Generalized ODIN: Detecting Out-of-distribution Image without Learning from Out-of-distribution Data

Yen-Chang Hsu[1], Yilin Shen[2], Hongxia Jin[2], Zsolt Kira[1]
[1]Georgia Institute of Technology, [2]Samsung Research America

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, many recent methods based on neural networks 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. We base our work on a popular method ODIN[1] [21], proposing two strategies for freeing it from the needs of tuning with OoD data, while improving its OoD detection performance. We specifically propose to decompose confidence scoring as well as a modified input pre-processing method. We show that both of these significantly help in detection performance. 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 ODIN-like 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) [13]. This problem has been studied for many years [12] and has been discussed in several views such as rejection [8, 5], anomaly detection [1], open set recognition [2], and uncertainty estimation [22, 23, 24]. In recent years, a popular neural network-based baseline is to use the max value of class posterior probabilities output from a softmax classifier, which can in some cases be a good indicator for distinguishing in-distribution and out-of-distribution inputs [13].

ODIN [21], based on a trained neural network classifier, provides two strategies, temperature scaling and input pre-processing, to make the max class probability a more effective score for detecting OoD data. Its performance has been further confirmed by [34], where 15 OoD detection methods are compared with a less biased evaluation protocol. ODIN out-performs popular strategies such as MC-Dropout [7], DeepEnsemble [18], PixelCNN++ [33], and OpenMax [3].

Despite its effectiveness, ODIN has a requirement that it needs OoD data to tune hyperparameters for both its strategies, leading to a concern that hyperparameters tuned with one out-of-distribution dataset might not generalize to others, discussed in [34]. In fact, other neural network-based methods [20, 38], which follow the same problem setting, have a similar requirement. [6, 14] push the idea of utilizing OoD data further by using a carefully chosen OoD dataset to regularize the learning of class posteriors so that OoD data have much lower confidence than in-distribution. Lastly, [19] uses a generative model to generate out-of-distribution data around the boundary of the in-distribution for learning.

Although the above works show that learning with OoD data is effective, the space of OoD data (ex: image pixel space) is usually too large to be covered, potentially causing a selection bias for the learning. Some previous works have done a similar attempt to learn without OoD data, such as [35], which uses word embeddings for extra supervision, and [25] which applies metric learning criteria. However,

[1]ODIN: Out-of-DIstribution detector for Neural networks [21]

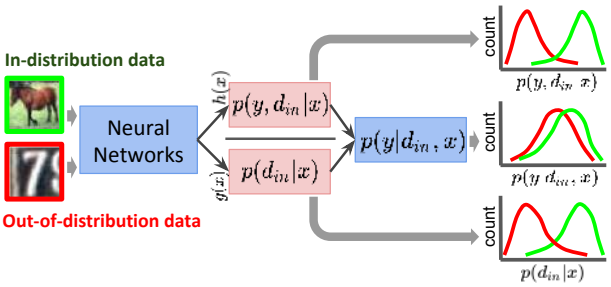


Figure 1: The concept of detecting out-of-distribution images by encouraging neural networks to output scores, $h(x)$ and $g(x)$, to behave like the decomposed factors in the conditional probability when the close-world assumption d_{in} is explicitly considered. Its elucidation is in Section 3.1. A small overlap between the green and red histograms means the x-axis a good scoring function for distinguishing OoD data from in-distribution. The extent of overlap is usually measured by AUROC, elaborated in Section 4.1.

both works report performance similar to ODIN, showing that learning without OoD data is a challenging setting.

In this work, we closely follow the setting of ODIN, proposing two corresponding strategies for the problem of learning without OoD data. First, we provide a new probabilistic perspective for decomposing confidence of predicted class probabilities. We specifically add a variable for explicitly adopting the closed world assumption, representing whether the data is in-distribution or not, and discuss its role in a decomposed conditional probability. Inspired by the probabilistic view, we use a dividend/divisor structure for a classifier, which encourages neural networks to behave similarly to the decomposed confidence effect. The concept is illustrated in Figure 1, and we note the dividend/divisor structure is closely related to temperature scaling except that the scale depends on the input instead of a tuned hyperparameter. Second, we build on the input preprocessing method from ODIN [21] 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 ODIN, achieving a performance close to, and in some cases surpassing, state-of-the-art methods [20] which use out-of-distribution data for tuning. Lastly, our systematical evaluation with DomainNet reveals the relative difficulties between two types of distribution shift: semantic shift and non-semantic shift, which are defined by whether a shift is related to the inclusion of new semantic categories.

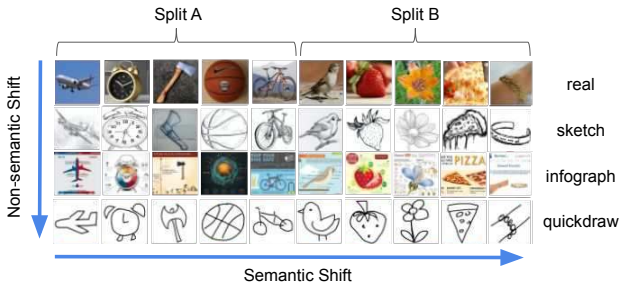


Figure 2: An example scheme of semantic shift and non-semantic shift. It is illustrated with DomainNet [31] images. The setting with two splits (A and B) will be used in our experiments, where only real-A is the in-distribution data.

In summary, the contribution of this paper is three-fold:

- A new perspective of decomposed confidence for motivating a set of classifier designs that consider the closed-world assumption.
- A modified input preprocessing method without tuning on OoD data.
- Comprehensive analysis with experiments under the setting of learning without OoD data.

2. Background

This work considers the OoD detection setting in classification problems. We begin with a dataset $D_{in} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, denoting in-distribution data $\boldsymbol{x}_i \in \mathbb{R}^k$ and categorical label $y_i \in \{\mathrm{y}\} = \{1..C\}$ for C classes. D_{in} is generated by sampling from a distribution $p_{in}(\mathbf{x}, \mathrm{y})$. 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 $p(\mathrm{y}|\boldsymbol{x})$.

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 $p_{out}(\mathbf{x}, \mathrm{y})$. Examples with this shift come from the same object class but are presented in different forms, such as cartoon or sketch images. Such shift is also a scenario be widely discussed in the problem of domain adaptation [30, 31]. In the case of semantic shift, the data is drawn from a distribution $p_{out}(\mathrm{x}, \bar{\mathrm{y}})$ with $\{\bar{\mathrm{y}}\} \cap \{\mathrm{y}\} = \emptyset$. In other words, the data is from a class not seen in the training set D_{in}. Figure 2 has an illustration.

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 $\boldsymbol{x} \sim p_{out}(\mathrm{x}, \bar{\mathrm{y}})$, or reject a low confidence prediction when $\boldsymbol{x} \sim p_{out}(\mathrm{x}, \mathrm{y})$? 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 [13] as a special case with a specific scoring function $S_{Base}(\boldsymbol{x}) = \max_{\mathrm{y}} p(\mathrm{y}|\boldsymbol{x})$, where $p(\mathrm{y}|\boldsymbol{x})$ 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 [21] and Mahalanobis [20]. These two methods will serve as strong baselines in our evaluation, especially since Mahalanobis has further been shown to have significant advantages over ODIN. 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 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 [21]. 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:

$$S^{\ell}_{Maha}(\boldsymbol{x}) = \max_i -(f^{\ell}(\boldsymbol{x}) - \mu^{\ell}_i)^T \Sigma^{-1}_{\ell} (f^{\ell}(\boldsymbol{x}) - \mu^{\ell}_i), \tag{2}$$

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

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

[36, 29, 13] observed that the softmax classifier tends to output a highly 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 [13]. 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 $p(\mathrm{y}|\boldsymbol{x})$, 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} in the classifier, rewriting it as the quotient of the joint class-domain probability and the domain probability using the rule of conditional probability:

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

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 $P(\mathrm{y}, d_{in}|\boldsymbol{x})$ 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 $p(\mathrm{y}|d_{in}, \boldsymbol{x})$ with Equation 4 gives a high probability (0.9),

demonstrating how overconfidence can result. Based on the form of Equation 4, we call $p(\mathrm{y}, d_{in}|\boldsymbol{x})$ and $p(d_{in}|\boldsymbol{x})$ the decomposed confidence scores.

One straightforward solution for the above issue is to learn a classifier to predict the joint probability $p(\mathrm{y}, d_{in}|\boldsymbol{x})$ by having both supervision on class y and domain d. Learning to predict $p(\mathrm{y}, d_{in}|\boldsymbol{x})$ is preferred over $p(d_{in}|\boldsymbol{x})$ because it can serve both purposes for predicting a class by $\arg\max_{y_{in}} p(\mathrm{y}, d_{in}|\boldsymbol{x})$ and rejecting a prediction by thresholding. This idea relates to the work of [14], 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. However, 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 $p(\mathrm{y}, d_{in}|\boldsymbol{x})$ and $p(d_{in}|\boldsymbol{x})$. 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 dividend/divisor structure in Equation 4 as the prior knowledge to design the structure of classifiers, providing classifiers a capacity to decompose the confidence of class probability.

In the dividend/divisor structure for classifiers, we define the logit $f_i(\boldsymbol{x})$ for class i, which is the division between two functions $h_i(\boldsymbol{x})$ and $g(\boldsymbol{x})$:

$$f_i(\boldsymbol{x}) = \frac{h_i(\boldsymbol{x})}{g(\boldsymbol{x})}, \tag{5}$$

The quotient $f_i(x)$ is then normalized by the exponential function (*i.e.* softmax) for outputting a class probability $p(\mathrm{y} = i|d_{in}, \boldsymbol{x})$, which is subject to cross-entropy loss.

With the exponential normalization effect of softmax, the cross-entropy loss can be minimized in two ways: increasing $h_i(\boldsymbol{x})$ or decreasing $g(\boldsymbol{x})$. In other words, when the data is not in the high-density region of in-distribution, $h_i(\boldsymbol{x})$ may tend towards smaller values. In such case, the $g(\boldsymbol{x})$ is encouraged to be small so that the resulting logits $f_i(\boldsymbol{x})$ can further minimize the cross-entropy loss. In the other case when the data is in the high density region, $h_i(\boldsymbol{x})$ generally can reach a higher value relatively easier, thus its corresponding $g(\boldsymbol{x})$ value is less encouraged to go small. The discussed interaction between $h_i(\boldsymbol{x})$ and $g(\boldsymbol{x})$ is the primary driving force to encourage $h_i(\boldsymbol{x})$ to behave similarly to $p(\mathrm{y} = i, d_{in}|\boldsymbol{x})$ and $g(\boldsymbol{x})$ to behave similarly to $p(d_{in}|\boldsymbol{x})$, in a way that the distributional overlap between the scores of OoD and in-distribution data is small, which is an intrinsic property of $p(y, d_{in}|\boldsymbol{x})$ and $p(d_{in}|\boldsymbol{x})$, illustrated in Figure 1.

3.1.1 Design Choices

Although the dividend/divisor structure provides a tendency, it does not necessarily guarantee the decomposed confidence effect to happen. The characteristic of $h_i(\boldsymbol{x})$ and $g(\boldsymbol{x})$ can largely affect how likely the decomposition could happen. Therefore we discuss a set of simple design choices to investigate whether such decomposition is generally obtainable.

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. The $\boldsymbol{w}$ and b represent the learnable weights. 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:

$$h_i^I(\boldsymbol{x}) = \boldsymbol{w}_i^T f^p(\boldsymbol{x}) + b_i; \tag{6}$$

$$h_i^E(\boldsymbol{x}) = -\|f^p(\boldsymbol{x}) - \boldsymbol{w}_i\|^2; \tag{7}$$

$$h_i^C(\boldsymbol{x}) = \frac{\boldsymbol{w}_i^T f^p(\boldsymbol{x})}{\|\boldsymbol{w}_i\|\|f^p(\boldsymbol{x})\|} \tag{8}$$

The overall neural network model f_θ therefore has two branches (h_i and g) after its penultimate layer (See Figure 1). 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 $S_{DeConf}(\boldsymbol{x}) = \max_i h_i(\boldsymbol{x})$ or $g(\boldsymbol{x})$.

Note that when $h_i(\boldsymbol{x}) = h_i^I(\boldsymbol{x})$ and $g(\boldsymbol{x}) = 1$, this method reduces to the baseline [13]. We call the three variants of our method DeConf-I, DeConf-E, and DeConf-C. For simplicity, the above names represent using $h_i(\boldsymbol{x})$ for the scores. The use of $g(\boldsymbol{x})$ will be indicated specifically.

3.1.2 Temperature Scaling

The $g(x)$ in Equation 5 can be immediately viewed as a learned temperature scaling function discussed in [28] and a concurrent report [37]. However, our experiment results strongly suggest that $g(x)$ is more than a scale. The $g(x)$ achieves an OoD detection performance significantly better than baselines in many experiments, indicating its potential in estimating the $p(d_{in}|\boldsymbol{x})$. More importantly, the temperature scaling is generally used as a numerical trick for learning a better embedding [40], softening the prediction [15], or calibrating the confidence [9]. Our work provides a probabilistic view for its effect, indicating such temperature might relate to how strong a classifier assumes a closed world as a prior.

3.2. A Modified Input Preprocessing Strategy

This section describes a modified version of the input preprocessing method proposed in ODIN [21]. The primary

purpose of the 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:

$$\hat{\boldsymbol{x}} = \boldsymbol{x} - \epsilon \text{sign}(-\nabla_{\boldsymbol{x}} S(\boldsymbol{x})) \tag{9}$$

In the original method [21] 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}:

$$\epsilon^* = \arg\max_{\epsilon} \sum_{x \in D_{in}^{val}} S(\hat{\boldsymbol{x}}) \tag{10}$$

Our searching criteria is still based on the same observation made by [21]. 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. 10 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 6 values: $[0.0025, 0.005, 0.01, 0.02, 0.04, 0.08]$. 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 [21, 20, 14, 34, 38, 19], except that we do not use OoD data for tuning the hyperparameters in the scoring function $S(\boldsymbol{x})$.

In-distribution Datasets: We use SVHN [27] and CIFAR-10/100 images with size 32x32 [17] for the classification task. Detecting OoD with CIFAR-100 classifier is generally harder than CIFAR-10 and SVHN, since a larger amount of classes usually involves a wider range of variance, and thus it 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 [21], which are TinyImageNet(crop), TinyImageNet(resize), LSUN(crop), LSUN(resize), iSUN, Uniform random images, and Gaussian random images. We further add SVHN, 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 that uses a generative model for OoD detection [32, 26, 4]. 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 [16], ResNet [11], and WideResNet [39] for the classifier backbone. DenseNet has 100 layers with a growth rate of 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 [10]. In the robustness analysis, the model may be indicated to have an extra regularization. In such case, we additional apply a dropout rate of 0.7 at the inputs for the dividend/divisor structure.

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 (FPR) of OoD data by varying a threshold. Thus it can be regarded as an averaged score. The second one is true negative rate

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 do not need any OoD data for tuning (see Section 2.1). The base network used in the table is DenseNet trained with CIFAR-10/100 (in-distribution data, or ID). All values are percentages averaged over three runs, and the best results are indicated in bold. Note that we only show the most common settings used in literature. The DeConf-C is selected since it shows the best robustness in our analysis, but it is not necessary to perform the best among all DeConf variants. Please see Figure 3 and Figure 4 for the summary. A more comprehensive version of the table is available in Supplementary.

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.**

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.

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 outperforms 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.

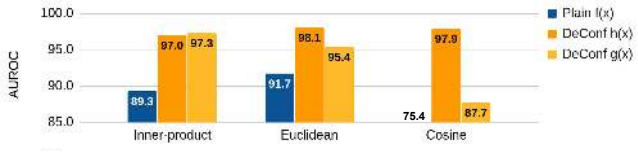
In Table 2 we further compare our results with the original ODIN [21] and Mahalanobis [20] methods which are tuned on each OoD dataset. We refer to the results of both original methods reported by [20] since it uses the same backbone network, OoD datasets, and metrics to evaluate OoD detection performance. 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 advantage from using OoD data. However, our DeConf-C* still outperforms the two original methods in many of the cases. The cross-setting comparison further supports the effectiveness of the proposed strategies.

Table 2: OoD detection with OoD data versus without OoD data with CIFAR-10/100 for the in-distribution (ID) data. The values of ODIN^{orig} and Maha^{orig} (abbreviation of Mahalanobis) are copied from the Mahalanobis paper [20] 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. All methods in this table use the same DenseNet for the backbone. Note that the performance with different network backbone may have a mild difference. For example, Maha^{orig} performs slightly better than DeConf-C* with ResNet-34.

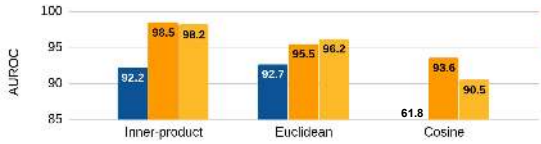
ID	OoD	AUROC	TNR@TPR95
		ODIN^{orig} / Maha^{orig}/ ODIN* / Maha* / DeConf-C*	
C-100	Imagenet(r)	85.2 / 97.4 / 91.1 / 96.4 / **98.6**	42.6 / 86.6 / 59.4 / 82.0 / **93.3**
	LSUN(r)	85.5 / 98.0 / 93.0 / 96.6 / **98.7**	41.2 / 91.4 / 64.0 / 82.6 / **93.8**
	SVHN	93.8 / **97.2** / 85.6 / 89.9 / 95.9	70.6 / **82.5** / 35.3 / 43.3 / 77.0
C-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**

Ablation Study: We study the effect of applying DeConf and our modified input preprocessing (IPP) strategy separately. In Figure 3, it shows that both $h_i(\boldsymbol{x})$ and $g(\boldsymbol{x})$ from all three variants (I, E, C) of the DeConf strategy help OoD detection performance with CIFAR-10 and SVHN classifiers, showing that the concept of DeConf is generally effective. However, the failure of DeConf-I and $g(x)$ with the CIFAR-100 classifier in Figure 4a may indicate these functions have different robustness and scalability, 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 in the case of CIFAR-100 (See Table 3). 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 of assigning a lower score for the data away from the high-density region of in-distribution data. We see this effect is reduced and has only a 1% accuracy drop when the extra regularization (dropout rate 0.7) is applied.

In Figure 5, 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.

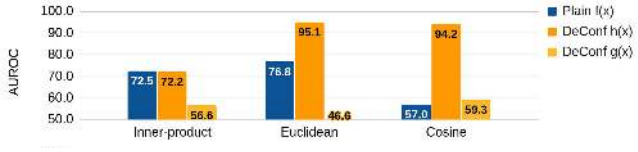


(a) CIFAR-10 classifier

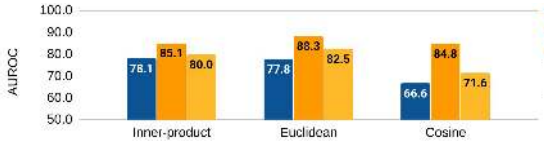


(b) SVHN classifier

Figure 3: An ablation study with three variants in our DeConf method (Section 3.1). *Plain* means $g(\boldsymbol{x}) = 1$ so that the dividend/divisor structure is turned off. Each bar in the figure is averaged with 24 experiments (8 OoD datasets listed in Table 1 with 3 repeats. Note that we use CIFAR-10 as OoD to replace the SVHN in the case of SVHN classifier). The backbone network is Resnet-34. The *plain* setting with inner-product is equivalent to a vanilla Resnet for classification. Overall, both scores from $h(x)$ and $g(x)$ are significant higher than random (AUROC=0.5) and corresponding *plain* baselines. Supplementary has breakdown results.



(a) CIFAR-100 classifier



(b) CIFAR-100 classifier with extra regularization (dropout 0.7)

Figure 4: An ablation study similar to Figure 3. This figure shows the performance of DeConf-I and all $g(\boldsymbol{x})$ are improved by adding extra regularization.

Robustness Study: This study investigates when the OoD detection method will or will not work. In Figure 6, 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 methods generally reach a higher performance in the high data regime. In Figure 6, 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

Figure 5: 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.

two DeConf variants. Lastly, Figure 7 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*.

Enhancing the Robustness: The overfitting issue may be the cause of low OoD detection performance for some of the DeConf variants and $g(x)$. In Figure 4b, the OoD detection performance is significantly increased with DeConf-I and all $g(x)$ when extra regularization (dropout rate 0.7) is applied. Figure 8 provides further analysis for DeConf-I and its $g(x)$ by varying the number of samples and classes in the training data. The performance with extra regularization is significantly better than the cases without it. Besides, the performance is also very similar between regularized $h_i(\boldsymbol{x})$ and $g(\boldsymbol{x})$, indicating that overfitting is an important issue. Lastly, we note that the DeConf-E and DeConf-C have a reduced performance with extra regularization in Figure 4b. This outcome might be because the dropout generally harms the distance calculation between centroids and data since part of the feature is masked. The results indicate that the design of (I, E, C) might not be optimal for the problem, leaving room for future work to find a robust pair of $h_i(\boldsymbol{x})$ and $g(\boldsymbol{x})$ for the OoD detection problem.

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 therefore use a larger scale image dataset, DomainNet [31], to repeat an evaluation similar to Table 1. DomainNet has high-resolution (180x180 to 640x880) images in 345 classes from six different domains. There are four domains in the dataset with class labels available when the experiments were conducted. They

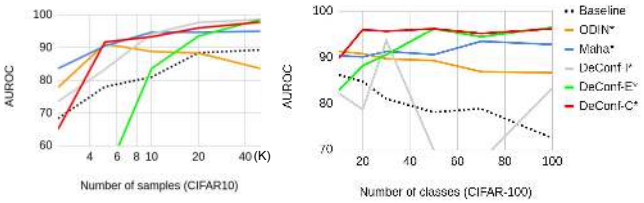


Figure 6: Robustness analysis of 6 OoD detection methods. The left figure has classifiers trained on a varied number of samples in CIFAR-10. The right figure has classifiers trained on a varied number of classes in CIFAR-100. Each point in the line is an average of the results on 8 OoD datasets. The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

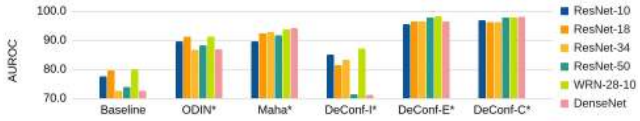


Figure 7: Robustness analysis using different neural network backbones. The in-distribution data is CIFAR-100. Each bar is averaged with the results on 8 OoD datasets.

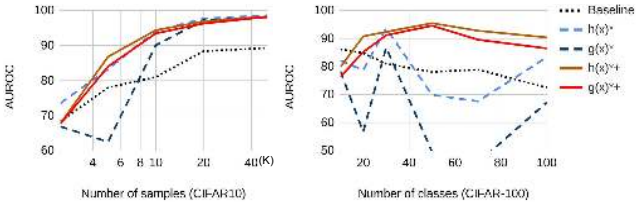


Figure 8: Robustness analysis for $h(x)$ and $g(x)$ from DeConf-I. The + sign represents the model trained with extra regularization (dropout rate 0.7).

are real, sketch, infograph, and quickdraw, resulting in different types of distribution shifts.

To create subsets with semantic shift, we separate the classes into 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. Figure 2 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 with initial learning rate of 0.01, and the images are center-cropped and resized to 224x224 in this experiment.

The results in Table 4 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

Table 3: The in-domain classification accuracy. The "+" means that the classifier is trained with extra regularization (dropout rate 0.7). The expanded version of this table is available in Supplementary.

Classifier	Model	Baseline	DeConf-I	DeConf-E	DeConf-C
CIFAR-10	DenseNet	95.2±0.1	94.9±0.1	95.0±0.1	95.0±0.1
CIFAR-100	DenseNet	77.0±0.2	75.8±0.4	76.4±0.1	75.9±0.1
SVHN	ResNet34	96.9±0.1	96.8±0.1	96.5±0.1	96.7±0.1
CIFAR-10	ResNet34	95.2±0.1	95.0±0.1	94.9±0.1	95.1±0.1
CIFAR-100	ResNet34	78.5±0.2	76.0±0.1	76.2±0.1	75.8±0.2
CIFAR-100$^+$	ResNet34	78.2±0.1	77.4±0.3	77.2±0.3	77.2±0.1
DomainNet (Real-A)	ResNet34	73.6±0.1	73.0±0.1	73.4±1.5	72.2±0.5

Table 4: 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 (S) $>$ Non-semantic shift (NS) $>$ Semantic $+$ Non-semantic shift.

OOD	Shift		AUROC	TNR@TPR95
	S	NS	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.**

shift. The semantic shift turns out to be the hardest one to detect. The second observation is the failure of Mahalanobis*. In most 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

In this paper, we propose two strategies, the decomposed confidence and the modified input preprocessing. These two simple modifications to ODIN lead to a significant change in the paradigm, which does not need OoD data for tuning the method. Our comprehensive analysis shows that our strategies are effective and even better in several 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

[1] J Andrews, Thomas Tanay, Edward J Morton, and Lewis D Griffin. Transfer representation-learning for anomaly detection. In *JMLR*, 2016.

[2] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1893–1902, 2015.

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

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

[5] Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Learning with rejection. In *International Conference on Algorithmic Learning Theory*, pages 67–82. Springer, 2016.

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

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

[8] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in neural information processing systems*, pages 4878–4887, 2017.

[9] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[10] 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*, pages 1026–1034, 2015.

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

[12] Martin E Hellman. The nearest neighbor classification rule with a reject option. *IEEE Transactions on Systems Science and Cybernetics*, 6(3):179–185, 1970.

[13] 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.

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

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[16] 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*, pages 4700–4708, 2017.

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

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

[19] 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)*, 2018.

[20] 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*, pages 7167–7177, 2018.

[21] 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.

[22] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7047–7058. Curran Associates, Inc., 2018.

[23] Andrey Malinin and Mark Gales. Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness. In *Advances in Neural Information Processing Systems*, pages 14520–14531, 2019.

[24] Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. Ensemble distribution distillation. In *International Conference on Learning Representations*, 2019.

[25] Marc Masana, Idoia Ruiz, Joan Serrat, Joost van de Weijer, and Antonio M Lopez. Metric learning for novelty and anomaly detection. *arXiv preprint arXiv:1808.05492*, 2018.

[26] 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.

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

[28] Lukas Neumann, Andrew Zisserman, and Andrea Vedaldi. Relaxed softmax: Efficient confidence auto-calibration for safe pedestrian detection. *NIPS MLITS Workshop*, 2018.

[29] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[30] Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):53–69, 2015.

[31] 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.

[32] 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.

[33] 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.

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

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

[36] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[37] Engkarat Techapanurak and Takayuki Okatani. Hyperparameter-free out-of-distribution detection using softmax of scaled cosine similarity. *arXiv preprint:1905.10628*, 2019.

[38] 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)*, pages 550–564, 2018.

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

[40] Xiao Zhang, Rui Zhao, Yu Qiao, Xiaogang Wang, and Hongsheng Li. Adacos: Adaptively scaling cosine logits for effectively learning deep face representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10823–10832, 2019.