Deep Verifier Networks: Verification of Deep Discriminative Models with Deep Generative Models

Tong Che\textsuperscript{*1} Xiaofeng Liu\textsuperscript{*2,3} Site Li\textsuperscript{3} Yubin Ge\textsuperscript{4} Ruixiang Zhang\textsuperscript{1} Caiming Xiong\textsuperscript{5} Yoshua Bengio\textsuperscript{1}

Abstract

AI Safety is a major concern in many deep learning applications such as autonomous driving. Given a trained deep learning model, an important natural problem is how to reliably verify the model’s prediction. In this paper, we propose a novel framework — deep verifier networks (DVN) to detect unreliable inputs or predictions of deep discriminative models, using separately trained deep generative models. Our proposed model is based on conditional variational auto-encoders with disentanglement constraints to separate the label information from the latent representation. We give both intuitive and theoretical justifications for the model. Our verifier network is trained independently with the prediction model, which eliminates the need of retraining the verifier network for a new model. We test the verifier network on both out-of-distribution detection and adversarial example detection problems, as well as anomaly detection problems in structured prediction tasks such as image caption generation. We achieve state-of-the-art results in all of these problems.

1. Introduction

Deep learning models provide state-of-the-art performance in various applications such as image classification (Krizhevsky et al., 2012), caption generation (Xu et al., 2015), sequence modeling (Chung et al., 2014) and machine translation (Wu et al., 2016). However, such performance is based on the assumption that the training and testing data are sampled from the same distribution (Goodfellow et al., 2016). Without this assumption, deep learning models can fail silently by producing high confidence incorrect predictions even on completely unrecognizable or irrelevant inputs (Amodei et al., 2016). For instance, the models trained on MNIST can produce 91% confidence on random noise images (Hendrycks & Gimpel, 2017). Generally speaking, the behavior of a trained deep learning model on a slightly different test distribution is unpredictable. One such problematic case is also shown in Fig. 1. Unfortunately, there is very little control over the test distribution in real-world deployments due to dynamically changing environments or malicious attacks (Guo et al., 2017). In fact, well calibrating the predictive uncertainty of DNNs is important for many production systems, e.g., authentication devices, medical diagnosis and self-driving vehicles (Lee et al., 2018b).

Being overconfident on out-of-distribution (OOD) inputs has raised concerns about the safety of artificial intelligence (AI) systems. Recent research efforts try to address these concerns by developing models that can identify anomalous inputs, i.e., OOD samples (Amodei et al., 2016). Formally, the OOD detection problem can be formulated as a binary classification problem where the objective is to decide whether a test sample is from the training distribution (i.e., in-distribution, ID) or from a different distribution (i.e., OOD).

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{A network trained on CIFAR-10 with ten-class softmax output will predict the resized 32x32x3 view of Vienna in ICML 2020’s website (OOD sample w.r.t. CIFAR-10) as the horse with high confidence.}
\end{figure}
In this paper, we propose to verify the predictions of deep discriminative models by using deep generative models that try to generate the input conditioned on the label selected by the discriminative model. We call this concept "deep verifier". The high-level idea is simple: we train an inverse verification model \( p(x|y) \) on the training data pairs \( (x, y) \). Intuitively speaking, for an input-output pair \( (x, y) \) with \( y \) picked by the predictive model, we verify whether the input \( x \) is consistent with \( y \), by estimating if \( p(x|y) \) is larger than a threshold. We design a density estimator of \( p(x|y) \) using modified conditional VAEs. To ensure that the class code \( y \) is not ignored as a conditioning variable, we impose a disentanglement constraint based on minimizing mutual information between latent variable \( z \) and the label \( y \). Although many different kinds of density estimators can be used in theory, we argue that the design of our model is robust to OOD samples and adversarial attacks, due to the use of latent variables with explicit and accurate density estimation.

Compared with previous approaches of OOD detection, our proposed method has four main advantages:

1. The verifier is trained independently of OOD distributions. Users do not need to figure out OOD samples before deployment of the system.

2. The verifier only needs to be trained once. No need to retrain the verifier for a new classifier.

3. The verifier can detect ordinary OOD samples and malicious adversarial attacks in a unified manner.

4. The framework is very general, so that it applies to structured prediction problems as well, such as image captioning.

We summarize the comparison of these four advantages with previous methods in Table 1.

The proposed solution achieves the state-of-the-art performance for detecting either OOD or adversarial samples in all tested classification scenarios, and can be generalized well for structured prediction tasks (e.g., image caption). In Sec 3.4, we analysed why DVN is useful for both OOD and Adversarial examples.

2. Related Work

Detecting the OOD samples in a low-dimensional space using traditional non-parametric density estimation, nearest neighbor and clustering analysis have been well-studied (Pimentel et al., 2014). However, they are usually unreliable in high-dimensional complex spaces, e.g., of images (Liang et al., 2018).

OOD detection with deep neural networks has recently been an active research topic. (Hendrycks & Gimpel, 2017) found that trained DNNs usually have higher maximum softmax output for in-distribution examples than anomalous one. A possible improvement of this baseline is to consider both the in-distribution and out-of-distribution training samples during training (Hendrycks, 2019). However, enumerating all possible OOD distributions before deployment is usually not possible.

(Liang et al., 2018) proposed that the difference between maximum probabilities in softmax distributions on ID/OOD samples can be made more significant by using adversarial perturbation pre-processing during training. (Devries & Taylor, 2018) augmented the classifier with a confidence estimation branch, and adjusted the objective using the predicted confidence score for training. (Lee et al., 2018a) trained a classifier simultaneously with a GAN, with an additional objective to encourage low confidence on generated samples. (Hendrycks, 2019) proposed to use real OOD samples instead of generated ones to train the detector. (Vyas et al., 2018) labels a part of training data as OOD samples to train the classifier, and they dynamically change the partition of ID and OOD samples. These improvements based on (Hendrycks & Gimpel, 2017) typically needs retrain a classifier with modified structures or optimization objectives. This can make it hard to maintain the original accuracy and is computationally expensive.

Recently, (Lee et al., 2018b) proposed a new framework of anomaly detection by first obtaining the class conditional Gaussian distribution using Gaussian discriminative analysis, and then define confidence score using the Mahalanobis distance between the sample and the closest class-conditional Gaussian distribution. By modeling each class of in-distribution samples independently, it showed remarkable results for OOD and adversarial attacks detection. Note however that their methods also needs the input pre-processing and model change. Besides, many previous methods (Liang et al., 2018; Vyas et al., 2018; Lee et al., 2018b) need OOD samples for hyper-parameter (e.g., threshold for verification) selection, and these are usually not accessible in the real world.

Recently, (Choi et al., 2018) proposed an unsupervised OOD detector by estimating the Watanabe-Akaike Information Criterion, which is in turn estimated using an ensemble.
of generative models. The goal of our model is different from WAIC in that rather than just detecting OOD samples, DVNs aim to verify the predictions of a supervised predictive model, i.e., estimating \( p(x|y) \) not just \( p(x) \). We argue that model \( p(x|y) \) is usually easier than directly model \( p(x) \) as the former distribution contains less modes. Another motivation for modelling \( p(x|y) \) instead of \( p(x) \) is that for an adversarial attack and its classifier prediction \((x',y')\), it is usually much easier to verify \( x' \) is not in \( p(x|y') \) than to verify \( x' \) is not in \( p(x) \).

### 3. Methodology

This paper targets the problem of verification of deep predictive models, as follows. Let \( x \in \mathcal{X} \) be an input and \( y \in \mathcal{Y} \) be the ground-truth value to be predicted. The in-distribution samples are sampled from the joint data-generating distribution \( p_m(x,y) = p_m(y|x)p_m(x) \). We propose to reverse the order of the prediction process of \( p(y|x) \) and try to compute the conditional probability \( p(x|y) \), where \( y \) is the label value guessed by the classifier to be verified (e.g., the one with the highest probability according to the deep network). We evaluate whether the input \( x \) is consistent with that \( y \).

The predictive model to be verified \( p_o(y|x) \) is trained on a dataset set drawn from the \( p_m(x,y) \), and may encounter samples from both \( p_m(x,y) \) and \( p_{oa}(x,y) \) (i.e., out-of-distribution or adversarial samples) at test time. Note there is some subtle difference between OOD (unlikely under \( p_m(x) \)) and adversarial examples (unlikely under the ground truth joint, but with high \( p_m(x) \), especially if a small amount of noise is allowed).

Our goal is to verify if the pair \((x,y)\) for \( y \) guessed by the predictive model given \( x \) is consistent with \( p_m(x,y) \). We train a verifier network \( q_o(x|y) \) as an approximation to the inverse posterior distribution \( p(x|y) \). Modelling \( p(x|y) \) instead of \( p(x) \) as a verification has many advantages: (1) Usually \( p(x) \) is much more diverse than the conditional distribution \( p(x|y) \), so modelling \( p(x|y) \) is much easier than modelling \( p(x) \). (2) Modelling \( p(x|y) \) allows us to provide a unified framework for verifying OODs, adversarial examples, and mis-classifications of the classifier.

#### 3.1. Basic Model

Our basic model is a conditional variational auto-encoder shown in Fig. 2. The model is composed of two deep neural networks, a stochastic encoder \( p_o(z|x) \) which takes input \( x \) to predict a latent variable \( z \) and a decoder \( p(x|z,y) \) which takes both latent variable \( z \) and the label \( y \) to reconstruct \( x \). One problem with training of conditional variational auto-encoders is that the decoder can ignore the effect of input label \( y \), passing all information through the continuous latent variable \( z \). This is not desirable as we want to use the decoder to model the conditional likelihood \( p(x|y) \), not \( p(x) \). Hence in this paper, we train the encoder so that it outputs a \( z \) which is approximately independent of \( y \). The encoder and decoder are thus jointly trained to maximize the evidence lower bound (ELBO):

\[
\log p(x|y) \geq \mathbb{E}_{q(z|x)}[\log p(x|z, y)] - KL(q(z|x)||p(z))
\]

The equality holds if and only if \( q(z|x) = p(z|x, y) \), where \( p(z|x, y) \) is the ground truth posterior. We note that the conditional GAN is not applicable here since its objective does not optimize the likelihood.

#### 3.2. Disentanglement constraints for anomaly detection

To achieve this independence, we propose to add a disentanglement penalty to minimize the mutual information between \( z \) and \( y \). Namely, besides the ELBO loss, we also minimize the mutual information estimator \( \hat{I}(z, y) \) together with the loss, yielding:

\[
L = -\mathbb{E}_{q(z|x)}[\log p(x|z, y) + \lambda \hat{I}(y, z)] + KL(q(z|x)||p(z))
\]

In this paper, we use deep Infomax (Hjelm et al., 2018) as the proxy for minimizing the mutual information (MI) between \( z \) and \( y \). The mutual information estimator is defined as:

\[
\hat{I}(z, y) = \mathbb{E}_{p(y,z)}[-s_+( -T(y, z))] - \mathbb{E}_{p(y)p(z)}[s_+( T(y, z))]
\]

where \( s_+ \) is the softplus function and \( T(y, z) \) is a discriminator network. Just like GAN discriminators, \( T \) is trained to maximize \( \hat{I}(y, z) \), in order to get a better lower-bound estimation of the (JS-version) mutual information, while \( L \) (and in particular the encoder and decoder) is optimized (considering \( T \) fixed) to minimize \( \hat{I}(y, z) \).
3.3. Measuring the likelihood as anomaly score

Our anomaly verification criterion is based on estimating the log-likelihood \( \log p(x|y) \) for test samples. Importance sampling is a possible solution to provide an unbiased estimate of \( p(x|y) \) when we have a VAE. Following IWAEn (Burda et al., 2015), the \( k \)-sample importance weighting estimate of the log-likelihood is a lower bound of the ground truth likelihood \( \mathcal{L}(x|y) = \mathbb{E}_{x \sim p(x|y)}[\log p(x|y)] \):

\[
\mathcal{L}_k(x|y) = \mathbb{E}_{z_1, \ldots, z_k \sim q(z|x)}[\log \frac{1}{k} \sum_{i=1}^{k} \frac{q(x, z_i|y)}{q(z_i|x)}].
\] (4)

where \( q(z) \) is a corrected density described below. We use the fact that \( \mathcal{L}_k(x|y) \rightarrow \mathcal{L}(x|y) \) as \( k \rightarrow \infty \) to estimate the likelihood. As will be discussed below, we want the decoder \( q(z|x,y) \) to be evaluated on the same input distribution for \( z \) as it is trained, which is not exactly the original Gaussian prior \( p(z) \), so we will form a refined estimator of the prior, denoted \( p^*(z) \). The quantities \( \mathcal{L}_k(x|y) \) form a monotonic series of lower bounds of the exact log-likelihood \( \log p(x|y) \), with \( \mathcal{L}_1 \leq \mathcal{L}_2 \leq \cdots \mathcal{L}_k \leq \log p(x|y) \). They have the property that when \( k \rightarrow \infty \), \( \mathcal{L}_k \rightarrow \log p(x|y) \). In our experiments we chose \( k=100 \) for a good approximation of the exact likelihood.

In our algorithm, the distribution of \( z \) actually fed into decoder \( p(x|z,y) \) during training is \( q(z) = \int q(z|x) p(x|y) \). However, this distribution \( q(z) \) can be drastically different from the Gaussian prior \( p(z) \). So instead of using the Gaussian prior \( p(z) \) for the decoder in Eq. 4, we use \( q(z) \) and estimate the corrected likelihood of \( x \) under this directed generative model, as \( p(x, z|y) = q(z)p(x|z,y) \). In order to estimate the density of \( q(z) \), we propose to train an additional discriminator \( D_z \) to distinguish \( p(z) \) and \( q(z) \). \( D_z \) is trained to discriminate the real distribution of latent variable \( q(z) = \int p_d(x) e(z|x) dx \) (where \( p_d(x) \) is the data distribution of \( x \), \( e(z|x) \) is the encoder network) and Gaussian prior distribution \( p(z) \), with ordinary GAN loss. Both \( q(z) \) and \( p(z) \) are easy to sample, so a discriminator is easy to train with the samples. In the GAN, the optimal discriminator \( D_z \) (Goodfellow, 2016) can be

\[
D_z = \frac{p(z)}{p(z) + q(z)}
\] (5)

After \( D_z \) is trained (in theory optimally) and since \( p(z) \) is known (i.e., Gaussian), we can estimate \( q(z) = \frac{1-D_z(z)}{D_z(z)} p(z) \).

We classify a sample \( x \) as an OOD sample if the log-likelihood is below the threshold \( \delta \) and the \( x \) is an in-distribution sample, otherwise

\[
x \in \begin{cases} 
\text{in-distribution (ID),} & \text{if } L_k(x) \geq \delta \\
\text{out-of-distribution (OOD),} & \text{otherwise}
\end{cases}
\] (6)

We set \( \delta \) to the threshold corresponding to 95% true positive rate (TPR), where the TPR refer to the probability of in-distribution validation samples are correctly verified as the in-distribution. Therefore, the threshold selection in our model is only tuned on in-distribution validation datasets, while most of previous methods also need the OOD samples for hyper-parameter validation (Liang et al., 2018; Lee et al., 2018b). We note that the distribution of OOD samples is usually not accessible before the system deployment.

3.4. Theoretical Justification

The loss function we optimize can be written as:

\[
L = L_1 + \lambda L_2 = \mathbb{E}_{x,y \sim p_d}[-\mathbb{E}_{q(z|x)}[\log p(x|z, y)]] + KL(q(z|x)||p(z)) + \lambda \mathbb{E}_{q(z|x)}[\hat{I}(y, z)]
\] (7)

where \( p(x|z, y) \) is the decoder we are training. In this section, we use the following convention. Symbol \( p \) means probability distributions induced by the decoder, and symbol \( q \) means probability distributions induced by the encoder. Also denote \( p_d \) for real data distributions. Specifically, we define joint distribution \( q(z, x, y) = q(z|x)p_d(x, y) \).

We have the following theorem that justifies the two parts of the above loss.

Theorem 1

(i) \(-L_1\) is a variational lower bound of \( \mathbb{E}_{x,y \sim p_d}[\log p(x|y)] \). The bound is tight when \( q \) is expressive enough and \( x, y \) are conditionally independent given \( x \).

(ii) If we have \( I(y, z) = 0 \), where \( (y, z) \sim \mathbb{E}_{x,y \sim p_d}[p_d(y|x)q(z|x)] \) (namely \( L_2 \approx 0 \)), and assume that the decoder is perfect in sense that \( p(x|y, z) = q(x|y, z) \), then we have our evaluation metric \( \mathbb{E}_{z \sim q(z)}[p(x|y, z)] = p_d(x|y) \). Namely, if \( I(y, z) = 0 \), and the decoder is trained to optimal, \( y \) no matter what the encoder looks like, the likelihood estimator we are using is \( \mathbb{E}_{z \sim q(z)}[p(x|y, z)] \) is equal to the groundtruth likelihood. This justifies why we need loss \( L_2 \).

Proof 3.1 For (i), we have:

\[
-L_1 = \mathbb{E}_{x,y \sim p_d}[\mathbb{E}_{q(z|x)}[\log p(x|z, y)] - KL(q(z|x)||p(z))] = \mathbb{E}_{x,y \sim p_d}[\mathbb{E}_{q(z|x)}[\log p(x, z|y)] - \log p(z|y)] - KL(q(z|x)||p(z)) \leq \mathbb{E}_{x,y \sim p_d}[\log p(x|y)]
\] (8)

1In this paper we assume \( q(z|x) = q(z|x, y) \), the motivation is during test time, \( y \) may be a wrong label, we don’t want it to confuse the encoder. See detailed ablation in our Appendix.
The bound is tight if \( \mathbb{E}[KL(q(z|x)||p(z|x,y))] = 0 \), which is equivalent to \( \mathbb{E}[KL(q(z|x)||p(z|x))] = 0 \) if \( z, y \) are conditionally independent given \( x \).

For (ii), we have that if \( y, z \) are independent, namely \( q(y,z) = p_d(y)q(z) \), we have: \( q(x,y,z) = q(x,y,z)/q(y,z) = q(z|x)p_d(x,y)/p_d(y)q(z) = q(z|x)p_d(x|y)/q(z) \). So we have:

\[
E_{z\sim q(z)}[p(x|y,z)] = E_{z\sim q(z)}[q(x|y,z)] = E_{z\sim q(z|x)}[p_d(x|y)] = p_d(x|y)
\]

3.5. Intuitive Justifications

We now present an intuitive justification for the above algorithm. First, consider the following part of our training loss:

\[
L_1 = E_{q(z|x)}[\log p(x|z,y)] - KL(q(z|x)||p(z)) \tag{9}
\]

It is well known that deep neural networks can generalize well for in-distribution samples, but their behavior out-of-distribution is less clear. Suppose \( x \) is an out-of-distribution sample, with \( y \) be the corresponding output of the classifier. Then the behavior of the stochastic encoder \( q(z|x) \) is undefined. We denote \( q(z) = \int q(z|x)p_d(x) \) the distribution to train \( q(x|y,z) \). There are two cases: (1) \( q(z|x) \) maps \( x \) to \( z \) with low density in \( q(z) \). This case can be easily detected because \( q(z) \) is easily computable. In this case the second term in Eq. 9 is a large negative number. (2) \( q(z|x) \) maps \( x \) to \( z \) with high density in \( q(z) \). Then since we train the decoder network with the input distribution \( q(z) \) and because \( y \) and \( z \) are approximately independent, so \( (z,y) \) looks like an in-distribution input for decoder \( p(x|z,y) \). Thus \( p(x|y,z) \) should map to some in-distribution \( x' \) with class label \( y \). Since input \( x \) is an OOD sample and reconstruction \( x' \) is an in-distribution sample, the reconstruction has to be bad. In this case, the first term in Eq. 9 is a large negative number. So in both cases, the log-likelihood score \( L_k \) derived from our model should be a large negative number. This is why our model is robust to both adversarial and OOD samples.

3.6. Can We Replace VAEs with Other Density Estimators?

In theory, we can use any other density estimator besides our modified conditional VAE (such as auto-regressive models and flow-based models) to estimate \( p(x|y) \). However, our experiments and previous observations suggest that these other models may have drawbacks that would make them less suitable for this task. The comparison with the DVN that based on PixelCNN (Van den Oord et al., 2016) and Glow (Kingma & Dhariwal, 2018) are compared in Table 2, which consistently inferior than our VAE solution. Auto-regressive models are quite slow and may ignore the conditioning label \( y \) (Bowman et al., 2015). Flow-based models were found to be less robust to adversarial examples, assigning higher likelihood on OOD samples than in-distribution samples (Nalisnick et al., 2018). We have intuitively explained in Sec. 3.5 why our modified cVAE based model does not suffer from the same problem as flow-based models, thanks to our disentanglement regularizer, which relies on the existence of a latent space.

4. Experimental results

In this section, we demonstrate the effectiveness of the proposed DVN on several classification benchmarks, and show its potential for the image captioning task. We choose the DenseNet (Huang et al., 2017) and ResNet (He et al., 2016) architectures as the backbones of our experiments.

For evaluation, we measure the following metrics as indicators of effectiveness in distinguishing in- vs out-of-distribution samples. Following the definitions in previous work, we denote the in-distribution images as positive samples, while the OOD ones as the negative samples.

- **True Negative Rate or False Positive Rate at 95% True Positive Rate (i.e., TNR@TPR95% or FPR@TPR95%).** Let TP, TN, FP, and FN denote true positive, true negative, false positive and false negative, respectively. We measure TNR = TN / (FP+TN) or FPR = FP / (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. The area under ROC is the probability that an in-distribution sample has a higher certainty score than an OOD sample.

- **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 the threshold.

- **Verification accuracy** is defined by \( 1 - \min_{\delta} \{ p_{\text{in}}(L \leq \delta)p(x \in p_{\text{in}}) + p_{\text{out}}(L > \delta)p(x \in p_{\text{out}}) \} \), where \( L \) is the predicted certainty score, \( p(x \in p_{\text{in}}) \) or \( p(x \in p_{\text{out}}) \) is the probability of positive or negative samples in the test set. It corresponds to the maximum classification probability over all possible thresholds. We note that AUROC, AUPR and verification accuracy are threshold (\( \delta \))-independent evaluation metrics.

4.1. Detecting out-of-distribution samples for classification

**Datasets.** The Street View Housing Numbers (SVHN) dataset (Netzer et al., 2011) consists of color images depicting house numbers, which range from 0 to 9. Images have a resolution of 32×32. For our tests, we use the official training set split which contains 73,257 images, and the test set
Deep Verifier Networks: Verification of Deep Discriminative Models with Deep Generative Models

We analyze how the hyperparameters generalize across different OOD datasets. We tune the hyperparameters on one of five OOD datasets and tested on Tiny-ImageNet or LSUN dataset. Given those OOD datasets, we split each of them to 10% of the images for validation, and 90% images as test set. Firstly, we minimize the FPR at TPR 95% on one of the OOD dataset's possible OOD pairs. Noticing that the threshold of the binary cross-entropy loss. The hyper-parameters in previous setups.

<table>
<thead>
<tr>
<th>In-Dist</th>
<th>OOD</th>
<th>Validation on OOD samples</th>
<th>Verification acc.</th>
<th>Validation on adversarial samples</th>
<th>Verification acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>SVHN</td>
<td>49.2/97.1</td>
<td>95.8</td>
<td>95.1/97.1/96.9</td>
<td>95.1/97.1/96.9</td>
</tr>
<tr>
<td>DenseNet T-ImageNet</td>
<td>LSUN</td>
<td>49.2/97.1</td>
<td>95.8</td>
<td>95.1/97.1/96.9</td>
<td>95.1/97.1/96.9</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>SVHN</td>
<td>49.2/97.1</td>
<td>95.8</td>
<td>95.1/97.1/96.9</td>
<td>95.1/97.1/96.9</td>
</tr>
<tr>
<td>DenseNet T-ImageNet</td>
<td>LSUN</td>
<td>49.2/97.1</td>
<td>95.8</td>
<td>95.1/97.1/96.9</td>
<td>95.1/97.1/96.9</td>
</tr>
<tr>
<td>SVHN</td>
<td>CIFAR-10</td>
<td>49.2/97.1</td>
<td>95.8</td>
<td>95.1/97.1/96.9</td>
<td>95.1/97.1/96.9</td>
</tr>
<tr>
<td>DenseNet T-ImageNet</td>
<td>LSUN</td>
<td>49.2/97.1</td>
<td>95.8</td>
<td>95.1/97.1/96.9</td>
<td>95.1/97.1/96.9</td>
</tr>
</tbody>
</table>

Figure 3. FPR (for OOD) and TPR (for ID) under different δ when using CIFAR-10 as the in-distribution dataset, and use Tiny-ImageNet(resize), LSUN and Gaussian/Uniform noise as OOD. CIFAR-10 only applicable to the TPR which use the dashed red line and indicated by the right axis while the other OOD datasets use the left FPR axis.

DVN is tuned on a distribution only. This corresponds to a more realistic scenario, since the OOD nature of real-world applications is usually uncontrollable.

Table 2. OOD verification results of image classification under different validation setups. All metrics are percentage and the best results are bolded. The backbone classifier in SUF (Lee et al., 2018b) and our DVN is ResNet34 (He et al., 2016), while ODIN (Liang et al., 2018; Lee et al., 2018b) use more powerful wide ResNet 40 with width 4 (Zagoruyko & Komodakis, 2016).

Figure 3. FPR (for OOD) and TPR (for ID) under different δ when using CIFAR-10 as the in-distribution dataset, and use Tiny-ImageNet(resize), LSUN and Gaussian/Uniform noise as OOD. CIFAR-10 only applicable to the TPR which use the dashed red line and indicated by the right axis while the other OOD datasets use the left FPR axis.

DVN is tuned on a distribution only. This corresponds to a more realistic scenario, since the OOD nature of real-world applications is usually uncontrollable.

Effects of the threshold and performance across datasets. How the hyper-parameters (e.g., δ) generalize across different OOD datasets is a challenging aspect of the system deployment. Most of the previous methods require a small set of OOD samples, with δ calibrated by evaluating the verification error at different values of δ. However, the more realistic scenario is that we do not have access to the OOD examples in the testing stage. A promising trend is improving the performance on an unknown OOD when using the model tuned on a similar OOD (Liang et al., 2018; Lee et al., 2018b). We argue that our
Deep Verifier Networks: Verification of Deep Discriminative Models with Deep Generative Models

Table 3. Test error rate of classification on CIFAR-10/100 using DenseNet as backbone. Our DVN does not re-train or modify the structure of the original trained classifier.

<table>
<thead>
<tr>
<th></th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODIN/SUF</td>
<td>4.81</td>
<td>22.37</td>
</tr>
<tr>
<td>DenseNet/DVN</td>
<td><strong>4.51</strong></td>
<td><strong>22.27</strong></td>
</tr>
</tbody>
</table>

Figure 4. Comparison with baseline MSP (Hendrycks & Gimpel, 2017) using DenseNet, with Tiny-ImageNet as in-distribution and LSUN as OOD.

Table 4. The performance of DVN w/o disentanglement of $y$ from $z$ with ResNet backbone, and using CIFAR-10/SVHN as in-distribution/OOD, respectively.

<table>
<thead>
<tr>
<th>$q(z)$</th>
<th>TNR@TPR95%</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td><strong>98.4</strong></td>
<td><strong>99.2</strong></td>
</tr>
<tr>
<td>-</td>
<td>62.6</td>
<td>84.7</td>
</tr>
</tbody>
</table>

Table 5. The performance of DVN w/o replace $p(z)$ with $q(z)$. We use ResNet backbone, and choose CIFAR-10/SVHN as in-distribution/OOD.

Ablation study. We also compared with the DVN that use pixel CNN or Glow rather than VAE as shown in Table 2. The pixelCNN/Glow-based DVN is consistently inferior than our solution.

Disentangling $y$ from $z$ is critical to our model. Table 4 validates the contribution of this manipulation w.r.t. both threshold dependent and independent metrics. One can see that the DVN with disentanglement significantly outperforms its counterparts without disentanglement. This also implies the DVN has successfully learned to sufficiently minimize the mutual information between $z$ and $y$ to circumvent the challenge of conditioning $x$ on $y$.

Since modeling $p(x|y)$ is the core of DVN, we cannot remove $y$. Here, we give another ablation study that without modify $p(z)$ with $q(z)$, are shown in . As shown in Table 5, there is a large margin between the DVN with or without disentanglement w.r.t. TNR@TPR95 and AUROC. The results demonstrate that disentanglement $y$ from $z$ is of essential important for our framework.

4.2. Detecting adversarial examples

Deal with popular adversarial attackers. To detect adversarial examples, we train our DenseNet and ResNet-based classification network and DVN using the training set of CIFAR-10, CIFAR-100 or SVHN datasets, and their corresponding test sets are used as the positive samples for the test. Following the setting in (Lee et al., 2018b), we applied several attack methods to generate the negative samples, such as basic iterative method (BIM) (Kurakin et al., 2016), Deepfool (Moosavi-Dezfooli et al., 2016), and Carlini-Wagner (CW) (Carlini & Wagner, 2017). The network structures are the same as for OOD verification.

We compare the DVN with the strategies in KD+PU (Feinman et al., 2017), LID (Ma et al., 2018), SUF (Lee et al., 2018b) in Table 6, and show that the DVN can achieve the
In this paper, we propose to enhance the performance of anomaly detection by verifying predictions of deep discriminative models using deep generative models. The idea is to train a conditional verifier network $q(x|y)$ as an approximation to the inverse posterior distribution. We propose Deep Verifier Networks (DVNs) which are based on a modified conditional variational auto-encoders with disentanglement constraints. We show our model is able to achieve state-of-the-art performance on benchmark OOD detection and adversarial example detection tasks.

For future work, it would be interesting to integrate DVNs to safe AI systems. For instance, ordinary image classifiers such as DenseNet have perfect accuracy for in-distribution queries, but their behaviors are undefined on adversarial queries. Robust image classifiers sacrifice some accuracy for robustness to adversarial examples. We can use DVNs to form a two-step prediction procedure: first using an or-

Table 7. OOD verification results of image caption under different validation setups. We use CUB-200, LSUN and COCO as the OOD of Oxford-102, while using Oxford-102, LSUN and COCO as OOD of CUB-200.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Methods</th>
<th>Negative Sample</th>
<th>Pre-process</th>
<th>Deep Fool</th>
<th>CW</th>
<th>BIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR 10</td>
<td>KD+PU</td>
<td>FGSM</td>
<td>-</td>
<td>68.34</td>
<td>53.21</td>
<td>3.10</td>
</tr>
<tr>
<td></td>
<td>LID</td>
<td>FGSM</td>
<td>-</td>
<td>70.86</td>
<td>71.50</td>
<td>94.55</td>
</tr>
<tr>
<td></td>
<td>SUF</td>
<td>FGSM</td>
<td>Yes</td>
<td>87.95</td>
<td>83.42</td>
<td>99.51</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>-</td>
<td>-</td>
<td>90.14</td>
<td>86.38</td>
<td>99.42</td>
</tr>
<tr>
<td>CIFAR 100</td>
<td>KD+PU</td>
<td>FGSM</td>
<td>-</td>
<td>65.30</td>
<td>58.08</td>
<td>66.86</td>
</tr>
<tr>
<td></td>
<td>LID</td>
<td>FGSM</td>
<td>-</td>
<td>69.68</td>
<td>72.36</td>
<td>68.62</td>
</tr>
<tr>
<td></td>
<td>SUF</td>
<td>FGSM</td>
<td>Yes</td>
<td>75.63</td>
<td>86.20</td>
<td>98.27</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>-</td>
<td>-</td>
<td>90.14</td>
<td>98.55</td>
<td>99.04</td>
</tr>
<tr>
<td>SVHN</td>
<td>KD+PU</td>
<td>FGSM</td>
<td>-</td>
<td>84.38</td>
<td>82.94</td>
<td>83.28</td>
</tr>
<tr>
<td></td>
<td>LID</td>
<td>FGSM</td>
<td>-</td>
<td>80.14</td>
<td>85.09</td>
<td>92.21</td>
</tr>
<tr>
<td></td>
<td>SUF</td>
<td>FGSM</td>
<td>Yes</td>
<td>93.47</td>
<td>96.95</td>
<td>99.12</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>-</td>
<td>-</td>
<td>94.14</td>
<td>97.35</td>
<td>99.12</td>
</tr>
<tr>
<td>CIFAR 10</td>
<td>KD+PU</td>
<td>FGSM</td>
<td>-</td>
<td>76.80</td>
<td>56.30</td>
<td>16.16</td>
</tr>
<tr>
<td></td>
<td>LID</td>
<td>FGSM</td>
<td>-</td>
<td>71.86</td>
<td>77.53</td>
<td>95.38</td>
</tr>
<tr>
<td></td>
<td>SUF</td>
<td>FGSM</td>
<td>Yes</td>
<td>78.06</td>
<td>93.90</td>
<td>98.91</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>-</td>
<td>-</td>
<td>82.45</td>
<td>95.51</td>
<td>99.07</td>
</tr>
<tr>
<td>CIFAR 100</td>
<td>KD+PU</td>
<td>FGSM</td>
<td>-</td>
<td>57.78</td>
<td>73.72</td>
<td>68.85</td>
</tr>
<tr>
<td></td>
<td>LID</td>
<td>FGSM</td>
<td>-</td>
<td>63.15</td>
<td>75.03</td>
<td>55.82</td>
</tr>
<tr>
<td></td>
<td>SUF</td>
<td>FGSM</td>
<td>Yes</td>
<td>81.95</td>
<td>90.96</td>
<td>96.38</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>-</td>
<td>-</td>
<td>85.22</td>
<td>93.38</td>
<td>97.72</td>
</tr>
<tr>
<td>SVHN</td>
<td>KD+PU</td>
<td>FGSM</td>
<td>-</td>
<td>67.28</td>
<td>76.58</td>
<td>84.88</td>
</tr>
<tr>
<td></td>
<td>LID</td>
<td>FGSM</td>
<td>-</td>
<td>72.20</td>
<td>86.73</td>
<td>95.39</td>
</tr>
<tr>
<td></td>
<td>SUF</td>
<td>FGSM</td>
<td>Yes</td>
<td>-</td>
<td>89.38</td>
<td>96.10</td>
</tr>
<tr>
<td></td>
<td>Our</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6. Comparison of AUROC (%) under different validation setups. The best results are bolded. We also compared the use of negative sample for training and input image pre-processing.

4.3. Detecting out-of-distribution samples for image captioning

For detecting OOD samples in the image captioning task, we choose Oxford-102 and CUB-200 as the in-distribution datasets. Oxford-102 contains 8,189 images of 102 classes of flower. CUB-200 contains 200 bird species with 11,788 images. Each of them has 10 descriptions that are provided by (Reed et al., 2016a). For these two datasets, we use 80% of the samples to train our caption generator, and the remaining 20% for testing in a cross-validation manner. The LSUN and Microsoft COCO datasets are used as our OOD dataset.

The captioner used in here is a classical image caption model (Xu et al., 2015). We choose the generator of GAN-INT-CLS (Reed et al., 2016b) as our decoder’s backbone, and replace its Normal distribution vector as the output of encoder $z$. A character level CNN-RNN model (Reed et al., 2016a) is used for the text embedding which produces the 1,024-dimension vector given the description, and then projected to a 128-dimension code $c$. We configure the encoder and decoder with four convolutional layers and the latent vector $z$ is a 100-dimension vector. The input of the discriminator is the concatenation of $z$ and $c$, which results in a 228-dimension vector. A two-layer fully connected network with sigmoid output unit is used as the discriminator. Table 7 summarizes the performance of DVN in image caption task and can be regarded as a powerful baseline.

5. Conclusion and Future Work

In this paper, we propose to enhance the performance of anomaly detection by verifying predictions of deep discriminative models using deep generative models. The idea is to train a conditional verifier network $q(x|y)$ as an approximation to the inverse posterior distribution. We propose Deep Verifier Networks (DVNs) which are based on a modified conditional variational auto-encoders with disentanglement constraints. We show our model is able to achieve state-of-the-art performance on benchmark OOD detection and adversarial example detection tasks.

For future work, it would be interesting to integrate DVNs to safe AI systems. For instance, ordinary image classifiers such as DenseNet have perfect accuracy for in-distribution queries, but their behaviors are undefined on adversarial queries. Robust image classifiers sacrifice some accuracy for robustness to adversarial examples. We can use DVNs to form a two-step prediction procedure: first using an or-
dinary classifier to get an initial prediction, then verify the prediction with DVN. If the prediction does not pass the verification, we switch to the robust image classifier. We believe our method would provide some advantages for safe AI systems.

References


Deep Verifier Networks: Verification of Deep Discriminative Models with Deep Generative Models

computer vision and pattern recognition, pp. 2574–2582, 2016.


