Deep Residual Flow for Out of Distribution Detection

Ev Zisselman
Department of Electrical Engineering
Technion
ev_zis@campus.technion.ac.il

Aviv Tamar
Department of Electrical Engineering
Technion
avivt@technion.ac.il

Abstract

The effective application of neural networks in the real-world relies on proficiently detecting out-of-distribution examples. Contemporary methods seek to model the distribution of feature activations in the training data for adequately distinguishing abnormalities, and the state-of-the-art method uses Gaussian distribution models. In this work, we present a novel approach that improves upon the state-of-the-art by leveraging an expressive density model based on normalizing flows. We introduce the residual flow, *a novel flow architecture that learns the residual distribution from a base Gaussian distribution. Our model is general, and can be applied to any data that is approximately Gaussian. For out of distribution detection in image datasets, our approach provides a principled improvement over the state-of-the-art. Specifically, we demonstrate the effectiveness of our method in ResNet and DenseNet architectures trained on various image datasets. For example, on a ResNet trained on CIFAR-100 and evaluated on detection of out-of-distribution samples from the ImageNet dataset, holding the true positive rate (TPR) at* 95%, *we improve the true negative rate (TNR) from* 56.7% *(current state-of-the-art) to* 77.5% *(ours).*

1. Introduction

Deep neural networks (DNNs) are powerful models that achieve high performance in various tasks in computer vision [26], speech and audio recognition [21], and language processing [7]. Leading DNN architectures are known to generalize well and achieve impressive performance when evaluated on samples drawn from the distribution observed at the training phase [7, 19, 22, 26, 37]. However, DNNs tend to behave unexpectedly when encountering input taken from an unfamiliar distribution. In such instances, an out-of-distribution (OOD) input causes the majority of models to mispredict, often with high confidence [18, 27, 29, 31, 40]. This behaviour poses a severe concern about the reliability of predictions made by DNNs and

hinders their applicability to real-world scenarios [1].

Contemporary work aimed at predicting classification uncertainty adopt an approach of constructing a confidence score based on characteristics of the feature space of trained neural networks. In [20], Hendrycks and Gimpel propose a baseline method, which taps into features of the penultimate layer and uses the soft-max score as the confidence score. Their method is further improved by Liang et al. [28], who incorporate the soft-max score with temperature scaling, alongside input pre-processing that emphasizes the score difference between in- and out-of-distribution samples. The current state-of-the-art is the method of Lee et al. [27], which models the feature distribution in different layers of a trained network by a Gaussian distribution under the LDA assumption (i.e., different mean but same covariance for different classes), and forms a confidence score for each layer based on the posterior distribution of the LDA model, averaged over different layers. Lee's method shows superior performance compared with previous methods; in some cases surpassing by a large margin [27].

Building on the observation that a Gaussian model of network activations is an effective confidence measure, in this work we ask: can we improve OOD detection performance by using more expressive distributions of network activations? In particular, there is no reason to expect that features in mid-layers of the network follow an exact Gaussian distribution, and we expect that a more expressive model should capture their distribution more accurately.

We present a new approach for OOD detection and propose a more expressive density function, based on deep normalizing flow, for modeling the distribution of the feature space of trained neural networks. As a prelude, we posit that training a linear flow on the feature space of neural networks is equal to fitting a Gaussian distribution, as proposed in [27]. Then, we leverage this property to propose a novel flow architecture that adds a non-linear residual to the linear flow to produce a more expressive mapping. The residual flow model is of independent interest, and should be effective for any data that is approximately Gaussian distributed. For out-of-distribution de-

tection in image classification, modeling the network activations as a residual from Gaussian distribution allows us a principled improvement over the state-of-the-art, and in some cases yields superior performance by a large margin. Furthermore, the proposed residual flow model enables class-conditional density learning that improves performance, even in cases of limited training examples from each class (as in CIFAR100). Lastly, to make in- and out-of-distribution samples more separable, we extend the input preprocessing ideas of [28, 27] to our flow-based model, and perturb test samples to increase their likelihood under our model. We show that this perturbation can increase the contrast between in- and out-of-distribution samples, leading to further performance improvement.

We demonstrate the effectiveness of our method using trained convolution neural networks such as DenseNet [22] and ResNet [19], trained on various datasets, and tested on various out-of-distribution examples. Our method outperforms the state-of-the-art method [27] for detecting out-of-distribution samples in all tested cases. For example, for a ResNet trained on CIFAR-100, we improve the true negative rate (TNR) of detecting samples from the LSUN dataset at a true positive rate (TPR) of 95% (i.e. 95% of the CIFAR-100 test images were correctly classified) from 38.4% [27] to 70.4% (ours), with all hyper-parameters tuned strictly from the training dataset. Our results demonstrate that the feature space of neural networks does not necessarily conform with a Gaussian distribution, and a more accurate model can significantly improve confidence estimates.

2. Background

We present preliminaries on normalizing flows and OOD detection.

2.1. Normalizing Flows for Density Estimation

Normalizing flows are an effective model for high-dimensional data distributions, originally studied in classical statistics [41, 42], and recently popularized in the deep learning community (e.g., NICE [11], RealNVP [12], and GLOW [24]). Let $x \in X$ denote data sampled from an unknown distribution $x \sim p_X(x)$. The main idea in normalizing flows is to represent $p_X(x)$ as a transformation of a Gaussian distribution $z \sim p_Z(z) = \mathcal{N}(0, I)$, i.e. $x = g(z)$. Moreover, we assume the mapping to be bijective $x = g(z) = f^{-1}(z)$. As such, the data log-likelihood is given by the change of variable formula:

$$\log(p_X(x)) = \log(p_Z(f(x))) \qquad (1)$$
$$+ \log\left(\left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|\right),$$

where $\frac{\partial f(x)}{\partial x^T}$ is the Jacobian of the map $f(x)$ at x. The functions f, g can be learned by maximum likelihood, where the

bijectivity assumption allows to train expressive mappings, such as deep neural networks by backpropagation. Further, given a sample x, its likelihood can be inferred from (1).

To achieve a tractable, yet flexible Jacobian for the map $f(x)$, the authors of NICE [11] and RealNVP [12] proposed to stack a sequence of simple bijective transformations, such that their Jacobian is a triangular matrix. This way, its log-determinant is simply determined by the sum of its diagonal elements. In NICE [11], the authors proposed the *additive coupling layer* for each transformation. This was further improved in RealNVP [12] which proposed the *affine coupling layer*. In each affine coupling transformation, the input vector $x \in \mathbb{R}^d$ is split into upper and lower halves, $x_1, x_2 \in \mathbb{R}^{d/2}$. These are plugged into the following transformation, referred to as a single flow-block f_i:

$$z_1 = x_1, \quad z_2 = x_2 \circ \exp(s_i(x_1)) + t_i(x_1), \qquad (2)$$

where $\circ$ denotes element-wise multiplication, and s_i and t_i are non-linear mappings (e.g., deep neural networks) that need not be invertible. Given the output z_1 and z_2, this affine transformation is trivially invertible by:

$$x_1 = z_1, \quad x_2 = (z_2 - t_i(z_1)) \circ \exp(-s_i(z_1)).$$

Let r denote a switch-permutation, which permutes the order of x_1 and x_2. A RealNVP flow comprises k reversible flow-blocks interleaved with switch-permutations,[1]

$$f_{RealNVP} = f_k \cdot r \dots f_2 \cdot r \cdot f_1.$$

According to the chain rule, the log-determinant of the Jacobian of the whole transformation f is computed by summing the log-determinant of the Jacobian of each f_i, making the likelihood computation (1) tractable.

In GLOW [24], additional permutations between flow-blocks are added, to reduce the structural constraint of separating the input into two halves:

$$f_{GLOW} = f_k \cdot p_{k-1} \dots f_3 \cdot p_2 \cdot f_2 \cdot p_1 \cdot f_1,$$

where p_i are either fixed (random) or learned permutation matrices. Since permutations are easily inverted and $|\det(p_i)| = 1$, the log-likelihood (1) remains tractable.

2.2. Out of Distribution detection

Consider a deep neural network classifier trained in the standard supervised learning setting (via labeled data). The OOD detection problem seeks to assign a confidence score to the classifier predictions, such that classification of OOD data would be given a lower score than in-distribution data.

---

[1]The RealNVP paper [12] also considered other types of permutations, such as checkerboard masks for 2-dimensional image input. Here, we focus on 1-dimensional data, and only consider the switch-permutation, which was first proposed in [11].

Liang et al. [28] applied temperature-scaling to the network's soft-max output as the confidence score. Let $\sigma_i(x)$ denote the network's logit output for class i and input x. Then the temperature-scaled (TS) score is:

$$S_{TS}(x;T) = \max_i \left(\frac{\exp(\sigma_i(x)/T)}{\sum_{j=1}^N \exp(\sigma_j(x)/T)} \right),$$

where T is the temperature. In addition, Liang et al. [28] proposed to pre-process the input x by modifying it in a direction that increases the soft-max score:

$$\tilde{x}_{TS}(x) = x - \epsilon \cdot \text{sign}\left(-\nabla_x \log S_{TS}(x;T) \right),$$

where the intuition is that in-distribution samples would be more susceptible to an informative pre-processing, leading to better discrimination between in- and out-of-distribution samples. The final method, termed ODIN is given by:

$$S_{ODIN}(x;T) = S_{TS}(\tilde{x}_{TS}(x);T).$$

Lee et al. [27] improve on the ODIN method by considering different layers of the network, and measuring the Mahalanobis distance from the average network activations. For some network layer l and class label c, let $\phi_l(x)$ denote the feature activations at layer l for input x.[2] Let $\hat{\mu}_{l,c}$ denote the empirical mean of feature activations for training data from class c, and let $\hat{\Sigma}_l$ denote the empirical covariance matrix of feature activations, calculated across all classes. Given a test example x, Lee et al. [27] calculate the score as the weighted Mahalanobis distance:

$$S_M(x) = \sum_l w_l \cdot \max_c \{ -(\phi_l(x) - \hat{\mu}_{l,c})^T \hat{\Sigma}_l^{-1} (\phi_l(x) - \hat{\mu}_{l,c}) \},$$

where w_l are weights. Using the Mahalanobis distance as a score is equivalent to modeling the feature space of every layer as a C class-conditional Gaussian distribution with a tied covariance $\hat{\Sigma}$, i.e., $P(\phi_l(x)|y=c) = \mathcal{N}(\phi_l(x)|\hat{\mu}_{l,c}, \hat{\Sigma})$, and measuring the score as the likelihood of the features (under the most likely class, and averaging over all layers).

Lee et al. [27] motivate the Mahalanobis score from a connection between the softmax output of the final layer and a generative classifier with a class-conditional Gaussian distribution model with tied covariance. This generative model is a special case of Gaussian discriminant analysis (GDA), also known as linear discriminant analysis (LDA).

Lee et al. [27] also propose a pre-processing method similar to ODIN, where

$$\tilde{x}_M(x) = x - \epsilon \cdot \text{sign}\left(\nabla_x \left(\phi_l(x) - \hat{\mu}_{l,\hat{c}} \right)^T \hat{\Sigma}_l^{-1} (\phi_l(x) - \hat{\mu}_{l,\hat{c}}) \right).$$

[2]For a convolutional neural network, [27] propose to take the average activation across the spatial dimensions for each channel. In this work we adopt this approach, but our method can be applied without change to the actual feature activations.

3. Residual Flow for OOD Detection

Our aim is to detect out of distribution (OOD) examples, equipped with an already trained neural network classifier at our disposal. This is achieved by learning the distribution of the feature space of various layers of the network, given valid, in-distribution inputs that were observed during the training phase. Motivated by the empirical success of the Gaussian distribution model of Lee at al. [27], in this section we propose a normalizing flow architecture that allows for a principled extension of the Gaussian model to non-Gaussian distributions. We hypothesize that the activations of general neural network layers do not necessarily follow a Gaussian distribution, and thus a more expressive model should allow for better OOD detection performance. Our model is composed of a linear component, which we show is equivalent to a Gaussian model, and a non-linear residual component, which allows to fit more expressive distributions using deep neural network flow architecture.

3.1. Linear Flow Model

We start by establishing a simple relation between the maximum-likelihood estimate of a Gaussian model (as in GDA) and linear flow. The next proposition shows that for a linear flow model, the maximum likelihood parameters are equivalent to the empirical mean and covariance of the data.

Proposition 1. *Let $X = \{x_1, x_2, ..., x_N\}$ be a dataset of vectors in $\mathbb{R}^d$, i.e $\forall i : x_i \in \mathbb{R}^d$. Consider a linear normalizing flow, i.e $X = AZ + b$, where $Z \sim \mathcal{N}(0, I)$, $A \in \mathbb{R}^{d \times d}$ and $b \in \mathbb{R}^d$. Let $p_{A,b}(x_i)$ denote the probability of x_i under this flow model. The parameters A, b that maximize the likelihood of the dataset X under this model satisfy: $b = \frac{1}{N} \sum_{n=1}^N x_i = \hat{\mu}$, the empirical mean and $AA^T = \frac{1}{N} \sum_{n=1}^N (x_i - \hat{\mu})(x_i - \hat{\mu})^T = \hat{\Sigma}$, the empirical covariance of the data X.*

Proof. Since X is a linear transformation of $Z \sim \mathcal{N}(0, I)$, the probability of X under this model is given by:

$$p_{A,b}(x_i) \sim \mathcal{N}(b, AA^T). \tag{3}$$

On the other hand, the maximum likelihood (ML) estimators $\tilde{\mu}, \tilde{\Sigma}$ for X under Gaussian distribution assumption are known to be the empirical mean and covariance [13]:

$$\tilde{\mu} = \frac{1}{N} \sum_{n=1}^N x_i = \hat{\mu}, \ \ \tilde{\Sigma} = \frac{1}{N} \sum_{n=1}^N (x_i - \hat{\mu})(x_i - \hat{\mu})^T = \hat{\Sigma}. \tag{4}$$

By combining (3) and (4) we get the desired results. $\qquad\square$

The linear flow transformation A can be obtained analytically by exploiting the spectral decomposition for the symmetric positive semi-definite (PSD) matrix $\hat{\Sigma} = QDQ^T$, where Q is an orthogonal matrix whose columns are the

eigen-vectors of $\hat{\Sigma}$ and D is a diagonal matrix whose entries are its eigen-values. The resulting invertible linear flow transformation for data X can be written as:

$$X = AZ + b, \quad Z = A^{-1}(X - b),$$

where $b = \hat{\mu}$, $A = QD^{\frac{1}{2}}$, $A^{-1} = D^{-\frac{1}{2}}Q^T$.[3] In the sequel, we propose an extension of the linear flow that adds non-linear components, which we term a residual flow model.

3.2. Residual Flow Model

In this section, we describe how to extend the linear flow model to include non-linear components. Rather than directly using a fully non-linear model like RealNVP or GLOW, as described in Section 2, we would like a model that can be viewed and trained as an extension to the linear model. This approach will allow a principled improvement over the Gaussian model of Lee et al. [27], which we already know to perform well.

We begin by composing a linear flow with a residual flow model:

$$f^{res} = p_k \cdot f_k^{non-lin} \cdot p_{k-1} \dots p_2 \cdot f_2^{non-lin} p_1 \cdot f_1^{non-lin} \cdot A^{-1},$$

with the following log determinant:

$$\log\left(\left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|\right) = \log\left(\left|\det\left(A^{-1}\right)\right|\right)$$
$$+ \sum_i \log\left(\left|\det\left(\frac{\partial f_i^{non-lin}(x)}{\partial x^T}\right)\right|\right).$$
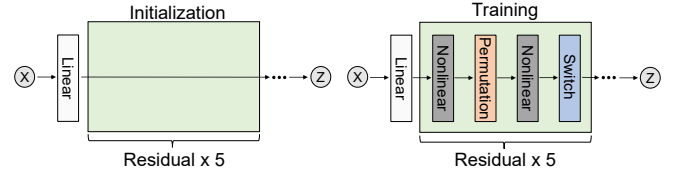
Note that, from Eq. (2), when s_i and t_i are set to zero, the non-linear terms $f_i^{non-lin}$ are reduced to the identity map. In this case, the permutation terms have no effect, as the components of z have identical and independent distributions. Thus, in this case, the residual flow f^{res} is equivalent to the linear flow $f^{lin} = A^{-1}$. Therefore, we can initialize the residual flow by fixing the networks s_i and t_i to be zero, and calculating A as described in Section 3.1, which is equivalent to fitting a Gaussian distribution model to our data. Subsequently, we can fine-tune the non-linear components in the model to obtain a better fit to the data. In practice, setting only the last layer of the networks s_i and t_i to zero is sufficient for the initialization step.[4]

Similar to the GLOW model [24], we found that the permutation terms p_i have an important contribution, by diversifying the inputs of the non-linear components. In our implementation, we alternate between fixed, initially random,[5]

(a) Residual Flow blocks during initialization and training.

(b) The complete Residual Flow architecture $Z = f(X)$.

Figure 1. Residual Flow architecture.

permutation matrices and switch permutation matrices to mediate the non-linear flow blocks. Concretely, p_i stands for a random permutation for odd i and switch permutation for even i. Figure 1 illustrates the proposed architecture, and the full implementation is described in Section 3.3.

3.2.1 Degenerate case

If the covariance matrix $\hat{\Sigma}$ is not full rank, then the multivariate normal distribution is degenerate: its vector elements are linearly dependent, and the covariance matrix does not correspond to a density over the d-dimensional space. In this case, Lee et al. [27] propose to use $\hat{\Sigma}^\dagger$, the pseudo-inverse of $\hat{\Sigma}$, to calculate the Mahalanobis distance:

$$-(X - \hat{\mu})^T \hat{\Sigma}^\dagger (X - \hat{\mu}),$$

which is equivalent to restricting attention to a subset of $k = rank(\hat{\Sigma})$ of the coordinates of X, such that the co-variance matrix of this subset is positive definite (PS); the remaining coordinates are regarded as an affine function of the selected coordinates. In our model we handle degenerate distributions with a similar approach: We set $Z = A^\dagger X$ to be a k-dimensional vector with a k-dimensional Gaussian distribution, using a dimensionality reduction transformation $A^\dagger \in \mathbb{R}^{k \times d}$.[6] We construct $A^\dagger = D^{-\frac{1}{2}}Q^T$ with $D^{-\frac{1}{2}} \in \mathbb{R}^{k \times k}$ and $Q^T \in \mathbb{R}^{k \times d}$, by considering the inverse root of the k non-zero eigen-values of $\hat{\Sigma}$ in $D^{-\frac{1}{2}}$ diagonal and their corresponding eigen-vectors in Q^T rows. Note that using $A^\dagger$ for degenerated vectors X yields the same Gaussian distribution as the pseudo-inverse used in [27]. In the rest of this paper we consider $A^\dagger$ as the linear flow transformation for degenerated vectors X. After this linear dimensionality reduction, we apply the residual flow model on the resulting k-dimensional vector Z as presented in Section 3.2. As a remark, the aforementioned treatment removes only linear dependencies among feature elements, and does not address non-linear dependencies. Practically,
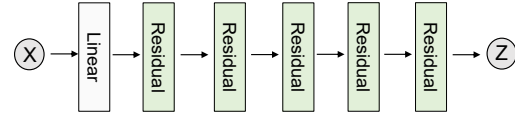
---

[3]To simplify notation, in the rest of this paper we assume that the empirical mean $\hat{\mu}$ is zero, achieved in practice by zero-centering the data.

[4]We found this to perform better in fine-tuning the non-linear terms, as most of the network is not initialized to zero and obtains large gradients in the initial training steps.

[5]The random permutation shuffles the preceding layer's input in a predetermined random order that remains consistent throughout training.

[6]Note that here $A^\dagger$ is not the inverse of A.

however, we found that this approach is sufficient for all the experiments we conducted.

Our residual flow model is a general normalizing flow architecture, and we expect it to work well when the data approximately fits a Gaussian distribution.

3.3. Residual Flow Applied to OOD Detection

We now describe an application of the residual flow that extends the Gaussian model of [27] for OOD detection. First, for each network layer l, we extract the mean activation in the training data for each class label $\mu_{l,c}$. Then, for each sample x in our training data, we extract the network activation in layer l, $\phi_l(x)$, and subtract from it the mean $\mu_{l,c}$ for the corresponding class, to obtain a centered feature training set $\hat{\phi}_l(x)$. Next, we fit a Gaussian distribution to the centered data by constructing a linear flow model for each layer as described in Section 3.1. We construct a single linear model *for all* classes, similar to the single covariance matrix in [27]. Finally, for each layer l, and for each class c, we train a residual flow model by training the non-linear flow blocks $f_i^{non-lin}$, as described in 3.2, and freeze the network weights in the linear block f_i^{lin}. As a stopping criteria for training the residual flow blocks, we use a separate validation set, and validate on the log-likelihood of the data. We found this approach to be effective for preventing overfitting in our experiments. This model, applied for OOD detection, already has good performance at the outset, leading to a better fit to the data distribution as training progresses.

Implementation details: We implement the model as a single linear flow block $f^{lin} = A^{-1}$, followed by 10 non-linear flow blocks $f^{non-lin}$, producing a map f^{res} totalling 11 flow blocks. As for the layers p_i, which interconnect the blocks $f^{non-lin}$, we alternate between switch and random permutation matrices. We use three fully connected layers per non-linear block (in each s_i and t_i) with leaky ReLU activation functions in the intermediate layers. We use a batch size of 256 and Adam [23] optimizer for learning the non-linear blocks with a learning rate of $10^{-5} - 10^{-6}$, chosen via a separate validation set of 10K examples.

3.4. Input pre-processing

Motivated by the success of input pre-processing in ODIN [28] and Mahalanobis [27], we propose an extension of this idea to our approach. Since the Mahalanobis pre-processing can be seen as maximizing the likelihood of the input under the Gaussian model, we similarly introduce the following input pre-processing stage for our flow model:

$$\tilde{x} = x + \epsilon \cdot \text{sign} \left(\nabla_x \log p(\phi_l(x); \hat{c}) \right), \quad (5)$$

where $\hat{c} = \arg\max_{c \in C} p(\phi_l(x); c)$ and $p(\phi_l(x); \hat{c})$ is the probability distribution of the feature space of the l-th layer of class $\hat{c}$, learned by our flow model. Note that this score aims to increase the probability of the in-distribution data.

Algorithm 1 Computing the Residual-Flow score S_l.

Input: Test sample x, weights of logistic regression detector α_l, noise ε and C residual-flow for each layer: $\{f_{l,c}^{res} : \forall l, c\}$

Initialize score vectors: $\mathbf{S}_{RF}(x) = [S_{l,c} : \forall l, c]$
for each layer $l \in 1, \ldots, L$ **do**
    Find the most probable class:
        $\hat{c} = \arg\max_c \ p_c(\phi_l(x) - \hat{\mu}_{l,c})$
    Add small noise to test sample:
        $\tilde{x} = x + \varepsilon \text{sign} \bigtriangledown_x p_{\hat{c}}(\phi_l(x) - \hat{\mu}_{l,\hat{c}})$
    Computing confidence score:
        $S_l = \max_c p_c (\phi_l(\tilde{x}) - \hat{\mu}_{l,c})$
end for
return Confidence score for test sample $\sum_l \alpha_l S_l$

3.5. OOD Detection Algorithm

In this section we describe the proposed procedure for OOD detection. Using the training set, we first train a collection of residual flows for each layer and each class $\{f_{l,c}^{res} : \forall l, c\}$ according to Section 3.2. Given a test example x, we extract the layers' activations for this example $\{\phi_l(x) : \forall l\}$, and calculate the most probable class for each layer $\hat{c}_l$. Using $\hat{c}_l$ we calculate the pre-processed input $\tilde{x}$, according to Eq. (5), and re-calculate the layers' activations $\{\phi_l(\tilde{x}) : \forall l\}$. The probability of the most probable class serves as a score of the layer $S_l = \max_c p_c (\phi_l(\tilde{x}) - \hat{\mu}_{l,c})$. Finally, the effective score is a weighted average of layers' scores $\sum_l \alpha_l S_l$. The weights are obtained using a similar strategy as in [27], where the weights of the layers α_l are computed by training a logistic regression detector on a validation set. The full algorithm is detailed in Algorithm 1.

3.6. Computational Overhead

It is important to evaluate the computational overhead of using a more expressive model for network activations. We compare our method to [27], and consider two cases: (i) During training: our initialization step is equivalent to the method of [27]. Thus, performance improvement comes at a cost of additional training time. Figure 2(b) shows the tradeoff between additional training iterations and performance gain. Note that the improvement monotonically increases with training iterations. (ii) During testing: In the test phase, both methods first calculate a forward pass of the test image through the classification network for feature extraction. Then, [27] calculates the Mahalanobis distance, while our method runs another forward pass of the residual flow networks. In our experiments, the forward pass of the classification network was the dominant complexity factor. This may change with a larger flow model, but in our experiments we did not require such. Thus, our performance advantage does not incur significant overhead.

4. Related Work

OOD detection has mostly been studied in the unlabelled setting, where the data contains only samples (e.g., images) but not class labels. Classical methods include one-class SVM [36] and support vector data description [43], and more recently, deep learning methods have become popular [5]. Methods such as [14, 3, 4, 6] extract features using unsupervised learning techniques, and feed them to classical OOD detection methods. Deep SVDD [33] learns a neural-network encoding that minimizes the volume of data around a predetermined point in feature space. Recently, Golan and El-Yaniv [17] proposed to learn features by applying a fixed set of geometric transformations to images, and training a deep network to classify which transformation was applied. Density estimation methods for detecting OOD examples have originally been studied in low dimensional space [32, 8, 16]. Recently, deep generative models such as generative adversarial networks, variational autoencoders, and deep energy-based models have been proposed for OOD detection in high-dimensional spaces [2, 39, 35, 44, 47, 38].

Our work focuses on the labelled setting, where a network trained for image classification is provided, along with the training data and labels. Hendrycks and Gimpel [20] proposes the soft-max output as a confidence score for OOD examples, and [15] compared this approach with the Monte-Carlo dropout ensemble method. Liang et al. [28] proposed ODIN, which combines temperature scaling and input pre-processing. The geometric transformations method of Golan and El-Yaniv [17] can also be applied to the labelled setting. The state-of-the-art is the method of Lee et al. [27] that uses the Mahalanobis distance in feature space. In our work we show that providing a better density model, leads to a marked improvement over Lee et al.'s results.

Concurrent with our work, several OpenReview postings suggested improvements to the method of [27]. Sastry et al. [34] propose a scoring function for OOD detection based on the correlation between different features of the same layer, using higher-order Gram matrices, which can be seen as a different form of incorporating higher-order statistics beyond the Gaussian model. Yu et al. [46] investigate the benefit of combining the global average of the feature maps with their spatial pattern information, while using the Gaussian model assumption. In principle, their approach can be combined with our improved flow-based density model.

5. Experiments

In our experiments, we aim to answer the following questions: (1) How does the residual flow model compare with conventional flow and Gaussian models? (2) How does our OOD detection method compare with state-of-the-art?

Our OOD detection evaluation follows the data sets and experiments in [27], and consists of 3 training data sets: CIFAR10, CIFAR100, and SVHN, and 4 out-of-distribution (OOD) data sets: CIFAR10, Tiny ImageNet, SVHN, and LSUN. In the supplementary material we provide additional experiments, which draw a comparison between residual flow, LDA (Mahalanobis) and the GDA model. The full residual flow implementation is available online.[7]

5.1. Residual Flow vs. Regular Flow

In this section we compare the performance of learning a residual flow model over learning regular non-linear flow model. First, we inspect the performance of the proposed approach on the task of distinguishing in- and out-of-distribution examples based on the first layer of ResNet, trained on CIFAR-100, where Tiny-ImageNet is used as OOD. In our comparison, we evaluate residual flow against regular non-linear flow and linear-flow/Mahalanobis density modeling. Figure 2(a) presents a receiver operating characteristic (ROC) curve [9] comparison of the three methods,[8] demonstrating the superiority of the residual flow model in modeling feature layer distribution of a neural network. Next, in Figure 2(b), we evaluate the area under the ROC (AUROC) curve as a function of training iterations. Note that the linear flow[9], as expected, converges to the same AUROC as the baseline Gaussian density model. The residual flow, however, starts at baseline performance (equivalent to the Gaussian model), and steadily improves upon it, as the non-linear components allow for better modelling of the data. The conventional non-linear flow, on the other hand, starts from a low AUROC score, rises erratically, and is not guaranteed to improve upon the baseline. The erratic behavior also makes it difficult to decide when to stop training. Indeed, we found this model to be much less stable in our evaluation.

5.2. OOD Detection Evaluation

We conduct a series of experiments to evaluate the performance in detecting out-of-distribution examples. These tests are used by contemporary state-of-the-art methods [20, 28, 27] to benchmark the efficacy of an algorithm in distinguishing abnormalities. We follow the practices presented in [27], in which already-trained neural networks are used in conjunction with conventional datasets. The experiments use DenseNet with 100 layers [22] and ResNet with 34 layers [19] as target networks, trained on one of the following datasets: CIFAR-10, CIFAR-100 [25] and SVHN [30]. Feature extraction is performed as proposed by Lee et al. [27]: At the outset, we extract the output of specific

---

[7]https://github.com/EvZissel/Residual-Flow

[8]Training the flow models throughout this paper (residual and regular) is conducted using a validation set of 10K samples that are portioned from the training set, and the stopping criterion is the overfit set-point at which the validation likelihood ceases to increase.

[9]The linear model is described in Supplementary material – Section 2.

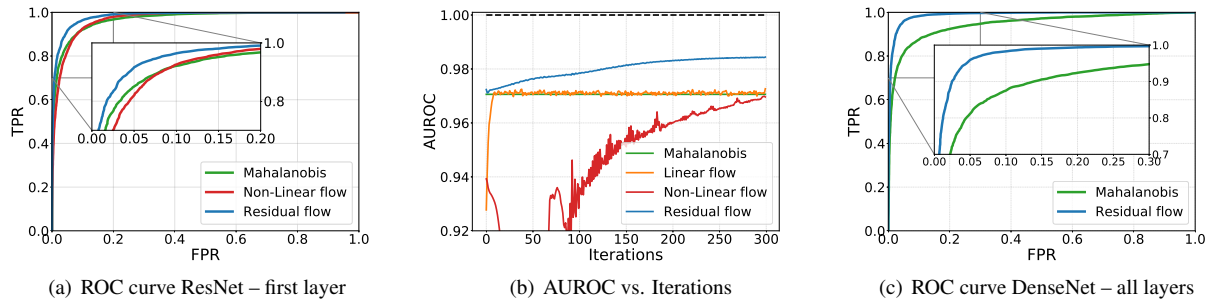| (a) ROC curve ResNet – first layer | (b) AUROC vs. Iterations | (c) ROC curve DenseNet – all layers |

Figure 2. (a) and (b) OOD detection using features taken from the first layer of ResNet trained on CIFAR-100, with TinyImageNet as OOD. (a) ROC curve comparison of Residual flow (AUROC = 98.4), Non-linear flow (AUROC = 97.0) and Mahalanobis (AUROC = 97.0) [27]. (b) AUROC comparison as a function of training iterations for different models. Note that by our initialization method, the residual flow starts at baseline performance of Mahalanobis. (c) ROC comparison of Mahalanobis (AUROC = 94.6) and Residual flow (AUROC = 98.9), using a weighted average of the features taken from layers of DenseNet trained on CIFAR-100, with LSUN as OOD.

layers from the target network and average over the spatial domain to produce a set of 1-dimensional feature vectors, whose size matches the number of feature maps in the corresponding layer. The selected layers are the terminal layers of every dense-block (or residual-block) of DenseNet (or ResNet). Next, we train a set of residual flow networks, each observing a different output layer of the target network (e.g. DenseNet) activated by an entire class of examples from its original dataset. A portion of the training set, 10K in total, is set aside as a validation set, to prevent overfit during training. The process repeats for all classes and for all end-block layers of the target network, yielding a set of trained residual flows. At the test phase, a score is calculated for every layer of the target network and the final confidence score is obtained using weights produced by training a logistic regression detector (see Algorithm 1).

The weights of the logistic regression decoder and the input pre-processing parameter, ϵ, are the hyperparameters of our model, tuned using a separate validation set of in- (positive class) and out-of-distribution (negative class) pairs, consisting of 1,000 images of each class. Similarly to Lee et al. [27], we also investigate performance when a validation set of OOD samples is not available, and in this case we tune the hyperparameters using validation sets of both in-distribution samples and corresponding adversarial samples generated by FGSM [18] as out-of-distribution samples.

The networks are tested using their original test set, with the introduction of OOD samples from either LSUN [45], CIFAR-10 [25], Tiny-ImageNet [10] or SVHN [30]. The following performance measures are evaluated: true negative rate (TNR) at 95% true positive rate (TPR), area under the receiver operating characteristic curve (AUROC), area under the precision-recall curve (AUPR), and detection accuracy. We compare our method to the state-of-the-art, which employs Mahalanobis score as a confidence score [27]. Note that to accommodate a fair comparison, we adopt the hyperparameter selection procedure presented in [27].

Table 1 aggregates the performance of our method compared to Mahalanobis for the task of OOD detection across all in- and out-of-distribution dataset pairs, when an OOD validation set is available. Table 2 compares the performance when the validation set is produced using FGSM, as described above. We present the detection performance measures of our method with and without input pre-processing (right and middle columns respectively), and compare it to Mahalanobis score method with input pre-processing (left column). Tables 1 and 2 demonstrably show that our method surpasses the current state-of-the-art, significantly outperforming the Mahalanobis approach in some cases – even without input pre-processing. For example, applying our method on ResNet trained on CIFAR-100 samples, when LSUN is used as OOD dataset, improves the AUROC from 66.2% to 82.0% (without input pre-processing) and 87.2% (with input pre-processing). In summary, the results in tables 1 and 2 demonstrate that better modeling of feature activations leads to better OOD detection. Figure 2(c) further demonstrates the contribution of our method compared to Mahalanobis [27]. We produce a ROC curve using ResNet trained on CIFAR-100, with LSUN dataset used as OOD. Note that the performance in Figure 2(c) was obtained without any pre-processing of the data. As seen from Figure 2(c), our method significantly outperforms the Mahalanobis score method.

6. Conclusions

We proposed an efficient method for detecting out-of-distribution inputs for trained neural networks, without retraining the network or modifying its underlying architecture, nor compromising its classification accuracy on in-distribution data. Key to our approach is a novel deep generative model – the residual flow, which is a principled extension of a Gaussian distribution model using a non-linear normalizing flow. This model, which is of independent interest, is most suitable for modelling distributions that

In-dist (model)	Out-of-dist	TNR at TPR 95%	AUROC	Detection accuracy	AUPR in	AUPR out
		Mahalanobis [27]/ Res-Flow without pre-processing / Res-Flow with pre-processing				
CIFAR-10 (DenseNet)	SVHN	85.8 / **94.9** / **94.9**	96.6 / **98.9** / **98.9**	91.9 / **95.3** / **95.3**	98.7 / **99.5** / **99.5**	88.8 / **97.5** / **97.5**
	ImageNet	95.3 / **96.4** / **96.4**	98.9 / **99.2** / **99.2**	95.2 / **96.0** / **96.0**	98.9 / **99.2** / **99.2**	98.7 / **99.2** / **99.2**
	LSUN	97.9 / **98.2** / **98.2**	99.3 / **99.5** / **99.5**	96.8 / **97.1** / **97.1**	99.3 / **99.6** / **99.6**	98.2 / **99.5** / **99.5**
CIFAR-100 (DenseNet)	SVHN	82.9 / 73.0 / **84.9**	96.1 / 95.2 / **97.5**	90.9 / 88.7 / **91.9**	98.5 / 97.5 / **99.0**	89.0 / 91.1 / **95.1**
	TinyImageNet	85.8 / **93.0** / **93.0**	96.6 / **98.5** / **98.5**	91.2 / **94.1** / **94.1**	96.9 / **98.5** / **98.5**	95.5 / **98.5** / **98.5**
	LSUN	83.6 / **96.3** / **96.3**	94.9 / **98.9** / **98.9**	89.9 / **95.7** / **95.7**	95.7 / **99.0** / **99.0**	93.0 / **98.8** / **98.8**
SVHN (DenseNet)	CIFAR-10	96.5 / **99.0** / **99.0**	98.9 / **99.5** / **99.5**	95.9 / **97.4** / **97.4**	95.6 / **97.8** / **97.8**	99.6 / **99.8** / **99.8**
	TinyImageNet	99.8 / **100.0** / **100.0**	99.9 / **100.0** / **100.0**	98.8 / **99.4** / **99.4**	99.6 / **99.8** / **99.8**	**100.0** / **100.0** / **100.0**
	LSUN	**100.0**/ **100.00** / **100.00**	99.9 / **100.0** / **100.0**	99.3 / **99.7** / **99.7**	99.7 / **99.9** / **99.9**	**100.0** / **100.0** / **100.0**
CIFAR-10 (ResNet)	SVHN	96.4 / 94.5 / **96.5**	**99.1** / 98.9 / **99.1**	**95.8** / 94.9 / **95.8**	**99.6** / **99.6** / **99.6**	**98.3** / 97.6 / **98.3**
	TinyImageNet	97.1 / **97.8** / **97.8**	99.5 / **99.6** / **99.6**	96.3 / **96.9** / **96.9**	99.5 / **99.6** / **99.6**	99.5 / **99.6** / **99.6**
	LSUN	98.9 / **99.0** / **99.0**	99.7 / **99.8** / **99.8**	97.7 / **97.8** / **97.8**	99.7 / **99.8** / **99.8**	99.7 / **99.8** / **99.8**
CIFAR-100 (ResNet)	SVHN	92.0 / 88.8 / **93.0**	98.4 / 97.8 / **98.5**	93.7 / 92.6 / **94.5**	**99.3** / 99.1 / **99.3**	96.4 / 95.3 / **97.1**
	TinyImageNet	90.8 / **95.0** / 94.6	98.2 / **98.9** / **98.9**	93.3 / **95.0** / **95.0**	98.1 / **98.9** / **98.9**	98.2 / **98.9** / 98.8
	LSUN	90.9 /**96.7** / 96.2	98.2 / **99.1** / 99.0	93.5 / **96.0** / 95.7	97.8 / **99.0** / 98.9	98.4 / **98.8** / 98.6
SVHN (ResNet)	CIFAR-10	98.5 / 99.3 / **99.4**	99.3 / **99.6** /**99.6**	96.9 / **97.7** / **97.7**	97.0 / **98.3** / **98.3**	99.7 / **99.9** / **99.9**
	TinyImageNet	99.9 / **100.0** / **100.0**	99.9 / **100.0** / 99.9	99.1 / **99.5** / 99.3	99.1 / **99.8** / 99.7	99.9 / **100.0** / **100.0**
	LSUN	99.9 / **100.0** / **100.0**	99.9 / **100.0** / **100.0**	99.5 / **99.7** / **99.7**	99.2 / **99.8** / **99.8**	99.9 / **100.0** / **100.0**

Table 1. A comparison between our method and Mahalanobis [27] on the task of out-of-distribution detection for image classification of various in- and out-of-distribution data sets. The hyper-parameters were tuned using a validation set of in- and out-of-distribution datasets. The values presented here are percentages and the best results are indicated in bold.

In-dist (model)	Out-of-dist	TNR at TPR 95%	AUROC	Detection accuracy	AUPR in	AUPR out
		Mahalanobis [27]/ Res-Flow without pre-processing / Res-Flow with pre-processing				
CIFAR-10 (DenseNet)	SVHN	88.7 / **91.3**/ 86.1	97.6 / **98.3** / 97.3	92.4 / **93.8** / 91.6	94.7 / **96.6** / 94.3	99.0 / **99.3** / 99.0
	TinyImageNet	88.6 / 96.0 / **96.1**	97.5 / **99.1** / **99.1**	92.2 / **95.6** / **95.6**	97.4 / **99.1** / **99.1**	97.7 / **99.2** / **99.2**
	LSUN	92.4 / 98.0 / **98.1**	98.3 / **99.5** / **99.5**	93.9 / 96.7 / **96.9**	98.4 / **99.5** / **99.5**	98.2 / 99.4 / **99.5**
CIFAR-100 (DenseNet)	SVHN	48.7 / **59.8** / 48.9	85.6 / **91.4** / 87.9	80.0 / **83.7** / 80.0	63.7 /**82.9**/ 74.9	93.3 /**96.1** / 94.3
	TinyImageNet	80.4 / **91.7** / 91.5	92.7 / **98.3** / 98.1	88.0 / **93.6** / 93.4	87.4 / **98.3** / 98.0	94.5 / **98.4** / 98.3
	LSUN	83.8 / 95.4 / **95.8**	95.0 / **98.9** / **98.9**	90.0 / 95.3 / **95.4**	93.0 / **99.0** / 98.9	95.7 / **98.8** / **98.8**
SVHN (DenseNet)	CIFAR-10	92.5 / **95.1** / 90.0	96.7 / **98.7** / 98.0	93.8 / **95.3** / 93.4	97.9 / 99.6 / **99.7**	93.5 / **95.2** / 93.6
	TinyImageNet	99.1 / 99.7 / **99.9**	99.5 / **99.9** / **99.9**	98.7 / **99.2** / 99.0	99.6 / **100.0** / **100.0**	99.2 / **99.8** / 99.6
	LSUN	99.7 / **100.0** / **100.0**	99.8 / **100.0** / 99.9	99.1 / **99.5** / 99.4	99.9 / 100.0 / **100.0**	99.6 / **99.8** / 99.7
CIFAR-10 (ResNet)	SVHN	87.5 / **91.0** / **91.0**	97.4 / **98.2** / **98.2**	91.8 / **93.8** / **93.8**	93.8 / **96.6** / **96.6**	98.9 / **99.1** / **99.1**
	TinyIageNet	93.1 / **98.0** / **98.0**	97.9 / **99.6** / **99.6**	94.1 / **97.0** / **97.0**	95.4 / **99.6** / **99.6**	98.4 / **99.6** / **99.6**
	LSUN	97.0 / **99.1** / **99.1**	99.2 / **99.8** / **99.8**	96.3 / **98.0** / **98.0**	98.6 / **99.8** / **99.8**	99.3 / **99.8** / **99.8**
CIFAR-100 (ResNet)	SVHN	66.5 / 57.2 / **74.1**	93.2 / 90.7 / **95.1**	85.9 / 83.8 / **88.7**	86.4 / 80.5 / **90.4**	96.6 / 95.4 / **97.5**
	TinyImageNet	56.7 / 71.6 / **77.5**	76.9 / 86.8 / **90.1**	77.6 / 84.3 / **87.1**	63.0 / 74.8 / **79.6**	83.7 / 90.4 / **93.1**
	LSUN	38.4 / 61.1 / **70.4**	66.2 / 82.0 / **87.2**	69.5 / 80.1 / **84.1**	54.6 / 70.0 / **75.9**	73.9 / 86.5 / **90.5**
SVHN (ResNet)	CIFAR-10	95.2 / **97.1** / 96.6	98.1 / **99.1** / 99.0	95.2 / **96.1** / 95.8	98.5 / **99.7** / **99.7**	95.2 / **96.7** / 96.5
	TinyImageNet	99.3 / **99.9** / **99.9**	99.4 / **99.9** / **99.9**	98.9 / **99.3** / 99.2	98.9 / **99.9** / **99.9**	98.3 / **99.7** / **99.7**
	LSUN	99.9 / **100.0** / **100.0**	99.9 / **100.0** / **100.0**	99.5 / **99.7** / 99.6	99.9 / **100.0** / **100.0**	98.8 / **99.7** / **99.7**

Table 2. A comparison between our method and Mahalanobis [27] on the task of out-of-distribution detection for image classification of various in- and out-of-distribution data sets. The hyper-parameters were tuned using strictly in-distribution and adversarial (FGSM) samples. The values presented here are percentages and the best results are indicated in bold.

are approximately Gaussian. Our method is general, and in principle can be applied to various data such as speech recognition and natural language processing. On deep networks trained for image classification, we obtain state-of-the-art out-of-distribution detection performance.

7. Acknowledgments

References

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016. 1

[2] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1), 2015. 6

[3] Jerone Andrews, Edward Morton, and Lewis Griffin. Detecting anomalous data using auto-encoders. *International Journal of Machine Learning and Computing*, 6:21, 01 2016. 6

[4] Van Loi Cao, Miguel Nicolau, and James Mcdermott. A hybrid autoencoder and density estimation model for anomaly detection. In *PPSN*, volume 9921, pages 717–726, 09 2016. 6

[5] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *CoRR*, abs/1901.03407, 2019. 6

[6] Jinghui Chen, Saket Sathe, Charu Aggarwal, and Deepak Turaga. Outlier detection with autoencoder ensembles. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 90–98. SIAM, 2017. 6

[7] Kyunghyun Cho, Caglar Gulcehre, Universite 'De Montreal, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, 2014. 1

[8] C Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, 16(1):41–46, 1970. 6

[9] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006. 6

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 7

[11] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *CoRR*, abs/1410.8516, 2014. 2, 11

[12] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *ArXiv*, abs/1605.08803, 2016. 2

[13] Scott R Eliason. *Maximum likelihood estimation: Logic and practice*. Number 96. Sage, 1993. 3

[14] Sarah M. Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recogn.*, 58(C):121–134, Oct. 2016. 6

[15] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in neural information processing systems*, pages 4878–4887, 2017. 6

[16] Amol Ghoting, Srinivasan Parthasarathy, and Matthew Eric Otey. Fast mining of distance-based outliers in high-dimensional datasets. *Data Mining and Knowledge Discovery*, 16(3):349–364, 2008. 6

[17] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations. In *Advances in Neural Information Processing Systems*, pages 9758–9769, 2018. 6

[18] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014. 1, 7

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 2, 6

[20] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. ICLR, 2017. 1, 6

[21] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdelrahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012. 1

[22] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 6

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 5

[24] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018. 2, 4

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 6, 7

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[27] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 7167–7177, 2018. 1, 2, 3, 4, 5, 6, 7, 8, 13, 14

[28] Shiyu Liang, Yixuan Li, and R Srikant. Principled detection of out-of-distribution examples in neural networks. ICLR, 2018. 1, 2, 3, 5, 6

[29] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017. 1

[30] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 6, 7

[31] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015. 1

[32] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014. 6

[33] Lukas Ruff, Robert A. Vandermeulen, Nico Görnitz, Lucas Deecke, Shoaib A. Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 4393–4402, 2018. 6

[34] Chandramouli S Sastry and Sageev Oore. Zero-shot out-of-distribution detection with feature correlations, 2020. 6

[35] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*, pages 146–157. Springer, 2017. 6

[36] Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471, July 2001. 6

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 1

[38] Yunfu Song and Zhijian Ou. Learning neural random fields with inclusive auxiliary generators. *ArXiv*, abs/1806.00271, 2018. 6

[39] Suwon Suh, Daniel H Chae, Hyon-Goo Kang, and Seungjin Choi. Echo-state conditional variational autoencoder for anomaly detection. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1015–1022. IEEE, 2016. 6

[40] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. 1

[41] Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013. 2

[42] Esteban G Tabak, Eric Vanden-Eijnden, et al. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010. 2

[43] David M. J. Tax and Robert P. W. Duin. Support vector data description. *Mach. Learn.*, 54(1):45–66, Jan. 2004. 6

[44] William Wang, Angelina Wang, Aviv Tamar, Xi Chen, and Pieter Abbeel. Safer classification by synthesis. *CoRR*, abs/1711.08534, 2017. 6

[45] 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. 7

[46] Sehun Yu, Donga Lee, and Hwanjo Yu. Out-of-distribution image detection using the normalized compression distance, 2020. 6

[47] Shuangfei Zhai, Yu Cheng, Weining Lu, and Zhongfei Zhang. Deep structured energy based models for anomaly detection. *CoRR*, abs/1605.07717, 2016. 6