# DETECTING OUT-OF-DISTRIBUTION SAMPLES USING LOW-ORDER DEEP FEATURE STATISTICS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

The ability to detect when an input sample was not drawn from the training distribution is an important desirable property of deep neural networks. In this paper, we show that a simple ensembling of first and second order deep feature statistics can be exploited to effectively differentiate in-distribution and out-of-distribution samples. Specifically, we observe that the mean and standard deviation within feature maps differs greatly between in-distribution and out-of-distribution samples. Based on this observation, we propose a simple and efficient plug-and-play detection procedure that does not require re-training, pre-processing or changes to the model. The proposed method outperforms the state-of-the-art by a large margin in all standard benchmarking tasks, while being much simpler to implement and execute. Notably, our method improves the true negative rate from 86.6% to 96.8% when 95% of in-distribution (CIFAR-100) are correctly detected using a DenseNet and the out-of-distribution dataset is TinyImageNet resize. The source code of our method will be made publicly available.

## 1 INTRODUCTION

In the past few years, deep neural networks (DNNs) (Goodfellow et al., 2016) have settled as the state-of-the art-techniques in many difficult tasks in a plurality of domains, such as image classification (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012; Han et al., 2018), and machine translation (Bahdanau et al., 2014; van den Oord et al., 2017). This recent progress has been mainly due to their high accuracy and good generalization ability when dealing with real-world data. Unfortunately, DNNs are also highly confident when tested against unseen samples, even if the samples are vastly different from the ones employed during training (Hendrycks & Gimpel, 2017). Moreover, several works have shown that such deep networks are easily fooled by minor perturbations to the input (Goodfellow et al., 2014; Su et al., 2017). Obtaining a calibrated confidence score from a deep neural network is a problem under continuous investigation (Hendrycks & Gimpel, 2017) and a major thread in artificial intelligence (AI) safety (Amodei et al., 2016). In fact, knowing when the model is wrong or inaccurate has a direct impact in many production systems, such as self-driving cars, authentication and disease identification (Akhtar & Mian, 2018; Goswami et al., 2018), to name a few.

Guo et al. (2017) showed that despite producing significantly low classification errors, DNNs confidence scores are not faithful estimates of the true certainty. Their experiments confirmed that depth, width, weight decay, and batch normalization are the main reasons for overconfident scores. Moreover, they demonstrated that a simple and yet powerful method of temperature scaling in the softmax scores is an effective way to improve calibrate a DNNs confidence. While calibrating the classifier's output to represent a faithful likelihood from the training (**in-distribution**) data has effective solutions, the problem of detecting whether or not the samples are generated from a known distribution (**out-of-distribution**), is still an open problem (Hendrycks & Gimpel, 2017).

One straightforward approach to calibrate the classifier's confidence in order to detect samples whose distribution differs from the training samples distribution is to train a secondary classifier that digests both in-distribution (ID) and out-of-distribution (OOD) data so that anomalies are scored differently from ID samples, as performed in Hendrycks et al. (2018). Re-training a network, however, can be computationally intensive and may even be intractable, since the number of OOD samples is virtually infinite. Other solutions rely on training both classification and generative neural net-

Table 1: Summary comparison of the characteristics of recent related methods. Test complexity refers to the required number of passes over the network. Training data is the number of samples for which the methods were calibrated against (with all standing for the whole training set). AUROC is the area under receiver characteristic curve (detailed in Section 4). Performance shown is for DenseNet trained on CIFAR-100 and using TinyImageNet (resized) as OOD dataset.

| Method | Input pre-proc. | Model change | Test Complexity | Training data | AUROC |
|---|---|---|---|---|---|
| Hendrycks & Gimpel (2017) | No | No | 1 | 1000 | 71.6 |
| Liang et al. (2018) | Yes | No | 3 | 1000 | 85.5 |
| Vyas et al. (2018) | Yes | Yes | 3 | all | 96.3 |
| Lee et al. (2018b) | Yes | Yes | 3 | all | 97.3 |
| Proposed | No | No | 1 | 1000 | **99.3** |

works for OOD using a multi-task loss (Lee et al., 2018a), or using unsupervised fully convolutional networks as done by Sabokrou et al. (2016) to detect OOD in video samples. All these methods, however, have a major drawback: they require re-training a modified model using a different loss function possibly with additional parameters, which increases the computational burden, and demands access to the entire original (and probably huge) training data.

In this work, we propose a novel OOD sample detection method that explores low-level statistics from feature layers. The statistics are obtained directly from the Batch Normalization layers (Ioffe & Szegedy, 2015), requiring no extra computations during training time, and no changes to the network architecture and loss functions. During test time, the proposed method extracts statistics from selected layers and combines them into an OOD-ness score, thus not requiring any preprocessing to the input image. Throughout this paper, we observe that the mean and standard deviation of a given channel in a layer differ greatly between ID and OOD samples, which naturally motivates their use as features to be employed by an OOD detector. By selecting the BN layers of a network, we are able to normalize the features according to the learned batch statistics, and combine them through a linear classifier. The effectiveness of the proposed method is validated in two state-of-the-art DNN architectures (DenseNet and ResNet) (Huang et al., 2017; He et al., 2015; 2016; Zagoruyko & Komodakis, 2016) that are trained for image classification tasks in popular datasets. The proposed approach achieves state-of-the-art performance, surpassing all competitors by a large margin in all tested scenarios, while being much more efficient. Notably, our method only requires one forward pass while Liang et al. (2018); Lee et al. (2018b); Vyas et al. (2018) require two forward and one backward passes.

The rest of the paper is organized as follows. Section 2 describes prior work on OOD samples detection. Section 3 introduces the proposed method, whereas Section 4 details all experiments and compares the results with state-of-the-art methods. Finally, we draw our conclusions in Section 5.

## 2 Previous work

In this section, we describe recent prior work on OOD detection methods. A summary of all methods described can be seen in Table 1.

**Hendrycks & Gimpel (2017)** proposed a baseline method based on the posterior distribution (i.e. softmax scores), that requires no re-training of the network. They showed that well-trained models tend to produce higher scores for ID samples than for OOD ones. Hence their method comprises of applying a threshold on the softmax-normalized output of a classifier. If the largest score is below the threshold, then the sample is considered OOD.

**Liang et al. (2018)** continued the aforementioned line of work and proposed the Out-of-Distribution detector for Neural networks (ODIN), which includes a temperature scaling $T \in \mathbb{R}_*^+$ to the softmax classifier as in Guo et al. (2017). The authors in ODIN argued that a good manipulation of $T$ eases the separation between in- and out-of-distribution samples. Allied to that, they also incorporated small perturbations to the input (inspired by Goodfellow et al. (2014)) whose goal is to increase the softmax score of the predicted class. Liang et al. (2018) found that this kind of perturbation has a stronger effect on ID samples than on OOD ones, increasing the separation between ID and

OOD samples. Their approach significantly outperforms the baseline method (Hendrycks & Gimpel, 2017) by a fair margin; however, it is three times slower than Hendrycks & Gimpel (2017) due to the two forward and one backward passes needed to compute the preprocessed input, while the previous method only requires one forward pass.

**Vyas et al. (2018)** describes a novel loss function, called margin entropy loss, over the softmax output that attempts to increase the distance between ID and OOD samples. During training, they partition the training data itself into ID and OOD by assigning samples labeled as certain classes as OOD and use the different partitions to train an ensemble of classifiers that are then used to detect OOD samples during test time. They also use the input pre-processing step proposed in Liang et al. (2018), including temperature scaling.

**Lee et al. (2018b)** is the most recent work on OOD detection that we have knowledge of. Despite having a similar working principle as ODIN, they showed that the posterior distribution defined by a generative classifier under Gaussian discriminant analysis is equivalent to that of the softmax classifier, and the generative classifier eases the separation between in- and out-of-distribution samples. Also, the confidence score is defined using the Mahalanobis distance between the sample and the closest class-conditional Gaussian distribution. In addition, they argue that abnormal samples can be better characterized in the feature space of DNNs rather than the output space of softmax-based posterior distribution used in previous works such as ODIN. Hence, they also pre-process the input similarly as in ODIN but instead of increasing the softmax score, it is the confidence score that is increased. To further improve the performance, they also considered intermediate generative classifiers for all layers in the network. Finally, their OOD sample detector is computed as an ensemble of confidence scores, chosen by training a logistic regression using validation samples. This method also shows remarkable results for detection of adversarial attacks and for incremental learning.

## 3 Proposed Solution

An OOD detector should incorporate information from the training data in a natural manner, without being directly influenced by the loss function, which is intrinsically related to the task which could be well-defined for ID samples but be meaningless for most OOD samples. Moreover, if the OOD method is more dependent on the training distribution, it should be able to be applied to a wide variety networks, and not be designed specifically for a given architecture.

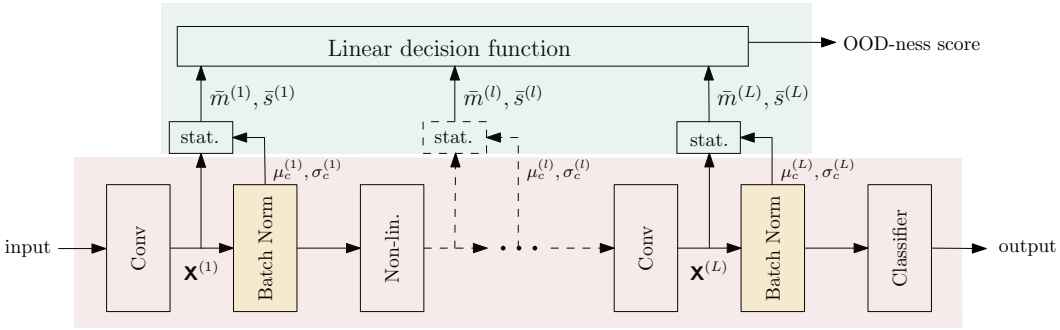
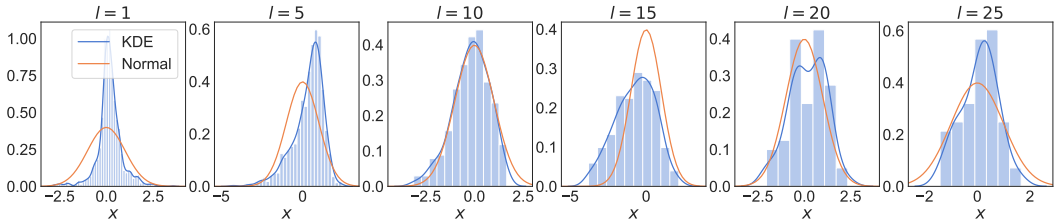


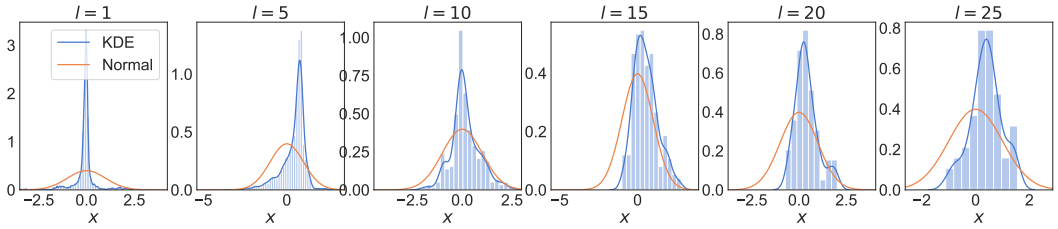

Figure 1: An illustration of the complete proposed model. At each BN layer, we extract the input, normalize it using the running statistics, and compute the first and second order features. The outputs are fed to a linear decision function to predict if the input sample is out-of-distribution or not.

Our method is based on a very simple observation. Input samples with different distributions generate statistically distinct feature spaces in a DNN. In other words, the deep features of an ID sample are distinct than those from an OOD one. Moreover, when using a normalization scheme, such as BN, the features are normalized by the statistics of the ID batch during training, possibly leading to feature statistics that are more similar to the batch statistics, as depicted in Figure 2.

The main problem then becomes how to summarize the feature space distribution for ID samples in a way that best discriminates between ID and OOD samples. In this work we show that using the first and second order statistics within each feature map performs remarkably well for this task.




(a) ID CIFAR-10




(b) OOD TinyImageNet resize

Figure 2: Comparison between the normalized BN latent space by its running statistics extracted from a WideResNet-28-10 and the normal distribution (using the BN running mean and variance) for BN layers $l = 1, 5, 10, 15, 20, 25$. The arbitrary ID CIFAR-10 sample (a) seems to be more similar to the batch stastitics than arbitrary OOD TinyImageNet resize sample (b) in the mid-layers. Bars: normalized feature histograms. Blue curve: KDE. Orange curve: $\mathcal{N}(0, 1)$ (always the same inside each column but scaled for visualization).

Figure 1 shows the proposed OOD detector, highlighting the feature statistics that are extracted for each channel at each BN layer. These statistics are then combined through a learned linear decision function, providing a measure of OOD-ness. The way we compute the per-channel statistics is formalized in Section 3.1. Finally, the linear classifier used to combine the statistics from the different layers is described in Section 3.2.

### 3.1 Low-order feature statistics

In the previous section, we motivated that characterizing the feature-space distributions might lead to a robust OOD detector. As a first approach, one could model these distributions using a non-parametric method to estimate the distribution of the features for each channel, which requires the computation of sufficient statistics using the training data or using a parametric method to fit the distribution (Bishop, 2007), which are both computationally intensive.

Here, we propose to use only the mean and standard deviation computed along the spatial dimension for each channel to summarize the per-channel distribution. As it will be shown, these two statistics are sufficient to distinguish between ID and OOD. Moreover, because the mean and standard deviation of each channel are normalized by the running mean and variance computed during training by the BN layers (Ioffe & Szegedy, 2015), these statistics can be naturally combined within each layer to produce effective features for OOD detection. We describe such a procedure in what follows.

Given the $l$-th BN layer with input tensor $\mathbf{X} \in \mathbb{R}^{C \times W \times H}$, the output $\mathcal{BN}^{(l)}_{c,i,j}$ at channel $c$ is

$$Z^{(l)}_{c,i,j} = \frac{X^{(l)}_{c,i,j} - \mu^{(l)}_c}{\sqrt{[\sigma^{(l)}_c]^2 + \epsilon}}, \quad \mathcal{BN}^{(l)}_{c,i,j} = Z^{(l)}_{c,i,j} \gamma^{(l)}_c + \beta^{(l)}_c, \tag{1}$$

where $\gamma^{(l)}_c > 0$ and $\beta^{(l)}_c$ are the per channel per layer learned scaling and shifting parameters, $\epsilon > 0$ is a small constant value for numerical stability, $\mu^{(l)}_c \in \mathbb{R}$ and $[\sigma^{(l)}_c]^2 \in \mathbb{R}^+$ are the mean and variance estimated through a moving average using the batch statistics during training, and $\mathbf{Z}^{(l)} \in \mathbb{R}^{C \times W \times H}$ is the normalized feature tensor. It is worth noting that the statistics are calculated independently for each channel $c$ at each layer $l$.

In this paper, we conjecture that low-order statistics computed from either $\boldsymbol{X}_{c,i,j}^{(l)}$ or $\boldsymbol{Z}_{c,i,j}^{(l)}$ can be used to discriminate between ID and OOD samples. Given the unnormalized input $\boldsymbol{X}_{c,i,j}^{(l)}$, we can compute the empirical mean $m_c^{(l)}$ and standard deviation $\sigma_c^{(l)}$ features for each channel $c$ as

$$m_c^{(l)} = \frac{1}{WH} \sum_{i,j} \boldsymbol{X}_{c,i,j}^{(l)}, \quad [s_c^{(l)}]^2 = \frac{1}{WH} \sum_{i,j} [\boldsymbol{X}_{c,i,j}^{(l)}]^2 - [m_c^{(l)}]^2, \tag{2}$$

and the features for the normalized tensor $\boldsymbol{Z}_{c,i,j}^{(l)}$ are defined as

$$\widehat{m}_c^{(l)} = \frac{1}{WH} \sum_{i,j} \boldsymbol{Z}_{c,i,j}^{(l)} = \frac{m_c^{(l)} - \mu_c{}^{(l)}}{\sigma_c^{(l)}} \tag{3}$$

$$[\widehat{s}_c^{(l)}]^2 = \frac{1}{WH} \sum_{i,j} [\boldsymbol{Z}_{c,i,j}^{(l)}]^2 - [\widehat{m}_c^{(l)}]^2 = \left[ \frac{\widehat{s}_c^{(l)}}{\sigma_c^{(l)}} \right]^2, \tag{4}$$

i.e., the normalized mean feature $\widehat{m}_c^{(l)}$ represents the difference between the sample mean $m_c^{(l)}$ and the BN running mean $\mu_c^{(l)}$ weighted by the running standard deviation $\sigma_c^{(l)}$, whereas $\widehat{s}_c^{(l)}$ is the normalized standard deviation by the BN running standard deviation $\sigma_c^{(l)}$.

### 3.2 Aggregation of feature statistics

**Intra-layers aggregation.** The features derived from low-order statistics as in equation 3 can be readily used to train a predictor for ID/OOD discrimination. Of course, if they were produced for every feature map in the network, this would result in a feature vector of very high dimension, typically tens of thousands. Instead, we propose to combine these features within each BN layer, so that in the end we obtain one pair of features per layer: average mean and average variance. Taking advantage of the fact that features are approximately normalized by BN's running statistics, we propose to simply average them for all channels within a layer. Thus, each layer yields the following features, for the normalized case:

$$\bar{m}^{(l)} = \frac{1}{C_l} \sum_c \widehat{m}_c^{(l)}, \quad [\bar{s}^{(l)}]^2 = \frac{1}{C_l} \sum_c [\widehat{s}_c^{(l)}]^2, \tag{5}$$

where $C_l$ is the number of channels in layer $l$. Note that doing this aggregation amounts to computing the mean and standard deviation of all normalized features at the given layer.

**Inter-layers ensemble and final classification.** Using all the features in equation 5, i.e., $\boldsymbol{f} = (\bar{m}^{(1)}, \bar{s}^{(1)}, ..., \bar{m}^{(L)}, \bar{s}^{(L)})$ we fit a simple logistic regression model $h(\boldsymbol{f}; \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta} \in \mathbb{R}^{2L+1}$. The parameters of the linear model are learned using a separate validation set formed with ID samples (positive examples) and OD samples (negative examples).

## 4 Experiments

In this section, we evaluate the effectiveness of the proposed method in state-of-the-art deep neural architectures, such as DenseNet (Huang et al., 2017) and Wide ResNet (Zagoruyko & Komodakis, 2016), on several computer vision benchmark datasets: CIFAR (Krizhevsky, 2009), TinyImageNet, a subset of ImageNet (Deng et al., 2009), LSUN (Yu et al., 2015), and iSUN (Xu et al., 2015). We also use Gaussian noise and uniform noise as synthetic datasets. Note that this evaluation protocol is the standard in recent OOD detection literature (Hendrycks & Gimpel, 2017; Liang et al., 2018; Lee et al., 2018b). The code to reproduce all results is publicly available[1].

### 4.1 Setup

**Datasets:** For backbone training, we use CIFAR-10 and CIFAR-100 datasets which have 10 and 100 classes respectively, both containing 50,000 images in the training set and 10,000 images in the

[1]Code for review avilable at `https://www.dropbox.com/s/o652ufpznatvx8u/ood-codes-review.zip?dl=0`.

test set. At test time, the test images from CIFAR-10 (CIFAR-100) are considered as ID (positive) samples. For OOD (negative) datasets, we test with datasets containing natural images, such as TinyImageNet resize and crop, LSUN resize and crop, and iSUN, as well as synthetic datasets, such as Gaussian noise, and uniform noise, which is the same dataset setup as in Liang et al. (2018) and Hendrycks & Gimpel (2017). This is summarized in Table 2. For all datasets, 1000 samples from the test set are separated in a validation set used for fitting the logistic regressor and hyper-parameter tuning, and the remaining samples (unseen for both backbone model and OOD detector) are used for testing.

Table 2: Summary of datasets. All generated samples have a size of $32 \times 32$.

| | | # samples | | Pre-processing | Observation |
|---|---|---|---|---|---|
| | | val | test | | |
| ID | CIFAR-10 | 1000 | 9000 | - | |
| ID | CIFAR-100 | 1000 | 9000 | - | |
| OOD | TinyImageNet (c) | 1000 | 9000 | Random patches | Subset of Imagenet containing 200 classes |
| OOD | TinyImageNet (r) | 1000 | 9000 | Downsampled | |
| OOD | LSUN (c) | 1000 | 9000 | Random patches | |
| OOD | LSUN (r) | 1000 | 9000 | Downsampled | |
| OOD | iSUN | 1000 | 7925 | - | Subset of SUN images |
| OOD | Gaussian Noise | 1000 | 9000 | 0.5 mean and unit variance clipped in $[0, 1]$ | Channel-independent |
| OOD | Uniform noise | 1000 | 9000 | Uniform distribution in $[0, 1]$ | |

**Backbone training:** Following Liang et al. (2018), we adopt the DenseNet (Huang et al., 2017) and Wide ResNet (Zagoruyko & Komodakis, 2016) architectures as our benchmark networks. For DenseNet, we use depth $L = 100$, growth rate $k = 12$, and zero dropout rate (DenseNet-BC-100-12). For Wide ResNet, we also follow the work of Liang et al. (2018), with $L = 28$ and widen factor of 10 (WRN-28-10). The hyperparameters of each network are identical to their original papers. For reproducibility's sake we summarize their training procedure next. All networks were trained using stochastic gradient descent with Nesterov momentum (Ruder, 2016) and an initial learning rate of 0.1. We train the DenseNet-BC-100-12 for 300 epochs, with batch size 64, momentum 0.9, weight decay of $10^{-4}$ and decay the learning rate by a factor of 10 after epochs 150 and 225. We train the WRN-28-10 for 200 epochs, with batch size 128, momentum 0.9, weight decay $0.0005$, and decay the learning rate by a factor of 10 after 60, 120, and 160 epochs. Table 3 summarizes the accuracy of each network over 4 independent runs each one initialized with a different random seed.

Table 3: Average test error rates (%) and standard deviation (in parenthesis) over 4 runs.

| Architecture | CIFAR-10 | CIFAR-100 |
|---|---|---|
| DenseNet-BC-100-12 | 4.8 (0.2) | 22.6 (0.3) |
| WRN-28-10 | 4.0 (0.1) | 19.1 (0.3) |

**Logistic regression:** The logistic regressor is trained considering only the validation partitions for ID (positive examples) and OOD (negative examples) datasets (see Table 2). Using both mean and standard deviation as input (from (5)), we have 50 features for WRN-28-10 models, and 198 features for DenseNet-BC-100-12 models. The training was performed using 5-fold cross-validation with the $\ell_2$ minimization and the regularization factor being chosen as the best one (according to the 5-folds) among 10 values linearly spaced in the range $10^{-4}$ and $10^4$.

**Evaluation metrics:** To evaluate the proposed method, we use the following metrics:

1. **True negative rate (TNR) at 95% true positive rate (TPR)**. Let TP, TN, FP, and FN be the true positive, true negative, false positive, and false negative, respectively. The TNR is defined as TN/(TN+FP) whereas TPR is defined as TP/(TP+FN).

2. **Area under the receiver operating characteristic curve (AUROC)**. AUROC is the area under the FPR=1-TNR against TPR curve.

### 4.2 PRELIMINARY EXPERIMENTS

In this section, several experiments showcasing different characteristics of the proposed method are performed. Focus will be given to understand how the proposed features behave and motivate the choices made in designing the OOD detector. All results shown consider averages and standard deviations of the performance metrics computed over all 4 backbone models trained for each architecture.

**Manifold visualization:** We applied t-SNE (L. van der Maaten, 2008) to visualize our high-dimensional feature space in order to see the similarities between ID/OOD samples. For this, we used the WRN-28-10 model and CIFAR-10 as ID dataset. We fitted the t-SNE using all ID and OOD datasets together using both mean and standard deviation features, and the result is shown in Figure 3 using a perplexity of 30. It is clear from the visualization that the proposed features are concentrated around well-defined clusters. Both synthetic OOD datasets have clear distinct behavior from the natural images ones, and it is straightforward to differentiate between them. TinyImageNet (c), LSUN (c) are similar and have some intersection with TinyImageNet (r), LSUN (r). Interestingly, the clustering seems to reflect the dataset generation method (reshaping or cropping). Nevertheless, one can see that the OOD samples are in different clusters than the ID (CIFAR-10) samples.

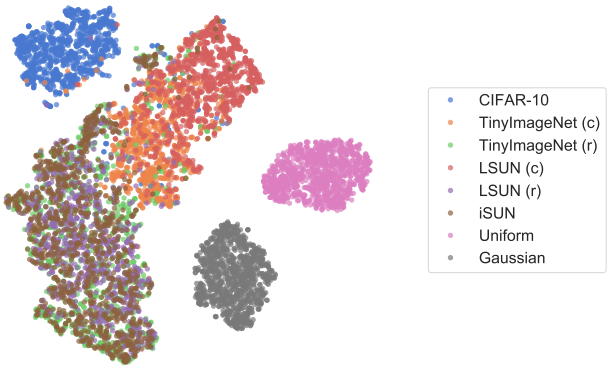



Figure 3: Manifold visualization of the proposed features (mean and std) from a WRN-28-10 model.

**Selecting OOD Dataset:** To understand to what extent a classifier trained considering one OOD dataset is able to generalize and detect samples from other OOD datasets, we trained the logistic regression considering as positive examples 1000 samples from the ID dataset (either CIFAR-10 or CIFAR-100) and as negative examples 1000 samples of a given OOD dataset. The obtained model was then evaluated on the remaining OOD datasets. This procedure was then repeated for each possible OOD dataset, and the averaged results are summarized in Table 4, where we also divided the OOD datasets in two groups: one containing only synthetic-noise images (Gaussian and uniform) and the other containing natural images (others). Classifiers fitted using only natural images (first row) are not able to generalize to synthetic samples (second column). The converse is also true, which seems to corroborate what was observed by the t-SNE visualization. We then tested combining both synthetic (from the Gaussian distribution) and real images in order to fit a single logistic regressor (last column) able to tackle both cases. This combination yields an improvement in generalization for both natural and synthetic images, with an improvement even for natural-only images, with the average TNR @ 95% TPR improving from 67.2% to 72.8%.

Table 4: Generalization to unseen OOD sets. We split datasets in two: Natural and Noise and averaged the performance. Results are TNR @ 95% TPR formatted as median (min - max).

| | | Test | |
|---|---|---|---|
| | | Natural | Noise |
| OOD set (fit) | Natural | 99.7 (23.6 - 100.0) | 99.1 (0.0 - 100.0) |
| | Noise | 65.9 (0.5 - 83.5) | 100.0 (100.0 - 100.0) |
| | Natural + Noise | 99.8 (23.8 - 100.0) | 100.0 (100.0 - 100.0) |

**Selecting Features:** Another question is what is the individual impact of the different proposed features (i.e., layers average mean and std). To evaluate this, results showing the average performance when changing the features used for fitting the classifier can be seen in Table 5. As can be seen, while the mean features help to detect samples from the fitted OOD dataset, it does not generalize well for different OOD datasets. On the other hand, the standard deviation features achieved very good performance on the target OOD dataset, but also generalize very well to unknown OOD datasets. Since we are interested in designing an OOD detector that is able to differentiate ID samples from any OOD sample, all our results from now on are presented using the averaged standard deviations per layer as the only features used to train the linear decision function. Also, following the discussion on the previous experiment, we will only fit to ID TinyImageNet (c) and OOD Gaussian noise, and evaluate on all remaining OOD datasets (unless specified otherwise).

Table 5: Generalization over unseen OOD datasets using different features (only mean, only std, and both). For each OOD dataset, we fit the linear model and tested against all OOD datasets. The results displayed are TNR @ 95% TPR averaged over the OOD datasets using WRN-28-10 as the backbone model and CIFAR-100 as ID dataset.

| Feat. | TinyImageNet (c) | TinyImageNet (r) | LSUN (r) | LSUN (c) | iSUN |
|---|---|---|---|---|---|
| $\bar{m}^{(l)}$ | 77.8 (20.4) | 82.6 (23.0) | 60.4 (33.6) | 80.4 (27.2) | 81.0 (26.1) |
| $\bar{s}^{(l)}$ | **94.2 (6.3)** | **95.5 (8.6)** | **74.0 (25.9)** | **96.3 (7.7)** | **95.8 (8.4)** |
| both | 90.5 (10.2) | 90.9 (13.5) | 67.8 (30.3) | 90.8 (14.6) | 90.0 (15.1) |

**Effect of feature normalization:** As shown in Table 6, normalizing the latent space using the BN statistics before computing the features has clear advantages.

Table 6: TNR @ 95% TPR for computing the averaged mean and standard deviation from unnormalized (first row) and normalized (second row) BN latent space.

| Norm. | TinyImageNet (c) | TinyImageNet (r) | LSUN (r) | LSUN (c) | iSUN |
|---|---|---|---|---|---|
| No | **100.0 (0.0)** | 80.6 (1.6) | 81.6 (1.7) | 78.1 (1.7) | **100.0 (0.0)** |
| Yes | **100.0 (0.0)** | **88.8 (3.9)** | **90.4 (3.7)** | **87.9 (4.2)** | **100.0 (0.0)** |

**Reduced validation data:** Often, ID and OOD validation samples to train the linear model are scarce. Moreover, the logistic regressor learned should be able to generalize well for a large number of unknown OOD samples even if trained with a restricted of number of OOD samples. Hence, instead of using 1000 validation samples to fit the logistic regression coefficients, we study if our method can correctly detect OOD samples even if a small number of samples is available. Figure 4 shows the TNR @ 95% TPR for WRN 28-10 trained on CIFAR-10 (CIFAR-100), where only 27, 45, 75, 150, 300, 700, 1.5k, 3k (all) test images (ID + TinyImageNet crop + Gaussian, equally divided) are used to fit 25 coefficients of our logistic regressor. Using our method, only 27 images (from each ID and OOD), are necessary to achieve an average of 87.6% of TNR @ 95% TPR.

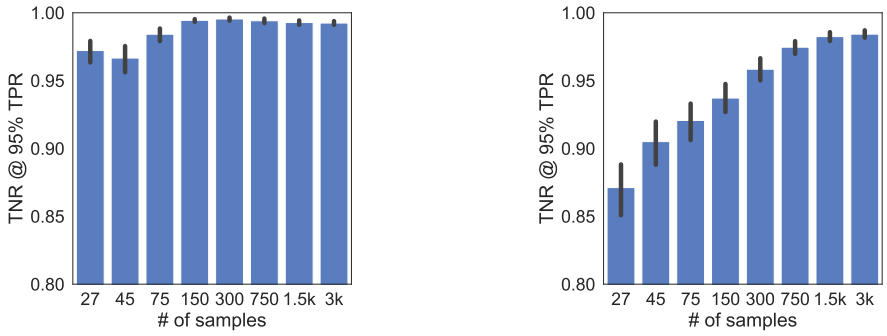
### 4.3 OOD METHODS COMPARISON

In this section, a comparison between four recent methods described in Section 2 and our proposed method is performed. Since Lee et al. (2018b) did not test using Wide ResNet models and the same datasets as in Liang et al. (2018); Vyas et al. (2018), here we only show the intersection between them: DenseNet BC 100-12 model, using CIFAR-10 (CIFAR-100) as in-distribution and TinyImageNet (resize) and LSUN (resize) as OOD distribution[2]. Extended results can be found in the appendix. For both Hendrycks & Gimpel (2017) and Liang et al. (2018) results, we reimplemented the method using their reference implementation[3,4]. For Liang et al. (2018), we employed the same procedure as described the authors to tune the methods parameters. A detailed description of the procedure can be found in Appendix B.1. For Lee et al. (2018b) and Vyas et al. (2018), we use

[2]At the time of submission, Lee et al. (2018b) was still unpublished work

[3]Baseline: `github.com/hendrycks/error-detection`

[4]ODIN: `github.com/ShiyuLiang/odin-pytorch`




(a) CIFAR-10

(b) CIFAR-100

Figure 4: Averaged TNR @ 95% TPR over the OOD datasets using only a few samples to fit the logistic regressor. The network employed is a WRN-28-10 where the logistic regressor coefficients were fitted using TinyImageNet (c) + Gaussian datasets using only $\bar{s}^{(l)}$. This experiment shows, in accordance with our results, that CIFAR100 is more difficult to differentiate from other OOD datasets than CIFAR10.

the values presented on their papers (Table 2 and Table 3, respectively). For the proposed method, as motivated in the previous section, we use the averages of normalized standard deviation features (equation 5) trained over the validation partition of the OOD set. For comparison sake with the other methds in the literature, we adjust the methods (when applicable) for each ID/OOD pair, including our proposed algorithm (column "pairs"). We also display the results for the proposed method when training the regressor only for Gaussian + TinyImageNet (c) as OOD datasets.

The results are compiled in Table 7. Notably, our method outperforms the baseline and ODIN methods by a large margin, and yields better results than Lee et al. (2018b) and Vyas et al. (2018) in all tested cases without requiring any preprocessing, or changing the backbone DNN.

Table 7: Comparison between previous work (Section 2) and our proposed method using the TNR @ 95% TPR metric for DenseNet-BC-100-12. The displayed values (%) are the mean over 4 different training runs of the backbone models.

| **Dataset** | **OOD** | **Baseline** | **ODIN** | **Vyas et al. (2018)** | **Lee et al. (2018b)** | **Ours** | **Ours** (Pair) |
|---|---|---|---|---|---|---|---|
| CIFAR-10 | TinyImageNet (r) | 30.3 | 86.5 | 97.1 | 95.3 | **97.1** | 100.0 |
| | LSUN (r) | 35.5 | 92.7 | 99.2 | 97.5 | **99.4** | 100.0 |
| CIFAR-100 | TinyImageNet (r) | 21.0 | 52.9 | 79.5 | 86.6 | **96.8** | 100.0 |
| | LSUN (r) | 26.7 | 55.7 | 83.8 | 91.2 | **98.2** | 100.0 |

When adjusting for the validation partition of the OOD set, we are able to correclty detect all OOD samples from the test partition of the same dataset. We argue that this is a limited measurement of quality in the sense that many practical applications have OOD samples that do not come from a single distribution mode, and it might be infeasible to collect data from the many different modes of the OOD distribution (some modes might even be unknown during training). This means that a good OOD detector should be able to correctly identify samples from OOD distibutions for which their hyperparameters have not been adjusted to.

In this experiment, whose results are shown in Table 8, instead of fitting the logistic regression to each ID/OOD pair we just fit it once to TinyImageNet (c) and Gaussian noise validation samples and test the obtained model on all the remaining OOD datasets. For ODIN, we tune its hyperparameters using the same grid search as described in B.1. Again, in all tested cases, our method outperforms ODIN by a large margin, showing better generalization to unseen OOD samples.

## 5 Conclusion

Deep neural networks trained to maximize classification performance in a training dataset are extremely adapted to said dataset. The statistics of activations throughout the network for samples

Table 8: Comparison between ODIN and our proposed OOD detector for several setups using image classification networks. All detector parameters were tuned for TinyImageNet (c) and Gaussian and are tested on others.

| | **Out-of-distribution dataset** | **TNR (95% TPR)** | **AUROC** |
|---|---|---|---|
| | | **ODIN / Ours** | |
| **DenseNet BC 100-12** CIFAR-10 | TinyImageNet (c) | 90.5 (2.1) / **100.0 (0.0)** | 98.0 (0.3) / **100.0 (0.0)** |
| | TinyImageNet (r) | 76.9 (4.7) / **96.6 (1.4)** | 95.8 (0.8) / **99.3 (0.3)** |
| | LSUN (c) | 82.9 (6.1) / **100.0 (0.0)** | 97.0 (0.7) / **100.0 (0.0)** |
| | LSUN (r) | 86.2 (3.7) / **99.2 (0.5)** | 97.2 (0.6) / **99.8 (0.1)** |
| | iSUN | 81.1 (4.6) / **98.4 (0.9)** | 96.4 (0.8) / **99.6 (0.2)** |
| | Uniform | 68.7 (40.5) / **100.0 (0.0)** | 91.6 (9.2) / **100.0 (0.0)** |
| | Gaussian | 75.3 (31.2) / **100.0 (0.0)** | 96.2 (1.8) / **100.0 (0.0)** |
| **DenseNet BC 100-12** CIFAR-100 | TinyImageNet (c) | 72.9 (4.1) / **100.0 (0.0)** | 95.4 (0.4) / **100.0 (0.0)** |
| | TinyImageNet (r) | 39.6 (4.5) / **95.3 (0.7)** | 88.3 (1.3) / **99.0 (0.2)** |
| | LSUN (c) | 54.7 (4.7) / **100.0 (0.0)** | 93.3 (0.5) / **100.0 (0.0)** |
| | LSUN (r) | 42.2 (3.3) / **98.1 (0.7)** | 89.8 (0.8) / **99.5 (0.1)** |
| | iSUN | 34.4 (3.6) / **96.2 (1.0)** | 87.5 (0.9) / **99.2 (0.2)** |
| | Uniform | 0.0 (0.0) / **100.0 (0.0)** | 67.3 (7.9) / **100.0 (0.0)** |
| | Gaussian | 0.0 (0.0) / **100.0 (0.0)** | 39.0 (15.0) / **100.0 (0.0)** |
| **WRN 28-10** CIFAR-10 | TinyImageNet (c) | 91.9 (2.0) / **99.9 (0.0)** | 98.5 (0.3) / **100.0 (0.0)** |
| | TinyImageNet (r) | 73.4 (1.9) / **97.1 (1.3)** | 91.9 (0.8) / **99.4 (0.2)** |
| | LSUN (c) | 91.8 (2.3) / **100.0 (0.0)** | 98.4 (0.4) / **100.0 (0.0)** |
| | LSUN (r) | 84.7 (1.8) / **99.4 (0.4)** | 95.8 (0.4) / **99.8 (0.1)** |
| | iSUN | 79.8 (1.8) / **98.6 (0.9)** | 94.4 (0.5) / **99.6 (0.2)** |
| | Uniform | 98.9 (1.1) / **100.0 (0.0)** | 99.3 (0.3) / **100.0 (0.0)** |
| | Gaussian | **100.0 (0.0)** / 100.0 (0.0) | 99.6 (0.3) / **100.0 (0.0)** |
| **WRN 28-10** CIFAR-100 | TinyImageNet (c) | 68.3 (3.0) / **100.0 (0.0)** | 94.6 (0.5) / **100.0 (0.0)** |
| | TinyImageNet (r) | 43.7 (2.3) / **96.8 (2.4)** | 87.7 (0.6) / **99.3 (0.5)** |
| | LSUN (c) | 38.8 (3.9) / **99.9 (0.0)** | 86.6 (2.3) / **99.9 (0.0)** |
| | LSUN (r) | 43.7 (1.3) / **98.2 (1.8)** | 88.5 (0.2) / **99.6 (0.4)** |
| | iSUN | 39.7 (2.0) / **97.5 (2.3)** | 87.2 (0.3) / **99.4 (0.4)** |
| | Uniform | 50.1 (49.0) / **99.8 (0.3)** | 78.5 (28.2) / **99.8 (0.3)** |
| | Gaussian | 25.3 (42.4) / **100.0 (0.0)** | 70.1 (26.2) / **100.0 (0.0)** |
| **WRN 40-4** CIFAR-10 | TinyImageNet (c) | 93.3 (1.1) / **100.0 (0.0)** | 98.3 (0.1) / **100.0 (0.0)** |
| | TinyImageNet (r) | 65.9 (4.0) / **96.9 (2.2)** | 89.7 (1.4) / **99.4 (0.3)** |
| | LSUN (c) | 93.0 (2.5) / **100.0 (0.0)** | 98.1 (0.3) / **100.0 (0.0)** |
| | LSUN (r) | 78.1 (2.2) / **98.6 (1.2)** | 93.9 (0.6) / **99.6 (0.2)** |
| | iSUN | 72.8 (3.0) / **97.9 (1.6)** | 92.5 (0.8) / **99.5 (0.3)** |
| | Uniform | 50.9 (32.3) / **100.0 (0.0)** | 84.5 (12.6) / **100.0 (0.0)** |
| | Gaussian | 55.8 (41.2) / **100.0 (0.0)** | 84.0 (15.2) / **100.0 (0.0)** |

from the training distribution (in-distribution) are remarkably stable. However, when a sample from a different distribution (out-of-distribution) is given to the network, its activation statistics depart greatly from those of in-distribution samples. Based on this observation, we proposed a very simple and efficient method to detect out-of-distribution samples. Our method is based on using averages of low-order statistics computed at the Batch Normalization layers of the network as features for a linear decision function. This procedure is much simpler and efficient than current state-of-the-art procedures, and outperforms them with a large margin. Moreover, our method generalizes remarkably well to unseen out-of-distribution datasets. Our results suggest that the proposed method all but solves the problem of out-of-distribution detection when the in- and out-of-distribution datasets are different. The most challenging scenario going forward remains when the two distributions are created by splitting the classes inside one training dataset. We suggest that future efforts in out-of-distribution detection should focus on this difficult task.

## REFERENCES

N. Akhtar and A. S. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, February 2018.

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

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate, September 2014. preprint arXiv:1409.0473.

C. M. Bishop. *Pattern recognition and machine learning*. Information Science and Statistics. Springer-Verlag, New York, USA, 1 edition, 2007.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, Florida, USA, June 2009. IEEE.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, pp. 2672–2680, Montreal, Canada, December 2014. Neural Information Processing Systems Foundation.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*, volume 1. MIT Press, 2016.

G. Goswami, N. K. Ratha, A. Agarwal, R. Singh, and M. Vatsa. Unravelling robustness of deep learning based face recognition against adversarial attacks. In *AAAI Conference on Artificial Intelligence*, pp. 6829–6836, New Orleans, USA, February 2018.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, volume 70, pp. 1321–1330, Sydney, Australia, August 2017. International Machine Learning Society.

K. J. Han, A. Chandrashekaran, J. Kim, and I. Lane. The CAPIO 2017 conversational speech recognition system, December 2018. preprint arXiv:1801.00059.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, December 2015. arXiv preprint arXiv:1512.03385.

K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, volume 9908, pp. 630–645, Amsterdam, the Netherlands, 2016.

D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, pp. 1–13, Toulon, France, April 2017.

D. Hendrycks, M. Mazeika, and T. Dietterich. Deep anomaly detection with outlier exposure, 2018. preprint.

G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, November 2012.

G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Computer Vision and Pattern Recognition*, Hawaii, USA, July 2017. IEEE.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, volume 37, pp. 448–456, Lille, France, July 2015. International Machine Learning Society.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, Canada, 2009.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, Nevada, USA, December 2012. Neural Information Processing Systems Foundation.

G. E. Hinton L. van der Maaten. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:9579–2605, November 2008.

K. Lee, H. Lee, K. Lee, and J. Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *International Conference on Learning Representations*, pp. 1–16, Vancouver, Canada, April 2018a.

K. Lee, K. Lee, H. Lee, and J. Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks, July 2018b. preprint arXiv 1807.03888.

S. Liang, Y. Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, pp. 1–27, Vancouver, Canada, April 2018.

S. Ruder. An overview of gradient descent optimization algorithms, September 2016. preprint arXiv:1609.04747.

M. Sabokrou, M. Fayyaz, M. Fathy, Z. Moayed, and R. Klette. Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes, September 2016. preprint arXiv:1609.00866.

J. Su, D. Vasconcellos Vargas, and S. Kouichi. One pixel attack for fooling deep neural networks, October 2017. preprint arVix:1710.08864.

A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. C. C. Rus, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis. Parallel WaveNet: Fast high-fidelity speech synthesis. Technical report, Google Deepmind, 2017.

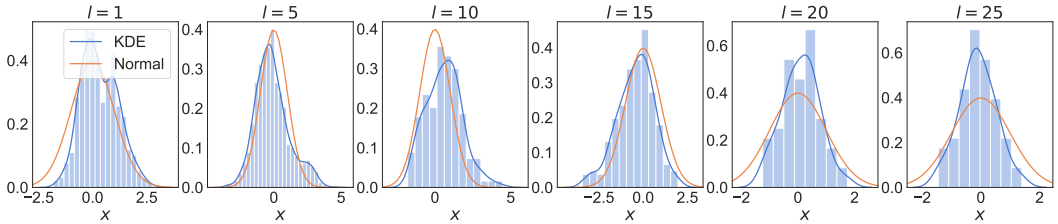
A. Vyas, N. Jammalamadaka, D. Das X. Zhu, B. Kaul, and T. L. Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classiers. In *European Conference on Computer Vision*, pp. 1–15, Munich, Germany, September 2018.

P. Xu, K. A. Ehinger, Y. Zhang, A. Finkelstein, S. R. Kulkarni, and J. Xiao. TurkerGaze: Crowdsourcing saliency with webcam based eye tracking, April 2015. preprint arXiv:1504.06755.

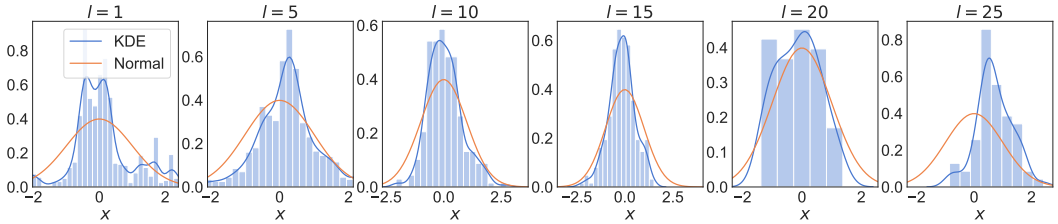
F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop, June 2015. preprint arXiv:1506.03365.

S. Zagoruyko and N. Komodakis. Wide residual networks. In *British Machine Vision Conference*, pp. 87.1–87.12, York, United Kingdom, September 2016. British Machine Vision Association.

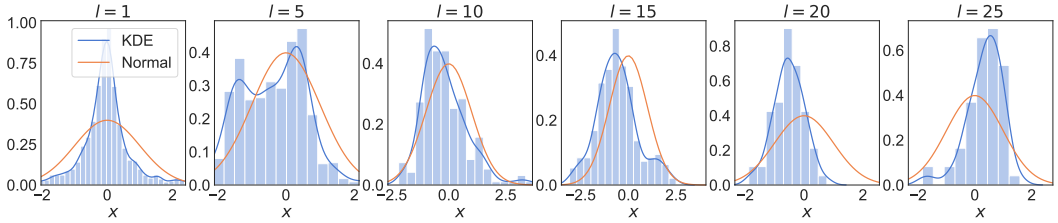
# A SUPPLEMENTARY RESULTS FOR SECTION 3



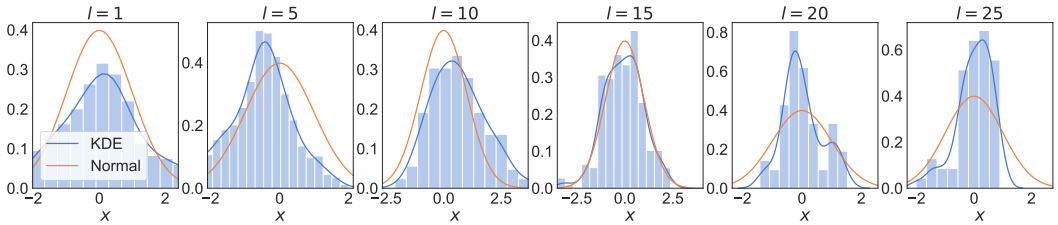

(a) CIFAR-10



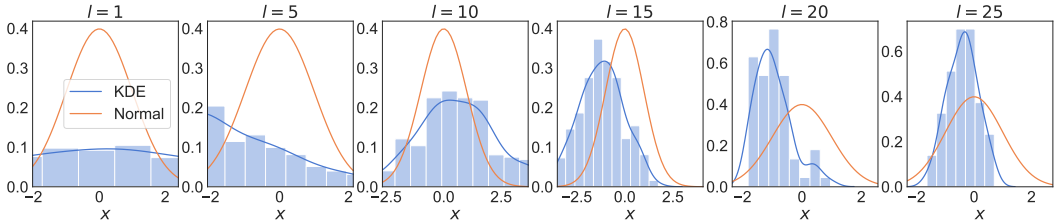

(b) TinyImageNet (c)




(c) TinyImageNet (r)




(d) iSUN



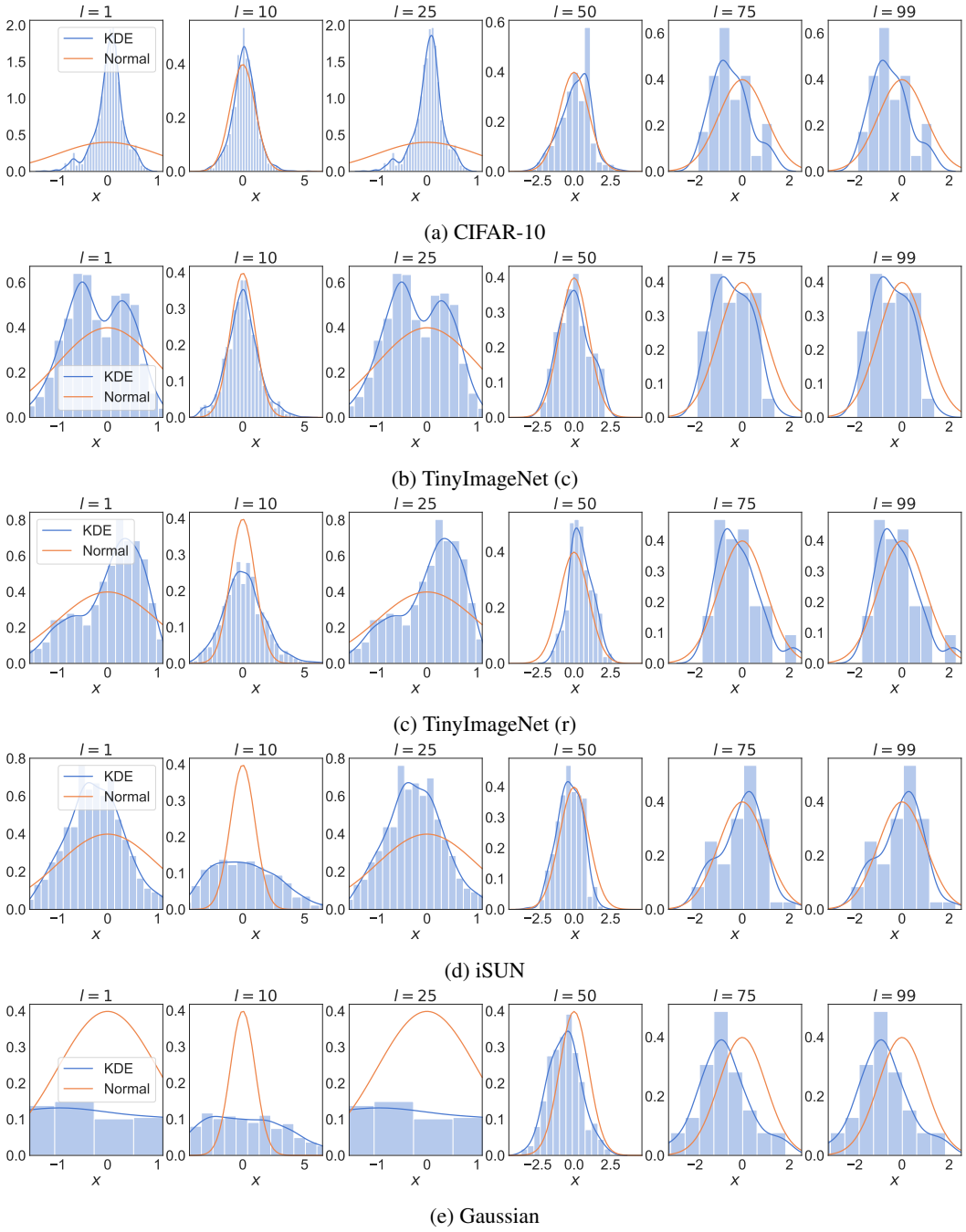

(e) Gaussian

Figure 5: More visualization results following the same procedure as in Figure 2 for a WRN-28-10 model.

# B SUPPLEMENTARY RESULTS FOR SECTION 4

## B.1 PARAMETER FINE-TUNING FOR ODIN

The temperature $T$ and noise magnitude $\epsilon$ hyperparameters were tuned using 1000 samples from the CIFAR test set as validation and evaluated on the remaining 9000 samples. The hyper-parameter




(a) CIFAR-10

(b) TinyImageNet (c)

(c) TinyImageNet (r)

(d) iSUN

(e) Gaussian

Figure 6: More visualization results following the same procedure as in Figure 2 but for a DenseNet-BC-100-12 model.

tuning is carried out on each pair of in- and out-of-distribution samples, which is the same procedure presented in Liang et al. (2018) and Lee et al. (2018b). For reproducibility, a grid search is employed considering $T \in \{1, 1000\}$ and $\epsilon$ with 21 linearly spaced values between 0 and 0.004 plus $[0.00005, 0.0005, 0.0011]$.

### B.2 ADDITIONAL BACKBONE MODEL

We also train a Wide ResNet with $L = 40$ and $k = 4$ (WRN-40-4) using the same training setup in Section 4, and the results for both training on CIFAR-10 and CIFAR-100 is depicted in Table **??**.

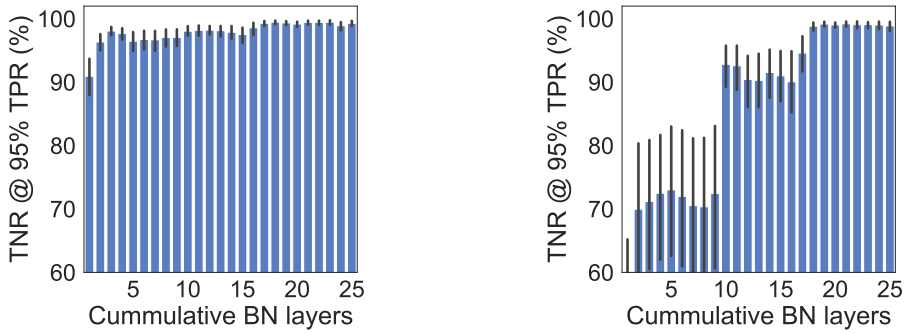
### B.3 OTHER EVALUATED METRICS

To compare all methods described in Section 2 to our proposed method, we also use the following additional metrics:

1. **Detection accuracy**. This metric is defined as the maximum classification probability over all possible thresholds $\delta$, considering that both in- and out-of-distribution have equal probability of appearing in the test set.
2. **Area under precision-recall curve (AUPR)**. The precision is evaluated as TP/(TP+FP) and recall in this case is the TPR. The AUPR-in (AUPR-out) is defined when in-(out-of)-distribution samples are considered as the positive ones.

### B.4 INTER-LAYER ENSEMBLE.

In this experiment, we evaluate the effect of ensembling across the BN layers, as described in Section 3.2, using a logistic regression. As shown in Figure 7, aggregating confidence scores of intermediate layers indeed improves the detection performance of OOD samples. The left-most bin indicates the TNR at 95% of TPR using only the last BN layer. As we go to the right, we aggregate the shallower BN layers, and the right-most bin indicates that all the BN layers are being used. Another important insight is that deeper layers (on the left of Figure 7) contribute more for the separation between in- and out-of-distribution samples.




(a) CIFAR-10

(b) CIFAR-100

Figure 7: TNR @ 95% TPR obtained when aggregating information from multiple layers. The leftmost bin corresponds to the deepest (last) layer, and the rightmost bin to the first BN layer in the network using a WRN-28-10 trained on CIFAR-10/CIFAR-100, the logistic regressor was fitted on TinyImageNet (c) + Gaussian and the results are an average over all OOD test datasets.

### B.5 OOD DETECTION COMPARISON

Table 9: Comparison between ODIN and our proposed OOD detector for several setups in image classification task. The hyperparameters were tuned for TinyImageNet (c) and Gaussian and tested on others, and we only use the standard deviation as feature. All values are in percentage. Higher values are better.

| | **Out-of-distribution dataset** | **TNR (95% TPR)** | **AUROC** | **AUPR In** | **AUPR Out** |
|---|---|---|---|---|---|
| | | **ODIN / Ours** | | | |
| **DenseNet BC 100-12** CIFAR-10 | TinyImageNet (c) | 90.5 (2.1) / **100.0 (0.0)** | 98.0 (0.3) / **100.0 (0.0)** | 98.5 (0.2) / **100.0 (0.0)** | 96.9 (0.6) / **100.0 (0.0)** |
| | TinyImageNet (r) | 76.9 (4.7) / **96.6 (1.4)** | 95.8 (0.8) / **99.3 (0.3)** | 96.4 (0.7) / **99.3 (0.3)** | 94.3 (1.2) / **99.3 (0.2)** |
| | LSUN (c) | 82.9 (6.1) / **100.0 (0.0)** | 97.0 (0.7) / **100.0 (0.0)** | 97.7 (0.5) / **100.0 (0.0)** | 95.7 (0.9) / **100.0 (0.0)** |
| | LSUN (r) | 86.2 (3.7) / **99.2 (0.5)** | 97.2 (0.6) / **99.8 (0.1)** | 97.8 (0.4) / **99.8 (0.1)** | 95.6 (1.1) / **99.7 (0.1)** |
| | iSUN | 81.1 (4.6) / **98.4 (0.9)** | 96.4 (0.8) / **99.6 (0.2)** | 97.3 (0.6) / **99.6 (0.1)** | 93.9 (1.6) / **99.5 (0.2)** |
| | Uniform | 68.7 (40.5) / **100.0 (0.0)** | 91.6 (9.2) / **100.0 (0.0)** | 95.0 (5.6) / **100.0 (0.0)** | 83.3 (13.1) / **100.0 (0.0)** |
| | Gaussian | 75.3 (31.2) / **100.0 (0.0)** | 96.2 (1.8) / **100.0 (0.0)** | 97.8 (1.1) / **100.0 (0.0)** | 89.2 (4.1) / **100.0 (0.0)** |
| **DenseNet BC 100-12** CIFAR-100 | TinyImageNet (c) | 72.9 (4.1) / **100.0 (0.0)** | 95.4 (0.4) / **100.0 (0.0)** | 96.5 (0.4) / **100.0 (0.0)** | 92.6 (0.4) / **100.0 (0.0)** |
| | TinyImageNet (r) | 39.6 (4.5) / **95.3 (0.7)** | 88.3 (1.3) / **99.0 (0.2)** | 89.8 (1.3) / **98.9 (0.2)** | 84.1 (1.3) / **99.0 (0.1)** |
| | LSUN (c) | 54.7 (4.7) / **100.0 (0.0)** | 93.3 (0.5) / **100.0 (0.0)** | 94.9 (0.4) / **100.0 (0.0)** | 90.2 (0.6) / **100.0 (0.0)** |
| | LSUN (r) | 42.2 (3.3) / **98.1 (0.7)** | 89.8 (0.8) / **99.5 (0.1)** | 91.6 (0.9) / **99.5 (0.1)** | 85.0 (0.8) / **99.5 (0.1)** |
| | iSUN | 34.4 (3.6) / **96.2 (1.0)** | 87.5 (0.9) / **99.2 (0.2)** | 90.4 (0.8) / **99.3 (0.2)** | 80.0 (1.1) / **99.1 (0.2)** |
| | Uniform | 0.0 (0.0) / **100.0 (0.0)** | 67.3 (7.9) / **100.0 (0.0)** | 79.7 (5.4) / **100.0 (0.0)** | 55.0 (5.5) / **99.9 (0.1)** |
| | Gaussian | 0.0 (0.0) / **100.0 (0.0)** | 39.0 (15.0) / **100.0 (0.0)** | 57.4 (13.2) / **100.0 (0.0)** | 41.2 (5.7) / **100.0 (0.0)** |
| **WRN 28-10** CIFAR-10 | TinyImageNet (c) | 91.9 (2.0) / **99.9 (0.0)** | 98.5 (0.3) / **100.0 (0.0)** | 98.8 (0.2) / **100.0 (0.0)** | 98.2 (0.4) / **100.0 (0.0)** |
| | TinyImageNet (r) | 73.4 (1.9) / **97.1 (1.3)** | 91.9 (0.8) / **99.4 (0.2)** | 89.0 (1.3) / **99.5 (0.2)** | 93.0 (0.6) / **99.4 (0.2)** |
| | LSUN (c) | 91.8 (2.3) / **100.0 (0.0)** | 98.4 (0.4) / **100.0 (0.0)** | 98.6 (0.3) / **100.0 (0.0)** | 97.9 (0.5) / **100.0 (0.0)** |
| | LSUN (r) | 84.7 (1.8) / **99.4 (0.4)** | 95.8 (0.4) / **99.8 (0.1)** | 94.6 (0.4) / **99.8 (0.1)** | 96.2 (0.5) / **99.8 (0.1)** |
| | iSUN | 79.8 (1.8) / **98.6 (0.9)** | 94.4 (0.5) / **99.6 (0.2)** | 93.4 (0.7) / **99.7 (0.1)** | 94.3 (0.6) / **99.6 (0.2)** |
| | Uniform | 98.9 (1.1) / **100.0 (0.0)** | 99.3 (0.3) / **100.0 (0.0)** | 99.5 (0.2) / **100.0 (0.0)** | 98.4 (0.4) / **100.0 (0.0)** |
| | Gaussian | **100.0 (0.0)** / 100.0 (0.0) | 99.6 (0.3) / **100.0 (0.0)** | 99.8 (0.2) / **100.0 (0.0)** | 98.9 (1.0) / **100.0 (0.0)** |
| **WRN 28-10** CIFAR-100 | TinyImageNet (c) | 68.3 (3.0) / **100.0 (0.0)** | 94.6 (0.5) / **100.0 (0.0)** | 95.4 (0.5) / **100.0 (0.0)** | 93.7 (0.5) / **100.0 (0.0)** |
| | TinyImageNet (r) | 43.7 (2.3) / **96.8 (2.4)** | 87.7 (0.6) / **99.3 (0.5)** | 88.9 (0.7) / **99.3 (0.5)** | 85.9 (0.6) / **99.3 (0.5)** |
| | LSUN (c) | 38.8 (3.9) / **99.9 (0.0)** | 86.6 (2.3) / **99.9 (0.0)** | 88.6 (2.4) / **99.9 (0.0)** | 84.3 (2.2) / **99.9 (0.0)** |
| | LSUN (r) | 43.7 (1.3) / **98.2 (1.8)** | 88.5 (0.2) / **99.6 (0.4)** | 89.8 (0.2) / **99.6 (0.3)** | 86.3 (0.5) / **99.5 (0.4)** |
| | iSUN | 39.7 (2.0) / **97.5 (2.3)** | 87.2 (0.3) / **99.4 (0.4)** | 89.6 (0.3) / **99.5 (0.4)** | 83.1 (0.8) / **99.3 (0.5)** |
| | Uniform | 50.1 (49.0) / **99.8 (0.3)** | 78.5 (28.2) / **99.8 (0.3)** | 84.7 (20.7) / **99.9 (0.2)** | 76.7 (25.4) / **99.7 (0.5)** |
| | Gaussian | 25.3 (42.4) / **100.0 (0.0)** | 70.1 (26.2) / **100.0 (0.0)** | 79.6 (19.2) / **100.0 (0.0)** | 65.1 (21.9) / **100.0 (0.0)** |
| **WRN 40-4** CIFAR-10 | TinyImageNet (c) | 93.3 (1.1) / **100.0 (0.0)** | 98.3 (0.1) / **100.0 (0.0)** | 98.8 (0.1) / **100.0 (0.0)** | 97.4 (0.2) / **100.0 (0.0)** |
| | TinyImageNet (r) | 65.9 (4.0) / **96.9 (2.2)** | 89.7 (1.4) / **99.4 (0.3)** | 87.0 (1.7) / **99.4 (0.3)** | 90.1 (1.3) / **99.4 (0.3)** |
| | LSUN (c) | 93.0 (2.5) / **100.0 (0.0)** | 98.1 (0.3) / **100.0 (0.0)** | 98.6 (0.2) / **100.0 (0.0)** | 97.0 (0.6) / **100.0 (0.0)** |
| | LSUN (r) | 78.1 (2.2) / **98.6 (1.2)** | 93.9 (0.6) / **99.6 (0.2)** | 92.6 (0.8) / **99.7 (0.2)** | 93.4 (0.5) / **99.6 (0.2)** |
| | iSUN | 72.8 (3.0) / **97.9 (1.6)** | 92.5 (0.8) / **99.5 (0.3)** | 91.7 (0.9) / **99.6 (0.2)** | 91.0 (0.9) / **99.4 (0.3)** |
| | Uniform | 50.9 (32.3) / **100.0 (0.0)** | 84.5 (12.6) / **100.0 (0.0)** | 82.5 (14.6) / **100.0 (0.0)** | 81.4 (12.4) / **100.0 (0.0)** |
| | Gaussian | 55.8 (41.2) / **100.0 (0.0)** | 84.0 (15.2) / **100.0 (0.0)** | 82.8 (17.8) / **100.0 (0.0)** | 80.6 (14.2) / **100.0 (0.0)** |
| **WRN 40-4** CIFAR-100 | TinyImageNet (c) | 72.8 (2.2) / **100.0 (0.0)** | 95.2 (0.4) / **100.0 (0.0)** | 96.0 (0.4) / **100.0 (0.0)** | 93.4 (0.4) / **100.0 (0.0)** |
| | TinyImageNet (r) | 42.5 (2.6) / **95.2 (2.1)** | 85.8 (1.7) / **99.0 (0.4)** | 85.2 (2.4) / **99.0 (0.4)** | 83.8 (1.5) / **99.1 (0.3)** |
| | LSUN (c) | 44.3 (1.7) / **100.0 (0.0)** | 89.7 (0.7) / **99.9 (0.0)** | 91.7 (0.7) / **100.0 (0.0)** | 86.7 (0.9) / **99.9 (0.0)** |
| | LSUN (r) | 46.1 (2.6) / **98.0 (1.3)** | 87.8 (1.4) / **99.5 (0.2)** | 87.7 (2.0) / **99.5 (0.2)** | 85.2 (1.1) / **99.5 (0.2)** |
| | iSUN | 40.0 (3.0) / **96.9 (1.7)** | 85.8 (1.5) / **99.3 (0.3)** | 87.0 (1.8) / **99.4 (0.2)** | 81.1 (1.7) / **99.2 (0.3)** |
| | Uniform | 28.5 (39.1) / **99.7 (0.4)** | 71.3 (26.7) / **99.9 (0.2)** | 78.4 (19.7) / **99.9 (0.2)** | 68.1 (21.2) / **99.8 (0.4)** |
| | Gaussian | 5.3 (6.8) / **100.0 (0.0)** | 77.7 (11.8) / **100.0 (0.0)** | 84.5 (9.1) / **100.0 (0.0)** | 66.9 (10.7) / **100.0 (0.0)** |

Table 10: Comparison between OOD detectors for several setups in image classification task. In each method, we tune all hyperparameters for each pair of in- and out-of-distribution datasets. All values are percentages. Higher values are better.

| | **Out-of-distribution dataset** | **TNR (95% TPR)** | **Detection accuracy** | **AUROC** | **AUPR In** | **AUPR Out** |
|---|---|---|---|---|---|---|
| | | | | **Baseline / ODIN / Ours** | | |
| | TinyImageNet (c) | 90.7 (2.0) / **100.0 (0.0)** | 94.2 (0.5) / **99.8 (0.0)** | 98.0 (0.3) / **100.0 (0.0)** | 98.5 (0.2) / **100.0 (0.0)** | 96.9 (0.6) / **100.0 (0.0)** |
| | TinyImageNet (r) | 70.4 (5.5) / **100.0 (0.0)** | 88.3 (0.7) / **99.7 (0.1)** | 94.6 (0.8) / **100.0 (0.0)** | 95.2 (0.6) / **100.0 (0.0)** | 92.9 (1.5) / **100.0 (0.0)** |
| | LSUN (c) | 92.7 (9.8) / **100.0 (0.0)** | 95.4 (2.6) / **99.8 (0.0)** | 98.3 (1.3) / **100.0 (0.0)** | 98.7 (1.0) / **100.0 (0.0)** | 97.6 (1.7) / **100.0 (0.0)** |
| **DenseNet BC 100-12** | LSUN (r) | 82.4 (3.2) / **100.0 (0.0)** | 91.8 (0.6) / **99.9 (0.0)** | 96.7 (0.5) / **100.0 (0.0)** | 97.3 (0.4) / **100.0 (0.0)** | 95.0 (0.9) / **100.0 (0.0)** |
| CIFAR-10 | iSUN | 78.0 (8.3) / **100.0 (0.0)** | 90.5 (1.8) / **99.9 (0.0)** | 95.9 (1.4) / **100.0 (0.0)** | 96.8 (1.1) / **100.0 (0.0)** | 93.2 (2.4) / **100.0 (0.0)** |
| | Uniform | 36.0 (40.8) / **100.0 (0.0)** | 73.1 (23.1) / **100.0 (0.0)** | 48.2 (48.2) / **100.0 (0.0)** | 64.3 (33.6) / **100.0 (0.0)** | 60.2 (29.6) / **100.0 (0.0)** |
| | Gaussian | 40.0 (42.4) / **100.0 (0.0)** | 73.5 (23.6) / **100.0 (0.0)** | 48.5 (48.5) / **100.0 (0.0)** | 64.5 (33.8) / **100.0 (0.0)** | 60.7 (30.2) / **100.0 (0.0)** |
| | TinyImageNet (c) | 75.0 (3.7) / **100.0 (0.0)** | 91.3 (0.6) / **99.8 (0.0)** | 95.8 (0.4) / **100.0 (0.0)** | 96.9 (0.3) / **100.0 (0.0)** | 93.0 (0.4) / **100.0 (0.0)** |
| | TinyImageNet (r) | 42.3 (6.2) / **100.0 (0.0)** | 82.3 (2.4) / **99.5 (0.1)** | 89.0 (2.2) / **100.0 (0.0)** | 90.4 (2.2) / **100.0 (0.0)** | 84.9 (2.3) / **100.0 (0.0)** |
| | LSUN (c) | 65.5 (2.2) / **100.0 (0.0)** | 89.2 (0.7) / **99.8 (0.0)** | 94.6 (0.3) / **100.0 (0.0)** | 95.8 (0.3) / **100.0 (0.0)** | 91.9 (0.3) / **100.0 (0.0)** |
| **DenseNet BC 100-12** | LSUN (r) | 46.1 (2.2) / **100.0 (0.0)** | 84.6 (1.4) / **99.9 (0.0)** | 90.8 (1.0) / **100.0 (0.0)** | 92.5 (1.1) / **100.0 (0.0)** | 86.3 (0.7) / **100.0 (0.0)** |
| CIFAR-100 | iSUN | 36.2 (4.7) / **100.0 (0.0)** | 81.5 (1.8) / **99.8 (0.0)** | 88.1 (1.7) / **100.0 (0.0)** | 90.8 (1.4) / **100.0 (0.0)** | 80.8 (2.1) / **100.0 (0.0)** |
| | Uniform | 0.0 (0.0) / **100.0 (0.0)** | 50.0 (0.0) / **100.0 (0.0)** | 0.0 (0.0) / **100.0 (0.0)** | 30.7 (0.0) / **100.0 (0.0)** | 30.7 (0.0) / **100.0 (0.0)** |
| | Gaussian | 0.0 (0.0) / **100.0 (0.0)** | 50.0 (0.0) / **100.0 (0.0)** | 0.0 (0.0) / **100.0 (0.0)** | 30.7 (0.0) / **100.0 (0.0)** | 30.7 (0.0) / **100.0 (0.0)** |
| | TinyImageNet (c) | 95.5 (2.1) / **100.0 (0.0)** | 95.7 (0.9) / **99.8 (0.0)** | 98.9 (0.2) / **100.0 (0.0)** | 99.1 (0.2) / **100.0 (0.0)** | 98.6 (0.4) / **100.0 (0.0)** |
| | TinyImageNet (r) | 80.5 (4.0) / **100.0 (0.0)** | 88.5 (1.5) / **99.6 (0.1)** | 93.6 (1.7) / **100.0 (0.0)** | 91.4 (2.7) / **100.0 (0.0)** | 94.7 (1.4) / **100.0 (0.0)** |
| | LSUN (c) | 97.2 (2.5) / **100.0 (0.0)** | 96.6 (1.4) / **99.9 (0.0)** | 99.3 (0.5) / **100.0 (0.0)** | 99.4 (0.4) / **100.0 (0.0)** | 99.1 (0.7) / **100.0 (0.0)** |
| **WRN 28-10** | LSUN (r) | 89.1 (2.0) / **100.0 (0.0)** | 92.2 (0.9) / **99.9 (0.0)** | 96.6 (0.6) / **100.0 (0.0)** | 95.5 (0.9) / **100.0 (0.0)** | 97.0 (0.6) / **100.0 (0.0)** |
| CIFAR-10 | iSUN | 85.3 (2.5) / **100.0 (0.0)** | 90.7 (1.1) / **99.8 (0.1)** | 95.6 (1.0) / **100.0 (0.0)** | 94.7 (1.4) / **100.0 (0.0)** | 95.7 (1.0) / **100.0 (0.0)** |
| | Uniform | 74.8 (43.2) / **100.0 (0.0)** | 86.4 (21.0) / **100.0 (0.0)** | 74.7 (43.1) / **100.0 (0.0)** | 82.5 (29.9) / **100.0 (0.0)** | 81.9 (29.6) / **100.0 (0.0)** |
| | Gaussian | 0.0 (0.0) / **100.0 (0.0)** | 50.0 (0.0) / **100.0 (0.0)** | 0.0 (0.0) / **100.0 (0.0)** | 30.7 (0.0) / **100.0 (0.0)** | 30.7 (0.0) / **100.0 (0.0)** |
| | TinyImageNet (c) | 79.7 (2.3) / **100.0 (0.0)** | 91.0 (0.4) / **99.7 (0.1)** | 96.6 (0.3) / **100.0 (0.0)** | 97.2 (0.3) / **100.0 (0.0)** | 95.8 (0.3) / **100.0 (0.0)** |
| | TinyImageNet (r) | 60.3 (5.6) / **99.9 (0.0)** | 84.5 (1.4) / **99.5 (0.1)** | 92.1 (1.2) / **100.0 (0.0)** | 92.7 (1.0) / **100.0 (0.0)** | 91.0 (1.4) / **100.0 (0.0)** |
| | LSUN (c) | 71.5 (1.3) / **100.0 (0.0)** | 88.3 (0.4) / **99.7 (0.1)** | 95.2 (0.3) / **100.0 (0.0)** | 95.8 (0.3) / **100.0 (0.0)** | 94.5 (0.5) / **100.0 (0.0)** |
| **WRN 28-10** | LSUN (r) | 61.0 (8.0) / **100.0 (0.0)** | 85.2 (1.7) / **99.8 (0.0)** | 92.5 (1.6) / **100.0 (0.0)** | 93.2 (1.4) / **100.0 (0.0)** | 91.2 (2.0) / **100.0 (0.0)** |
| CIFAR-100 | iSUN | 53.5 (5.0) / **100.0 (0.0)** | 83.0 (1.3) / **99.6 (0.1)** | 90.6 (1.2) / **100.0 (0.0)** | 92.2 (1.0) / **100.0 (0.0)** | 87.7 (1.4) / **100.0 (0.0)** |
| | Uniform | 62.6 (41.4) / **100.0 (0.0)** | 89.0 (14.4) / **100.0 (0.0)** | 87.1 (19.2) / **100.0 (0.0)** | 91.2 (13.2) / **100.0 (0.0)** | 83.6 (21.8) / **100.0 (0.0)** |
| | Gaussian | 0.7 (1.1) / **100.0 (0.0)** | 62.9 (16.5) / **100.0 (0.0)** | 33.4 (37.2) / **100.0 (0.0)** | 54.2 (26.2) / **100.0 (0.0)** | 45.3 (19.3) / **100.0 (0.0)** |
| | TinyImageNet (c) | 95.2 (1.1) / **100.0 (0.0)** | 95.5 (0.3) / **99.8 (0.0)** | 98.5 (0.1) / **100.0 (0.0)** | 98.9 (0.1) / **100.0 (0.0)** | 97.6 (0.2) / **100.0 (0.0)** |
| | TinyImageNet (r) | 73.4 (4.8) / **100.0 (0.0)** | 86.1 (2.0) / **99.6 (0.1)** | 92.1 (1.7) / **100.0 (0.0)** | 90.3 (2.2) / **100.0 (0.0)** | 92.3 (1.5) / **100.0 (0.0)** |
| | LSUN (c) | 94.9 (5.2) / **100.0 (0.0)** | 96.0 (1.6) / **99.9 (0.0)** | 98.5 (0.8) / **100.0 (0.0)** | 98.9 (0.6) / **100.0 (0.0)** | 97.7 (1.4) / **100.0 (0.0)** |
| **WRN 40-4** | LSUN (r) | 84.5 (3.2) / **100.0 (0.0)** | 90.6 (1.3) / **99.9 (0.0)** | 95.4 (1.0) / **100.0 (0.0)** | 94.6 (1.4) / **100.0 (0.0)** | 95.0 (0.9) / **100.0 (0.0)** |
| CIFAR-10 | iSUN | 78.9 (3.1) / **100.0 (0.0)** | 88.6 (1.3) / **99.8 (0.1)** | 94.0 (1.2) / **100.0 (0.0)** | 93.5 (1.5) / **100.0 (0.0)** | 92.8 (1.1) / **100.0 (0.0)** |
| | Uniform | 0.0 (0.0) / **100.0 (0.0)** | 50.0 (0.0) / **100.0 (0.0)** | 0.0 (0.0) / **100.0 (0.0)** | 30.7 (0.0) / **100.0 (0.0)** | 30.7 (0.0) / **100.0 (0.0)** |
| | Gaussian | 0.0 (0.0) / **100.0 (0.0)** | 50.0 (0.0) / **100.0 (0.0)** | 0.0 (0.0) / **100.0 (0.0)** | 30.7 (0.0) / **100.0 (0.0)** | 30.7 (0.0) / **100.0 (0.0)** |
| | TinyImageNet (c) | 80.6 (2.2) / **100.0 (0.0)** | 91.3 (0.8) / **99.7 (0.1)** | 96.5 (0.4) / **100.0 (0.0)** | 97.2 (0.4) / **100.0 (0.0)** | 94.7 (0.4) / **100.0 (0.0)** |
| | TinyImageNet (r) | 52.0 (5.2) / **100.0 (0.0)** | 80.7 (2.3) / **99.5 (0.1)** | 87.6 (2.2) / **100.0 (0.0)** | 86.4 (2.8) / **100.0 (0.0)** | 86.3 (2.0) / **100.0 (0.0)** |
| | LSUN (c) | 74.0 (1.8) / **100.0 (0.0)** | 89.8 (0.8) / **99.6 (0.1)** | 95.7 (0.4) / **100.0 (0.0)** | 96.4 (0.4) / **100.0 (0.0)** | 94.3 (0.2) / **100.0 (0.0)** |
| **WRN 40-4** | LSUN (r) | 57.5 (4.6) / **100.0 (0.0)** | 83.3 (1.6) / **99.9 (0.0)** | 90.1 (1.7) / **100.0 (0.0)** | 89.6 (2.2) / **100.0 (0.0)** | 88.2 (1.8) / **100.0 (0.0)** |
| CIFAR-100 | iSUN | 51.1 (6.1) / **100.0 (0.0)** | 81.2 (2.0) / **99.7 (0.0)** | 88.0 (2.2) / **100.0 (0.0)** | 88.4 (2.5) / **100.0 (0.0)** | 84.4 (2.7) / **100.0 (0.0)** |
| | Uniform | 37.2 (37.5) / **100.0 (0.0)** | 79.6 (13.8) / **100.0 (0.0)** | 78.7 (21.6) / **100.0 (0.0)** | 84.3 (15.2) / **100.0 (0.0)** | 74.3 (20.4) / **100.0 (0.0)** |
| | Gaussian | 31.8 (19.9) / **100.0 (0.0)** | 85.1 (8.4) / **100.0 (0.0)** | 87.3 (10.0) / **100.0 (0.0)** | 91.2 (7.2) / **100.0 (0.0)** | 78.7 (11.9) / **100.0 (0.0)** |