Learning Out-of-distribution Detection without Out-of-distribution Data

Anonymous authors
Paper under double-blind review

Abstract

Deep neural networks have attained remarkable performance when applied to data that comes from the same distribution as that of the training set, but can significantly degrade otherwise. Therefore, detecting whether an example is out-of-distribution (OOD) is crucial to enable a system that can reject such samples or alert users. Recent works have made significant progress on OOD benchmarks consisting of small image datasets. However, such methods rely on training or tuning with both in-distribution and out-of-distribution data. The latter is generally hard to define *a-priori*, and its selection can easily bias the learning. In this work, we focus on the feasibility of learning OOD detection without OOD data, proposing two strategies for the problem. We specifically propose to decompose confidence scoring as well as a modified input pre-processing method. We show that both of these significantly help detection performance, all without tuning to any out-of-distribution data during training. Our further analysis on a larger scale image dataset shows that the two types of distribution shifts, specifically semantic shift and non-semantic shift, present a significant difference in the difficulty of the problem, providing an analysis of when the proposed strategies do or do not work.

1 Introduction

State-of-the-art machine learning models, specifically deep neural networks, are generally designed for a static and closed world. The models are trained under the assumption that the input distribution at test time will be the same as the training distribution. In the real world, however, data distributions shift over time in a complex, dynamic manner. Even worse, new concepts (*e.g.* new categories of objects) can be presented to the model at any time. Such within-class distribution shift and unseen concepts both may lead to catastrophic failures since the model still attempts to make predictions based on its closed-world assumption. These failures are therefore often silent in that they do not result in explicit errors in the model.

The above issue had been formulated as a problem of detecting whether an input data is from in-distribution (*i.e.* the training distribution) or out-of-distribution (*i.e.* a distribution different from the training distribution). A common baseline is to use the max value of class posterior probabilities output from a neural network classifier, which can in some cases be a good indicator for distinguishing in-distribution and out-of-distribution inputs (Hendrycks & Gimpel, 2017). However, they also show that such probabilities tend to be overconfident. Since then, several works (Liang et al., 2017; Lee et al., 2018b; Vyas et al., 2018; Ren et al., 2019) follow a similar setting and evaluation metric, achieving significant improvement on the problem. However, the success of the above methods requires tuning hyperparameters with out-of-distribution data. In response, Dhamija et al. (2018) and Hendrycks et al. (2019) push this idea further by using a carefully chosen out-of-distribution dataset to regularize the learning of class posteriors so that out-of-distribution data have a much lower confidence than in-distribution. Lastly, Lee et al. (2018a) use a generative model to generate out-of-distribution data around the boundary of the in-distribution. All of the above strategies show the effectiveness of using out-of-distribution data during learning.

The risk of having out-of-distribution data for learning or tuning comes from the bias of the selection, leading to a concern that hyperparameters tuned with one out-of-distribution dataset might not generalize to others. Shafaei et al. (2019) investigate this concern and provide a less biased evaluation procedure by enumerating all combinations of known out-of-distribution datasets (for tuning) and unknown out-of-distribution datasets (for testing).

One natural choice for avoiding the mentioned bias is to not use out-of-distribution data for learning or tuning. Therefore in this work we focus on the setting where only in-distribution data are available during both training and validation stages. Furthermore, we consider a standard classification setting, in that the class labels and data are the only available information (in contrast, *e.g.* , to the work by Shalev et al. (2018) which uses extra supervision from word embeddings for learning). To our knowledge, this is the first study focused on such a minimal setting, investigating the feasibility of learning out-of-distribution detection without out-of-distribution data.

Contribution: We propose two simple strategies for the problem under the above setting. First, inspired by the observation of classifier overconfidence (Hendrycks & Gimpel, 2017), we provide a new probabilistic perspective of re-scaling the logits for decomposing confidence (predicted class probabilities) during the training. We specifically add a binary variable, representing whether the data is in-distribution or not, and use this variable to define the probability of the class and this variable conditioned on the data. We show this decomposes into two terms, both of which can be learned and used to define a new classifier. Our formulation is general and we show several existing methods reduce to specific cases for it. Second, we build on the input preprocessing method from ODIN (Liang et al., 2017) and develop an effective strategy to tune its perturbation magnitude (which is a hyperparameter of the preprocessing method) with only in-distribution data. We then perform extensive evaluations on benchmark image datasets such as CIFAR10/100, TinyImageNet, LSUN, SVHN, as well as a larger scale dataset DomainNet, for investigating the conditions under which the proposed strategies **do** or **do not** work. The results show that the two strategies can significantly improve upon baseline methods, achieving a performance close to, and in several cases surpassing, state-of-the-art methods (Lee et al., 2018b) which use out-of-distribution data for tuning. Lastly, our systematical evaluation with DomainNet reveals the relative difficulties between two types of distribution shifts: semantic shift and non-semantic shift, which are defined by whether a shift is related to the inclusion of new semantic categories.

2 BACKGROUND

This work considers the OOD detection setting in classification problems. We begin with a dataset $D_{in} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, denotion in-distribution data $\boldsymbol{x}_i \in \mathbb{R}^k$ and categorical label $y_i \in \{\mathrm{y}_{in}\} = \{1..C\}$ for C classes. D_{in} is generated by sampling from a distribution $p_{in}(\mathrm{x}, \mathrm{y}_{in})$. We then have a discriminative model $f_\theta(\boldsymbol{x})$ with parameters θ learned with the in-domain dataset D_{in}, predicting the class posterior probability $p(\mathrm{y}_{in}|\boldsymbol{x})$.

When the learned classifier f_θ is deployed in the open world, it may encounter data drawn from a different distribution p_{out} such that $p_{out} \neq p_{in}$. Sampling from all possible distributions p_{out} that might be encountered is generally intractable especially when the dimension k is large, such as in the cases of image data. Note also that we can conceptually categorize the type of differences into non-semantic shift and semantic shift. Data with non-semantic shift is drawn from the distribution $p_{out}(\mathbf{x}, \mathrm{y}_{in})$. Examples with this shift come from the same object class but are presented in different forms, such as cartoon or sketch images. In the case of semantic shift, the data is drawn from a distribution $p_{out}(\mathrm{x}, \mathrm{y}_{out})$ with $\{\mathrm{y}_{out}\} \cap \{\mathrm{y}_{in}\} = \emptyset$. In other words, the data is from a class not seen in the training set D_{in}.

The above separation leads to two natural questions that must be answered for a model to work in an open world: How can the model avoid making a prediction when encountering an input $\boldsymbol{x} \sim p_{out}(\mathrm{x}, \mathrm{y}_{out})$, or reject a low confidence prediction when $\boldsymbol{x} \sim p_{out}(\mathrm{x}, \mathrm{y}_{in})$? In this work, we propose to introduce an *explicit* binary domain variable $\mathrm{d} \in \{d_{in}, d_{out}\}$ in order to represent this decision, with d_{in} meaning that the input is $\boldsymbol{x} \sim p_{in}$ while d_{out} meaning $\boldsymbol{x} \nsim p_{in}$ (or equivalently $\boldsymbol{x} \sim p_{out}$). Note that while generally the model cannot distinguish between the two cases we defined, we can still show that both of the questions above can be answered by estimating this single variable d.

The ultimate goal, then, is to find a scoring function $S(\boldsymbol{x})$ which correlates to the domain posterior probability $p(\mathrm{d}|\boldsymbol{x})$, in that a higher score s from $S(\boldsymbol{x})$ indicates a higher probability of $p(d_{in}|\boldsymbol{x})$. The binary decision now can be made by applying a threshold on s. Selecting such a threshold is subject to the application requirement or the performance metric calculation protocol. With the above notation, we can view the baseline method (Hendrycks & Gimpel, 2017) as a special case with a specific scoring function $S_{Base}(\boldsymbol{x}) = \max_{\mathrm{y}_{in}} p(\mathrm{y}_{in}|\boldsymbol{x})$, where $p(\mathrm{y}_{in}|\boldsymbol{x})$ is obtained from a standard neural network classifier f_θ trained with cross entropy loss. However, $S(\boldsymbol{x})$ can become a learnable parameterized function, and different OOD methods can then be categorized by specific parameterizations and learning procedures. A key differentiator between methods is whether the parameters are learned with or without OOD data.

2.1 Related Methods

This section describes the two methods that are the most related to our work: ODIN (Liang et al., 2017) and Mahalanobis (Lee et al., 2018b). These two methods will serve as strong baselines in our evaluation, since Shafaei et al. (2019) reported that ODIN achieved the strongest results among 15 methods (which doesn't include Mahalanobis), out-performing popular strategies such as MC-Dropout (Gal & Ghahramani, 2016), DeepEnsemble (Lakshminarayanan et al., 2017), PixelCNN++ (Salimans et al., 2017), and OpenMax (Bendale & Boult, 2016) with a less biased evaluation method. Mahalanobis has further been shown to have significant advantages over ODIN (Lee et al., 2018b). Note that both ODIN and Mahalanobis start from a vanilla classifier f_θ trained on D_{in}, then have a scoring function $S(\boldsymbol{x}; f_\theta)$ which has extra parameters to be tuned. In their original work, those parameters are specifically tuned for each OOD dataset. Here we will describe the methods to use them without tuning on OOD data.

ODIN comprises two strategies: temperature scaling and input preprocessing. The temperature scaling is applied to its scoring function which has $f_i(\boldsymbol{x})$ for the logit of i class:

$$S_{ODIN}(\boldsymbol{x}) = \max_i \frac{\exp\left(f_i(\boldsymbol{x})/T\right)}{\sum_{j=1}^{C} \exp\left(f_j(\boldsymbol{x})/T\right)} \tag{1}$$

Although ODIN originally involved tuning the hyperparameter T with out-of-distribution data, it was also shown that a large T value can generally be preferred, suggesting that the gain is saturated after 1000 (Liang et al., 2017). We follow this guidance and fix $T = 1000$ in our experiments.

Mahalanobis comprises two parts as well: Mahalanobis distance calculation and input preprocessing. The score is calculated with Mahalanobis distance as follows:

$$S^\ell_{Maha}(\boldsymbol{x}) = \max_i -(f^\ell(\boldsymbol{x}) - \mu^\ell_i)^T \Sigma_\ell^{-1} (f^\ell(\boldsymbol{x}) - \mu^\ell_i) \tag{2}$$

$$S_{Maha}(\boldsymbol{x}) = \sum_\ell \alpha_\ell S^\ell_{Maha}(\boldsymbol{x}) \tag{3}$$

The $f^\ell(\boldsymbol{x})$ represents the output features at the ℓth-layer of neural networks, while μ_i and Σ are the class mean representation and the covariance matrix, correspondingly. The hyperparameter is α_ℓ. In the original method, α_ℓ is regressed with a small validation set containing both in-distribution and out-of-distribution data. Therefore they have a set of α_ℓ tuned for each OOD dataset. As a result, for the baseline that does not tune on OOD data we use uniform weighting $S_{Maha}(\boldsymbol{x}) = \sum_\ell S^\ell_{Maha}(\boldsymbol{x})$.

Note that both methods use the input preprocessing strategy, which has a hyperparameter to be tuned. In their original works, this hyperparameter is tuned for each OOD dataset as well. Therefore we develop a version that does not require tuning with out-of-distribution data.

3 Approach

3.1 The Decomposed Confidence

Hendrycks & Gimpel (2017) observed that the softmax classifier tends to output a high confident prediction, reporting that "random Gaussian noise fed into an MNIST image classifier gives a predicted class probability of 91%". They attribute this to the use of the softmax function which is a smooth approximation of an indicator function, hence tending to give a spiky distribution instead of a uniform distribution over classes. We acknowledge this view and further consider it as a limitation in the design of the softmax classifier. To address this limitation, our inspiration starts from reconsidering its outputs, the class posterior probability $p(\mathrm{y}_{in}|\boldsymbol{x})$, which does not consider the domain d at all. In other words, current methods condition on domain $\mathrm{d} = d_{in}$ based on the implicit closed world assumption. Thus, we use our explicit variable d in the classifier, rewriting it as the quotient of the joint class-domain probability and the domain probability using the rule of conditional probability:

$$p(\mathrm{y}_{in}|d_{in}, \boldsymbol{x}) = \frac{p(\mathrm{y}_{in}, d_{in}|\boldsymbol{x})}{p(d_{in}|\boldsymbol{x})} \tag{4}$$

Equation 4 provides a probabilistic view of why classifiers tend to be overconfident. Consider an example $\boldsymbol{x} \sim p_{out}$: It is natural to expect that the joint probability $P(\mathrm{y}_{in}, d_{in}|\boldsymbol{x})$ is low (e.g. 0.09) for its maximum value among C classes. One would also expect its domain probability $p(d_{in}|\boldsymbol{x})$ is low (e.g. 0.1). Therefore, calculating $p(\mathrm{y}_{in}|d_{in}, \boldsymbol{x})$ with Equation 4 gives a high probability (0.9), demonstrating how overconfidence can result. Based on the form of Equation 4, we call $p(\mathrm{y}_{in}, d_{in}|\boldsymbol{x})$ and $p(d_{in}|\boldsymbol{x})$ the decomposed confidence scores.

One straight-forward solution for the above issue is to learn a classifier to predict the joint probability $p(\mathrm{y}_{in}, d_{in}|\boldsymbol{x})$ by having both supervision on class y_{in} and domain d. Learning to predict $p(\mathrm{y}_{in}, d_{in}|\boldsymbol{x})$ is preferred over $p(d_{in}|\boldsymbol{x})$ because it can serve both purposes for predicting a class by $\arg\max_{y_{in}} p(\mathrm{y}_{in}, d_{in}|\boldsymbol{x})$ and rejecting a prediction by thresholding. This idea relates to the work of Hendrycks et al. (2019), which adds an extra loss term to penalize a predicted non-uniform class probability when an out-of-distribution data is given to the classifier. In other words, this strategy requires out-of-distribution data for regularizing the training.

Without having supervision on domain d (*i.e.* without out-of-distribution data), there is no principled way to learn $p(\mathrm{y}_{in}, d_{in}|\boldsymbol{x})$. This situation is similar to unsupervised learning (or self-supervised learning) in that we need to insert assumptions or prior knowledge about the task for learning. In our case, we use the structure in Equation 4 as the prior knowledge to re-design the structure of classifiers, leading to the decomposed confidence property.

We begin with describing a softmax function with logit $f_i(\boldsymbol{x})$ for class i. The $f_i(\boldsymbol{x})$ is defined by two functions $h_i(\boldsymbol{x})$ and $g(\boldsymbol{x})$:

$$p(\mathrm{y}_{in} = i|d_{in}, \boldsymbol{x}) = \frac{\exp f_i(\boldsymbol{x})}{\sum_{j=1}^{C} \exp f_j(\boldsymbol{x})}, \text{ with } f_i(\boldsymbol{x}) = \frac{h_i(\boldsymbol{x})}{g(\boldsymbol{x})}. \tag{5}$$

The $h_i(\boldsymbol{x})$ is a class-specific function similar to $p(\mathrm{y}_{in} = i, d_{in}|\boldsymbol{x})$, while the $g(\boldsymbol{x})$ is shared among all classes like $p(d_{in}|\boldsymbol{x})$. The quotient of the two scores is then normalized by the exponential function (*i.e.* softmax) for outputting a class probability, which is subject to cross entropy loss. With Equation 5, the cross entropy loss can be minimized in two ways: increasing $h_i(\boldsymbol{x})$ or decreasing $g(\boldsymbol{x})$. Here we make an assumption: There exist a pair of functions for $h_i(\boldsymbol{x})$ and $g(\boldsymbol{x})$ so that $h_i(\boldsymbol{x})$ will have a harder time to output a large value for the data not in the high density region of in-distribution, but $g(x)$ can easily output a small value (e.g. close to zero) for that case, making the logit $f_i(\boldsymbol{x})$ still able to generate a high $p(\mathrm{y}_{in} = i|d_{in}, \boldsymbol{x})$ for minimizing cross entropy loss.

There are several simple design choices for $h_i(\boldsymbol{x})$ and $g(\boldsymbol{x})$ which satisfy the above assumption. Specifically we have $g(\boldsymbol{x}) = \sigma(BN(\boldsymbol{w}_g f^p(\boldsymbol{x}) + b_g))$, which uses features $f^p(\boldsymbol{x})$ from the penultimate layer of neural networks sequentially through another linear layer, batch normalization (BN, optional for a faster convergence), and a sigmoid function σ. For $h_i(\boldsymbol{x})$, we investigate three similarity measures, including inner-product (I), negative Euclidean distance (E), and cosine similarity (C) for $h_i^I(\boldsymbol{x})$, $h_i^E(\boldsymbol{x})$, and $h_i^C(\boldsymbol{x})$, correspondingly:

$$h_i^I(\boldsymbol{x}) = \boldsymbol{w}_i f^p(\boldsymbol{x}) + b_i; \quad h_i^E(\boldsymbol{x}) = -\|f^p(\boldsymbol{x}) - \boldsymbol{w}_i\|^2; \quad h_i^C(\boldsymbol{x}) = \frac{\boldsymbol{w}_i f^p(\boldsymbol{x})}{\|\boldsymbol{w}_i\|\|f^p(\boldsymbol{x})\|} \tag{6}$$

The overall neural network model f_θ therefore has two branches (h_i and g) after its penultimate layer. At training time, the model calculates the logit f_i followed by the softmax function with cross entropy loss on top of it. At testing time, the class prediction can be made by either calculating $\arg\max_i f_i(\boldsymbol{x})$ or $\arg\max_i h_i(\boldsymbol{x})$ (both will give the same predictions). For out-of-distribution detection, we use the scoring function $S_{DeConf}(\boldsymbol{x}) = \max_i h_i(\boldsymbol{x})$.

Note that when $h_i(\boldsymbol{x}) = h_i^I(\boldsymbol{x})$ and $g(\boldsymbol{x}) = 1$, this method reduces to the baseline (Hendrycks & Gimpel, 2017). Additionally, the method in a concurrent work done by Techapanurak & Okatani (2019) can be regarded as a specific way to instantiate our framework when $h_i(\boldsymbol{x}) = h_i^C(\boldsymbol{x})$. The above compatibility indicates that the proposed framework is general. Lastly, we call the three variants of our method DeConf-I, DeConf-E, and DeConf-C based the three different similarity measures (I, E, C) used in $h_i(\boldsymbol{x})$.

3.2 A Modified Input Preprocessing strategy

This section describes a modified version of the input preprocessing method proposed in ODIN (Liang et al., 2017). The main purpose of modification is making the search of the perturbation magnitude ϵ to not rely on out-of-distribution data. The perturbation of input is given by:

$$\hat{\boldsymbol{x}} = \boldsymbol{x} - \epsilon \text{sign}(-\nabla_{\boldsymbol{x}} S(\boldsymbol{x})) \tag{7}$$

In the original method (Liang et al., 2017) the best value of ϵ is searched with a half-half mixed validation dataset of $D_{in}^{val} \sim p_{in}$ and $D_{out}^{val} \sim p_{out}$ over a list of 21 values. The perturbed images $\hat{\boldsymbol{x}}$ are fed into the classification model f_θ for calculating the score $S(\boldsymbol{x})$. The performance of each magnitude is evaluated with the benchmark metric (TNR@TPR95, described later) and the best one is selected. This process repeats for each out-of-distribution dataset, and therefore the original method results in a number of ϵ values equal to the number of out-of-distribution datasets in the benchmark.

In our method, we search for the ϵ^* which maximizes the score $S(\boldsymbol{x})$ with only the in-distribution validation dataset D_{in}^{val}:

$$\epsilon^* = \arg\max_{\epsilon} \sum_{x \in D_{in}^{val}} S(\hat{\boldsymbol{x}}) \tag{8}$$

Our searching criteria is still based on the same observation made by Liang et al. (2017). They observe that the in-distribution images tend to have their score s increased more than the out-of-distribution images when the input perturbation is applied. We therefore use Eq. 8 since we argue that an ϵ which makes a large score increase for in-distribution data should be sufficient to create a distinction in score. Our method also does not even require class labels although it is available in D_{in}^{val}. More importantly, our method selects only one ϵ based on D_{in}^{val} without access to the benchmark performance metric (e.g. TNR@TPR95), greatly avoiding the hyperparameter from fitting to a specific benchmark score. Lastly, we perform the search of ϵ on a much coarser grid, which has only 5 values: $[0.0025, 0.005, 0.01, 0.02, 0.04]$. Therefore, our search is much faster. Although overshooting is possible (e.g. the maximum value is at the middle of two scales in the grid) due to the coarser grid, it can be mitigated by reducing the found magnitude by one scale (*i.e.* divide it by two). This simple strategy consistently gains or maintains the performance on varied scoring functions, such as S_{Base}, S_{DeConf}, S_{ODIN}, and S_{Maha}.

The method in this section is orthogonal to all the methods evaluated in this work. For convenience, we will add a * after the name of other methods to indicate a combination, for example Baseline* and DeConf-C*.

4 Experiments

4.1 Experimental Settings

Overall procedure: In all experiments, we first train a classifier f_θ on an in-distribution training set, then tune the hyperparameters (*e.g.* the perturbation magnitude ϵ) on an in-distribution validation set without using its class labels. At testing time, the OOD detection scoring function $S(\boldsymbol{x})$ calculates the scores s from the outputs of f_θ). The scores s is calculated for both in-distribution validation set D_{in}^{val} and out-of-distribution dataset $D_{out} \sim p_{out}$. The scores s are then sent to a performance metric calculation function. The above procedure is the same as related works in this line of research (Liang et al., 2017; Lee et al., 2018b; Hendrycks et al., 2019; Shafaei et al., 2019; Vyas et al., 2018;

Table 1: Performance of four OOD detection methods. All methods in the table have no access to OOD data during training and validation. ODIN* and Mahalanobis* are modified versions that don't need any OOD data for tuning (see Section 2.1). The base network used in the table is DenseNet. All values are percentages averaged over three runs and the best results are indicated in bold. We only show DeConf-C* instead of having all the variants for readability.

ID	OOD	AUROC	TNR@TPR95
		Baseline / ODIN* / Mahalanobis* / DeConf-C*	
CIFAR-100	Imagenet(c)	79.0 / 90.5 / 92.4 / **97.6**	25.3 / 56.0 / 63.5 / **87.8**
	Imagenet(r)	76.4 / 91.1 / 96.4 / **98.6**	22.3 / 59.4 / 82.0 / **93.3**
	LSUN(c)	78.6 / 89.9 / 81.2 / **95.3**	23.0 / 53.0 / 31.6 / **75.0**
	LSUN(r)	78.2 / 93.0 / 96.6 / **98.7**	23.7 / 64.0 / 82.6 / **93.8**
	iSUN	76.8 / 91.6 / 96.5 / **98.4**	21.5 / 58.4 / 81.2 / **92.5**
	SVHN	78.1 / 85.6 / 89.9 / **95.9**	18.9 / 35.3 / 43.3 / **77.0**
	Uniform	65.0 / 91.4 / **100.** / 99.9	2.95 / 66.1 / **100.** / **100.**
	Gaussian	48.0 / 62.0 / **100.** / 99.9	0.06 / 33.3 / **100.** / **100.**
CIFAR-10	Imagenet(c)	92.1 / 88.2 / 96.3 / **98.7**	50.0 / 47.8 / 81.2 / **93.4**
	Imagenet(r)	91.5 / 90.1 / 98.2 / **99.1**	47.4 / 51.9 / 90.9 / **95.8**
	LSUN(c)	93.0 / 91.3 / 92.2 / **98.3**	51.8 / 63.5 / 64.2 / **91.5**
	LSUN(r)	93.9 / 92.9 / 98.2 / **99.4**	56.3 / 59.2 / 91.7 / **97.6**
	iSUN	93.0 / 92.2 / 98.2 / **99.4**	52.3 / 57.2 / 90.6 / **97.5**
	SVHN	88.1 / 89.6 / 98.0 / **98.8**	40.5 / 48.7 / 90.6 / **94.0**
	Uniform	95.4 / 98.9 / **99.9** / **99.9**	59.9 / 98.1 / **100.** / **100.**
	Gaussian	94.0 / 98.6 / **100.** / 99.9	48.8 / 92.1 / **100.** / **100.**

Lee et al., 2018a), except that we do not use OOD data for tuning the hyperparameters in the scoring function $S(\boldsymbol{x})$.

In-distribution Datasets: We use CIFAR-10/100 images with size 32x32 (Krizhevsky et al., 2009) for the classification task. Detecting OOD with CIFAR-100 classifier is generally harder than CIFAR-10, since a larger amount of classes usually involves a wider range of variance thus has a higher tendency to treat random data (*e.g.* Gaussian noise) as in-distribution. For that reason we use CIFAR-100 in our ablation and robustness study.

Out-of-distribution Datasets: We include all the OOD datasets used in ODIN (Liang et al., 2017), which are TinyImageNet(crop), TinyImageNet(resize), LSUN(crop), LSUN(resize), iSUN, Uniform random images, and Gaussian random images. We further add SVHN (Netzer et al., 2011), a colored street numbers image dataset, to serve as a difficult OOD dataset. The selection is inspired by the finding in the line of works which uses a generative model for OOD detection (Ren et al., 2019; Nalisnick et al., 2018; Choi & Jang, 2018). Those works report that a generative model of CIFAR-10 assigns higher likelihood to SVHN images, indicating a hard case for OOD detection.

Networks and Training Details: We use DenseNet (Huang et al., 2017), ResNet (He et al., 2016), and WideResNet (Zagoruyko & Komodakis, 2016) for the classifier backbone. DenseNet has 100 layers with growth rate 12. It is trained with batch size 64 for 300 epochs with weight decay 0.0001. The ResNet and WideResNet-28-10 are trained with batch size 128 for 200 epochs with weight decay 0.0005. In both training, the optimizer is SGD with momentum 0.9, and the learning rate starts with 0.1 and decreases by factor 0.1 at 50% and 75% of the training epochs. Note that we do not apply weight decay for the weights in the $h_i(\boldsymbol{x})$ function of DeConf classifier since they work as the centroids for classes, and those weights are initialized with He-initialization (He et al., 2015).

Evaluation Metrics: We use the two most widely adopted metrics in the OOD detection literature. The first one is the area under the receiver operating characteristic curve (AUROC), which plots the true positive rate (TPR) of in-distribution data against the false positive rate (TPR) of OOD data by varying a threshold. Thus it can be regarded as an averaged score. The second one is true negative rate at 95% true positive rate (TNR@TPR95), which simulates an application requirement that the recall of in-distribution data should be 95%. Having a high TNR under a high TPR is much more challenging than having a high AUROC score, thus TNR@TPR95 can discern between high-performing OOD detectors better.

Table 2: OOD detection with OOD data versus without OOD data. The values of ODIN^{orig} and Maha^{orig} (abbreviation of Mahalanobis) are copied from the Mahalanobis paper (Lee et al., 2018b) which are tuned with OOD data. The values of ODIN*, Maha*, and DeConf-C* are copied from Table 1 of our paper which do not have any access to OOD data during training. All methods in this table use the same DenseNet for backbone. Note that the Mahalanobis paper (Lee et al., 2018b) only has three OOD datasets for the table, so we only compare with those cases.

ID	OOD	AUROC	TNR@TPR95
		ODIN^{orig} / Maha^{orig}/ ODIN* / Maha* / DeConf-C*	
CIFAR-100	Imagenet(r)	85.2 / 97.4 / 91.1 / 96.4 / **98.6**	42.6 / 86.6 / 59.4 / 82.0 / **93.3**
	LSUN(r)	85.5 / 98.0 / 93.0 / 96.6 / **98.7**	41.2 / 91.4 / 64.0 / 82.6 / **93.8**
	SVHN	93.8 / **97.2** / 85.6 / 89.9 / 95.9	70.6 / **82.5** / 35.3 / 43.3 / 77.0
CIFAR-10	Imagenet(r)	98.5 / 98.8 / 90.1 / 98.2 / **99.1**	92.4 / 95.0 / 51.9 / 90.9 / **95.8**
	LSUN(r)	99.2 / 99.3 / 92.9 / 98.2 / **99.4**	96.2 / 97.2 / 59.2 / 91.7 / **97.6**
	SVHN	95.5 / 98.1 / 89.6 / 98.0 / **98.8**	86.2 / 90.8 / 48.7 / 90.6 / **94.0**

4.2 Results and Discussion

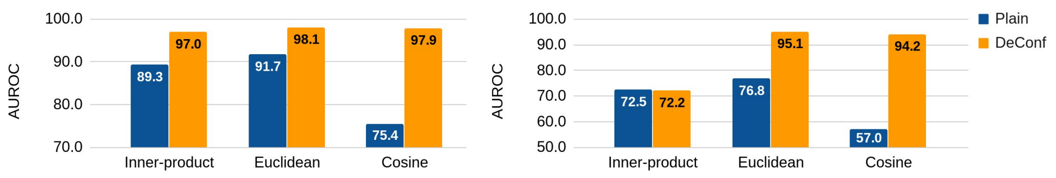
OOD benchmark performance: We show an overall comparison for methods that train without OOD data in Table 1 with 8 OOD benchmark datasets. The ODIN* and Mahalanobis* are significantly better than the baseline, while DeConf-C* still outperform them with a significant margin. These results clearly show that learning OOD detection without OOD data is feasible, and the two methods we proposed in Sections 3.1 and 3.2 combined are very effective for this purpose.

In Table 2 We further compare our results with the original ODIN (Liang et al., 2017) and Mahalanobis (Lee et al., 2018b) methods which are tuned on each OOD dataset. We refer to the results of both original methods reported by Lee et al. (2018b) since it uses the same backbone network, OOD datasets, and metrics to evaluate OOD detection performance. In the comparison we find our ODIN* and Mahalanobis* perform worse than the ODIN^{orig} and Mahalanobis^{orig} in a major fraction of the cases. The result is not surprising because the original methods gain the advantage from using OOD data. However, our DeConf-C* still outperforms the two original methods with a margin in most of the cases. The cross-setting comparison further supports the effectiveness of proposed strategies.

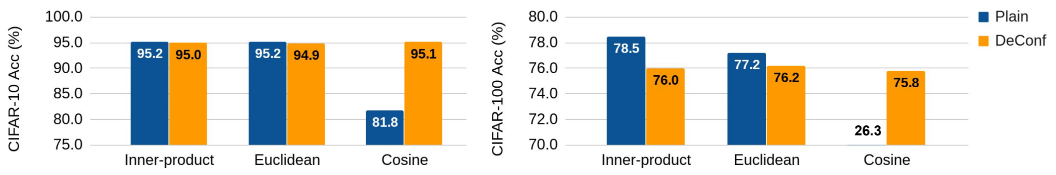
Ablation Study: We study the effect of applying DeConf and our modified input preprocessing (IPP) strategy separately in Figure 1 and 2. Figure 1 shows that all three variants (I, E, C) of the DeConf strategy help OOD detection performance with CIFAR-10 classifier, proving that the concept of DeConf is generally useful. However, the failure of DeConf-I with CIFAR-100 classifier may indicate the three variants have different robustness, which we will investigate in the next section. One downside of using the DeConf strategy is that the accuracy of the classifier may slightly reduce (Fig. 1b). This could be a natural consequence of having an alternative term, *i.e.* $g(\boldsymbol{x})$, in the model to fit the loss function. This may cause the lack of a high score for $h_i(\boldsymbol{x})$, instead assigning a lower score for the data away from the high density region of in-distribution data.

In Figure 2, the results show that tuning the perturbation magnitude with only in-distribution data is an effective strategy, allowing us to reduce the required supervision for learning. The supervision here means the binary label for in/out-of-distribution.

Robustness Study: This study investigates when the OOD detection method will or will not work. In Figure 3, it shows that the number of in-distribution training data can largely affect the performance of the OOD detector. Mahalanobis has the lowest data requirement, but the DeConf method generally reaches a higher performance in the high data regime. In Figure 3 we also examine scalability by varying the number of classes in the in-distribution data. In this test, DeConf-E* and DeConf-C* show the best scalability. Overall, DeConf-C* is more robust than the other two DeConf variants. Lastly, Figure 4 shows that high performing methods such as DeConf-E*, DeConf-C*, and Mahalanobis* are not sensitive to the type and depth of neural networks. Therefore, the number of in-distribution samples and classes are the main factors that affect OOD detection performance.



(a) OOD detection performance with/without DeConf strategy (no input preprocessing). Left: CIFAR-10 classifier; Right: CIFAR-100 classifier.



(b) In-domain classification accuracy

Figure 1: An ablation study with three variants in our DeConf method (Section 3.1). *Plain* means $g(\boldsymbol{x}) = 1$ so that the confidence decomposing effect is turned off. Each bar in (a) is averaged with the results on 8 OOD datasets listed in Table 1. The backbone network is Resnet-34. Note that the *plain* setting with inner-product is equivalent to a vanilla Resnet for classification. Overall, our DeConf boosts the OOD detection performance while losing a small fraction of classification accuracy on in-distribution data.

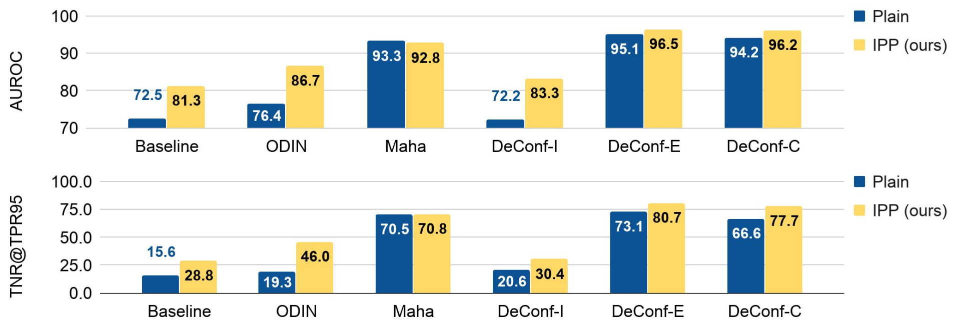


Figure 2: The OOD detection performance of our input preprocessing (IPP) strategy, which selects the perturbation magnitude with only in-distribution data. The setting *plain* means the IPP is turned off. The in-distribution data is CIFAR-100. The backbone network is Resnet-34. Each value is averaged with the results on 8 OOD datasets listed in Table 1. Each method has its own scoring function $S(\boldsymbol{x})$ (See Section 2.1 and 3), causing IPP to perform at varied levels of performance gain.

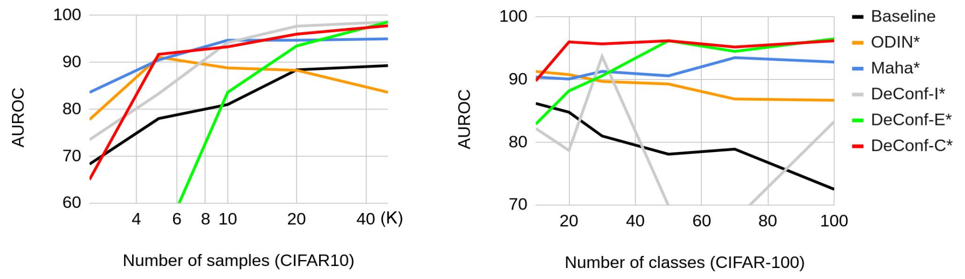


Figure 3: Robustness analysis of 6 OOD detection methods. The left figure has classifiers trained on varied number of samples in CIFAR-10. The right figure has classifiers trained on varied number of classes in CIFAR-100. Each point in the line is an average of the results on 8 OOD datasets. The backbone network is Resnet-34. Please see Section 4.2 for a detailed discussion.

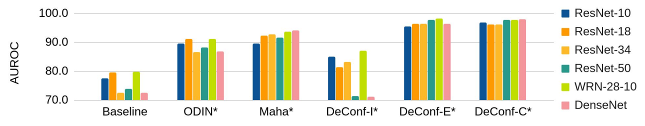


Figure 4: Robustness analysis using different neural network backbones. The in-distribution data is CIFAR-100. Each bar is averaged with the results on 8 OOD datasets.

Table 3: Performance of four OOD detection methods using DomainNet. The in-distribution is the real-A subset. Each value is averaged over three runs. The type of distribution shift presents a trend of difficulty to the OOD detection problem: Semantic shift > Non-semantic shift > Semantic + Non-semantic shift.

OOD	Type of shift		AUROC	TNR@TPR95
	Semantic	Non-semantic	Baseline / ODIN* / Maha* / DeConf-C*	
real-B	✓		**75.1** / 69.9 / 53.6 / 69.8	15.3 / **15.4** / 5.09 / 14.0
sketch-A		✓	75.5 / 80.7 / 59.5 / **84.5**	20.1 / 31.2 / 7.30 / **37.5**
sketch-B	✓	✓	81.8 / 85.7 / 60.4 / **89.1**	25.2 / 36.8 / 7.55 / **44.1**
infograph-A		✓	79.6 / 82.7 / 81.5 / **89.0**	23.5 / 27.8 / 21.6 / **45.4**
infograph-B	✓	✓	82.1 / 85.3 / 80.9 / **90.9**	24.8 / 31.7 / 21.9 / **49.6**
quickdraw-A		✓	78.8 / 96.4 / 67.4 / **96.9**	21.1 / 79.9 / 3.38 / **83.1**
quickdraw-B	✓	✓	80.5 / 96.9 / 66.1 / **97.4**	22.1 / 83.6 / 2.38 / **86.6**
Uniform	✓	✓	54.7 / 75.6 / **99.8** / 99.3	1.65 / 5.37 / **100.** / **100.**
Gaussian	✓	✓	71.3 / 95.5 / **99.9** / 99.4	0.64 / 46.9 / **100.** / **100.**

4.3 Semantic Shift versus Non-semantic Shift

One interesting aspect of out-of-distribution data that has not been explored is the separation of semantic and non-semantic shift. We therfore use a larger scale image dataset, DomainNet (Peng et al., 2018), to repeat an evaluation similar to Table 1. DomainNet has high resolution images in 345 classes from 6 different domains. There are 4 domains in the dataset with class labels available. They are real, sketch, infograph, and quickdraw, resulting in different types of distribution shifts.

To create subsets with semantic shift, we separate the classes to two splits. Split A has class indices from 0 to 172 while split B has 173 to 344. Our experiment uses real-A for in-distribution and has the other subsets for out-of-distribution. With the definition given in Section 2, real-B has a semantic shift from real-A, while sketch-A has a non-semantic shift. Sketch-B therefore has both types of distribution shift. Appendix Figure 5 illustrates the setup. The classifier learned on real-A uses a Resnet-34 backbone. Its training setting is described in Section 4.1 except that the networks are trained for 100 epochs and the images are center-cropped and resized to 224x224 in this experiment.

The results in Table 3 reveal two interesting trends. The first one is that the OOD datasets with both types of distribution shifts are easier to detect, followed by non-semantic shift. The semantic shift turns out to be the hardest one to be detected. The second observation is the failure of Mahalanobis*. In most of the cases it is even worse than Baseline, except detecting random noise. In contrast, ODIN* has performance gain in most of the cases, but has less gain with random noise. Our DeConf-C* still performs the best, showing that its robustness and scalability is capable of handling a more realistic problem setting, although there is still large room for improvement.

5 Conclusion

This paper shows methods and results that strongly suggest that learning OOD detection without OOD data is a viable option. Our comprehensive analysis shows that our two strategies, the decomposed confidence and modified input preprocessing, are effective and even better in most cases than the methods tuned for each OOD dataset. Our further analysis using a larger scale image dataset shows that the data with only semantic shift is harder to detect, pointing out a challenge for future works to address.

REFERENCES

Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1563–1572, 2016.

Hyunsun Choi and Eric Jang. Generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

Akshay Raj Dhamija, Manuel Günther, and Terrance Boult. Reducing network agnostophobia. In *Advances in Neural Information Processing Systems*. 2018.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *International Conference on Learning Representations (ICLR)*, 2017.

Dan Hendrycks, Mantas Mazeika, and Thomas G Dietterich. Deep anomaly detection with outlier exposure. *International Conference on Learning Representations (ICLR)*, 2019.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pp. 6402–6413, 2017.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *International Conference on Learning Representations (ICLR)*, 2018a.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pp. 7167–7177, 2018b.

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

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? *arXiv preprint arXiv:1810.09136*, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. *arXiv preprint arXiv:1812.01754*, 2018.

Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark A DePristo, Joshua V Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. *arXiv preprint arXiv:1906.02845*, 2019.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *International Conference on Learning Representations (ICLR)*, 2017.

Alireza Shafaei, Mark Schmidt, and James Little. A less biased evaluation of ood sample detectors. In *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2019.

Gabi Shalev, Yossi Adi, and Joseph Keshet. Out-of-distribution detection using multiple semantic label representations. In *Advances in Neural Information Processing Systems*, pp. 7375–7385, 2018.

Engkarat Techapanurak and Takayuki Okatani. Hyperparameter-free out-of-distribution detection using softmax of scaled cosine similarity. *CoRR*, abs/1905.10628, 2019. URL http://arxiv.org/abs/1905.10628.

Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, Bharat Kaul, and Theodore L Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 550–564, 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

APPENDICES

A ADDITIONAL FIGURES

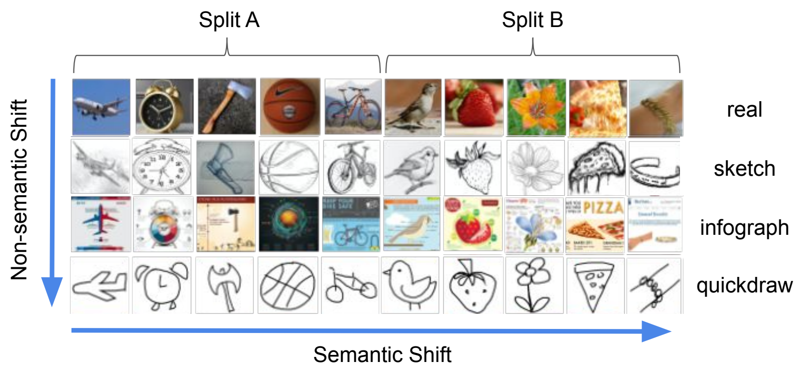


Figure 5: Semantic shift versus Non-semantic shift. We setup the two types of distribution shifts using DomainNet (Peng et al., 2018)

B ADDITIONAL RESULTS

Table 4: OOD detection performance with DomainNet. In this table we use whole *real* set (345 classes) for the in-distribution data. The remaining five domains are OOD data. The backbone networks is Resnet-34 trained from scratch. Each value is averaged with 3 runs. The values in parentheses are standard deviation. The type of distribution shift in this experiment is categorized as non-semantic shift.

OOD	AUROC	TNR@TPR95
	Baseline / ODIN* / Mahalanobis* / DeConf-C*	
painting	**71.4**(0.1) /62.4(0.2) /57.4(1.0) /66.7(0.2)	**15.6**(0.2) /5.55(0.2) /5.00(0.0) /7.40(0.5)
sketch	75.8(0.1) /81.8(0.3) /63.8(0.1) /**85.5**(0.3)	19.7(0.5) /30.6(0.8) /8.21(0.1) /**37.2**(1.5)
clipart	69.8(0.2) /82.6(0.1) /73.6(0.4) /**85.0**(0.0)	15.2(0.4) /35.3(0.5) /15.7(0.2) /**41.7**(0.6)
infograph	77.4(0.0) /82.1(0.3) /80.4(0.8) /**88.5**(0.2)	18.7(0.4) /25.7(1.1) /20.5(1.0) /**43.5**(2.0)
quickdraw	78.8(0.8) /**97.6**(0.1) /71.8(5.3) /97.4(0.2)	19.4(1.7) /**88.4**(0.8) /3.43(0.8) /87.5(1.7)
Uniform	74.3(28.) /88.6(13.) /**99.8**(0.0) /97.0(1.1)	38.5(41.) /66.4(46.) /**100.**(0.0) /91.1(6.9)
Gaussian	57.9(19.) /89.3(12.) /**99.9**(0.0) /98.1(0.1)	0.08(0.1) /66.6(47.) /**100.**(0.0) /99.9(0.1)

Table 5: Performance of Six OOD detection methods on 8 benchmark datasets. This is a full version of Table 1 which use DenseNet for the backbone networks. The values in parentheses are standard deviation.

ID	OOD	AUROC
		Baseline / ODIN* / Maha* / DeConf-I* / DeConf-E* / DeConf-C*
CIFAR-100	Imagenet(c)	79.0(2.2) /90.5(1.1) /92.4(0.3) /84.4(2.3) /95.1(0.5) /97.6(0.2)
	Imagenet(r)	76.4(3.2) /91.1(1.3) /96.4(0.2) /81.2(3.6) /97.4(0.3) /98.6(0.2)
	LSUN(c)	78.6(1.1) /89.9(0.5) /81.2(0.6) /91.7(0.3) /90.1(0.3) /95.3(0.4)
	LSUN(r)	78.2(2.4) /93.0(0.8) /96.6(0.2) /84.1(2.1) /97.8(0.2) /98.7(0.0)
	iSUN	76.8(2.7) /91.6(1.1) /96.5(0.2) /82.1(2.9) /97.4(0.2) /98.4(0.0)
	SVHN	78.1(3.5) /85.6(0.0) /89.9(0.2) /89.7(0.4) /94.0(0.6) /95.9(0.7)
	Uniform	65.0(22.) /91.4(10.) /100.(0.0) /48.5(16.) /99.9(0.0) /99.9(0.0)
	Gaussian	48.0(28.) /62.0(38.) /100.(0.0) /6.79(4.9) /99.9(0.0) /99.9(0.0)
CIFAR-10	Imagenet(c)	92.1(1.0) /88.2(4.2) /96.3(0.1) /98.2(0.0) /98.0(0.2) /98.7(0.1)
	Imagenet(r)	91.5(1.4) /90.1(4.1) /98.2(0.0) /98.4(0.0) /98.2(0.2) /99.1(0.1)
	LSUN(c)	93.0(0.5) /91.3(2.0) /92.2(0.4) /98.4(0.0) /98.6(0.2) /98.3(0.2)
	LSUN(r)	93.9(0.4) /92.9(2.9) /98.2(0.0) /98.6(0.0) /98.8(0.0) /99.4(0.1)
	iSUN	93.0(0.7) /92.2(3.4) /98.2(0.0) /98.6(0.0) /98.8(0.0) /99.4(0.0)
	SVHN	88.1(4.8) /89.6(0.3) /98.0(0.3) /98.2(0.2) /98.4(0.6) /98.8(0.1)
	Uniform	95.4(0.7) /98.9(0.7) /99.9(0.0) /99.2(0.5) /99.9(0.0) /99.9(0.0)
	Gaussian	94.0(2.9) /98.6(1.7) /100.(0.0) /99.1(0.3) /99.9(0.0) /99.9(0.0)

ID	OOD	TNR@TPR95
		Baseline / ODIN* / Maha* / DeConf-I* / DeConf-E* / DeConf-C*
CIFAR-100	Imagenet(c)	25.3(2.8) /56.0(3.1) /63.5(2.1) /31.0(3.4) /74.6(2.8) /87.8(1.7)
	Imagenet(r)	22.3(3.1) /59.4(3.7) /82.0(1.6) /21.4(4.0) /87.6(1.7) /93.3(1.2)
	LSUN(c)	23.0(2.2) /53.0(1.0) /31.6(1.3) /59.6(1.9) /51.0(1.0) /75.0(1.9)
	LSUN(r)	23.7(2.5) /64.0(3.0) /82.6(1.8) /21.1(3.3) /89.8(1.5) /93.8(0.3)
	iSUN	21.5(2.8) /58.4(4.1) /81.2(1.4) /17.6(3.3) /87.3(1.2) /92.5(0.2)
	SVHN	18.9(4.9) /35.3(2.9) /43.3(2.7) /52.0(0.6) /67.1(3.4) /77.0(5.0)
	Uniform	2.95(4.1) /66.1(46.) /100.(0.0) /0.0(0.0) /100.(0.0) /100.(0.0)
	Gaussian	0.06(0.0) /33.3(47.) /100.(0.0) /0.0(0.0) /100.(0.0) /100.(0.0)
CIFAR-10	Imagenet(c)	50.0(2.8) /47.8(15.) /81.2(0.8) /92.0(0.2) /90.1(1.5) /93.4(1.2)
	Imagenet(r)	47.4(4.4) /51.9(16.) /90.9(0.5) /93.6(0.2) /91.7(1.6) /95.8(0.9)
	LSUN(c)	51.8(3.1) /63.5(7.8) /64.2(0.6) /92.5(0.4) /93.3(1.5) /91.5(1.2)
	LSUN(r)	56.3(3.6) /59.2(18.) /91.7(0.3) /94.9(0.2) /95.7(0.1) /97.6(0.5)
	iSUN	52.3(3.6) /57.2(18.) /90.6(0.7) /94.6(0.3) /95.4(0.2) /97.5(0.3)
	SVHN	40.5(6.9) /48.7(3.2) /90.6(1.7) /91.4(1.1) /92.1(3.4) /94.0(0.6)
	Uniform	59.9(12.) /98.1(2.6) /100.(0.0) /99.9(0.0) /100.(0.0) /100.(0.0)
	Gaussian	48.8(26.) /92.1(11.) /100.(0.0) /99.9(0.0) /100.(0.0) /100.(0.0)

Table 6: Performance of Six OOD detection methods on 8 benchmark datasets. The backbone networks for this table is Resnet-34. The values in parentheses are standard deviation.

ID	OOD	AUROC
		Baseline / ODIN* / Maha* / DeConf-I* / DeConf-E* / DeConf-C*
CIFAR-100	Imagenet(c)	78.9(0.1) /84.8(0.6) /93.4(0.3) /88.2(0.6) /95.2(0.6) /95.3(0.6)
	Imagenet(r)	75.1(0.8) /85.7(0.2) /96.3(0.1) /84.6(1.0) /97.0(0.4) /95.9(0.7)
	LSUN(c)	78.8(0.6) /80.3(1.3) /79.8(0.3) /93.8(0.3) /92.6(0.2) /93.8(0.3)
	LSUN(r)	76.2(1.4) /86.6(0.8) /96.3(0.2) /85.9(1.8) /97.0(0.7) /96.1(0.5)
	iSUN	75.2(1.4) /85.9(0.8) /95.8(0.2) /84.7(1.4) /96.6(0.6) /95.7(0.5)
	SVHN	75.1(2.5) /80.2(2.0) /80.9(1.1) /89.2(2.6) /93.8(0.8) /93.2(1.1)
	Uniform	69.0(13.) /96.7(2.5) /100.(0.0) /79.3(8.3) /99.9(0.0) /99.9(0.0)
	Gaussian	51.5(1.8) /93.7(1.7) /99.9(0.0) /60.8(23.) /99.9(0.0) /99.9(0.0)
CIFAR-10	Imagenet(c)	90.0(0.9) /81.2(2.4) /94.2(0.1) /98.2(0.2) /98.2(0.1) /96.0(0.2)
	Imagenet(r)	87.3(1.3) /81.1(2.9) /96.5(0.1) /98.1(0.3) /98.1(0.3) /96.1(0.5)
	LSUN(c)	92.0(1.7) /77.9(4.6) /87.7(0.2) /98.8(0.1) /98.5(0.0) /97.2(0.1)
	LSUN(r)	91.6(1.2) /88.5(2.0) /97.2(0.1) /98.9(0.2) /99.0(0.1) /98.0(0.1)
	iSUN	90.1(1.4) /86.1(2.5) /96.5(0.2) /98.8(0.2) /98.9(0.1) /97.6(0.1)
	SVHN	87.7(2.4) /63.9(4.3) /87.8(1.6) /96.8(0.4) /96.1(1.4) /97.8(0.3)
	Uniform	85.9(10.) /93.3(4.5) /99.9(0.0) /99.6(0.1) /99.9(0.0) /99.9(0.0)
	Gaussian	89.9(10.) /97.1(2.0) /99.9(0.0) /99.7(0.0) /99.9(0.0) /99.9(0.0)
ID	**OOD**	**TNR@TPR95**
		Baseline / ODIN* / Maha* / DeConf-I* / DeConf-E* / DeConf-C*
CIFAR-100	Imagenet(c)	24.1(0.6) /44.0(2.2) /68.2(1.4) /42.6(2.7) /73.4(3.7) /72.6(3.7)
	Imagenet(r)	19.4(0.1) /45.5(1.4) /82.6(0.8) /30.4(3.0) /84.3(2.7) /76.5(3.8)
	LSUN(c)	21.9(0.4) /34.8(2.4) /27.7(1.4) /66.1(2.2) /59.7(0.7) /65.7(2.3)
	LSUN(r)	19.8(1.6) /48.2(3.0) /81.8(1.4) /29.4(5.2) /84.6(4.0) /76.8(3.3)
	iSUN	17.7(0.5) /45.3(2.8) /80.4(0.8) /27.1(4.3) /83.0(3.1) /75.3(3.3)
	SVHN	16.6(1.5) /27.5(5.0) /25.7(2.6) /43.7(10.) /60.8(5.3) /55.1(7.1)
	Uniform	5.63(7.0) /76.4(27.) /100.(0.0) /4.11(5.8) /100.(0.0) /100.(0.0)
	Gaussian	0.0(0.0) /46.6(20.) /100.(0.0) /0.06(0.0) /100.(0.0) /100.(0.0)
CIFAR-10	Imagenet(c)	54.6(2.6) /53.7(3.1) /74.6(0.6) /90.8(1.5) /91.1(0.9) /81.1(1.7)
	Imagenet(r)	48.3(3.2) /53.1(4.3) /85.1(0.6) /90.5(1.8) /90.8(1.8) /81.4(2.4)
	LSUN(c)	59.9(4.7) /50.9(6.1) /53.6(1.0) /93.9(0.5) /92.4(0.5) /87.3(1.0)
	LSUN(r)	57.5(4.4) /68.1(4.2) /87.4(0.8) /95.8(1.0) /96.0(0.7) /90.9(0.9)
	iSUN	53.7(3.8) /62.8(5.0) /84.6(0.9) /95.1(1.0) /95.3(0.5) /88.8(1.1)
	SVHN	44.5(8.1) /29.7(6.2) /46.2(4.8) /84.5(2.5) /78.8(7.6) /89.5(2.1)
	Uniform	27.9(20.) /74.5(20.) /100.(0.0) /100.(0.0) /100.(0.0) /100.(0.0)
	Gaussian	52.7(40.) /87.1(9.3) /100.(0.0) /100.(0.0) /100.(0.0) /100.(0.0)