Unsupervised Anomaly Detection by Robust Density Estimation

Boyang Liu, Pang-Ning Tan, Jiayu Zhou

Department of Computer Science and Engineering, Michigan State University
liuboya2, ptan, jiayuz@msu.edu

Abstract

Density estimation is a widely used method for unsupervised anomaly detection. However, the presence of anomalies in training data may severely impact the density estimation process, thereby hampering the use of more sophisticated density estimation methods such as those based on deep neural networks. In this work, we propose RobustRealNVP, a robust deep density estimation framework for unsupervised anomaly detection. Our approach differs from existing flow-based models from two perspectives. First, RobustRealNVP discards data points with low estimated densities during optimization to prevent them from corrupting the density estimation process. Furthermore, it imposes Lipschitz regularization to ensure smoothness in the estimated density function. We demonstrate the robustness of our algorithm against anomalies in training data from both theoretical and empirical perspectives. The results show that our algorithm outperforms state-of-the-art unsupervised anomaly detection methods.

Introduction

Anomaly detection (AD) is the task of finding unusual observations, whose characteristics are considerably different from the majority of the data. Applications of AD can be found in diverse domains, including cybersecurity, finance, and healthcare. While there have been extensive methods developed for AD, density-based methods such as kernel density estimation, local outlier factor (LOF), and their variants have found success due to their simplicity, ease of use, and ability to detect diverse types of anomalies under various settings (Breunig et al. 2000). However, despite their success, current density-based methods are mostly designed to operate in the original feature space, which hampers their applicability to high-dimensional data, as density estimation has high sample complexity and computational costs with increasing number of dimensions (Tsybakov 2008). Moreover, if the data lies in some low-dimensional, nonlinear manifold, performing density estimation and anomaly detection in the original feature space may lead to inferior performance.

With the emergence of deep learning, deep generative models have been widely used to perform density estimation in complex, high-dimensional datasets. For example, flow-based models (Dinh, Sohl-Dickstein, and Bengio 2016; Dinh, Krueger, and Bengio 2014) learn an invertible function to facilitate exact inference and sampling of data points for density estimation. By carefully designing the network structure, it can successfully learn the underlying data distribution. Unlike other deep generative models such as generative adversarial networks (Goodfellow et al. 2014) and variational autoencoders (Kingma and Welling 2013), flow-based models explicitly learn the density function, which makes it a powerful tool for inferring out-of-distribution data.

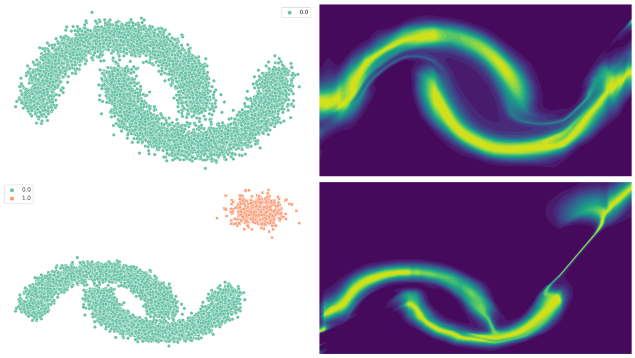

Figure 1: Effect of anomalies on density estimation for flow-based models. The upper left figure represents clean data while the lower left figure represents data contaminated by anomalies (red points). The right column shows the density functions estimated by Real NVP. Observe that Real NVP assigns large density values to the clustered anomalies.

However, due to the high capacity of the DNNs, the density function estimated by the flow-based model can be corrupted by anomalies. To illustrate this, Figure 1 shows a toy example in which the normal data lies in a two-moon manifold. When the training data is clean, the flow-based model accurately captures the true density function, as shown in the upper right figure. However, when training data is contaminated with anomalies, the flow-based model will attempt to learn the density of the anomalies as well, as shown in the lower right figure. Thus, a more robust approach is needed to estimate the density function in order to enhance the performance of flow-based models in unsupervised AD tasks.

This paper presents a deep unsupervised AD method called RobustRealNVP. Unlike Real NVP (Dinh, Sohl-Dickstein, and Bengio 2016), which is a popular flow-based

model, our framework ignores low-density points when learning the density function. We provide theoretical analysis to show that the robustness of RobustRealNVP depends on smoothness of its estimated density function. This result is significant as the original Real NVP formulation has no performance guarantees on contaminated data. We use Lipschitz regularization to enforce smoothness of the estimated density function. To the best of our best knowledge, none of the previous works have employed Lipschitz regularization in flow-based models or for density estimation.

In summary, our major contributions are as follows:

- We propose a robust flow-based deep density estimation method for unsupervised anomaly detection.
- We present theoretical analysis to demonstrate its robustness to contaminated training data. Specifically, we show its convergence to an ϵ-approximated stationary point of Real NVP trained on clean (anomaly-free) data.
- We perform extensive experiments to show that our framework outperforms various state-of-the-art methods.

Related Work

Density-based methods are widely used in unsupervised AD (Chandola, Banerjee, and Kumar 2009). They typically employ non-parametric methods such as k-nearest-neighbor and kernel density estimation (Tsybakov 2008) to estimate the density function but are susceptible to anomalies. (Humbert et al. 2020) proposed an approach that provides robustness guarantees of kernel density estimation by using the median of the means to replace the empirical mean of kernel sums. Another approach is to replace the ℓ_2-loss of the estimation with a Huber loss (Kim and Scott 2012). However, these approaches are not scalable to high-dimensional data.

The success of deep learning has inspired the development of deep models such as generative adversarial networks (GAN) (Goodfellow et al. 2014), variational autoencoders (VAE) (Kingma and Welling 2013), and flow-based models (Dinh, Krueger, and Bengio 2014) for density estimation. Unlike GAN and VAE, flow-based models explicitly learn the density function and directly optimize the likelihood function using the change of variable formula. Many methods are proposed to improve the efficiency and expressive power of flow-based models (Chen et al. 2019; Kingma and Dhariwal 2018). While these methods can be applied to detect out-of-distribution data (Zisselman and Tamar 2020), they mostly assume the availability of clean training data.

Robust optimization aims to optimize an objective function assuming a small fraction of the data is corrupted (Huber 1992). However, previous works mostly focused on linear models (Bhatia, Jain, and Kar 2015; Bhatia et al. 2017; Shen and Sanghavi 2019) or are limited to convex optimization problems (Prasad et al. 2018). SEVER (Diakonikolas et al. 2019) is a highly generalizable optimization algorithm with agnostic corruption guarantees. However, it cannot be efficiently applied to DNNs due to its high space complexity.

Lipschitz regularization can be used to optimize a loss function to ensure that the learned function is Lipschitz continuous (smooth). It has been used to regularize the discriminator in GAN (Gulrajani et al. 2017; Miyato et al. 2018; Qin, Mitra, and Wonka 2020) and to defend against attacks in adversarial learning (Hein and Andriushchenko 2017). One way to constrain a function to be Lipschitz continuous is to penalize its gradient norm (Gulrajani et al. 2017). An alternative method is to use adversarial training for Lipschitz regularization (Terjék 2019), but the method is inefficient since it requires finding adversarial samples for every data point. Another approach, specifically designed for neural networks, is spectral normalization, which constrains the Lipschitz constant in a layer-wise fashion (Miyato et al. 2018) in order to make the function smooth.

Preliminaries

We first review the flow-based model before introducing our proposed framework. Given n samples $\{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$, the flow-based model aims to learn an invertible mapping from a simple distribution such as Gaussian to a more complex target distribution by maximizing the likelihood function of the observed data, $\max_\theta p_\theta(\mathbf{X})$, where θ is the network parametesr. Let $p(z)$ be some simple density function and $z = f(x)$, where f is a bijective transformation function. According to the change of variable formula:

$$p_X(x) = p_Z(f(x)) \left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|,$$

where $\left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|$ is the determinant of the Jacobian matrix of f at x. By defining $g = f^{-1}$, we have the following sample generation process. First, we generate z from $p(z)$ and use the sample to generate $x = g(z)$. This requires a network architecture that satisfies the following two criteria. First, it must be easy to calculate the determinant to ensure backpropagation can be applied to optimize the network. Second, f must be invertible, so that f^{-1} can be used to generate the data x. One such transformation is the affine coupling transformation proposed in Real NVP (Dinh, Sohl-Dickstein, and Bengio 2016). Given a D-dimensional input x and $d < D$, the output y is defined as:

$$\begin{aligned} y_{1:d} &= x_{1:d} \\ y_{d+1:D} &= x_{d+1:D} \odot \exp\left[s\left(x_{1:d}\right)\right] + t\left(x_{1:d}\right), \end{aligned} \quad (1)$$

where s and t are functions that map $\mathbb{R}^D \rightarrow \mathbb{R}^{D-d}$ and can be implemented using DNNs. The transformation is invertible and its Jacobian is a triangular matrix, whose determinant can be easily computed by multiplying the diagonal elements of the matrix. Let $\mathcal{N}$ be the set of normal observations. The goal is to maximize the following objective:

$$\begin{aligned} &\max_f \log\left(\prod\nolimits_{x\in\mathcal{N}} p_Z(f(x)) \left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|\right) \\ &= \max_f \sum\nolimits_{x\in\mathcal{N}} \log(p_Z(f(x)) + \log\left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|, \end{aligned}$$

which is equivalent to minimizing the following loss:

$$\min_f -\left[\sum\nolimits_{x\in\mathcal{N}} \log(p_Z(f(x)) + \log\left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|\right].$$

After learning f, the density of x can be inferred as follows:

$$\tilde{p}(x) = p_Z\left(f(x)\right)\left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|.$$

Methodology

Our goal is to develop a robust density-based unsupervised AD approach amenable to training data that may contain anomalies. In this setting, our objective function is:

$$-\min_{f} \sum\nolimits_{x\in\mathcal{N}\cup\mathcal{A}} \log(p_Z(f(x)) + \log\left(\left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|\right), \tag{2}$$

where $\mathcal{A}$ is the set of anomalies. Optimizing (2) directly may lead to poor results due to the adverse impact of anomalies.

Robustness for Density Estimation

Let $\epsilon \in [0, 0.5]$ be the anomaly ratio, i.e., proportion of anomalies in training data. Given a set of normal instances $\mathcal{N}$ from a density function p and a set of anomalies $\mathcal{A}$ from an arbitrary density function, our goal is to learn $\Psi : \mathcal{N} \cup \mathcal{A} \rightarrow \hat{p}$, such that the gradient norm of the negative log-likelihood for $\mathcal{N}$ is minimized. If $\hat{p}$ is parameterized by θ, then the learning objective is:

$$\min_{\theta} \left\| -\sum\nolimits_{x\in\mathcal{N}} \nabla_\theta \log \hat{p}_\theta(x) \right\|.$$

Optimizing the objective function is challenging as we have no prior knowledge which instances belong to $\mathcal{N}$ or $\mathcal{A}$.

We propose a robust gradient estimation approach to address this problem. Consider a gradient descent approach for minimizing the negative log-likelihood function. Given a data point, x_i, its corresponding gradient is given by

$$\mathbf{g}_i = -\nabla_\theta \log \hat{p}_\theta(\mathbf{x}_i) = -\frac{1}{\hat{p}_\theta(\mathbf{x}_i)} \frac{\partial \hat{p}(\mathbf{x}_i)}{\partial \theta}.$$

Given n data points, let $\mathbf{G} = [g_1^T, g_2^T ... g_n^T] \in \mathbb{R}^{n\times d}$ be the contaminated gradient matrix and $\Gamma : \mathbb{R}^{n\times d} \rightarrow \mathbb{R}^d$ be an aggregation function. The model parameters can be updated using gradient descent as follows:

$$\theta^{(t+1)} = \theta^{(t)} - \eta^{(t)} \Gamma(\mathbf{G}).$$

If $\Gamma(\cdot)$ is the empirical mean, then the update formula reduces to standard gradient descent. However, since $\mathbf{G}$ contains gradients from both normal data and anomalies, $\Gamma(G)$ cannot be guaranteed to be close to the gradient of the clean data. Thus, our goal here is to design a robust function Γ that optimizes the following objective to ensure that the estimated density is not corrupted by anomalies:

$$\min_\Gamma \|\Gamma(G) - \mu\|^2,$$

where $\mu = \frac{1}{|\mathcal{N}|} \sum_{i\in\mathcal{N}} \mathbf{g}_i$ is the gradient of clean data.

The question is: If there exists a small constant ζ such that the gradient difference is upper bounded by ζ in every gradient descent step, i.e., $\forall t : \|\Gamma(G)^{(t)} - \mu^{(t)}\|^2 \leq \zeta$, can we guarantee that the stationary point will have a bounded gradient norm? The following proposition gives such a guarantee on the bounded gradient norm upon convergence.

Proposition 1 (Convergence of Biased SGD) *Let ϕ be the objective function and θ be the variable to be optimized. Under mild Lipschitz assumptions (Diakonikolas et al. 2019; Ajalloeian and Stich 2020), denote ζ as the maximum l_2 norm of the difference between the clean mini-batch gradient μ and corrupted mini-batch gradient, $\Gamma(\mathbf{G})$: $\|\mu - \Gamma(\mathbf{G})\| \leq \zeta$. By using the biased gradient estimation $\Gamma(\mathbf{G})$, SGD converges to the ζ-approximated stationary points: $\mathbb{E}\left(|\nabla\phi(\theta_t)|\right) = \mathbb{O}(\zeta)$.*

Proposition 1 comes from a series of previous work (Fan et al. 2020; Ajalloeian and Stich 2020; Bernstein et al. 2018; Hu et al. 2020; Diakonikolas et al. 2019), and has been widely studied in optimization community. For the sake of completeness, we include the above proposition here to show that it is enough to design a robust mean estimation method to guarantee that the final solution will have small gradient norm in terms of its clean (anomaly-free) objective. The proof of this proposition can be found in the Appendix.

Algorithm 1: Robust Gradient for Density Estimation

Input: corrupted gradient matrix $\mathbf{G} \in \mathbb{R}^{n\times d}$, where n = $|\mathcal{A}|+|\mathcal{N}|$, anomaly ratio ϵ, current parameter estimate, $\theta^{(t)}$
Output: Estimated mean of clean gradient, $\hat{\mu}^{(t)} \in \mathbb{R}^d$
1. For each row $\mathbf{g}_i$ in $\mathbf{G}$, calculate its predicted density, $\hat{p}_{\theta^{(t)}}(\mathbf{x}_i)$
2. Choose the ϵ-fraction rows in $\mathbf{G}$ with smallest $\hat{p}_{\theta^{(t)}}(\mathbf{x}_i)$
3. Remove the selected rows from $\mathbf{G}$
4. Return the empirical mean of the remaining rows as $\hat{\mu}^{(t)}$.

Robust Gradient Method for Density Estimation

Our robust aggregation function for $\Gamma(G)$ works as follows. Given a mini-batch $\mathcal{B}$ of size m, we sort the individual losses of instances in $\mathcal{B}$ and discard the instances with the top-k largest loss. We will show that this strategy ensures robustness guarantee of our algorithm. Before going into details, we first introduce the following Lipschitz assumption.

Assumption 1 *The density function learned by the DNN, $\hat{p}(x)$, is $\hat{L}$-smooth respect to its parameter θ, i.e., $\forall\theta_{\mathbf{1}}, \theta_{\mathbf{2}}, \mathbf{x}$: $\|\hat{p}_{\theta_{\mathbf{1}}}(\mathbf{x}) - \hat{p}_{\theta_{\mathbf{2}}}(\mathbf{x})\| \leq \hat{L}\|\theta_{\mathbf{1}} - \theta_{\mathbf{2}}\|$.*

The above assumption ensures that the gradient norm is upper bound by $\hat{L}$. By applying the $\hat{L}$-smooth assumption on $\hat{p}$, we can bound the individual gradient norm as follows:

$$\|\mathbf{g}_i\| = \|-\nabla_\theta \log \hat{p}_\theta(\mathbf{x}_i)\| = \frac{1}{\hat{p}_\theta(\mathbf{x}_i)} \left\| \frac{\partial \hat{p}_\theta(\mathbf{x}_i)}{\partial \theta} \right\| \leq \frac{\hat{L}}{\hat{p}_\theta(\mathbf{x}_i)}.$$

Since the loss $-\log \hat{p}_\theta(x)$ monotonically decreases with increasing $\frac{1}{\hat{p}_\theta(x)}$, thus, at each iteration, our algorithm simply needs to sort the current estimated density $\hat{p}_{\theta^{(t)}}(x)$ of each data point and discards the ones with low densities. The mean gradient, $\hat{\mu}^{(t)}$, is then computed from the remaining data points, which will be used to perform the gradient descent update: $\theta^{(t+1)} = \theta^{(t)} - \eta^{(t)}\hat{\mu}^{(t)}$. This is summarized by the pseudocode given in Algorithm 1.

We will first show that Algorithm 1 has robustness guarantee by using the following lemma:

Lemma 1 (Mean Gradient Estimation Error) *Let $\mathbf{g}_i^{(t)}$ be the gradient of the i_{th} datum at iteration t, $\mathcal{N}$ be the clean*

data, and $\hat{\mu}^{(t)}$ be the output of Algorithm 1 at iteration t. The following guarantee holds on the gradient estimation error:

$$\left\| \frac{1}{|\mathcal{N}|} \sum\nolimits_{\mathcal{N}} \mathbf{g}_i^{(t)} - \hat{\mu}^{(t)} \right\|^2 = \mathbb{O}(\epsilon)\hat{L}.$$

For brevity, we only provide a sketch of the proof for the lemma. A more detailed proof is given in the Appendix. Suppose $|\mathbf{g}|_{i\in\mathcal{N}} \leq \frac{1}{\hat{p}_i}\hat{L}$, and let $\max_{i\in\mathcal{N}} \frac{1}{\hat{p}_i} = C$. Since we discard an ϵ-fraction of the data with small $\hat{p}_i$, there are two possibilities: either all the training anomalies are discarded or some training anomalies still remain as their $|g_i|$ values are less than $C\hat{L}$. In both cases, our approach is to apply empirical mean estimation to the remaining gradient vectors, which has limited magnitude (i.e., bounded by $C\hat{L}$). When $\hat{L}$ is small, since the magnitude of the remaining gradients (which may include anomalies) are limited, the impact of each remaining anomaly on the mean gradient estimation is limited as well, not exceeding $C\hat{L}$. In the worst case, if no anomalies are discarded, this will introduce $\mathbb{O}(\epsilon)$ error. Putting them together, we obtain the above error rate of $\mathbb{O}(\epsilon)\hat{L}$, assuming C is a constant.

The above lemma suggests that the gradient estimation error is controlled by the Lipschitz constant when ϵ is small. Combined with Proposition 1, if we can control the Lipschitz constant to be small (i.e. having a smooth loss surface), then the presence of anomalies in training data will not significantly alter the density estimation of clean data.

The preceding analysis suggests that the following two assumptions must hold for the gradient estimation to be robust. First, $\hat{p}_{i\in\mathcal{N}}$ must be large enough to ensure C is small. This is a reasonable assumption as the data contain mostly normal observations. Second, the predicted density function should be smooth enough to avoid the ill-effects of anomalies. If the anomalies are sparse, i.e., not clustered, then including them in the training data has little effect on the estimated density since the density of the anomalies is small. However, clustered anomalies can systematically bias the density estimation as they have relatively high densities. To avoid the effects of such concentrated anomalies, we propose a spectral normalization approach to smooth the learned density.

Spectral Normalization

The spectral normalization (SN) method (Miyato et al. 2018) is motivated by the following inequality $\|g_1 \circ g_2\|_{\text{Lip}} \leq \|g_1\|_{\text{Lip}} \cdot \|g_2\|_{\text{Lip}}$, where $\|.\|_{\text{Lip}}$ represents the Lipschitz constant. A neural network can be treated as a composition of a series of linear transformation and activation functions. Since most activation functions are Lipschitz continuous (e.g., the Lipschitz constant of RELU function is 1), if the Lipschitz constant at every linear layer is bounded, then it is also bounded for the entire network.

For a linear function $\phi : \mathbf{h} \rightarrow \mathbf{Wh}$, where $\mathbf{W}$ is a linear transformation matrix, its Lipschitz constant is upper bounded by the operator norm of the matrix $\mathbf{W}$, i.e., $\|\mathbf{W}\|_{op}$. As a result, the Lispchitz function for the entire network is upper bounded by $\prod_{i=1}^{v} \|\mathbf{W}_i\|_{op}$, where v is the number of layers (for RELU networks). In the original paper of SN, each layer is normalized as follows: $\mathbf{W}_{SN} = \frac{\mathbf{W}}{\|\mathbf{W}\|_{op}}$ to ensure that the function is 1-Lipschitz. However, a 1-Lipschitz may over-smooth the function and lead to underfitting of the density function. In this work, we propose the following SN to satisfy the k-Lipschitz constraint: $\mathbf{W}_{SN} = \frac{\mathbf{W}}{(\|\mathbf{W}\|_{op}/k)}$. Similar to (Miyato et al. 2018), we approximate the operator norm using power iterations in our experiments.

Lipschitz Regularization for Flow-based Model

SN assumes that a DNN can be decomposed into a series of linear transformation and activation functions. However, for flow-based models, the network is no longer a simple decomposition of linear transformation and activation function. While SN can be used to guarantee the learned functions s and t in Equation (1) are k-smooth, there has yet been any proof showing the entire transformation is k-smooth. In this section, we show that constraining s and t is sufficient to achieve the k-smoothness. Our rationale for this is based on the following two lemmas on spectral normalization:

Lemma 2 *If $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a differentiable function everywhere with a Jacobian matrix $\mathbf{J}$, then $\|\mathbf{J}\|_{op} \leq L$ if and only if f is L-smooth.*

Lemma 3 $\|[\mathbf{A}, \mathbf{B}]\|_{op} \leq \|\mathbf{A}\|_{op} + \|\mathbf{B}\|_{op}$, *where $[\cdot, \cdot]$ denotes matrix concatenation.*

We analyze the applicability of SN to the transformation given in Equation (1) by using the preceding two lemmas. The Jacobian of the transformation is as follows:

$$\mathbf{J} = \frac{\partial y}{\partial x^T} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & \text{diag}\left(\exp\left[s\left(x_{1:d}\right)\right]\right) \end{bmatrix}.$$

The matrix can be further decomposed into the sum of a diagonal and an off-diagonal matrix. Thus

$$|\mathbf{J}|_{op} \leq \left| \begin{bmatrix} \mathbb{I}_d & 0 \\ 0 & \text{diag}\left(\exp\left[s\left(x_{1:d}\right)\right]\right) \end{bmatrix} \right|_{op} + \left| \begin{bmatrix} 0 & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & 0 \end{bmatrix} \right|_{op}$$
$$= \max(1, \max_{i\in[1,2,\ldots,D-d]} \exp[s(x_{1:d})]_i) + \left| \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} \right|_{op}.$$

Note that the first term is well bounded if the output of function s is well bounded. In our experiments, we use the tanh function as the last layer of function s. Thus, the first term will be bounded by $\exp(1)$. For the second term, according to Equation (1), we have

$$\left|x_{d+1:D} \odot \exp\left(s\left(x_{1:d}\right)\right) + t\left(x_{1:d}\right)\right|_{\text{Lip}}$$
$$\leq \left|x_{d+1:D} \odot \exp\left(s\left(x_{1:d}\right)\right)\right|_{\text{Lip}} + \left|t\left(x_{1:d}\right)\right|_{\text{Lip}}.$$

Since s and t are bounded by a Lipschitz constant using SN, assuming $\|x\|$ is bounded by B and the last layer of s is a tanh function, then, according to the inequality $\|g_1 \circ g_2\|_{\text{Lip}} \leq \|g_1\|_{\text{Lip}} \cdot \|g_2\|_{\text{Lip}}$, it is easy to see that the transformation $y_{d+1:D} = x_{d+1:D} \odot \exp\left(s\left(x_{1:d}\right)\right) + t\left(x_{1:d}\right)$ has a bounded Lipschitz constant $B * \exp(1) * |s|_{\text{Lip}} + |t|_{\text{Lip}}$. Thus the operator norm $\left| \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} \right|_{op}$ is also bounded. Finally,

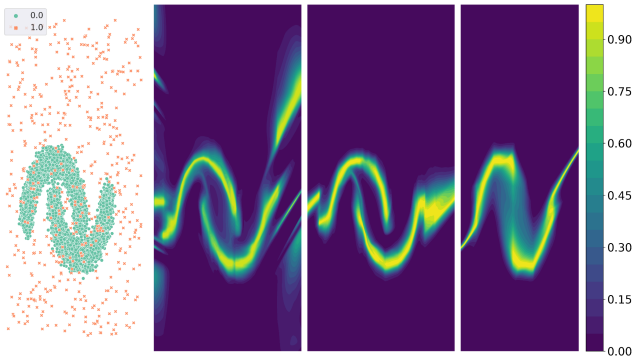

Figure 2: Effect of uniform anomalies (red points) on density estimation. The plots show the true data distribution, followed by the estimated density functions of Real NVP, TrimRealNVP, and RobustRealNVP (from left to right).

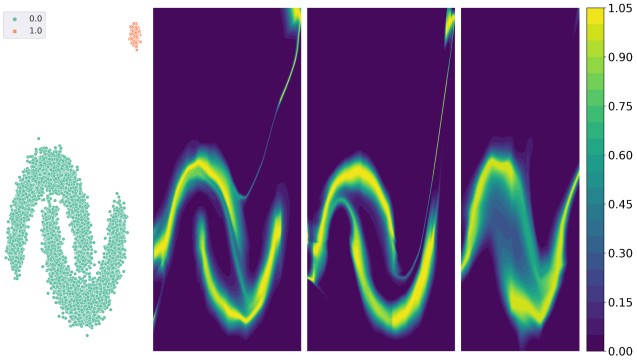

Figure 3: Effect of clustered anomalies (red points) on density estimation. The plots show the true data distribution, followed by the estimated density functions of Real NVP, TrimRealNVP, and RobustRealNVP (from left to right).

by Lemma 2, we conclude that application of SN to the functions s and t will guarantee that the flow model is Lipschitz regularized, which in turn, guarantees that the learned density function are not affected by clustered anomalies.

Anomaly Detection from Trained Network

By discarding data points with low estimated densities during training and adding spectral normalization to the flow-based model, a robust density estimation function can be trained. During testing, the density for each test point can be inferred by applying the trained network. The anomaly score for each test point is given by its estimated density, in which the higher the density, the lower is its anomaly score.

Experimental Evaluation

This section presents the empirical studies to validate the effectiveness of our proposed approach.

Experiments on Synthetic Data

Lemma 1 suggests that the robustness of the estimated density function is influenced by smoothness of the prediction function, which depends on the distribution of anomalies. If the anomalies are uniformly distributed, then spectral normalization (SN) is not necessary. However, if the anomalies are clustered, then SN is needed to provide robustness guarantees. To validate this, we generate a synthetic data, where the normal observations form a two-moon manifold structure, while the anomalies are generated under the following two settings: (1) **Uniform**, where the anomalies are randomly generated from a uniform distribution. The Lipschitz constant for the density function of anomalies is 0. (2) **Clustered**, where the anomalies are generated from a Gaussian distribution. The Lipschitz constant of the density function for anomalies is large. In both settings, we set the anomaly ratio $\epsilon = 0.05$. We compared the performance of our algorithm, **RobustRealNVP**, against **Real NVP** (Dinh, Sohl-Dickstein, and Bengio 2016). We also investigated a variation of our framework without Lipschitz regularization via SN and termed this approach as **TrimRealNVP**. We expect the performance of Real NVP to be severely hampered by the presence of anomalies in both settings. For anomalies that are uniformly distributed, we expect both TrimRealNVP and RobustRealNVP to perform equally well since they are both designed to alleviate the effect of anomalies during density estimation. For clustered anomalies, we expect RobustRealNVP to perform the best since TrimRealNVP does not guarantee smoothness of the predicted density function.

The results shown in Figure 2 are consistent with our expectation. While TrimRealNVP successfully alleviates the effect of uniform anomalies, it is largely affected by clustered anomalies. The estimated density functions of Real NVP are also impacted by both types of anomalies. RobustRealNVP can effectively handle both types of anomalies since it uses SN to increase smoothness of its loss landscape, which in turn, makes the gradient estimation more robust.

Experiments on Real-World Data

Datasets We perform experiments on two benchmark datasets: (1) Stony Brook ODDS library (Rayana 2016), which contains 16 benchmark outlier detection data. (2) CIFAR10, which is an image dataset with high-dimensional features. Since the results on CIFAR10 depend on the feature extraction step, for a fair comparison, we first apply the VGG19 network pre-trained on ImageNet and use the output before its final layer as the extracted features. We then apply PCA to reduce its dimensionality from 4096 to 128.

Baseline Methods We compare **RobustRealNVP** against the following baseline methods: (1) **AE** (autoencoder) and **VAE** (variational autoencoder). (2) **Deep-SVDD (Ruff et al. 2018)**, a deep 1-class SVM method. (3) **SO-GAAL (Liu et al. 2019)**, a GAN-based anomaly detection approach. (4) **OCSVM (Chen, Zhou, and Huang 2001)**, a shallow 1-class SVM method. (5) **LOF (Breunig et al. 2000)**, a local density-based AD method. (6) **IF (Liu, Ting, and Zhou 2008)** (isolation forest), an ensemble tree-based method.

Experiment Settings For ODDS dataset, we use 60% of the data for training and 40% for testing. For CIFAR10 dataset, 80% of the data are reserved for training while the remaining 20% for testing. CIFAR10 contains images from

	RobustRealNVP	Real NVP	AE	VAE	SO-GAAL	Deep-SVDD	OCSVM	LOF	IF
vowels	**0.959±0.010**	0.889±0.038	0.879±0.020	0.503±0.045	0.637±0.197	0.206±0.035	0.765±0.036	0.947±0.014	0.776±0.017
pima	**0.678±0.024**	0.652±0.031	0.669±0.013	0.648±0.015	0.613±0.049	0.395±0.034	0.594±0.026	0.610±0.034	0.661±0.020
letter	**0.930±0.020**	0.915±0.017	0.829±0.031	0.521±0.042	0.601±0.060	0.465±0.039	0.557±0.038	0.845±0.026	0.621±0.030
cardio	0.737±0.028	0.712±0.039	0.867±0.020	**0.944±0.006**	0.473±0.075	0.505±0.056	0.936±0.002	0.684±0.027	0.925±0.009
arrhythmia	0.786±0.039	0.780±0.038	0.802±0.044	**0.811±0.034**	0.538±0.042	0.635±0.063	0.782±0.028	0.777±0.026	0.799±0.023
musk	0.991±0.011	0.794±0.074	0.998±0.003	0.994±0.002	0.234±0.193	0.829±0.048	**1.000±0.000**	0.353±0.054	0.997±0.003
mnist	0.818±0.009	0.820±0.015	0.802±0.009	0.778±0.009	0.795±0.025	0.538±0.048	**0.835±0.012**	0.698±0.013	0.805±0.007
satimage-2	0.943±0.014	0.915±0.021	0.818±0.069	0.966±0.008	0.789±0.177	0.739±0.088	**0.998±0.003**	0.428±0.109	0.996±0.005
satellite	**0.709±0.010**	0.690±0.007	0.575±0.068	0.538±0.016	0.640±0.070	0.631±0.016	0.650±0.014	0.570±0.005	0.706±0.026
mammo	0.841±0.018	0.855±0.013	0.853±0.015	0.864±0.014	0.204±0.026	0.272±0.009	**0.881±0.015**	0.768±0.024	0.873±0.019
thyroid	0.961±0.011	0.956±0.005	0.928±0.020	0.839±0.011	**0.984±0.005**	0.704±0.027	0.960±0.006	0.898±0.017	0.979±0.006
annthyroid	**0.880±0.025**	0.864±0.024	0.675±0.022	0.589±0.021	0.679±0.022	0.591±0.014	0.599±0.013	0.711±0.022	0.829±0.015
ionosphere	0.900±0.020	**0.914±0.015**	0.821±0.010	0.763±0.015	0.783±0.080	0.735±0.053	0.812±0.039	0.879±0.022	0.842±0.021
pendigits	0.748±0.023	0.730±0.024	0.685±0.073	0.931±0.006	0.257±0.053	0.613±0.071	0.935±0.003	0.472±0.029	**0.943±0.013**
shuttle	0.995±0.003	0.825±0.032	0.921±0.013	0.987±0.001	0.571±0.316	0.531±0.290	0.985±0.001	0.529±0.017	**0.997±0.001**
glass	0.786±0.039	0.747±0.056	0.570±0.152	0.626±0.134	0.420±0.112	0.756±0.114	0.522±0.207	0.756±0.141	**0.706±0.058**
Avg rank	**2.937**	4.375	5	5.5	7.4375	8.34375	4.3125	6.6525	3.0625

Table 1: Average and standard deviation of AUC scores for the ODDS datasets (across 5 different random seeds). The last row corresponds to average rank of each method.

	RobustRealNVP	RealNVP	AE	VAE	SO-GAAL	Deep-SVDD	OCSVM	LOF	IF
airplane-u	**0.786±0.012**	0.784±0.012	0.582±0.010	0.583±0.012	0.534±0.036	0.554±0.031	0.582±0.007	0.743±0.008	0.582±0.010
auto-u	**0.818±0.028**	0.782±0.022	0.486±0.010	0.505±0.010	0.490±0.072	0.569±0.011	0.486±0.008	0.750±0.008	0.502±0.008
bird-u	**0.694±0.070**	0.691±0.062	0.556±0.007	0.562±0.005	0.484±0.015	0.512±0.118	0.565±0.010	0.633±0.015	0.552±0.016
cat-u	**0.772±0.077**	0.726±0.076	0.484±0.009	0.499±0.015	0.526±0.021	0.581±0.023	0.517±0.008	0.582±0.005	0.503±0.010
deer-u	**0.801±0.006**	0.781±0.033	0.604±0.010	0.612±0.010	0.431±0.072	0.619±0.023	0.611±0.007	0.723±0.009	0.597±0.013
dog-u	**0.760±0.044**	0.704±0.050	0.444±0.003	0.450±0.006	0.535±0.065	0.624±0.035	0.490±0.017	0.587±0.008	0.441±0.010
frog-u	**0.807±0.022**	0.797±0.007	0.599±0.016	0.592±0.015	0.532±0.061	0.523±0.034	0.607±0.008	0.821±0.006	0.587±0.007
horse-u	**0.786±0.013**	0.769±0.057	0.504±0.014	0.503±0.011	0.510±0.052	0.545±0.009	0.534±0.008	0.670±0.018	0.511±0.017
truck-u	0.770±0.033	**0.838±0.027**	0.597±0.010	0.598±0.011	0.477±0.110	0.633±0.031	0.605±0.018	0.778±0.006	0.603±0.008
ship-u	0.798±0.006	**0.802±0.009**	0.493±0.016	0.489±0.006	0.410±0.055	0.658±0.041	0.505±0.006	0.738±0.009	0.506±0.016
airplane-c	**0.752±0.018**	0.748±0.023	0.611±0.013	0.619±0.010	0.512±0.035	0.464±0.116	0.616±0.008	0.687±0.009	0.619±0.021
auto-c	**0.850±0.020**	0.843±0.014	0.469±0.016	0.466±0.007	0.447±0.054	0.496±0.113	0.448±0.012	0.748±0.017	0.469±0.017
bird-c	0.640±0.025	0.639±0.030	0.580±0.010	0.561±0.016	0.485±0.035	0.517±0.058	0.581±0.017	**0.644±0.006**	0.579±0.008
cat-c	0.462±0.017	0.451±0.027	0.427±0.008	0.417±0.005	**0.545±0.072**	0.489±0.059	0.449±0.004	0.450±0.005	0.426±0.015
deer-c	**0.825±0.019**	0.808±0.009	0.676±0.008	0.669±0.001	0.456±0.056	0.544±0.111	0.643±0.010	0.761±0.005	0.656±0.012
dog-c	0.500±0.021	0.487±0.023	0.410±0.012	0.404±0.005	**0.585±0.085**	0.481±0.055	0.429±0.006	0.416±0.017	0.418±0.016
frog-c	**0.826±0.015**	0.788±0.020	0.593±0.008	0.599±0.016	0.436±0.070	0.507±0.102	0.614±0.009	0.814±0.005	0.588±0.013
horse-c	**0.575±0.023**	0.550±0.025	0.462±0.005	0.460±0.003	0.512±0.114	0.451±0.090	0.490±0.002	0.558±0.013	0.475±0.022
truck-c	**0.657±0.030**	0.664±0.013	0.623±0.015	0.622±0.006	0.503±0.047	0.518±0.102	0.622±0.012	0.650±0.011	0.627±0.011
ship-c	**0.671±0.025**	0.653±0.023	0.450±0.012	0.446±0.013	0.492±0.064	0.515±0.132	0.464±0.002	0.598±0.005	0.472±0.003

Table 2: Average and standard deviation of AUC scores for the CIFAR10 dataset (across 5 different random seeds). The first column shows the category chosen as normal class. 'class'-c means the training anomalies are clustered whereas 'class'-u means the anomalies are uniformly sampled from other classes.

10 classes. We select one class as the normal class (with 5000 samples) and then create anomalies from the remaining classes in two ways: **i)** Anomalies are randomly sampled from other classes. **ii)** Anomalies are selected from one specific class. The training anomaly ratio is set to be 0.1. All the models are evaluated on a balanced CIFAR10 test set, containing 1000 samples for each class. All experiments are repeated with 5 different random seeds. For unsupervised AD, hyperparameter tuning is trickier as there is no clean validation data available. Instead, we fixed the network structure for all the methods to ensure a fair comparison. Details of the hyperparameters used are given in the Appendix. Results are reported using AUC score as our evaluation metric.

Results for ODDS data Table 1 shows the AUC scores for various methods on the ODDS datasets. While no method consistently outperforms all others given the diversity of the datasets, RobustRealNVP has the highest average rank compared to other baselines. IF also achieves highly competitive results. The baseline deep learning methods (i.e. SO-GAAL, Deep-SVDD) yield relatively poor results since most of them are designed for clean training data, which is unavailable in an unsupervised AD setting.

Results for CIFAR10 Table 2 shows the AUC scores for various methods when different categories of images are chosen as normal class. RobustRealNVP outperforms all the baseline methods in 15 out of 20 experiments. Similar to the

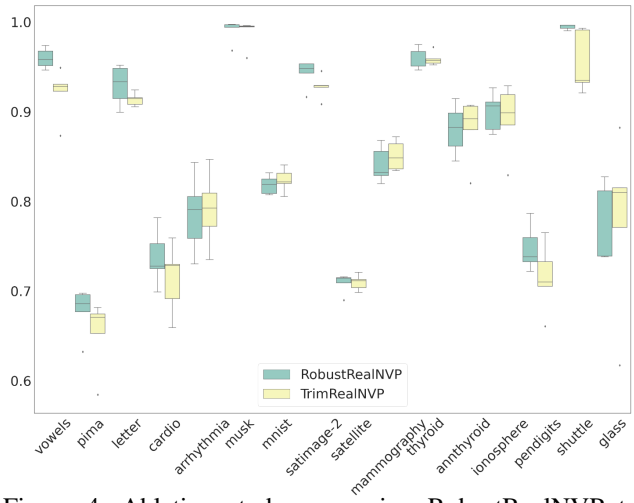

Figure 4: Ablation study comparing RobustRealNVP to TrimRealNVP, its variant without spectral normalization.

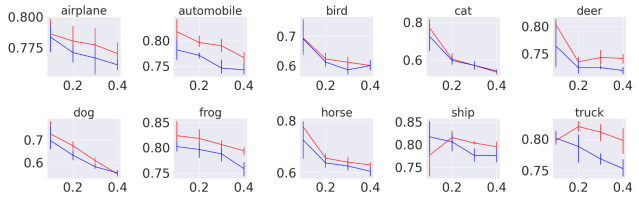

Figure 5: Sensitivity analysis when varying ϵ for uniform anomalies. Red line is RobustRealNVP, blue is Real NVP.

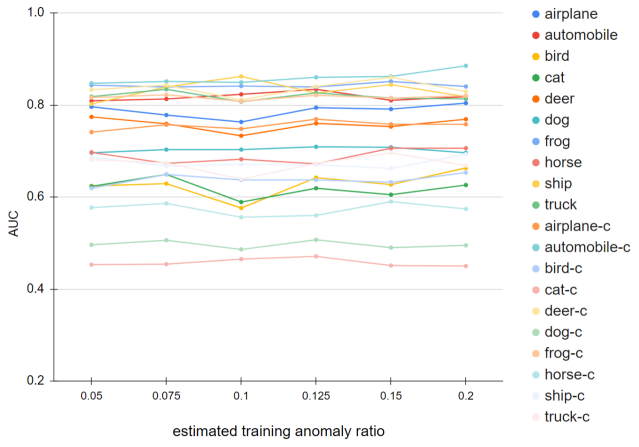

Figure 6: Sensitivity analysis when varying estimated ϵ from 0.05 to 0.2. The ground truth ϵ is 0.1.

ODDS results, the deep learning baselines do not perform well when trained on contaminated data. For Deep-SVDD, the AUC scores reported in Table 2 are slightly lower than the values reported in the original paper (Ruff et al. 2018) as our experiments were performed on contaminated instead of clean training data. Nevertheless, even if we use the results reported in the Deep-SVDD paper, which uses clean training data, RobustRealNVP still outperforms Deep-SVDD on all 10 CIFAR10 datasets with uniform anomalies. Furthermore, Deep-SVDD also performs poorly on data with concentrated anomalies.

Ablation Study for Spectral Normalization We perform ablation study using the ODDS dataset to show the effectiveness of spectral normalization (SN). In Figure 4, we see that RobustRealNVP outperforms TrimRealNVP in 8 out of the 16 datasets and tied in at least 4 of the remaining datasets. These results along with the synthetic experiment results demonstrated the effectiveness of SN in our framework.

Sensitivity Analysis In practice, the anomaly ratio ϵ is often unknown, which makes it hard to determine the number of data points to be discarded. To investigate the robustness of RobustRealNVP when ϵ is over- or underestimated, we perform sensitivity analysis on the CIFAR10 datasets by varying the estimated contamination ratio from 0.05 to 0.2 (note that the true anomaly ratio is 0.1). The results shown in Figure 6 suggest that our method remains stable under most settings even though the anomaly ratio was overestimated or underestimated by as high as 15%. In addition, we also perform sensitivity analysis on the CIFAR10 datasets when the true anomaly ratio is varied from 0.1 to 0.4. The results for Real NVP and RobustRealNVP shown in Figure 5 suggest that RobustRealNVP performs consistently better than the Real NVP in most settings.

Conclusions

This paper presents RobustRealNVP, a deep, density-based unsupervised anomaly detection method that is robust against training contamination. We demonstrate the effectiveness of our framework from both theoretical and empirical perspectives. For future work, we will investigate the applicability of the framework to other flow-based models.

Acknowledgments

This research was supported in part by the grant National Science Foundation IIS-1939368, IIS-1749940, Office of Naval Research N00014-20-1-2382, National Institute on Aging RF1AG072449. Any use of trade, firm or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

References

Ajalloeian, A.; and Stich, S. U. 2020. Analysis of SGD with Biased Gradient Estimators. *arXiv preprint arXiv:2008.00051*.

Bernstein, J.; Wang, Y.-X.; Azizzadenesheli, K.; and Anandkumar, A. 2018. signSGD: Compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*.

Bhatia, K.; Jain, P.; Kamalaruban, P.; and Kar, P. 2017. Consistent robust regression. In *Advances in Neural Information Processing Systems*, 2110–2119.

Bhatia, K.; Jain, P.; and Kar, P. 2015. Robust regression via hard thresholding. In *Advances in Neural Information Processing Systems*, 721–729.

Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; and Sander, J. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 93–104.

Chandola, V.; Banerjee, A.; and Kumar, V. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3): 1–58.

Chen, R. T.; Behrmann, J.; Duvenaud, D.; and Jacobsen, J.-H. 2019. Residual flows for invertible generative modeling. *arXiv preprint arXiv:1906.02735*.

Chen, Y.; Zhou, X. S.; and Huang, T. S. 2001. One-class SVM for learning in image retrieval. In *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*, volume 1, 34–37. IEEE.

Diakonikolas, I.; Kamath, G.; Kane, D.; Li, J.; Steinhardt, J.; and Stewart, A. 2019. Sever: A robust meta-algorithm for stochastic optimization. In *International Conference on Machine Learning*, 1596–1606.

Dinh, L.; Krueger, D.; and Bengio, Y. 2014. Nice: Nonlinear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

Fan, J.; Zhang, Q.; Zhu, J.; Zhang, M.; Yang, Z.; and Cao, H. 2020. Robust deep auto-encoding Gaussian process regression for unsupervised anomaly detection. *Neurocomputing*, 376: 180–190.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. 2017. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*.

Hein, M.; and Andriushchenko, M. 2017. Formal guarantees on the robustness of a classifier against adversarial manipulation. *arXiv preprint arXiv:1705.08475*.

Hu, Y.; Zhang, S.; Chen, X.; and He, N. 2020. Biased Stochastic Gradient Descent for Conditional Stochastic Optimization. *arXiv preprint arXiv:2002.10790*.

Huber, P. J. 1992. Robust estimation of a location parameter. In *Breakthroughs in statistics*, 492–518. Springer.

Humbert, P.; Bars, B. L.; Minvielle, L.; and Vayatis, N. 2020. Robust Kernel Density Estimation with Median-of-Means principle. *arXiv preprint arXiv:2006.16590*.

Kim, J.; and Scott, C. D. 2012. Robust kernel density estimation. *The Journal of Machine Learning Research*, 13(1): 2529–2565.

Kingma, D. P.; and Dhariwal, P. 2018. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Liu, F. T.; Ting, K. M.; and Zhou, Z.-H. 2008. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, 413–422. IEEE.

Liu, Y.; Li, Z.; Zhou, C.; Jiang, Y.; Sun, J.; Wang, M.; and He, X. 2019. Generative adversarial active learning for unsupervised outlier detection. *IEEE Transactions on Knowledge and Data Engineering*.

Miyato, T.; Kataoka, T.; Koyama, M.; and Yoshida, Y. 2018. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Prasad, A.; Suggala, A. S.; Balakrishnan, S.; and Ravikumar, P. 2018. Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485*.

Qin, Y.; Mitra, N.; and Wonka, P. 2020. How does lipschitz regularization influence GAN training? In *European Conference on Computer Vision*, 310–326. Springer.

Rayana, S. 2016. ODDS Library. Stony Brook University, Department of Computer Sciences, http://odds.cs.stonybrook.edu.

Ruff, L.; Vandermeulen, R.; Goernitz, N.; Deecke, L.; Siddiqui, S. A.; Binder, A.; Müller, E.; and Kloft, M. 2018. Deep one-class classification. In *International conference on machine learning*, 4393–4402.

Shen, Y.; and Sanghavi, S. 2019. Learning with bad training data via iterative trimmed loss minimization. In *International Conference on Machine Learning*, 5739–5748. PMLR.

Terjék, D. 2019. Adversarial Lipschitz Regularization. In *International Conference on Learning Representations*.

Tsybakov, A. B. 2008. *Introduction to nonparametric estimation*. Springer Science & Business Media.

Zisselman, E.; and Tamar, A. 2020. Deep residual flow for out of distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13994–14003.