TRAINING CONFIDENT NEURAL CLASSIFIERS FOR DETECTING OUT-OF-DISTRIBUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

The problem of detecting whether a test sample is from in-distribution (i.e., training distribution by a classifier) or out-of-distribution sufficiently different from it arises in many real-world machine learning applications. However, the state-of-art deep neural networks are known to be highly overconfident in their predictions, i.e., do not distinguish in- and out-of-distributions. Recently, to handle this issue, several threshold-based detectors have been proposed given pre-trained neural classifiers. However, the performance of prior works highly depends on how to train the classifiers since they only focus on improving inference procedures. In this paper, we develop a novel training method for classifiers so that such inference algorithms can work better. In particular, we suggest two additional terms added to the original loss (e.g., cross entropy). The first one forces samples from out-of-distribution less confident by the classifier and the second one is for (implicitly) generating most effective training samples for the first one. In essence, our method jointly trains both classification and generative neural networks for out-of-distribution. We demonstrate its effectiveness using deep convolutional neural networks on various popular image datasets.

1 INTRODUCTION

Deep neural networks (DNNs) have demonstrated state-of-the-art performance on many classification tasks, e.g., speech recognition (Hannun et al., 2014), image classification (Girshick, 2015), video prediction (Villegas et al., 2017) and medical diagnosis (Caruana et al., 2015). Even though DNNs achieve high accuracy, it has been addressed (Lakshminarayanan et al., 2017; Guo et al., 2017) that they are typically overconfident in their predictions. For example, DNNs trained to classify MNIST images often produce high confident probability 91% even for random noise (see the work of (Hendrycks & Gimpel, 2016)). Since evaluating the quality of their predictive uncertainty is hard, deploying them in real-world systems raises serious concerns in AI Safety (Amodei et al., 2016), e.g., one can easily break a secure authentication system that can be unlocked by detecting the gaze and iris of eyes using DNNs (Shrivastava et al., 2017).

The overconfidence issue of DNNs is highly related to the problem of detecting out-of-distribution: detect whether a test sample is from in-distribution (i.e., training distribution by a classifier) or out-of-distribution sufficiently different from it. Formally, it can be formulated as a binary classification problem. Let an input $\mathbf{x} \in \mathcal{X}$ and a label $y \in \mathcal{Y} = \{1, \dots, K\}$ be random variables that follow a joint data distribution $P_{\texttt{in}}(\mathbf{x}, y) = P_{\texttt{in}}(y|\mathbf{x}) P_{\texttt{in}}(\mathbf{x})$. We assume that a classifier $P_\theta(y|\mathbf{x})$ is trained on a dataset drawn from $P_{\texttt{in}}(\mathbf{x}, y)$, where θ denotes the model parameter. We let $P_{\texttt{out}}(\mathbf{x})$ denote an out-of-distribution which is 'far away' from in-distribution $P_{\texttt{in}}(\mathbf{x})$. Our problem of interest is determining if input $\mathbf{x}$ is from $P_{\texttt{in}}$ or $P_{\texttt{out}}$, possibly utilizing a well calibrated classifier $P_\theta(y|\mathbf{x})$. In other words, we aim to build a detector, $g(\mathbf{x}) : \mathcal{X} \to \{0, 1\}$, which assigns label 1 if data is from in-distribution, and label 0 otherwise.

There have been recent efforts toward developing efficient detection methods where they mostly have studied simple threshold-based detectors utilizing a pre-trained classifier (Hendrycks & Gimpel, 2016; Liang et al., 2017). For each input $\mathbf{x}$, it measures some confidence score $q(\mathbf{x})$ based on a pre-trained classifier, and compares the score to some threshold $\delta > 0$. Then, the detector assigns label 1 if the confidence score $q(\mathbf{x})$ is above δ, and label 0, otherwise. Specifically, (Hendrycks & Gimpel, 2016) defined the confidence score as a maximum value of the predictive distribution,

and (Liang et al., 2017) further improved the performance by using temperature scaling (Guo et al., 2017) and adding small controlled perturbations to the input data. Although such inference methods are computationally simple, their performances highly depend on the pre-trained classifier. Namely, they fail to work if the classifier does not separate the maximum value of predictive distribution well enough with respect to $P_{\texttt{in}}$ and $P_{\texttt{out}}$. Ideally, a classifier should be trained to separate all class-dependent in-distributions as well as out-of-distribution in the output space. As another line of research, Bayesian probabilistic models (Li & Gal, 2017; Louizos & Welling, 2017) and ensembles of classifiers (Lakshminarayanan et al., 2017) were also investigated. However, training or inferring those models are computationally more expensive. This motivates our approach of developing a new training method for the more plausible simple classifiers. Our direction is orthogonal to the Bayesian and ensemble approaches, where one can also combine them for even better performance.

Contribution. In this paper, we develop such a training method for detecting out-of-distribution $P_{\texttt{out}}$ better without losing its original classification accuracy. First, we consider a new loss function, called *confidence loss*. Our key idea on the proposed loss is to additionally minimize the Kullback-Leibler (KL) divergence from the predictive distribution on out-of-distribution samples to the uniform one in order to give less confident predictions on them. Then, in- and out-of-distributions are expected to be more separable. However, optimizing the confidence loss requires training samples from out-of-distribution, which are often hard to sample: a priori knowledge on out-of-distribution is not available or its underlying space is too huge to cover. To handle the issue, we consider a new generative adversarial network (GAN) (Goodfellow et al., 2014) for generating most effective samples from $P_{\texttt{out}}$. Unlike the original GAN, the proposed GAN generates 'boundary' samples in the low-density area of $P_{\texttt{in}}$. Finally, we design a joint training scheme minimizing the classifier's loss and new GAN loss alternatively, i.e., the confident classifier improves the GAN, and vice versa, as training proceeds. Here, we emphasize that the proposed GAN does not need to generate explicit samples under our scheme, and instead it implicitly encourages training a more confident classifier.

We demonstrate the effectiveness of the proposed method using deep convolutional neural networks such as AlexNet (Krizhevsky, 2014) and VGGNet (Szegedy et al., 2015) for image classification tasks on CIFAR (Krizhevsky & Hinton, 2009), SVHN (Netzer et al., 2011), ImageNet (Deng et al., 2009), and LSUN (Yu et al., 2015) datasets. The classifier trained by our proposed method drastically improves the detection performance of all threshold-based detectors (Hendrycks & Gimpel, 2016; Liang et al., 2017) in all experiments. In particular, VGGNet with 13 layers trained by our method improves the true negative rate (TNR), i.e., the fraction of detected (LSUN) out-of-distribution samples, compared to the baseline: $14.0\% \rightarrow 37.8\%$ and $46.3\% \rightarrow 99.9\%$ on CIFAR-10 and SVHN, respectively, when 95% of in-distribution samples are correctly detected. We also provide visual understandings on the proposed method using the image datasets. We believe that our method can be a strong guideline when other researchers will pursue these tasks in the future.

2 TRAINING CONFIDENT NEURAL CLASSIFIERS

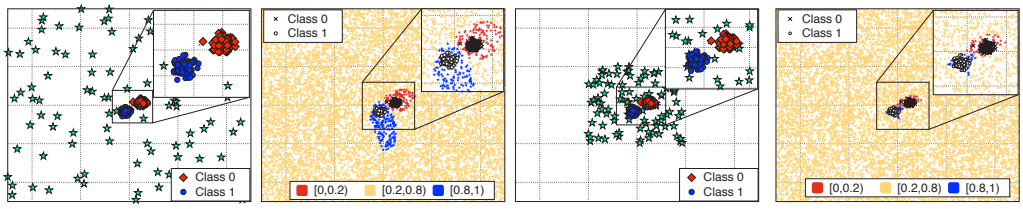
2.1 CONFIDENT CLASSIFIER FOR OUT-OF-DISTRIBUTION

We propose a new loss function to train a classifier which can map the samples from in- and out-of-distribution into the output space separately. Without loss of generality, suppose that the cross entropy loss is used for training. Then, we define the following, termed confidence loss:

$$\min_{\theta} \; \mathbb{E}_{P_{\texttt{in}}(\widehat{\mathbf{x}},\widehat{y})}\big[-\log P_\theta\left(y=\widehat{y}|\widehat{\mathbf{x}}\right)\big] + \mathbb{E}_{P_{\texttt{out}}(\mathbf{x})}\big[KL\left(\mathcal{U}\left(y\right) \parallel P_\theta\left(y|\mathbf{x}\right)\right)\big], \tag{1}$$

where KL denotes the Kullback-Leibler (KL) divergence and $\mathcal{U}\left(y\right)$ is the uniform distribution. It is highly intuitive as the new loss forces the predictive distribution on out-of-distribution samples to be closer to the uniform one, i.e., zero confidence, while that for samples from in-distribution still follows the label-dependent probability. In other words, the proposed loss is designed for assigning higher maximum prediction values to in-distribution samples than out-of-distribution ones. Here, a caveat is that adding the KL divergence term might degrade the classification performance. However, we found that it is not the case due to the high expressive power of deep neural networks, while in- and out-of-distributions become more separable with respect to the maximum prediction value by optimizing the confidence loss (see Section 3.1 for supporting experimental results).

We remark that minimizing a similar KL loss was studied recently for different purposes (Lee et al., 2017; Pereyra et al., 2017). Training samples for minimizing the KL loss is explicitly given in



(a) (b) (c) (d)

Figure 1: Illustrating the behavior of classifier under different out-of-distribution training datasets. We generate the out-of-distribution samples from (a) 2D box $[-50, 50]^2$, and show (b) the corresponding decision boundary of classifier. We also generate the out-of-distribution samples from (c) 2D box $[-20, 20]^2$, and show (d) the corresponding decision boundary of classifier.

their settings while we might not. Ideally, one has to sample all (almost infinite) types of out-of-distribution to minimize the KL term in (1), or require some prior information on testing out-of-distribution for efficient sampling. However, this is often infeasible and fragile. To address the issue, we suggest to sample out-of-distribution close to in-distribution, which could be more effective in improving the detection performance, without any assumption on testing out-of-distribution.

In order to explain our intuition in details, we consider a binary classification task on a simple example, where each class data is drawn from a Gaussian distribution and entire data space is bounded by 2D box $[-50, 50]^2$ for visualization. We apply the confidence loss to simple fully-connected neural networks (2 hidden layers and 500 hidden units for each layer) using different types of out-of-distribution training samples. First, as shown in Figure 1(a), we construct an out-of-distribution training dataset of 100 (green) points using rejection sampling on the entire data space $[-50, 50]^2$. Figure 1(b) shows the decision boundary of classifier optimizing the confidence loss on the corresponding dataset. One can observe that a classifier still shows overconfident predictions (red and blue regions) near the labeled in-distribution region. On the other hand, if we construct a training out-of-distribution dataset of 100 points from $[-20, 20]^2$, i.e., closer to target, in-distribution space (see Figure 1(c)), a classifier produces confident predictions only on the labeled region and zero confidence on the remaining in the entire data space $[-50, 50]^2$ as shown in Figure 1(d). This implies that training out-of-distribution samples nearby the in-distribution region could be more effective in improving the detection performance. Our underlying intuition is that the effect of boundary of in-distribution region might propagate to the entire out-of-distribution space. Our experimental results in Section 3.1 also support this: realistic images are more useful as training out-of-distribution than synthetic datasets (e.g., Gaussian noise) for improving the detection performance when we consider an image classification task. This motivates us to develop a new generative adversarial network (GAN) for generating such effective out-of-distribution samples.

2.2 Adversarial generator for out-of-distribution

The GAN framework proposed by (Goodfellow et al., 2014) consists of two main components: discriminator D and generator G. The generator maps a latent variable $\mathbf{z}$ from a prior distribution $P_{\texttt{pri}}(\mathbf{z})$ to generated outputs $G(\mathbf{z})$, and discriminator $D : \mathcal{X} \to [0, 1]$ represents a probability that sample $\mathbf{x}$ is from a target distribution. Suppose that we want to recover the in-distribution $P_{\texttt{in}}(x)$ using the generator G. Then, one can optimize the following min-max objective for forcing $P_G \approx P_{\texttt{in}}$:

$$\min_G \max_D \; \mathbb{E}_{P_{\texttt{in}}(\mathbf{x})}\big[\log D(\mathbf{x})\big] + \mathbb{E}_{P_{\texttt{pri}}(\mathbf{z})}\big[\log\big(1 - D(G(\mathbf{z}))\big)\big]. \tag{2}$$

However, unlike the original GAN, we want to make the generator recover an effective out-of-distribution $P_{\texttt{out}}$ instead of $P_{\texttt{in}}$. Inspired by our intuition in Section 2.1. we assume that the density of out-of-distribution is (i) inversely proportional to that of in-distribution in its high-density area, i.e., if $P_{\texttt{in}}(\mathbf{x}) > \varepsilon$ for some ε, and (ii) very marginal, say $\eta \approx 0$, otherwise. Then, we first derive

$$\begin{aligned} KL(P_G \parallel P_{\texttt{out}}) &= -\mathcal{H}(P_G(\mathbf{x})) - \mathbb{E}_{P_G(\mathbf{x})}\big[\log P_{\texttt{out}}(\mathbf{x})\big] \\ &= \underbrace{-\mathcal{H}(P_G(\mathbf{x})) - \mathbb{E}_{P_G(\mathbf{x})}\big[\mathbb{I}(P_{\texttt{in}}(\mathbf{x}) > \varepsilon)\log P_{\texttt{out}}(\mathbf{x})\big]}_{\text{(a)}} \underbrace{-\mathbb{E}_{P_G(\mathbf{x})}\big[\mathbb{I}(P_{\texttt{in}}(\mathbf{x}) \le \varepsilon)\log P_{\texttt{out}}(\mathbf{x})\big]}_{\text{(b)}}, \end{aligned}$$

where $\mathcal{H}(\cdot)$ denotes the entropy and $\mathbb{I}$ is the indicator function. Using the assumption (i), minimizing the part (a) with respect to the generator G becomes

$$\begin{aligned}
&\min_G \quad -\mathcal{H}\left(P_G\left(\mathbf{x}\right)\right) - \mathbb{E}_{P_G(\mathbf{x})}\left[\mathbb{I}\left(P_{\texttt{in}}\left(\mathbf{x}\right) > \varepsilon\right)\log P_{\texttt{out}}\left(\mathbf{x}\right)\right] \\
&= \min_G \quad -\mathcal{H}\left(P_G\left(\mathbf{x}\right)\right) + \mathbb{E}_{P_G(\mathbf{x})}\left[\mathbb{I}\left(P_{\texttt{in}}\left(\mathbf{x}\right) > \varepsilon\right)\log P_{\texttt{in}}\left(\mathbf{x}\right)\right] \\
&\approx \min_G \quad -\mathcal{H}\left(P_G\left(\mathbf{x}\right)\right) + \mathbb{E}_{P_G(\mathbf{x})}\left[\mathbb{I}\left(KL\left(\mathcal{U}\left(y\right) \parallel P_\theta\left(y|\mathbf{x}\right)\right) > \varepsilon\right) KL\left(\mathcal{U}\left(y\right) \parallel P_\theta\left(y|\mathbf{x}\right)\right)\right],
\end{aligned}$$

where θ is the model parameter of a classifier trained on in-distribution and the approximation is inspired by our design of the confidence loss (1), i.e., $P_{\texttt{in}}(\mathbf{x}) \approx \exp\left(KL\left(\mathcal{U}\left(y\right) \parallel P_\theta\left(y|\mathbf{x}\right)\right)\right)$. We remark that this approximation is also closely related to the inception score (Salimans et al., 2016) which is popularly used as a quantitative measure of visual fidelity of the samples. Next, using the assumption (ii), minimizing the part (b) with respect to the generator G becomes

$$\begin{aligned}
&\min_G \quad -\mathbb{E}_{P_G(\mathbf{x})}\left[\mathbb{I}\left(P_{\texttt{in}}\left(\mathbf{x}\right) \leq \varepsilon\right)\log P_{\texttt{out}}\left(\mathbf{x}\right)\right] \\
&= \min_G \quad \mathbb{E}_{P_G(\mathbf{x})}\left[\mathbb{I}\left(P_{\texttt{in}}\left(\mathbf{x}\right) \leq \varepsilon\right)\right] \cdot \log\frac{1}{\eta} \\
&\approx \quad \min_G \max_D \quad \mathbb{E}_{P_{\texttt{in}}(\mathbf{x})}\left[\log D\left(\mathbf{x}\right)\right] + \mathbb{E}_{P_G(\mathbf{x})}\left[\log\left(1 - D\left(\mathbf{x}\right)\right)\right],
\end{aligned}$$

since minimizing the probability that the generator produces samples in the low-density area can be approximated by forcing $P_G \approx P_{\texttt{in}}$, i.e., the original GAN loss. Finally, by combining all above approximations, we propose the following new GAN loss for minimizing $KL\left(P_G \parallel P_{\texttt{out}}\right)$:

$$\begin{aligned}
\min_G \max_D \quad & -\mathcal{H}\left(P_G\left(\mathbf{x}\right)\right) + \mathbb{E}_{P_G(\mathbf{x})}\left[\mathbb{I}\left(KL\left(\mathcal{U}\left(y\right) \parallel P_\theta\left(y|\mathbf{x}\right)\right) > \varepsilon\right) KL\left(\mathcal{U}\left(y\right) \parallel P_\theta\left(y|\mathbf{x}\right)\right)\right] \\
& + \mathbb{E}_{P_{\texttt{in}}(\mathbf{x})}\left[\log D\left(\mathbf{x}\right)\right] + \mathbb{E}_{P_G(\mathbf{x})}\left[\log\left(1 - D\left(\mathbf{x}\right)\right)\right], \qquad (3)
\end{aligned}$$

which encourages the generator to produce the samples which are on the low-density boundary of the in-distribution space. We also provide its experimental evidences in Section 3.2.

However, maximizing the entropy $\mathcal{H}\left(P_G\left(\mathbf{x}\right)\right)$ in the objective (3) is technically challenging since a GAN does not model the generator distribution explicitly. To handle the issue, we leverage the pull-away term (PT) (Zhao et al., 2017):

$$-\mathcal{H}\left(P_G\left(\mathbf{x}\right)\right) \backsimeq \mathcal{PT}\left(P_G\left(\mathbf{x}\right)\right) = \frac{1}{M(M-1)}\sum_{i=1}^{M}\sum_{j\neq i}\left(\frac{G\left(\mathbf{z}_i\right)^\top G\left(\mathbf{z}_j\right)}{\|G\left(\mathbf{z}_i\right)\|\|G\left(\mathbf{z}_j\right)\|}\right)^2,$$

where $\mathbf{z}_i, \mathbf{z}_j \sim P_{\texttt{pri}}\left(\mathbf{z}\right)$ and M is the number of samples. One can note that it corresponds to the squared cosine similarity of generated samples. By minimizing the squared cosine similarity, one can increase the entropy. We also remark that (Dai et al., 2017) consider a similar GAN generating samples from out-of-distribution for the purpose of semi-supervised learning. The authors assume the existence of a pre-trained density estimation model such as PixelCNN++ (Salimans et al., 2017) for in-distribution, but such a model might not exist and be expensive to train in general. Instead, we use much simpler confident classifiers for approximating the density. Hence, under our fully-supervised setting, our GAN is much easier to train and more suitable.

2.3 Joint training method of confident classifier and adversarial generator

In the previous section, we suggest training the proposed GAN using a pre-trained confident classifier. We remind that the converse is also possible, i.e., the motivation of having such a GAN is for training the classifier. Under this relation between two models, we propose a joint training scheme where the confident classifier improves the proposed GAN, and vice versa, as training proceeds. Specifically, we suggest the following joint objective function:

$$\begin{aligned}
\min_G \max_D \min_\theta \quad & \underbrace{\mathbb{E}_{P_{\texttt{in}}(\widehat{\mathbf{x}},\widehat{y})}\left[-\log P_\theta\left(y = \widehat{y}|\widehat{\mathbf{x}}\right)\right]}_{\text{(c)}} \\
& \underbrace{+\mathbb{E}_{P_G(\mathbf{x})}\left[\mathbb{I}\left(KL\left(\mathcal{U}\left(y\right) \parallel P_\theta\left(y|\mathbf{x}\right)\right) > \varepsilon\right) KL\left(\mathcal{U}\left(y\right) \parallel P_\theta\left(y|\mathbf{x}\right)\right)\right]}_{\text{(d)}} \\
& \underbrace{+\mathbb{E}_{P_{\texttt{in}}(\widehat{\mathbf{x}})}\left[\log D\left(\widehat{\mathbf{x}}\right)\right] + \mathbb{E}_{P_G(\mathbf{x})}\left[\log\left(1 - D\left(\mathbf{x}\right)\right)\right] + \mathcal{PT}\left(P_G\left(\mathbf{x}\right)\right).}_{\text{(e)}} \qquad (4)
\end{aligned}$$

The classifier's confidence loss corresponds to (c) + (d), and the proposed GAN loss corresponds to (d) + (e), i.e., they share the KL term (d) under joint training. To optimize the above objective efficiently, we propose an alternating algorithm, which optimizes model parameters $\{\theta\}$ of classifier and GAN models $\{G, D\}$ alternatively as shown in Algorithm 1. Since the algorithm monotonically decreases the objective function, it is guaranteed to converge.

Algorithm 1 Alternating minimization for detecting and generating out-of-distribution.

repeat

$/*$ Update proposed GAN $*/$

Sample $\{\mathbf{z}_1, \dots, \mathbf{z}_M\}$ and $\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ from prior $P_{\texttt{pri}}(\mathbf{z})$ and and in-distribution $P_{\texttt{in}}(\mathbf{x})$, respectively, and update the discriminator D by ascending its stochastic gradient of

$$\frac{1}{M}\sum_{i=1}^{M}\Big[\log D(\mathbf{x}_i) + \log\left(1 - D\left(G\left(\mathbf{z}_i\right)\right)\right)\Big].$$

Sample $\{\mathbf{z}_1, \dots, \mathbf{z}_M\}$ from prior $P_{\texttt{pri}}(\mathbf{z})$, and update the generator G by descending its stochastic gradient of

$$\frac{1}{M}\sum_{i=1}^{M}\Big[\log\left(1 - D\left(G\left(\mathbf{z}_i\right)\right)\right)\Big] + \frac{1}{M(M-1)}\sum_{i=1}^{M}\sum_{j\neq i}\left(\frac{G\left(\mathbf{z}_i\right)^{\top}G\left(\mathbf{z}_j\right)}{\|G\left(\mathbf{z}_i\right)\|\|G\left(\mathbf{z}_j\right)\|}\right)^2$$
$$+\frac{1}{M}\sum_{i=1}^{M}\Big[\mathbb{I}\left(KL\left(\mathcal{U}(y) \parallel P_\theta\left(y|G\left(\mathbf{z}_i\right)\right)\right) > \varepsilon\right) KL\left(\mathcal{U}(y) \parallel P_\theta\left(y|G\left(\mathbf{z}_i\right)\right)\right)\Big].$$

$/*$ Update confident classifier $*/$

Sample $\{\mathbf{z}_1, \dots, \mathbf{z}_M\}$ and $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_M, y_M)\}$ from prior $P_{\texttt{pri}}(\mathbf{z})$ and in-distribution $P_{\texttt{in}}(\mathbf{x}, y)$, respectively, and update the classifier θ by descending its stochastic gradient of

$$\frac{1}{M}\sum_{i=1}^{M}\Big[-\log P_\theta\left(y = y_i|\mathbf{x}_i\right) + \mathbb{I}\left(KL\left(\mathcal{U}(y) \parallel P_\theta\left(y|G\left(\mathbf{z}_i\right)\right)\right) > \varepsilon\right) KL\left(\mathcal{U}(y) \parallel P_\theta\left(y|G\left(\mathbf{z}_i\right)\right)\right)\Big].$$

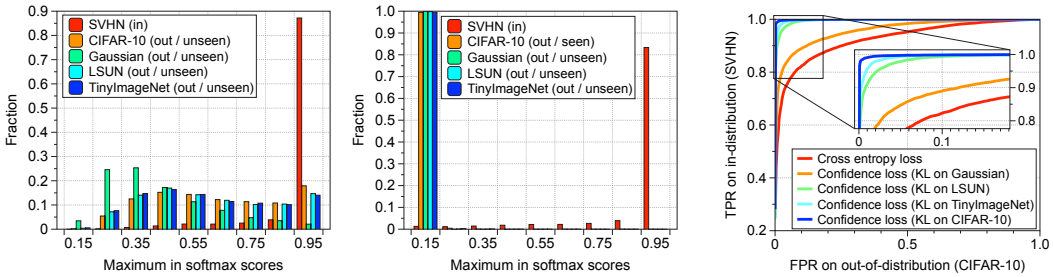
until *convergence*

3 Experimental Results

We demonstrate the effectiveness of our proposed method using various datasets: CIFAR (Krizhevsky & Hinton, 2009), SVHN (Netzer et al., 2011), ImageNet (Deng et al., 2009), LSUN (Yu et al., 2015) and synthetic (Gaussian) noise distribution. We train convolutional neural networks (CNNs) including VGGNet (Szegedy et al., 2015) and AlexNet (Krizhevsky, 2014) for classifying CIFAR-10 and SVHN datasets. The corresponding test dataset is used as the in-distribution (positive) samples to measure the performance. We use realistic images and synthetic noises as the out-of-distribution (negative) samples. For evaluation, we measure the following metrics using threshold-based detectors (Hendrycks & Gimpel, 2016; Liang et al., 2017): the true negative rate (TNR) at 95% true positive rate (TPR), the area under the receiver operating characteristic curve (AUROC), the area under the precision-recall curve (AUPR), and the detection accuracy, where larger values of all metrics indicate better detection performances. Due to the space limitation, more explanations about datasets, metrics and network architectures are given in Appendix A.

3.1 Effects of confidence loss

We first verify the effect of confidence loss (1) trained by some explicit, say seen, out-of-distribution datasets. First, we compare the quality of confidence level by applying various training losses. Specifically, the softmax classifier is used and simple CNNs (two convolutional layers followed by three fully-connected layers) is trained by minimizing the standard cross entropy loss on SVHN dataset. We also apply the confidence loss to the model by additionally optimizing the KL divergence term using CIFAR-10 dataset (as training out-of-distribution). In Figure 2(a) and 2(b), we report distributions of the maximum prediction value in softmax scores to evaluate the separation quality between in-distribution (i.e., SVHN) and out-of-distributions. It is clear that there exists a



(a) Cross entropy loss (b) Confidence loss (c) ROC curve

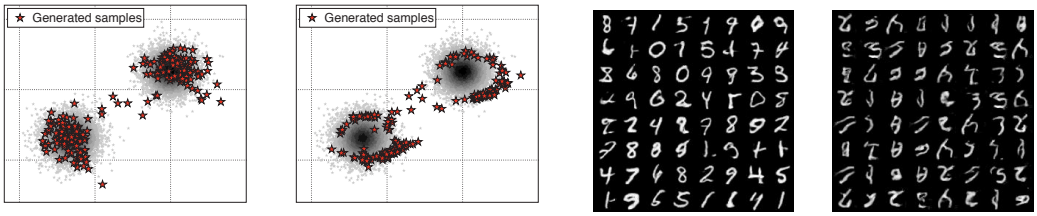
Figure 2: Fraction of the maximum prediction value in softmax scores of trained by (a) cross entropy loss and (b) confidence loss: the x-axis and y-axis represent the maximum prediction value and the fraction of images receiving the corresponding score, respectively. The receiver operating characteristic (ROC) curve under different losses is reported in (c): the red curve corresponds to the ROC curve of a model trained by optimizing the naive cross entropy loss on SVHN data, whereas other ones correspond to the ROC curves of models trained by optimizing the confidence loss. For all experiments in (a), (b) and (c), we commonly use the SVHN dataset for in-distribution.

In-dist	Out-of-dist	Classification accuracy	TNR at TPR 95%	AUROC	Detection accuracy	AUPR in	AUPR out
		Baseline (cross entropy loss) / Our (confidence loss)					
SVHN	CIFAR-10 (seen)	93.82 / **94.23**	47.4 / **99.9**	62.6 / **99.9**	78.6 / **99.9**	71.6 / **99.9**	91.2 / **99.4**
	TinyImageNet (unseen)		49.0 / **100.0**	64.6 / **100.0**	79.6 / **100.0**	72.7 / **100.0**	91.6 / **99.4**
	LSUN (unseen)		46.3 / **100.0**	61.8 / **100.0**	78.2 / **100.0**	71.1 / **100.0**	90.8/ **99.4**
	Gaussian (unseen)		56.1 / **100.0**	72.0 / **100.0**	83.4 / **100.0**	77.2 / **100.0**	92.8 / **99.4**
CIFAR-10	SVHN (seen)	80.14 / **80.56**	13.7 / **99.8**	46.6 / **99.9**	66.6 / **99.8**	61.4 / **99.9**	73.5 / **99.8**
	TinyImageNet (unseen)		**13.6** / 10.1	**39.6** / 31.8	**62.6** / 58.6	**58.3** / 55.3	**71.0** / 66.1
	LSUN (unseen)		**14.0** / 10.8	**40.7** / 34.8	**63.2** / 60.2	**58.7** / 56.4	**71.5** / 68.0
	Gaussian (unseen)		2.8 / **3.5**	10.2 / **14.1**	50.0 / 50.0	48.1 / **49.4**	39.9 / **47.0**

Table 1: Performance of the baseline detector using VGGNet. All values are percentages and boldface values indicate relative the better results. For each in-distribution, we minimize the KL divergence term in (1) using training samples from an out-of-distribution dataset denoted by "seen", where other "unseen" out-of-distributions were also used for testing.

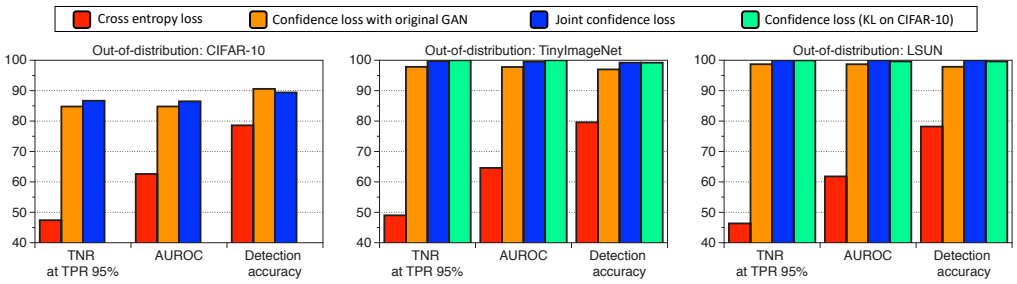
better separation between the SVHN test set (red bar) and other ones when the model is trained by the confidence loss. Here, we emphasize that the maximum prediction value is also low on even untrained (unseen) out-of-distributions, e.g., TinyImageNet, LSUN and synthetic datasets. Therefore, it is expected that one can distinguish in- and out-of-distributions more easily when a classifier is trained by optimizing the confidence loss. To verify that, we obtain the ROC curve using the baseline detector (Hendrycks & Gimpel, 2016) that computes the maximum value of predictive distribution on a test sample and classifies it as positive (i.e., in-distribution) if the confidence score is above some threshold. Figure 2(c) shows the ROC curves when we optimize the KL divergence term on various datasets. One can observe that realistic images such as TinyImageNet (aqua line) and LSUN (green line) are more useful than synthetic datasets (orange line) for improving the detection performance. This supports our intuition that out-of-distribution samples close to in-distribution could be more effective in improving the detection performance as we discussed in Section 2.1.

We indeed evaluate the performance of the baseline detector for out-of-distribution using large-scale CNNs, i.e., VGGNets with 13 layers, under various training scenarios, where more results on AlexNet and ODIN detector (Liang et al., 2017) can be found in Appendix B (the overall trends of results are similar). For optimizing the confidence loss (1), SVHN and CIFAR-10 training datasets are used for optimizing the KL divergence term for the cases when the in-distribution is CIFAR-10 and SVHN, respectively. Table 1 shows the detection performance for each in- and out-of-distribution pair. When the in-distribution is SVHN, the classifier trained by our method drastically improves the detection performance across all out-of-distributions without hurting its original classification performance. However, when the in-distribution is CIFAR-10, the confidence loss does not help the detection performance in overall, where we expect that this is because the trained/seen SVHN out-

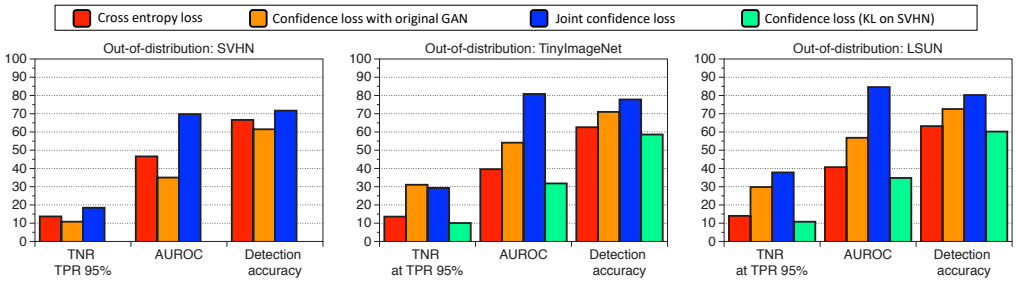


(a) (b) (c) (d)

Figure 3: The generated samples from (a)/(c) original GAN and (b)/(d) proposed GAN. In (a)/(b), the grey area is the 2D histogram of training in-distribution samples drawn from a mixture of two Gaussian distributions and red points indicate generated samples by GANs.



(a) In-distribution: SVHN



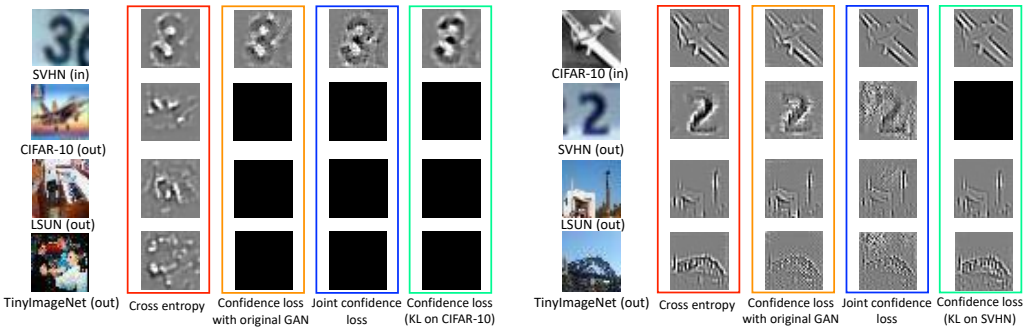
(b) In-distribution: CIFAR-10

Figure 4: Performances of the baseline detector under various training losses. For fair comparisons, we only plot the performances for unseen out-of-distributions, where those for seen out-of-distributions (used for minimizing the KL divergence term in (1)) can be found in Table 1.

of-distribution does not effectively cover all tested out-of-distributions. Our joint confidence loss (4), which was designed under the intuition, resolves the issue of the CIFAR-10 (in-distribution) classification case in Table 1 (see Figure 4(b)).

3.2 Effects of adversarial generator and joint confidence loss

In this section, we verify the effect of the proposed GAN in Section 2.2 and evaluate the detection performance of the joint confidence loss (4). To verify that the proposed GAN can produce the samples nearby the low-density boundary of the in-distribution space, we first compare the generated samples by original GAN and proposed GAN on a simple example where the target distribution is a mixture of two Gaussian distributions. For both the generator and discriminator, we use fully-connected neural networks with 2 hidden layers. For our method, we use a pre-trained classifier which minimizes the cross entropy on target distribution samples and the KL divergence on out-of-distribution samples generated by rejection sampling on a bounded 2D box. As shown in Figure 3(a), the samples of original GAN cover the high-density area of the target distribution while those of proposed GAN does its boundary one (see Figure 3(b)). We also compare the generated samples of original and proposed GANs on MNIST dataset (LeCun et al., 1998), which consists of handwritten digits. For this experiment, we use deep convolutional GANs (DCGANs) (Radford et al.,



(a) In-distribution: SVHN (b) In-distribution: CIFAR-10

Figure 5: Guided gradient (sensitivity) maps of the top-1 predicted class with respect to the input image under various training losses.

2015). In this case, we use a pre-trained classifier which minimizes the cross entropy on MNIST training samples and the KL divergence on synthetic Gaussian noises. As shown in Figure 3(c) and 3(d), samples of original GAN looks more like digits than those of proposed GAN. Somewhat interestingly, the proposed GAN still generates some new digit-like images.

We indeed evaluate the performance of our joint confidence loss utilizing the proposed GAN. To this end, we use VGGNets (as classifiers) and DCGANs (as GANs). We also test a variant of confidence loss which optimizes the KL divergence term on samples from a pre-trained original GAN (implicitly) modeling the in-distribution. One can expect that samples from the original GAN can be also useful for improving the detection performance since it may have bad generalization properties (Arora et al., 2017) and also generate a few samples on the low-density boundary as like the proposed GAN. Figure 4 shows the performance of the baseline detector for each in- and out-of-distribution pair. First, observe that the joint confidence loss (blue bar) outperforms the confidence loss with some explicit out-of-distribution datasets (green bar). This is quite remarkable since the former is trained only using in-distribution datasets, while the latter utilizes additional out-of-distribution datasets. We also remark that our methods significantly outperform the baseline cross entropy loss (red bar) in all cases without harming its original classification performances (see Table 6 in Appendix B). Interestingly, the confidence loss with the original GAN (orange bar) is often (but not always) useful for improving the detection performance, whereas that with the proposed GAN (blue bar) still outperforms it in all cases.

Finally, we also provide visual interpretations of models using the guided gradient maps (Springenberg et al., 2014). Here, the gradient can be interpreted as an importance value of each pixel which influences on the classification decision. As shown in Figure 5, the model trained by the cross entropy loss shows sharp gradient maps for both samples from in- and out-of-distributions, whereas models trained by the confidence losses do only on samples from in-distribution. For the case of SVHN in-distribution, all confidence losses gave almost zero gradients, which matches to the results in Figure 4(a): their detection performances are almost perfect. For the case of CIFAR-10 distribution, one can now observe that there exists some connection between gradient maps and detection performances. This is intuitive because for detecting samples from out-of-distributions better, the classifier should look at more pixels as similar importance and the KL divergence term forces it. We think that our visualization results might give some ideas in future works for developing better inference methods for detecting out-of-distribution under our models.

4 Conclusion

In this paper, we aim to develop a training method for neural classification networks for detecting out-of-distribution better without losing its original classification accuracy. In essence, our method jointly trains two models for detecting and generating out-of-distribution by minimizing their losses alternatively. Although we primarily focus on image classification in our experiments, our method can be used for any classification tasks using deep neural networks. It is also interesting future directions applying our methods for other related tasks: network calibration (Guo et al., 2017), ensemble method (Lakshminarayanan et al., 2017) and semi-supervised learning (Dai et al., 2017).

REFERENCES

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *International Conference on Machine Learning (ICML)*, 2017.

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan Salakhutdinov. Good semi-supervised learning that requires a bad gan. In *Advances in neural information processing systems (NIPS)*, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

Ross Girshick. Fast r-cnn. In *International Conference on Computer Vision (ICCV)*, 2015.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems (NIPS)*, 2014.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations (ICLR)*, 2016.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2014.

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems (NIPS)*, 2017.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Kimin Lee, Changho Hwang, KyoungSoo Park, and Jinwoo Shin. Confident multiple choice learning. In *International Conference on Machine Learning (ICML)*, 2017.

Yingzhen Li and Yarin Gal. Dropout inference in bayesian neural networks with alpha-divergences. In *International Conference on Machine Learning (ICML)*, 2017.

Shiyu Liang, Yixuan Li, and R Srikant. Principled detection of out-of-distribution examples in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

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*, volume 2011, pp. 5, 2011.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2015.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems (NIPS)*, 2016.

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

Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. 2017.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to generate long-term future via hierarchical prediction. In *International Conference on Machine Learning (ICML)*, 2017.

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. In *International Conference on Learning Representations (ICLR)*, 2017.

A EXPERIMENTAL SETUPS IN SECTION 3

Datasets. We train deep models such as VGGNet (Szegedy et al., 2015) and AlexNet (Krizhevsky, 2014) for classifying CIFAR-10 and SVHN datasets: the former consists of 50,000 training and 10,000 test images with 10 image classes, and the latter consists of 73,257 training and 26,032 test images with 10 digits.[1] The corresponding test dataset are used as the in-distribution (positive) samples to measure the performance. We use realistic images and synthetic noises as the out-of-distribution (negative) samples: the TinyImageNet consists of 10,000 test images with 200 image classes from a subset of ImageNet images. The LSUN consists of 10,000 test images of 10 different scenes. We downsample each image of TinyImageNet and LSUN to size 32×32. The Gaussian noise is independently and identically sampled from a Gaussian distribution with mean 0.5 and variance 1. We clip each pixel value into the range $[0, 1]$.

Detailed CNN structure and training. The simple CNN that we use for evaluation shown in Figure 2 consists of two convolutional layers followed by three fully-connected layers. Convolutional layers have 128 and 256 filters, respectively. Each convolutional layer has a 5×5 receptive field applied with a stride of 1 pixel each followed by max pooling layer which pools 2×2 regions at strides of 2 pixels. AlexNet (Krizhevsky, 2014) consists of five convolutitonal layers followed by three fully-connected layers. Convolutional layers have 64, 192, 384, 256 and 256 filters, respectively. First and second convolutional layers have a 5×5 receptive field applied with a stride of 1 pixel each followed by max pooling layer which pools 3×3 regions at strides of 2 pixels. Other convolutional layers have a 3×3 receptive field applied with a stride of 1 pixel followed by max pooling layer which pools 2×2 regions at strides of 2 pixels. Fully-connected layers have 2048, 1024 and 10 hidden units, respectively. Dropout (Hinton et al., 2012) was applied to only fully-connected layers of the network with the probability of retaining the unit being 0.5. All hidden units are ReLUs. Figure 6 shows the detailed structure of VGGNet (Szegedy et al., 2015) with three fully-connected layers and 10 convolutional layers. Each ConvReLU box in the figure indicates a 3×3 convolutional layer followed by ReLU activation. Also, all max pooling layers have 2×2 receptive fields with stride 2. Dropout was applied to only fully-connected layers of the network with the probability of retaining the unit being 0.5. For all experiments, the softmax classifier is used, and each model is trained by optimizing the objective function using Adam learning rule (Kingma & Ba, 2014) with a mini-batch size of 128. The learning rate is chosen from $\{0.001, 0.0005, 0.0002\}$ and the threshold is chosen from $\varepsilon \in \{0, 0.05, \ldots 0.95, 1\}$. We drop the learning rate by 0.2 at 60, 120 and 160 epochs and models are trained for total 200 epochs. The best test result is reported for each method.

Performance metrics. We measure the following metrics using threshold-based detectors:

- **True negative rate (TNR) at 95% true positive rate (TPR).** Let TP, TN, FP, and FN denote true positive, true negative, false positive and false negative, respectively. We measure TNR = TN / (FP+TN), when TPR = TP / (TP+FN) is 95%.
- **Area under the receiver operating characteristic curve (AUROC).** The ROC curve is a graph plotting TPR against the false positive rate = FP / (FP+TN) by varying a threshold.
- **Area under the precision-recall curve (AUPR).** The PR curve is a graph plotting the precision = TP / (TP+FP) against recall = TP / (TP+FN) by varying a threshold. AUPR-IN (or -OUT) is AUPR where in- (or out-of-) distribution samples are specified as positive.
- **Detection accuracy.** This metric corresponds to the maximum classification probability over all possible thresholds δ: $1 - \min_\delta \left\{ P_{\texttt{in}} \left(q\left(\mathbf{x}\right) \leq \delta \right) P\left(\mathbf{x} \text{ is from } P_{\texttt{in}}\right) + P_{\texttt{out}} \left(q\left(\mathbf{x}\right) > \delta \right) P\left(\mathbf{x} \text{ is from } P_{\texttt{out}}\right) \right\}$, where $q(\mathbf{x})$ is a confident score such as a maximum value of softmax. We assume that both positive and negative examples have equal probability of appearing in the test set, i.e., $P\left(\mathbf{x} \text{ is from } P_{\texttt{in}}\right) = P\left(\mathbf{x} \text{ is from } P_{\texttt{out}}\right) = 0.5$.

Note that AUROC, AUPR and detection accuracy are threshold-independent evaluation metrics.

Generating samples on a simple example. As shown in Figure 3(a) and Figure 3(b), we compare the generated samples by original GAN and proposed GAN on a simple example where the target distribution is a mixture of two Gaussian distributions. For both the generator and discriminator, we use fully-connected neural networks with 2 hidden layers and 500 hidden units for each layer. For

[1] We do not use the extra SVHN dataset for training.

Figure 6: Detailed structure of VGGNet with 13 layers.

all layers, we use ReLU activation function. We use a 100-dimensional Gaussian prior for the latent variable $\mathbf{z}$. For our method, we pre-train the simple fully-connected neural networks (2 hidden layers and 500 ReLU units for each layer) by minimizing the cross entropy on target distribution samples and the KL divergence on out-of-distribution samples generated by rejection sampling on bounded 2D box $[-10, 10]^2$. The threshold ε is set to 0.1. We use ADAM learning rule (Kingma & Ba, 2014) with a mini-batch size of 400. The initial learning rate is set to 0.002, and we train for total 100 epochs.

Generating samples on MNIST. As shown in Figure 3(c) and Figure 3(d), we compare the generated samples of original and proposed GANs on MNIST dataset, which consists of greyscale images, each containing a digit 0 to 9 with 60,000 training and 10,000 test images. We expand each image to size $3 \times 32 \times 32$. For both the generator and discriminator, we use deep convolutional GANs (DCGANs) (Radford et al., 2015). The discriminator and generator consist of four convolutional and deconvolutional layers, respectively. Convolutional layers have 128, 256, 512 and 1 filters, respectively. Each convolutional layer has a 4×4 receptive field applied with a stride of 2 pixel. The second and third convolutional layers are followed by batch normalization (Ioffe & Szegedy, 2015). For all layers, we use LeakyReLU activation function. Deconvolutional layers have 512, 256, 128 and 1 filters, respectively. Each deconvolutional layer has a 4×4 receptive field applied with a stride of 2 pixel followed by batch normalization (Ioffe & Szegedy, 2015) and ReLU activation. For our method, we use a pre-trained simple CNNs (two convolutional layers followed by three fully-connected layers) by minimizing the cross entropy on MNIST training samples and the KL divergence on synthetic Gaussian noise. Convolutional layers have 128 and 256 filters, respectively. Each convolutional layer has a 5×5 receptive field applied with a stride of 1 pixel each followed by max pooling layer which pools 2×2 regions at strides of 2 pixels. The threshold is set to 0. We use ADAM learning rule (Kingma & Ba, 2014) with a mini-batch size of 128. The initial learning rate is set to 0.0002, and we train for total 50 epochs.

B MORE EXPERIMENTAL RESULTS

In this section, we provide more experimental results using AlexNet (Krizhevsky, 2014) (see Table 2) and ODIN detector (Liang et al., 2017) (see Table 3). We remark that they show similar trends. For the ODIN detector, the temperature is set to 1000 and perturbation magnitude is set to 0.0012. Also, we report the detection performances of the baseline and ODIN detectors when we optimize the KL divergence term using Gaussian noises and explicit out-of-distribution samples in Table 4 and 5. Finally, Table 6 reports the classification accuracy of VGGNets on CIFAR-10 and SVHN datasets under various training losses shown in Figure 4.

In-dist	Out-of-dist	Classification accuracy	TNR at TPR 95%	AUROC	Detection accuracy	AUPR in	AUPR out
		Baseline (cross entropy loss) / Our (confidence loss)					
SVHN	CIFAR-10 (seen)	92.14 / **93.77**	42.0 / **99.9**	88.0 / **100.0**	83.4 / **99.8**	88.7 / **99.9**	87.3 / **99.3**
	TinyImageNet (unseen)		45.6 / **99.9**	89.4 / **100.0**	84.3 / **99.9**	90.2 / **100.0**	88.6 / **99.3**
	LSUN (unseen)		44.6 / **100.0**	89.8 / **100.0**	84.5 / **99.9**	90.8 / **100.0**	88.4 / **99.3**
	Gaussian (unseen)		58.6 / **100.0**	94.2 / **100.0**	88.8 / **100.0**	95.5 / **100.0**	92.5 / **99.3**
CIFAR-10	SVHN (seen)	**76.58** / 76.18	12.8 / **99.6**	71.0 / **99.9**	73.2 / **99.6**	74.3 / **99.9**	70.7 / **99.6**
	TinyImageNet (unseen)		**10.3** / 10.1	**59.2** / 52.1	**64.2** / 62.0	**63.6** / 59.8	**64.4** / 62.3
	LSUN (unseen)		**10.7** / 8.1	**56.3** / 51.5	**64.3** / 61.8	**62.3** / 59.5	**65.3** / 61.6
	Gaussian (unseen)		**6.7** / 1.0	**49.6** / 13.5	**61.3** / 50.0	**58.5** / 43.7	**59.5** / 32.0

Table 2: Performance of the baseline detector using AlexNet. All values are percentages and boldface values indicate relative the better results. For each in-distribution, we minimize the KL divergence term in (1) using training samples from an out-of-distribution dataset denoted by "seen", where other "unseen" out-of-distributions were also used for testing.

Model (In-dist)	Out-of-dist	Classification accuracy	TNR at TPR 95%	AUROC	Detection accuracy	AUPR in	AUPR out
		Baseline (cross entropy loss) / Our (confidence loss)					
AlexNet (SVHN)	CIFAR-10 (seen)	92.14 / **93.77**	55.5 / **99.9**	89.1 / **99.2**	82.4 / **99.8**	85.9 / **100.0**	89.0 / **99.2**
	TinyImageNet (unseen)		59.5 / **99.9**	90.5 / **99.3**	83.8 / **99.9**	87.5 / **100.0**	90.4 / **99.3**
	LSUN (unseen)		61.5 / **100.0**	91.8 / **99.3**	84.8 / **99.9**	90.5 / **100.0**	91.3 / **99.3**
	Gaussian (unseen)		82.6 / **100.0**	97.0 / **99.3**	91.6 / **100.0**	97.4 / **100.0**	96.4 / **99.3**
AlexNet (CIFAR-10)	SVHN (seen)	**76.58** / 76.18	37.1 / **99.6**	86.7 / **99.6**	79.3 / **99.6**	88.1 / **99.9**	84.2 / **99.6**
	TinyImageNet (unseen)		**11.4** / 8.4	**69.1** / 65.6	**64.4** / 61.8	**71.4** / 68.6	**64.6** / 60.7
	LSUN (unseen)		**13.3** / 7.1	**71.9** / 67.1	**75.3** / 63.7	**67.2** / 72.0	**65.3** / 60.5
	Gaussian (unseen)		**3.8** / 0.0	**70.9** / 57.2	**69.3** / 40.4	**78.1** / 56.1	**60.7** / 40.7
VGGNet (SVHN)	CIFAR-10 (seen)	93.82 / **94.23**	70.0 / **99.9**	93.4 / **99.4**	87.4 / **99.9**	92.3 / **100.0**	92.9 / **99.4**
	TinyImageNet (unseen)		86.9 / **100.0**	95.4 / **99.4**	89.5 / **100.0**	95.3 / **100.0**	94.7 / **99.4**
	LSUN (unseen)		86.4 / **100.0**	95.5 / **99.4**	89.4 / **100.0**	95.9 / **100.0**	94.8 / **99.4**
	Gaussian (unseen)		85.9 / **100.0**	97.3 / **99.4**	92.5 / **100.0**	97.7 / **100.0**	96.5 / **99.4**
VGGNet (CIFAR-10)	SVHN (seen)	80.14 / **80.56**	41.8 / **99.8**	88.8 / **99.8**	81.7 / **99.8**	90.5 / **100.0**	85.7 / **99.8**
	TinyImageNet (unseen)		**19.2** / 10.9	**74.1** / 69.0	**67.6** / 64.5	**75.8** / 70.9	**71.1** / 64.3
	LSUN (unseen)		**20.8** / 12.2	**76.5** / 71.3	**69.8** / 66.4	**78.6** / 73.1	**72.8** / 66.3
	Gaussian (unseen)		4.4 / **7.5**	58.0 / **69.6**	60.4 / **66.3**	66.5 / **75.4**	52.4 / **62.1**

Table 3: Performance of the ODIN detector (Liang et al., 2017) using VGGNet and AlexNet. All values are percentages and boldface values indicate relative the better results. For each in-distribution, we minimize the KL divergence term in (1) using training samples from an out-of-distribution dataset denoted by "seen", where other "unseen" out-of-distributions were also used for testing.

Model (In-dist)	Out-of-dist	Classification accuracy	TNR at TPR 95%	AUROC	Detection accuracy	AUPR in	AUPR out
		Baseline (cross entropy loss) / Our (confidence loss)					
AlexNet (SVHN)	CIFAR-10 (seen)	92.14 / **93.90**	42.0 / **99.9**	88.0 / **100.0**	83.4 / **99.8**	88.7 / **100.0**	87.3 / **99.2**
	TinyImageNet (unseen)		45.6 / **99.9**	89.4 / **100.0**	84.3 / **99.8**	90.2 / **100.0**	88.6 / **99.2**
	LSUN (unseen)		44.6 / **100.0**	89.8 / **100.0**	84.5 / **100.0**	90.8 / **100.0**	88.4 / **99.2**
	Gaussian		58.6 / **100.0**	94.2 / **100.0**	88.8 / **100.0**	95.5 / **100.0**	92.5 / **99.2**
AlexNet (CIFAR-10)	SVHN (seen)	**76.58** / 76.11	12.8 / **99.5**	71.0 / **99.9**	73.2 / **99.5**	74.3 / **99.8**	70.7 / **99.7**
	TinyImageNet (unseen)		10.3 / **13.1**	59.2 / **61.6**	64.2 / **64.5**	63.6 / **64.5**	64.4 / **66.7**
	LSUN (unseen)		10.7 / **13.1**	56.3 / **65.9**	64.3 / **67.4**	62.3 / **68.5**	65.3 / **67.8**
	Gaussian (unseen)		6.7 / **100.0**	49.6 / **100.0**	61.3 /**100.0**	58.5 / **100.0**	59.5 / **99.7**
VGGNet (SVHN)	CIFAR-10 (seen)	93.82 / **93.89**	47.4 / **99.9**	62.6 / **99.9**	78.6 / **99.8**	71.6 / **99.9**	91.2 / **99.5**
	TinyImageNet (unseen)		49.0 / **99.9**	64.6 / **99.9**	79.6 / **99.9**	72.7 / **99.9**	91.6 / **99.5**
	LSUN (unseen)		46.3 / **100.0**	61.8 / **100.0**	78.2 / **100.0**	71.1/ **100.0**	90.8/ **99.5**
	Gaussian (unseen)		56.1 / **100.0**	72.0 / **100.0**	83.4 / **100.0**	77.2 / **100.0**	92.8 / **99.5**
VGGNet (CIFAR-10)	SVHN (seen)	80.14 / **80.56**	13.7 / **99.8**	46.6 / **99.7**	66.6 / **99.7**	61.4 / **99.9**	73.5 / **99.8**
	TinyImageNet (unseen)		13.6 / **15.9**	39.6 / **44.6**	62.6 / **64.8**	58.3 / **60.1**	71.0 / **73.3**
	LSUN (unseen)		14.0 / **17.3**	40.7 / **49.4**	63.2 / **67.4**	58.7 / **62.4**	71.5 / **75.1**
	Gaussian (unseen)		2.8 / **100.0**	10.2 / **100.0**	50.0 / **100.0**	48.1 / **100.0**	39.9 / **99.8**

Table 4: Performance of the baseline detector using VGGNet and AlexNet. All values are percentages and boldface values indicate relative the better results. For each in-distribution, we minimize the KL divergence term in (1) using Gaussian noises and training samples from an out-of-distribution dataset denoted by "seen", where other "unseen" out-of-distributions were also used for testing.

Model (In-dist)	Out-of-dist	Classification accuracy	TNR at TPR 95%	AUROC	Detection accuracy	AUPR in	AUPR out
		Baseline (cross entropy loss) / Our (confidence loss)					
AlexNet (SVHN)	CIFAR-10 (seen)	92.14 / **93.90**	55.5 / **99.9**	89.1 / **99.2**	82.4 / **99.8**	85.9 / **100.0**	89.0 / **99.2**
	TinyImageNet (unseen)		59.5 / **99.9**	90.5 / **99.2**	83.8 / **99.9**	87.5 / **100.0**	90.4 / **99.2**
	LSUN (unseen)		61.5 / **100.0**	91.8 / **99.2**	84.8 / **100.0**	90.5 / **100.0**	91.3 / **99.2**
	Gaussian (unseen)		82.6 / **100.0**	97.0 / **99.2**	91.6 / **100.0**	97.4 / **100.0**	96.4 / **99.2**
AlexNet (CIFAR-10)	SVHN (seen)	**76.58** / 76.11	37.1 / **99.5**	86.7 / **99.6**	79.3 / **99.5**	88.1 / **99.9**	84.2 / **99.7**
	TinyImageNet (unseen)		11.4 / **13.1**	**69.1** / 68.1	**64.4** / 63.5	**71.4** / 69.4	64.6 / **65.0**
	LSUN (unseen)		**13.3** / 13.2	71.9 / **73.3**	75.3 / **77.9**	67.2 / **76.7**	65.3 / **67.8**
	Gaussian (unseen)		3.8 / **100.0**	70.9 / **99.8**	69.3 / **100.0**	78.1 / **100.0**	60.7 / **99.8**
VGGNet (SVHN)	CIFAR-10 (seen)	93.82 / **93.89**	70.0 / **99.9**	93.4 / **99.5**	87.4 / **99.8**	92.3 / **100.0**	92.9 / **99.5**
	TinyImageNet (unseen)		86.9 / **100.0**	95.4 / **99.5**	89.5 / **99.9**	95.3 / **100.0**	94.7 / **99.5**
	LSUN (unseen)		86.4 / **100.0**	95.5 / **99.5**	89.4 / **100.0**	95.9 / **100.0**	94.8 / **99.5**
	Gaussian (unseen)		85.9 / **100.0**	97.3 / **99.5**	92.5 / **100.0**	97.7 / **100.0**	96.5 / **99.5**
VGGNet (CIFAR-10)	SVHN (seen)	80.14 / **80.56**	41.8 / **99.8**	88.8 / **99.8**	81.7 / **99.7**	90.5 / **99.9**	85.7 / **99.8**
	TinyImageNet (unseen)		19.2 / **23.3**	74.1 / **78.6**	67.6 / **71.5**	75.8 / **80.3**	71.1 / **75.8**
	LSUN (unseen)		20.8 / **27.0**	76.5 / **83.2**	69.8 / **75.8**	78.6 / **85.6**	72.8 / **79.5**
	Gaussian (unseen)		4.4 / **100.0**	58.0 / **99.8**	60.4 / **100.0**	66.5 / **100.0**	52.4 / **99.8**

Table 5: Performance of the ODIN detector using VGGNet and AlexNet. All values are percentages and boldface values indicate relative the better results. For each in-distribution, we minimize the KL divergence term in (1) using Gaussian noises and training samples from an out-of-distribution dataset denoted by "seen", where other "unseen" out-of-distributions were also used for testing.

In-distribution	Cross entropy	Confidence loss with original GAN	Joint confidence loss	Confidence loss with explicit out-of-distribution samples
CIFAR-10	80.14	80.27	80.38	80.56
SVHN	93.82	94.13	93.80	94.23

Table 6: Classification test set accuracy of VGGNets on CIFAR-10 and SVHN datasets under various training losses.