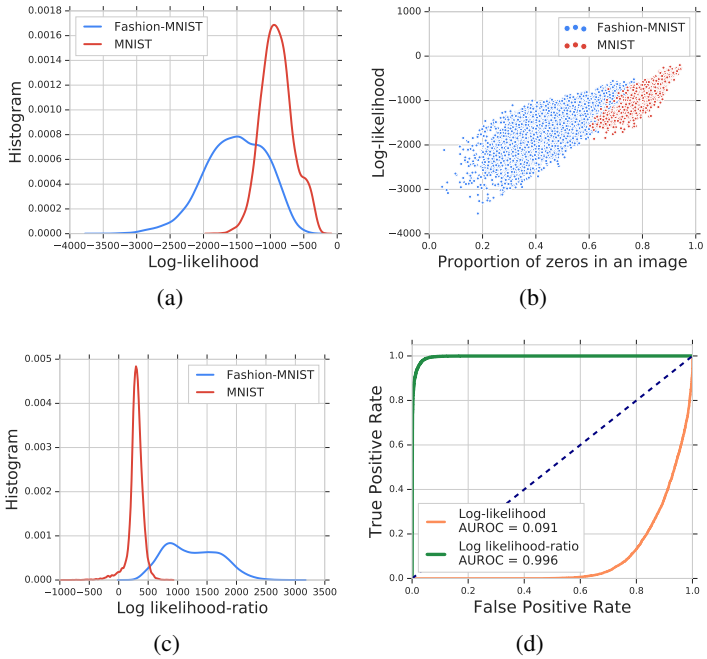
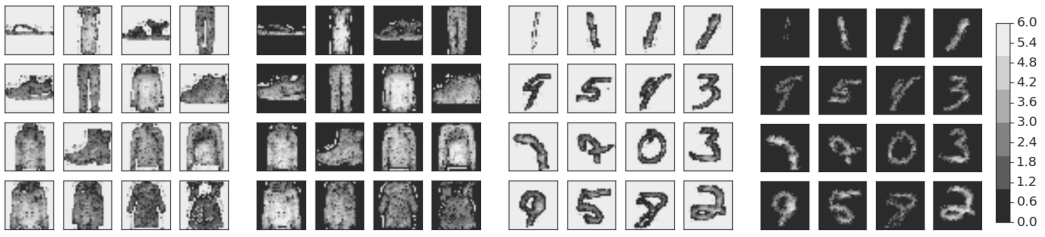


Figure 2: (a) Log-likelihood of MNIST images (OOD) is higher than that of Fashion-MNIST images (in-distribution). (b) Log-likelihood is highly correlated with the background (proportion of zeros in an image). (c) Log-likelihood ratio is higher for Fashion-MNIST (in-dist) than MNIST (OOD). (d) Likelihood ratio significantly improves the AUROC of OOD detection from 0.091 to 0.996.

Which pixels contribute the most to the likelihood (ratio)? To qualitatively evaluate the difference between the likelihood and the likelihood ratio, we plot their values for each pixel for Fashion-MNIST and MNIST images. This allows us to visualize which pixels contribute the most to the two terms respectively. Figure 3 shows a heatmap, with lighter (darker) gray colors indicating higher (lower) values. Figures 3(a,b) show that the likelihood value is dominated by the "background" pixels, whereas likelihood ratio focuses on the "semantic" pixels. Figures 3(c,d) confirm that the background pixels cause MNIST images to be assigned high likelihood, whereas likelihood ratio focuses on the semantic pixels. We present additional qualitative results in Appendix B. For instance, Figure S2 shows that images with the highest likelihood-ratios are those with prototypical Fashion-MNIST icons, e.g. "shirts" and "bags", highly contrastive with the background, while images with the lowest likelihood-ratios are those with rare patterns, e.g. dress with stripes or sandals with high ropes.



(a) Likelihood (b) Likelihood-Ratio (c) Likelihood (d) Likelihood-Ratio

Figure 3: The log-likelihood of each pixel in an image $\log p_{\boldsymbol{\theta}}(x_d|\boldsymbol{x}_{<d})$, and the log likelihood-ratio of each pixel $\log p_{\boldsymbol{\theta}}(x_d|\boldsymbol{x}_{<d}) - \log p_{\boldsymbol{\theta}_0}(x_d|\boldsymbol{x}_{<d}), d = 1 \ldots, 784.$, for 16 Fashion-MNIST images (a, b) and MNIST images (c, d). Lighter gray color indicates larger value (see colorbar). Note that the range of log-likelihood (negative value) is different from that of log likelihood-ratio (mostly positive value). For the ease of visualization, we unify the colorbar by adding a constant to the log-likelihood score. The images are randomly sampled from the test dataset and sorted by their likelihood $p_\theta(\mathbf{x})$. Looking at which pixels contribute the most to each quantity, we observe that the likelihood value is dominated by the "background" pixels on both Fashion-MNIST and MNIST, whereas likelihood ratio focuses on the "semantic" pixels.

We compare our method with other baselines. The classifier-based baseline methods are built using LeNet architecture. Table 1a shows that our method achieves the highest AUROC↑, AUPRC↑, and the lowest FPR80↓. The method using Mahalanobis distance performs better than other baselines. Note that the binary classifier between in-distribution and perturbed in-distribution does not perform as well as our method, possibly due to the fact that while the features learned by the discriminator can be good for detecting perturbed inputs, they may not generalize well for OOD detection. The generative model approach based on $p(\boldsymbol{x})$ captures more fundamental features of the data generation process than the discriminative approach.

For the experiment using CIFAR-10 as in-distribution and SVHN as OOD, we apply the same training procedure using the PixelCNN++ model and choose hyperparameters using grayscaled CIFAR-10 which was shown to be OOD by Nalisnick et al. (2018). See Appendix B.1 for model details. Looking at the results in Table 2, we observe that the OOD images from SVHN have higher likelihood than the in-distribution images from CIFAR-10, confirming the observations of Nalisnick et al. (2018), with AUROC of 0.095. Our likelihood-ratio method significantly improves the AUROC to 0.931. Figure S3 in Appendix B shows additonal qualitative results. For detailed results including other baseline methods, see Table S2 in Appendix B.3.

5.2 OOD detection for genomic sequences

Dataset for detecting OOD genomic sequences We design a new dataset for evaluating OOD methods. As bacterial classes are discovered gradually over time, in- and out-of-distribution data can be naturally separated by the time of discovery. Classes discovered before a cutoff time can be regarded as in-distribution classes, and those discovered afterward, which were unidentified at the cutoff time, can be regarded as OOD. We choose two cutoff years, 2011 and 2016, to define the training, validation, and test splits (Figure 4). Our dataset contains of 10 in-distribution classes, 60 OOD classes for validation, and 60 OOD classes for testing. Note that the validation OOD dataset is only used for hyperparameter tuning, and the validation OOD classes are disjoint from the test OOD

Table 1: AUROC↑, AUPRC↑, and FPR80↓ for detecting OOD inputs using likelihood and likelihood-ratio method and other baselines on (a) Fashion-MNIST vs. MNIST datasets and (b) genomic dataset. The up and down arrows on the metric names indicate whether greater or smaller is better. μ in the parentheses indicates the background model is tuned only using noise perturbed input, and (μ and λ) indicates the background model is tuned by both perturbation and L_2 regularization. Numbers in front and inside of the brackets are mean and standard error respectively based on 10 independent runs with random initialization of network parameters and random shuffling of training inputs. For ensemble models, the mean and standard error are estimated based on 10 bootstrap samples from 30 independent runs, which can be underestimations of the true standard errors.

	AUROC↑	AUPRC↑	FPR80↓
Likelihood	0.089 (0.002)	0.320 (0.000)	1.000 (0.001)
Likelihood Ratio (ours, μ)	0.973 (0.031)	0.951 (0.063)	0.005 (0.008)
Likelihood Ratio (ours, μ, λ)	**0.994 (0.001)**	**0.993 (0.002)**	**0.001 (0.000)**
$p(\hat{y}\|\boldsymbol{x})$	0.734 (0.028)	0.702 (0.026)	0.506 (0.046)
Entropy of $p(y\|\boldsymbol{x})$	0.746 (0.027)	0.726 (0.026)	0.448 (0.049)
ODIN	0.752 (0.069)	0.763 (0.062)	0.432 (0.116)
Mahalanobis distance	0.942 (0.017)	0.928 (0.021)	0.088 (0.028)
Ensemble, 5 classifiers	0.839 (0.010)	0.833 (0.009)	0.275 (0.019)
Ensemble, 10 classifiers	0.851 (0.007)	0.844 (0.006)	0.241 (0.014)
Ensemble, 20 classifiers	0.857 (0.005)	0.849 (0.004)	0.240 (0.011)
Binary classifier	0.455 (0.105)	0.505 (0.064)	0.886 (0.126)
$p(\hat{y}\|\boldsymbol{x})$ with noise class	0.877 (0.050)	0.871 (0.054)	0.195 (0.101)
$p(\hat{y}\|\boldsymbol{x})$ with calibrations	0.904 (0.023)	0.895 (0.023)	0.139 (0.044)
WAIC, 5 models	0.221 (0.013)	0.401 (0.008)	0.911 (0.008)

(a)

	AUROC↑	AUPRC↑	FPR80↓
Likelihood	0.626 (0.001)	0.613 (0.001)	0.661 (0.002)
Likelihood Ratio (ours, μ)	0.732 (0.015)	0.685 (0.017)	0.534 (0.031)
Likelihood Ratio (ours, μ, λ)	**0.755 (0.005)**	**0.719 (0.006)**	**0.474 (0.011)**
$p(\hat{y}\|\boldsymbol{x})$	0.634 (0.003)	0.599 (0.003)	0.669 (0.007)
Entropy of $p(y\|\boldsymbol{x})$	0.634 (0.003)	0.599 (0.003)	0.617 (0.007)
Adjusted ODIN	0.697 (0.010)	0.671 (0.012)	0.550 (0.021)
Mahalanobis distance	0.525 (0.010)	0.503 (0.007)	0.747 (0.014)
Ensemble, 5 classifiers	0.682 (0.002)	0.647 (0.002)	0.589 (0.004)
Ensemble, 10 classifiers	0.690 (0.001)	0.655 (0.002)	0.574 (0.004)
Ensemble, 20 classifiers	0.695 (0.001)	0.659 (0.001)	0.570 (0.004)
Binary classifier	0.635 (0.016)	0.634 (0.015)	0.619 (0.025)
$p(\hat{y}\|\boldsymbol{x})$ with noise class	0.652 (0.004)	0.627 (0.005)	0.643 (0.008)
$p(\hat{y}\|\boldsymbol{x})$ with calibration	0.669 (0.005)	0.635 (0.004)	0.627 (0.006)
WAIC, 5 models	0.628 (0.001)	0.616 (0.001)	0.657 (0.002)

(b)

Table 2: CIFAR-10 vs SVHN results: AUROC↑, AUPRC↑, FPR80↓ for detecting OOD inputs using likelihood and our likelihood-ratio method.

	AUROC↑	AUPRC↑	FPR80↓
Likelihood	0.095 (0.003)	0.320 (0.001)	1.000 (0.000)
Likelihood Ratio (ours, μ)	0.931 (0.032)	0.888 (0.049)	0.062 (0.073)
Likelihood Ratio (ours, μ, λ)	0.930 (0.042)	0.881 (0.064)	0.066 (0.123)

classes. To mimic sequencing data, we fragmented genomes in each class into short sequences of 250 base pairs, which is a common length that current sequencing technology generates. Among all the short sequences, we randomly choose 100,000 sequences for each class for training, validation, and test. Additional details about the dataset, including pre-processing and the information for the in- and out-of-distribution classes, can be found in Appendix C.1.

Likelihood ratio method for detecting OOD sequences We build an LSTM model for estimating the likelihood $p(\boldsymbol{x})$ based on the transition probabilities $p(x_d|\boldsymbol{x}_{<d})$, $d = 1, \ldots, D$. In particular, we feed the one-hot encoded DNA sequences into an LSTM layer, followed by a dense layer and a softmax function to predict the probability distribution over the 4 letters of $\{A, C, G, T\}$, and train the model using only the in-distribution training data. We evaluate the likelihood for sequences in the OOD test dataset under the trained model, and compare those with the likelihood for sequences in the in-distribution test dataset. The AUROC↑, AUPRC↑, and FPR80↓ scores are 0.626, 0.613, and 0.661 respectively (Table 1b).

We train a background model by using the perturbed in-distribution data and optionally adding L_2 regularization to the model weights. Hyperparameters are tuned using validation dataset which contains in-distribution and validation OOD classes, and the validation OOD classes are disjoint from test OOD classes. Contrasting against the background model, the AUROC↑, AUPRC↑, and FPR80↓ for the likelihood-ratio significantly improve to 0.755, 0.719, and 0.474, respectively (Table 1b, Figure 5b). Compared with the likelihood, the AUROC and AUPRC for likelihood-ratio increased 20% and 17% respectively, and the FPR80 decreased 28%. Furthermore, Figure 5a shows that the likelihood ratio is less sensitive to GC-content, and the separation between in-distribution and OOD distribution becomes clearer. We evaluate other baseline methods on the test dataset as well. For classifier-based baselines, we construct CNNs with one convolutional layer, one max-pooling layer, one dense layer, and a final dense layer with softmax activation for predicting class probabilities, as in Alipanahi et al. (2015); Busia et al. (2018); Ren et al. (2018b). Comparing our method to the baselines in Table 1b, our method achieves the highest AUROC, AUPRC, and the lowest FPR80 scores on the test dataset. Ensemble method and ODIN perform better than other baseline methods. Comparing with the Fashion-MNIST and MNIST experiment, the Mahalanobis distance performs worse for detecting genomic OOD possibly due to the fact that Fashion-MNIST and MNIST images are quite distinct while in-distribution and OOD bacteria classes are interlaced under the same taxonomy (See Figure S5 for the phylogenetic tree of the in-distribution and OOD classes).

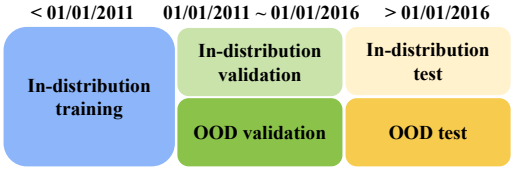
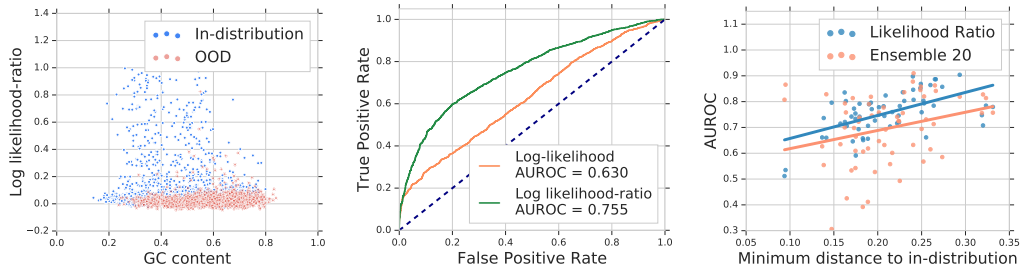


Figure 4: The design of the training, validation, and test datasets for genomic sequence classification including in and OOD data.



(a) (b) (c)

Figure 5: (a) The likelihood-ratio score is roughly independent of the GC-content which makes it less susceptible to background statistics and better suited for OOD detection. (b) ROCs and AUROCs for OOD detection using likelihood and likelihood-ratio. (c) Correlation between the AUROC of OOD detection and distance to in-distribution classes using Likelihood Ratio and the Ensemble method.

OOD detection correlates with its distance to in-distribution We investigate the effect of the distance between the OOD class to the in-distribution classes, on the performance of OOD detection. To measure the distance between the OOD class to the in-distribution, we randomly select representative genome from each of the in-distribution classes and OOD classes. We use the state-of-the-art alignment-free method for genome comparison, d_2^S (Ren et al., 2018a; Reinert et al., 2009), to compute the genetic distance between each pair of the genomes in the set. This genetic distance is calculated based on the similarity between the normalized nucleotide word frequencies (k-tuples) of the two genomes, and studies have shown that this genetic distance reflects true evolutionary distances between genomes (Chan et al., 2014; Bernard et al., 2016; Lu et al., 2017). For each of the OOD classes, we use the minimum distance between the genome in that class to all genomes in the in-distribution classes as the measure of the genetic distance between this OOD class and the in-distribution. Not surprisingly, the the AUROC for OOD detection is positively correlated with the genetic distance (Figure 5c), and an OOD class far away from in-distribution is easier to be detected. Comparing our likelihood ratio method and one of the best classifier-based methods, ensemble method, we observe that our likelihood ratio method has higher AUROC for different OOD classes than ensemble method in general. Furthermore, our method has a higher Pearson correlation coefficient (PCC) of 0.570 between the minimum distance and AUROC for Likelihood Ratio method, than the classifier-based ensemble method with 20 models which has PCC of 0.277. The dataset and code for the genomics study is available at `https://github.com/google-research/google-research/tree/master/genomics_ood`.

6 Discussion and Conclusion

We investigate deep generative model-based methods for OOD detection and show that the likelihood of auto-regressive models can be confounded by background statistics, providing an explanation to the failure of PixelCNN for OOD detection observed by recent work (Nalisnick et al., 2018; Hendrycks et al., 2018; Shafaei et al., 2018). We propose a likelihood ratio method that alleviates this issue by contrasting the likelihood against a background model. We show that our method effectively corrects for the background components, and significantly improves the accuracy of OOD detection on both image datasets and genomic datasets. Finally, we create and release a realistic genomic sequence dataset for OOD detection which highlights an important real-world problem, and hope that this serves as a valuable OOD detection benchmark for the research community.

Acknowledgments

We thank Alexander A. Alemi, Andreea Gane, Brian Lee, D. Sculley, Eric Jang, Jacob Burnim, Katherine Lee, Matthew D. Hoffman, Noah Fiedel, Rif A. Saurous, Suman Ravuri, Thomas Colthurst, Yaniv Ovadia, the Google Brain Genomics team, and Google TensorFlow Probability team for helpful feedback and discussions.

References

Ahlgren, N. A., Ren, J., Lu, Y. Y., Fuhrman, J. A., and Sun, F. Alignment-free oligonucleotide frequency dissimilarity measure improves prediction of hosts from metagenomically-derived viral sequences. *Nucleic acids research*, 45(1):39–53, 2016.

Alemi, A. A., Fischer, I., and Dillon, J. V. Uncertainty in the variational information bottleneck. *arXiv preprint arXiv:1807.00906*, 2018.

Alipanahi, B., Delong, A., Weirauch, M. T., and Frey, B. J. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature Biotechnology*, 33(8):831, 2015.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

Bailey, T. L. and Elkan, C. The value of prior knowledge in discovering motifs with MEME. In *ISMB*, volume 3, pp. 21–29, 1995.

Bernard, G., Chan, C. X., and Ragan, M. A. Alignment-free microbial phylogenomics under scenarios of sequence divergence, genome rearrangement and lateral genetic transfer. *Scientific Reports*, 6: 28970, 2016.

Bishop, C. M. Novelty Detection and Neural Network Validation. *IEEE Proceedings-Vision, Image and Signal processing*, 141(4):217–222, 1994.

Bishop, C. M. Regularization and complexity control in feed-forward networks. In *Proceedings International Conference on Artificial Neural Networks ICANN*, volume 95, pp. 141–148, 1995a.

Bishop, C. M. Training with noise is equivalent to Tikhonov regularization. *Neural computation*, 7 (1):108–116, 1995b.

Blauwkamp, T. A., Thair, S., Rosen, M. J., Blair, L., Lindner, M. S., Vilfan, I. D., Kawli, T., Christians, F. C., Venkatasubrahmanyam, S., Wall, G. D., et al. Analytical and clinical validation of a microbial cell-free dna sequencing test for infectious disease. *Nature Microbiology*, 4(4):663, 2019.

Brady, A. and Salzberg, S. L. Phymm and PhymmBL: metagenomic phylogenetic classification with interpolated Markov models. *Nature Methods*, 6(9):673, 2009.

Bulatov, Y. NotMNIST dataset, 2011. URL http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html.

Busia, A., Dahl, G. E., Fannjiang, C., Alexander, D. H., Dorfman, E., Poplin, R., McLean, C. Y., Chang, P.-C., and DePristo, M. A deep learning approach to pattern recognition for short DNA sequences. *bioRxiv*, pp. 353474, 2018.

Chan, C. X., Bernard, G., Poirion, O., Hogan, J. M., and Ragan, M. A. Inferring phylogenies of evolving sequences without multiple sequence alignment. *Scientific reports*, 4:6504, 2014.

Choi, H., Jang, E., and Alemi, A. A. WAIC, but why? Generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

Eckburg, P. B., Bik, E. M., Bernstein, C. N., Purdom, E., Dethlefsen, L., Sargent, M., Gill, S. R., Nelson, K. E., and Relman, D. A. Diversity of the human intestinal microbial flora. *Science*, 308 (5728):1635–1638, 2005.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

Gupta, A., Müller, A. T., Huisman, B. J., Fuchs, J. A., Schneider, P., and Schneider, G. Generative recurrent networks for de novo drug design. *Molecular informatics*, 37(1-2):1700111, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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

Hendrycks, D., Mazeika, M., and Dietterich, T. G. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

Hildebrand, F., Meyer, A., and Eyre-Walker, A. Evidence of selection upon genomic GC-content in bacteria. *PLoS genetics*, 6(9):e1001107, 2010.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997.

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

Killoran, N., Lee, L. J., Delong, A., Duvenaud, D., and Frey, B. J. Generating and designing DNA with deep generative models. *arXiv preprint arXiv:1712.06148*, 2017.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*, 2018.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, K., Lee, H., Lee, K., and Shin, J. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325*, 2017.

Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS*, 2018.

Liang, S., Li, Y., and Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

Lu, Y. Y., Tang, K., Ren, J., Fuhrman, J. A., Waterman, M. S., and Sun, F. CAFE: a C elerated A lignment-F r E e sequence analysis. *Nucleic Acids Research*, 45(W1):W554–W559, 2017.

Luhn, H. P. Key word-in-context index for technical literature (kwic index). *American Documentation*, 11(4):288–295, 1960.

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

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. Hybrid models with deep and invertible features. In *ICML*, 2019.

Nayfach, S., Shi, Z. J., Seshadri, R., Pollard, K. S., and Kyrpides, N. C. New insights from uncultivated genomes of the global human gut microbiome. *Nature*, pp. 1, 2019.

Nguyen, A., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 427–436, 2015.

Olivecrona, M., Blaschke, T., Engkvist, O., and Chen, H. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017.

Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J. V., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019.

Patil, K. R., Haider, P., Pope, P. B., Turnbaugh, P. J., Morrison, M., Scheffer, T., and McHardy, A. C. Taxonomic metagenome sequence assignment with structured output models. *Nature Methods*, 8 (3):191, 2011.

Ponsero, A. J. and Hurwitz, B. L. The promises and pitfalls of machine learning for detecting viruses in aquatic metagenomes. *Frontiers in Microbiology*, 10:806, 2019.

Reinert, G., Chew, D., Sun, F., and Waterman, M. S. Alignment-free sequence comparison (I): statistics and power. *Journal of Computational Biology*, 16(12):1615–1634, 2009.

Ren, J., Bai, X., Lu, Y. Y., Tang, K., Wang, Y., Reinert, G., and Sun, F. Alignment-free sequence analysis and applications. *Annual Review of Biomedical Data Science*, 1:93–114, 2018a.

Ren, J., Song, K., Deng, C., Ahlgren, N. A., Fuhrman, J. A., Li, Y., Xie, X., and Sun, F. Identifying viruses from metagenomic data by deep learning. *arXiv preprint arXiv:1806.07810*, 2018b.

Rosen, G. L., Reichenberger, E. R., and Rosenfeld, A. M. NBC: the naive Bayes classification tool webserver for taxonomic classification of metagenomic reads. *Bioinformatics*, 27(1):127–129, 2010.

Salimans, T., Karpathy, A., Chen, X., Kingma, D. P., and Bulatov, Y. PixelCNN++: A PixelCNN implementation with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017.

Shafaei, A., Schmidt, M., and Little, J. J. Does your model know the digit 6 is not a cat? a less biased evaluation of" outlier" detectors. *arXiv preprint arXiv:1809.04729*, 2018.

Sueoka, N. On the genetic basis of variation and heterogeneity of DNA base composition. *Proceedings of the National Academy of Sciences*, 48(4):582–592, 1962.

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. Conditional image generation with PixelCNN decoders. In *NeurIPS*, 2016.

Wagstaff, K. L. Machine learning that matters. In *ICML*, 2012.

Yarza, P., Richter, M., Peplies, J., Euzeby, J., Amann, R., Schleifer, K.-H., Ludwig, W., Glöckner, F. O., and Rosselló-Móra, R. The All-Species Living Tree project: a 16S rRNA-based phylogenetic tree of all sequenced type strains. *Systematic and Applied Microbiology*, 31(4):241–250, 2008.

Zhou, J. and Troyanskaya, O. G. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature Methods*, 12(10):931, 2015.

Zhu, Z., Ren, J., Michail, S., and Sun, F. Metagenomic unmapped reads provide important insights into human microbiota and disease associations. *bioRxiv*, pp. 504829, 2018.

Zou, J., Huss, M., Abid, A., Mohammadi, P., Torkamani, A., and Telenti, A. A primer on deep learning in genomics. *Nature Genetics*, pp. 1, 2018.

Appendix

A Additional details about our proposed likelihood ratio method

The pseudocode for generating input perturbations for training the background model is described in Algorithm 1.

Algorithm 1 Input perturbation for background model training

Inputs: D-dimensional input $\boldsymbol{x} = x_1 \dots x_D$, Mutation rate μ and vocabulary $\mathcal{A}$. Note that we assume inputs to be discrete, i.e. $x_d \in \mathcal{A}$, where $\mathcal{A} = \{A, C, G, T\}$ for genomic sequences and $\mathcal{A} = \{0, \dots, 255\}$ for images.
Output: perturbed input $\tilde{\boldsymbol{x}}$
Generate a D-dimensional vector $\mathbf{v} = v_1 \dots, v_D$, where $v_d \in \{0, 1\}$ are independent and identically distributed according to a Bernoulli distribution with rate μ.
for index $d \in \{1, \dots, D\}$ **do**
  if $v_d = 1$ **then**
    Sample $\tilde{x}_d$ from the set $\mathcal{A}$ with equal probability.
  else
    Set $\tilde{x}_d = x_d$.
  end if
end for

The complete pseudocode for our method is described in Algorithm 2. The runtime of our method is two times of the standard generative model runtime.

Algorithm 2 OOD detection using Likelihood Ratio

Fit a model $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ using in-distribution dataset $\mathcal{D}_{\text{in}}$.
Fit a background model $p_{\boldsymbol{\theta}_0}(\boldsymbol{x})$ using perturbed input data $\widetilde{\mathcal{D}}_{\text{in}}$ (generated using Algorithm 1) and (optionally) model regularization techniques.
Compute the likelihood ratio statistic $\mathsf{LLR}(\boldsymbol{x})$
Predict OOD if $\mathsf{LLR}(\boldsymbol{x})$ is small.

B Supplementary materials for the experiments on images

B.1 Model details

Following existing literature (Nalisnick et al., 2018; Choi et al., 2018), we evaluate our method using two experiments for detecting OOD images: (a) Fashion-MNIST as in-distribution and MNIST as OOD, (b) CIFAR-10 as in-distribution and SVHN as OOD. For the experiment of Fashion-MNIST vs. MNIST, we train a generative model using the training set of Fashion-MNIST, and use the test set of Fashion-MNIST and the test set of MNIST as OOD as the final test dataset. The same rule is applied for CIFAR-10 vs. SVHN experiment. Since SVHN test set has more inputs than CIFAR-10 test set, we randomly select the same number of inputs for evalaution.

For each experiment, we train a PixelCNN++ (Salimans et al., 2017; Van den Oord et al., 2016) model on the in-distribution data using maximum likelihood. For Fashion-MNIST dataset, the model uses 2 blocks of 5 gated ResNet layers with 32 convolutional 2D filters (concatenate ELU activation function, and without weight normalization and dropout), and 1 component in the logitstic mixture, and is trained for 50,000 steps with initial learning rate of 0.0001 with exponential decay rate of 0.999995 per step, batch size of 32, and Adam optimizer with momentum parameter $\beta_1 = 0.95$ and $\beta_2 = 0.9995$. For CIFAR-10 dataset, the model uses 2 blocks of 5 gated ResNet layers with 160 convolutional 2D filters, and 10 components in the logitstic mixture, and is trained for 600,000 steps with the same learning rate, batch size, and optimizer as the above. The bits per dimension of the

PixelCNN++ models are 2.92 and 1.64 for in-distribution images Fashion-MNIST and OOD images MNIST respectively, and 3.20 and 2.15 for in-distribution images CIFAR-10 and OOD images SVHN, respectively.

For the background model, we train a PixelCNN++ model with the same architecture on perturbed inputs obtained by randomly flipping input pixel values to one of the 256 possible values following an independent and identical Bernoulli distribution with rate μ (see Algorithm 1). The mutation rate μ for adding perturbations to input in the background model training is a hyperparameter to our method. For tuning the hyperparameters, we use independent datasets, NotMNIST (Bulatov, 2011) for Fashion-MNIST vs. MNIST experiment, and gray-scaled CIFAR-10 for CIFAR-10 vs. SVHN experiment, as the validation OOD dataset. We choose the optimal mutation rate μ based on the AUROC for OOD detection using the validation dataset. We test the mutation rate μ from the range of $[0.1, 0.2, 0.3]$. For Fashion-MNIST vs. MNIST, the optimal mutation rate tuned using NotMNIST as OOD dataset is $\mu = 0.3$ for most of the 10 independent runs. We also add L_2 regularization to model weights. We let the L_2 coefficient λ ranges from $\lambda = [0, 10, 100]$ and test different combinations of μ and λ on the background model training. We find that adding L_2 regularization helps to improve the final test AUROC only slightly from 0.973 to 0.996. The optimal combination is $\mu = 0.3$ and $\lambda = 100$ for most of the 10 independent runs of random initialization of network parameters and random shuffling of training inputs. Table S1a shows one of those, which results in AUROC of 0.996 in the final test dataset. For CIFAR-10 vs. SVHN experiment, we observe that tuning for both mutation rate and L_2 regularization achieves similar performance with tuning for mutation rate only. The optimal mutation rate is $\mu = 0.1$ for most of the 10 independent runs (data not shown).

In the case where no independent OOD data (such as NotMNIST) is available for hyperparameter tuning, we can use randomly mutated in-distribution input at mutation rate 2%, to mimic the OOD input. The mutation is added using the same procedure as that for training the background model. The optimal hyperparameter setting is $\mu = 0.1$ and $\lambda = 100$, which achieves AUROC of 0.958 in the final test dataset. The results show that the hyperparameters for the background model are easy to tune. Under the situation where indepedent OOD data are not available, using only simulated OOD data we are able to achieve reasonable performance.

We found that the performance of our method LLR can be slightly affected by different PixelCNN++ network setups. We tested both versions of PixelCNN++ where input images ranging from 0 to 255 are (a) directly fed into the networks (b) re-scaled to the range of -1 to 1 and fed into the networks. The two setups give different initializatiobs of the network. We found that the version without rescaling achieves slightly better performance in terms of AUROC possibly because it learns a better background model based on its initialization. The AUROC↑, AUPRC↑, and FPR80↓ for Fashion-MNIST vs. MNIST in Table 1a are based on the version without rescaling. Using the version with rescaling, the numbers for the three evaluation metrics are 0.936 (0.003), 0.891 (0.004), 0.025 (0.003), without changing model parameters (the same 5 gated ResNet with 32 convolutional 2D filters, batch size, learning rate, optimizer as before), and training for 600,000 steps. For CIFAR-10 vs. SVHN experiment, we found rescaling helps to produce more stable results. So we report AUROC↑, AUPRC↑, and FPR80↓ based on the version with rescaling in Table S2.

Table S1: Hyperparameter tuning of mutation rate μ and L_2 coefficient λ of the background model of our likelihood ratio method for Fashion-MNIST vs. MNIST experiment. (a) AUROC is evaluated based on in-distribution Fashion-MNIST validation dataset and NotMNIST dataset. Note that MNIST is not used for hyperparameter turning. (b) The same as (a) but tuning using simulated OOD inputs without exposing to any NotMNIST or MNIST images. The simulated OOD inputs are generated by permuting the in-distribution inputs at the mutation rate 2%.

	$\mu = 0.1$	0.2	0.3
$\lambda = 0$	0.358	0.426	**0.795**
10	0.433	0.432	0.489
100	0.414	0.777	**0.798**

	$\mu = 0.1$	0.2	0.3
$\lambda = 0$	**0.984**	0.971	0.856
10	**0.989**	0.977	0.971
100	**0.989**	0.851	0.857

To compare with classifier-based baseline methods, we build convolutional neural networks (CNNs). We used LeNet (LeCun et al., 1998) architecture for Fashion-MNIST images. The model composes two convolutional layers with 32 and 64 2D filters respectively of size 3 by 3 with ReLU activation function, a max pooling layers of size 2 by 2, a dropout layer with the rate of 0.25, a dense layer of

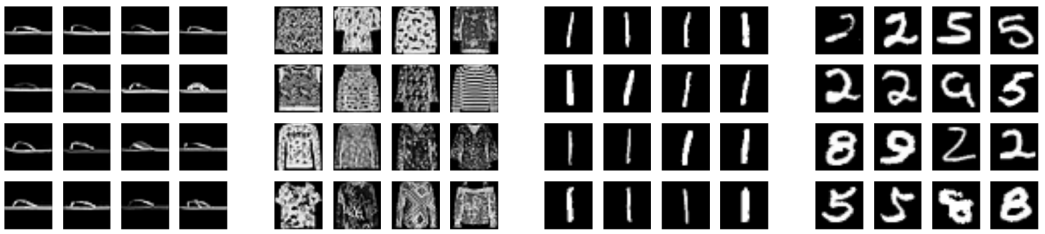
128 units with ReLU activation function, another dropout layer with the rate of 0.25 based on the flattened output from the previous layer, and a final dense layer with the softmax activation function used for generating the class probabilities. Model parameters are trained for 12 epochs with batch size of 128, learning rate of 0.002, and Adam optimizer. The prediction accuracy on test data is 0.923 for in-distribution Fashion-MNIST images. For CIFAR-10 images, we build ResNet-20 V1 architecture with ReLU activations (He et al., 2016). Model parameters are trained for 120 epochs with batch size of 7, Adam optimizer, and a learning rate schedule that multiplied an initial learning rate of 0.0007 by 0.1, 0.01, 0.001, and 0.0005 at steps 80, 120, 160, and 180 respectively. Training inputs are randomly distorted using horizontal flips and random crops preceded by 4-pixel padding as described in He et al. (2016). The prediction accuracy on test data is 0.911 for in-distribution CIFAR-10 images. Both LeNet and ResNet-20 V1 model architectures are consistent with those in Ovadia et al. (2019).

For the baseline methods 6-8 that are based on perturbed inputs, the perturbation rate is tuned from the range of $[10^{-5}, 10^{-4}, \ldots, 10^{-1}]$ based on validation in-distribution dataset and an independent dataset that is different from the final OOD test dataset, e.g. NotMNIST for Fashion-MNIST vs. MNIST experiment, and grayscaled CIFAR-10 for CIFAR-10 vs. SVHN experiment. Similarly the hyperparameters in ODIN method, the temperature scaling to logits and perturbations to inputs, are tuned based on the same validation dataset. The temperature is tuned in the range of $[1, 5, 10, 100, 1000]$, and the perturbation is tuned in the range of $[0, 10^{-8}, 10^{-7}, \ldots, 10^{-1}]$.

B.2 Supplementary figures

Images with the highest and lowest likelihood in Fashion-MNIST and MNIST dataset are shown in Figure S1. Images with the highest likelihoods are mostly "sandals" in Fashion-MNIST dataset and "1"s in MNIST dataset that have a large proportion of zeros. Images with the highest and lowest likelihood-ratio are shown in Figure S2. Images with the highest likelihood-ratios are those with prototypical Fashion-MNIST icons such as "shirts" and "bags", highly contrastive with the background, while images with the lowest likelihood-ratios are those with rare patterns, such as dress with stripes or sandals with high ropes.

Figure S3 shows qualitative results on CIFAR-10, displaying the per-pixel likelihood and likelihood-ratio as a heatmap. Similar to the trends in Figure 3, we observe that "background" pixels cause some CIFAR-10 images to be assigned high likelihood.

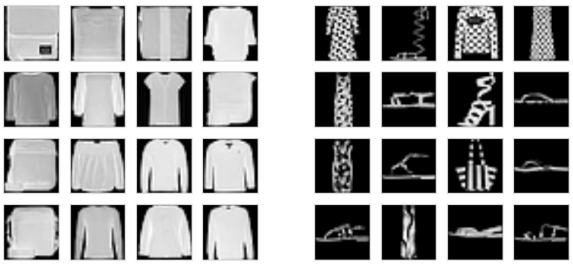


(a) FashionMNIST: highest log-likelihood. (b) FashionMNIST: lowest log-likelihood. (c) MNIST: highest log-likelihood. (d) MNIST: lowest log-likelihood.

Figure S1: FashionMNIST images with (a) the highest log-likelihood, and (b) the lowest log-likelihood. MNIST images with (c) the highest log-likelihood, and (d) the lowest log-likelihood.

B.3 Supplementary tables

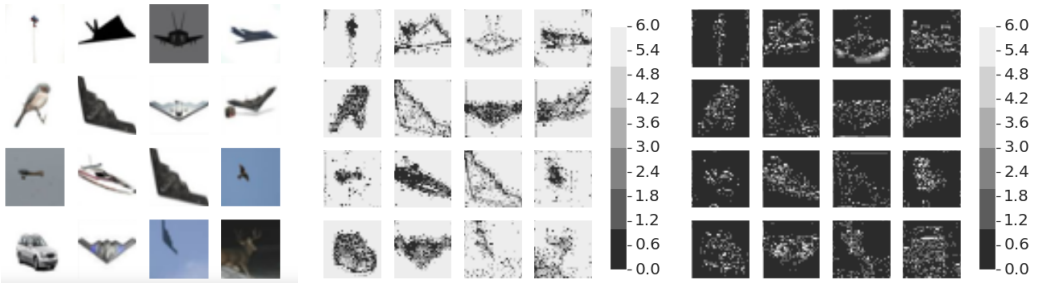
The AUROC↑, AUPRC↑, and FPR80↓ for CIFAR-10 (in-distribution) vs. SVHN (OOD) experiment is shown in Table S2. Pure likelihood performs poorly for OOD detection with AUROC 0.095, and likelihood-ratio improves the performance significantly, achieving AUROC 0.931. Classifier-based ensemble methods perform well with AUROCs ranging from 0.937 to 0.946. Note that our likelihood-ratio method is completely unsupervised whereas classifier methods require labels. Using likelihood-ratios on class-conditional generative models might further improve performance.



(a) FashionMNIST: highest log-likelihood ratio.

(b) FashionMNIST: lowest log-likelihood ratio.

Figure S2: FashionMNIST images with (a) the highest and (b) the lowest log-likelihood-ratios.



(a) (b) (c)

Figure S3: Examples of CIFAR-10 images (a), and their corresponding log-likelihood of each pixel in an image $\log p_{\boldsymbol{\theta}}(x_d|\boldsymbol{x}_{<d})$ (b), and the log likelihood-ratio of each pixel $\log p_{\boldsymbol{\theta}}(x_d|\boldsymbol{x}_{<d}) - \log p_{\boldsymbol{\theta}_0}(x_d|\boldsymbol{x}_{<d})$, are plotted as 32×32 images. Lighter (darker) gray color indicates larger (smaller) value. Note that the range of log-likelihood (negative value) is different from that of log likelihood-ratio (mostly positive value). For the ease of visualization, we unify the colorbar by adding a constant to the log-likelihood score. We picked the images that have the highest log-likelihood $p_{\boldsymbol{\theta}}(\mathbf{x})$.

Table S2: AUROC↑, AUPRC↑, FPR80↓ for detecting OOD inputs using likelihood and likelihood-ratio method and other baselines on CIFAR-10 vs. SVHN datasets. Note that the generative-model based approaches (including our LLR method) are completely unsupervised, which puts them at a slight disadvantage, compared to classifier-based methods which have access to labels. Numbers in front and inside of the brackets are mean and standard error respectively based on 10 independent runs with random initialization of network parameters and random shuffling of training inputs. For ensemble models, mean and standard error are estimated based on 10 bootstrap samples from 30 independent runs, which can be underestimations of the true standard errors.

	AUROC↑	AUPRC↑	FPR80↓
Likelihood	0.095 (0.003)	0.320 (0.001)	1.000 (0.000)
Likelihood Ratio (ours, μ)	0.931 (0.032)	0.888 (0.049)	0.062 (0.073)
Likelihood Ratio (ours, μ, λ)	0.930 (0.042)	0.881 (0.064)	0.066 (0.123)
$p(\hat{y}\|\boldsymbol{x})$	0.910 (0.011)	0.871 (0.019)	0.094 (0.030)
Entropy of $p(\hat{y}\|\boldsymbol{x})$	0.920 (0.013)	0.890 (0.021)	0.139 (0.020)
ODIN	0.938 (0.091)	0.926 (0.103)	0.086 (0.179)
Mahalanobis distance	0.728 (0.108)	0.711 (0.118)	0.469 (0.178)
Ensemble, 5 classifiers	0.937 (0.010)	0.906 (0.017)	0.037 (0.013)
Ensemble, 10 classifiers	0.943 (0.004)	0.915 (0.008)	0.023 (0.002)
Ensemble, 20 classifiers	0.946 (0.002)	0.916 (0.004)	0.020 (0.001)
Binary classifier	0.508 (0.027)	0.505 (0.021)	0.948 (0.107)
$p(y\|x)$ with noise class	0.923 (0.011)	0.892 (0.016)	0.064 (0.026)
$p(y\|x)$ with calibration	0.809 (0.043)	0.735 (0.046)	0.396 (0.101)
WAIC, 5 models	0.146 (0.089)	0.343 (0.038)	0.956 (0.062)

C Supplementary materials for the experiments on genomic sequences

C.1 Dataset design

We downloaded 11,672 bacteria genomes from National Center for Biotechnology Information (NCBI, https://www.ncbi.nlm.nih.gov/genome/browse#!/prokaryotes/) on September 2018. For each genome we pooled its taxonomy information from the species level, to the genus, the family, the order, the class, and the phylum level. High taxonomy levels such as the phylum level represents broad classification, while low taxonomy levels like the species and genus give a refined classification. To provide a precise classification, we use different genera as class labels, as has been done in previous studies (Brady & Salzberg, 2009; Ahlgren et al., 2016). We filtered genomes that have missing genus information, or have ambiguous genus names. A genus usually contains genomes from different species, subspecies, or strains.

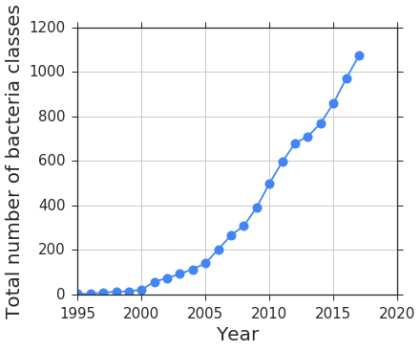


Figure S4: Cumulative number of new bacteria classes discovered over the years (NCBI microbial genomes browser, September 2018.)

Different bacterial classes were discovered gradually over the years (Figure S4). Grouping classes by years is a natural way to mimic the in-distribution and OOD examples. Given a certain cutoff year, the classes discovered before the year cutoff can be regarded as in-distribution classes, and those after the year cutoff can be regarded as the OODs. In particular, we define the year that a class was first discovered as the earliest year when any of the genomes belonging to this class was discovered. We choose two cutoff years, 2011 and 2016, to define the training dataset for in-distribution, validation datasets for in-distribution and OOD, and test datasets for in-distribution and OOD (Figure 4). Genomes belonging to classes that were first discovered before 01/01/2011 are used for generating the training dataset for in-distribution. Genomes belonging to new classes that were first discovered between 01/01/2011 and 01/01/2016 are used for generating the validation dataset for OOD. Genomes belonging to the old classes but sequenced and released between 01/01/2011 and 01/01/2016 are used for generating the validation dataset for in-distribution. Similarly, genomes belonging to the new classes that were first discovered after 01/01/2016 are used for generating test dataset for OOD, while genomes belonging to the old classes that were sequenced and released after 01/01/2016 are used for generating the test dataset for in-distribution. This setting avoids overlaps among genomes from training, validation, and test datasets. It is possible that different bacteria genomes may share similar gene regions, but those are rare for genomes from different genera and hence, we ignore this effect in our study. The bacteria class names are listed in Table S3.

We designed a dataset containing 10 in-distribution classes, 60 OOD classes for validation, and 60 OOD classes for test. The classes were chosen since they are the most common classes and have the largest sample sizes. The in-distribution and OOD classes are interlaced under the same taxonomy (Figure S5). To mimic the real sequencing data, we fragmented genomes in each class into short sequences of length 250 base pairs, which is a common length that the current sequencing technology generates. Among all the short sequences, we randomly choose 100,000 sequences for each class for the training, validation, and test datasets.

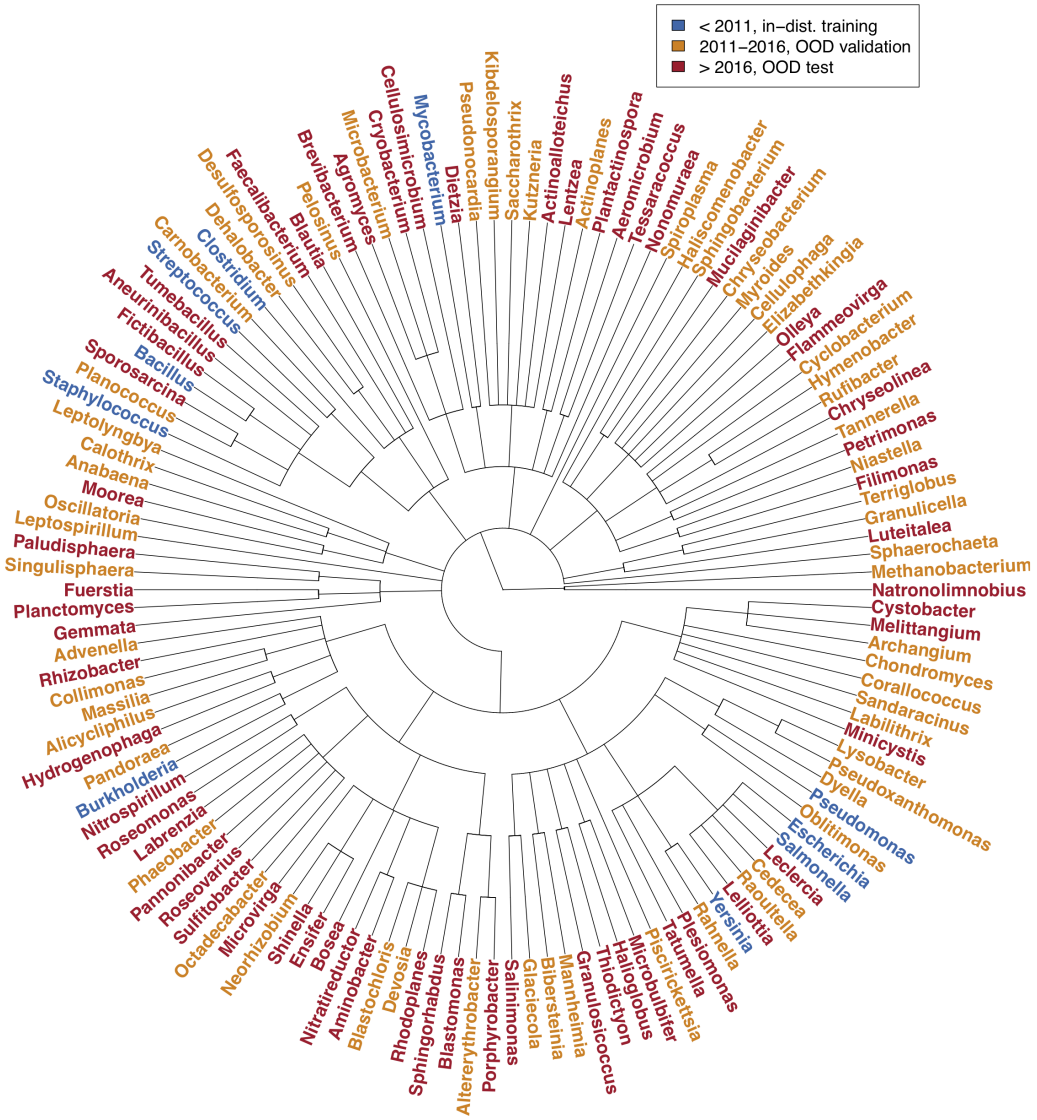


Figure S5: The phylogenetic tree of the 10 in-distribution, 60 OOD validation, and 60 OOD test bacteria classes. Note that the in-distribution and OOD classes are interlaced under the same taxonomy.

C.2 Model details

For generative models of genomic sequences, we build a LSTM model (Hochreiter & Schmidhuber, 1997) to estimate the probability distribution of the next position given the history $p(x_d|\boldsymbol{x}_{<d})$. In particular, we feed the one-hot encoded DNA sequences into an LSTM layer, followed by a dense layer and a softmax function to predict the probability distribution over the 4 letters of $\{A, C, G, T\}$. The model is trained using the in-distribution training data only. The size of the hidden layer in the LSTM was tuned via the in-distribution validation dataset and the final model uses 2,000 hidden units. The model is trained for 900,000 steps using learning rate of 0.0005, batch size of 100, and Adam optimizer. The accuracy for predicting next character is 0.45 for in-distribution inputs.

We train a background model by using the perturbed in-distribution data and (optionally) adding L_2 regularization to model weights. We search the optimal mutation rate μ from the range of $\mu = [0.01, 0.05, 0.1, 0.2]$, and evaluate the AUROC of 2,000 in-distribution and the same number of OOD inputs in the validation dataset. Note that the set of OOD classes in the validation dataset is different

Table S3: The bacterial classes used in the genomic dataset for in- and out-of- distributions.

In-distribution training	Bacillus, Burkholderia, Clostridium, Escherichia, Mycobacterium, Pseudomonas, Salmonella, Staphylococcus, Streptococcus, Yersinia
OOD validation	Actinoplanes, Advenella, Alicycliphilus, Altererythrobacter, Anabaena, Archangium, Bibersteinia, Blastochloris, Calothrix, Carnobacterium, Cedecea, Cellulophaga, Chondromyces, Chryseobacterium, Collimonas, Corallococcus, Cyclobacterium, Dehalobacter, Desulfosporosinus, Devosia, Dyella, Elizabethkingia, Glaciecola, Granulicella, Haliscomenobacter, Hymenobacter, Kibdelosporangium, Kutzneria, Labilithrix, Leptolyngbya, Leptospirillum, Lysobacter, Mannheimia, Massilia, Methanobacterium, Microbacterium, Myroides, Neorhizobium, Niastella, Oblitimonas, Octadecabacter, Oscillatoria, Pandoraea, Pelosinus, Phaeobacter, Piscirickettsia, Planococcus, Pseudonocardia, Pseudoxanthomonas, Rahnella, Raoultella, Rufibacter, Saccharothrix, Sandaracinus, Singulisphaera, Sphaerochaeta, Sphingobacterium, Spiroplasma, Tannerella, Terriglobus
OOD testing	Actinoalloteichus, Aeromicrobium, Agromyces, Aminobacter, Aneurinibacillus, Blastomonas, Blautia, Bosea, Brevibacterium, Cellulosimicrobium, Chryseolinea, Cryobacterium, Cystobacter, Dietzia, Ensifer, Faecalibacterium, Fictibacillus, Filimonas, Flammeovirga, Fuerstia, Gemmata, Granulosicoccus, Halioglobus, Hydrogenophaga, Labrenzia, Leclercia, Lelliottia, Lentzea, Luteitalea, Melittangium, Microbulbifer, Microvirga, Minicystis, Moorea, Mucilaginibacter, Natronolimnobius, Nitratireductor, Nitrospirillum, Nonomuraea, Olleya, Paludisphaera, Pannonibacter, Petrimonas, Planctomyces, Plantactinospora, Plesiomonas, Porphyrobacter, Rhizobacter, Rhodoplanes, Roseomonas, Roseovarius, Salinimonas, Shinella, Sphingorhabdus, Sporosarcina, Sulfitobacter, Tatumella, Tessaracoccus, Thiodictyon, Tumebacillus

from that in the test dataset. We tune hyperparameters without exposure to the final test OOD classes. The optimal μ is 0.2 with AUROC of 0.763 in validation data and 0.727 in the test dataset. We also test if L_2 regularization helps for training the background model. Evaluating AUROC of OOD detection under different combinations of $\mu = [0.01, 0.05, 0.1, 0.2]$ and $\lambda = [0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}]$, we observe that AUROC is generally high for most of the combinations of the two hyperparameters except for some extreme cases (Table S4a), when both μ and λ are too high ($\mu \geq 10^{-3}$ and $\lambda \geq 0.2$) such that the model fails to learn informative patterns, or both are too small ($\mu \leq 10^{-6}$ and $\lambda \leq 0.05$) such that the background model is too similar to the in-distribution specific model. The optimal combination is $\mu = 0.1$ and $\lambda = 10^{-4}$, achieving AUROCs of 0.775 in validation dataset and 0.755 in the test dataset.

Table S4: Hyperparameter tuning of mutation rate μ and L_2 coefficient λ of the background model of our likelihood-raio method for genomic dataset. (a) Effects of the mutation rate μ and L_2 coefficient λ on the AUROC↑ for OOD detection of genomic sequences on the validation dataset containing 2,000 in-distribution and the same number of OOD inputs. When tuning only on mutation rate μ, the optimal value is $\mu = 0.2$. When tuning on both μ and λ, the optimal values are $\mu = 0.1$ and $\lambda = 10^{-4}$. (b) The same as (a) but tuning using simulated OOD inputs. The simulated OOD inputs are generated by permuting the in-distribution inputs at the mutation rate 10%. The trend of the AUROC under different combintations of hyperparameters are similar with that using real OOD inputs.

	$\mu = 0.01$	0.05	0.1	0.2
$\lambda = 0$	0.551	0.664	0.719	**0.763**
10^{-6}	0.694	0.753	0.767	0.767
10^{-5}	0.747	0.761	0.771	0.768
10^{-4}	0.768	0.774	**0.775**	0.764
10^{-3}	0.762	0.755	0.748	0.706

	$\mu = 0.01$	0.05	0.1	0.2
$\lambda = 0$	0.579	0.662	0.744	**0.779**
10^{-6}	0.568	0.653	0.742	0.726
10^{-5}	0.638	0.663	0.689	0.755
10^{-4}	0.749	0.754	**0.797**	0.775
10^{-3}	0.762	0.777	0.750	0.741

We further study if the hyperparameters can be tuned without using the OOD inputs. We use the perturbed in-distribution validation data as simulated OOD inputs. We choose the mutation rate as 10%, because the average identity between bacteria is estimated 96.4% at the genus level, and 90.1% at the family level (Yarza et al., 2008). Using the mutated in-distribution data to mimic OODs, we compute the likelihood-ratio for the in-distribution and the simulated OOD. The optimal ranges of the hyperparameters under which high AUROC are similar with the previous choice based on the real OODs. The optimal mutation rate when tuning without L_2 regularization ($\lambda = 0$), and the optimal combination of the two hyperparameters, are the same as that tuned using real OOD input.

In order to compare with the classifier-based baselines, we build a classifier using convolutional neural networks (CNNs), which are commonly used in both image and genomic sequence classification problems (Alipanahi et al., 2015; Zhou & Troyanskaya, 2015; Busia et al., 2018; Ren et al., 2018b). For genomic sequences, we feed one-hot encoded DNA sequence composed by $\{A, C, G, T\}$ into a convolutional layer, followed by a max-pooling layer and a dense layer. The output is then transformed to class probabilities using a softmax function. The number of filters, the filter size, and the number of neurons in the dense layer were tuned using the in-distribution validation dataset. This resulted in 1,000 convolutional filters of length 20 and 1,000 neurons in the dense layer. The accuracy of the classifier on the validation dataset is 0.8160. Baseline methods 6-8 are based on perturbed in-distribution inputs, so the mutation rate is a hyperparameter for tuning. We use the same validation dataset as above, and tune the mutation rate ranging from $[0.0, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5]$. For ODIN method, we tune the temperature and the input perturbation in the ranges of $[1, 5, 10, 100, 1000]$ and $[0, 0.0001, 0.001, 0.01, 0.1]$, respectively.

C.3 Supplementary tables

Table S5 shows the minimum genetic distance between each of the OOD classes and in-distribution classes and its corresponding AUROC for OOD detection using Likelihood Ratio and classifier-based ensemble method with 20 models. We discovered that the AUROC for OOD detection is correlated with the genetic distance (Figure 5c). The Pearson Correlation Coefficient are 0.570 for Likelihood Ratio method, and 0.277 for the ensemble method. The results confirm that in general a OOD class far away from the in-distribution is easier to be detected.

Table S5: Minimum genetic distance between each of the 60 OOD classes and in-distribution classes and their corresponding AUROCs for OOD detection.

OOD Class	Min distance	AUROC↑ (Likelihood Ratio)	AUROC↑ (Ensemble 20)
Sulfitobacter	0.331	0.779	0.757
Tumebacillus	0.323	0.784	0.814
Blautia	0.320	0.708	0.828
Roseovarius	0.319	0.747	0.756
Moorea	0.293	0.904	0.819
Natronolimnobius	0.273	0.857	0.585
Fuerstia	0.266	0.887	0.841
Chryseolinea	0.265	0.887	0.863
Faecalibacterium	0.264	0.784	0.713
Gemmata	0.260	0.864	0.615
Aneurinibacillus	0.255	0.661	0.731
Olleya	0.253	0.793	0.820
Planctomyces	0.252	0.804	0.568
Nitratireductor	0.250	0.759	0.694
Filimonas	0.249	0.869	0.858
Sphingorhabdus	0.249	0.852	0.845
Mucilaginibacter	0.241	0.807	0.910
Paludisphaera	0.240	0.846	0.632
Petrimonas	0.240	0.898	0.878
Flammeovirga	0.230	0.803	0.836
Granulosicoccus	0.229	0.836	0.756
Minicystis	0.225	0.783	0.493
Labrenzia	0.224	0.720	0.724
Microvirga	0.224	0.746	0.725
Porphyrobacter	0.222	0.716	0.741
Cellulosimicrobium	0.217	0.760	0.601
Agromyces	0.214	0.704	0.558
Melittangium	0.209	0.824	0.692
Cystobacter	0.208	0.745	0.632
Blastomonas	0.207	0.782	0.777
Pannonibacter	0.201	0.778	0.647
Ensifer	0.201	0.750	0.758
Nonomuraea	0.197	0.727	0.527
Halioglobus	0.193	0.771	0.746
Salinimonas	0.192	0.796	0.819
Microbulbifer	0.190	0.790	0.791
Roseomonas	0.189	0.706	0.618
Plantactinospora	0.189	0.656	0.412
Shinella	0.183	0.651	0.606
Aeromicrobium	0.183	0.658	0.392
Rhodoplanes	0.179	0.792	0.730
Fictibacillus	0.179	0.742	0.738
Bosea	0.178	0.693	0.722
Rhizobacter	0.175	0.591	0.665
Lentzea	0.175	0.733	0.609
Brevibacterium	0.175	0.754	0.543
Thiodictyon	0.173	0.773	0.698
Plesiomonas	0.172	0.646	0.814
Tessaracoccus	0.170	0.742	0.533
Actinoalloteichus	0.165	0.706	0.425
Sporosarcina	0.164	0.802	0.758
Aminobacter	0.163	0.721	0.793
Luteitalea	0.162	0.835	0.700
Nitrospirillum	0.157	0.715	0.635
Dietzia	0.147	0.796	0.306
Tatumella	0.141	0.660	0.828
Cryobacterium	0.138	0.736	0.554
Hydrogenophaga	0.137	0.661	0.627
Lelliottia	0.095	0.535	0.866
Leclercia	0.094	0.512	0.807