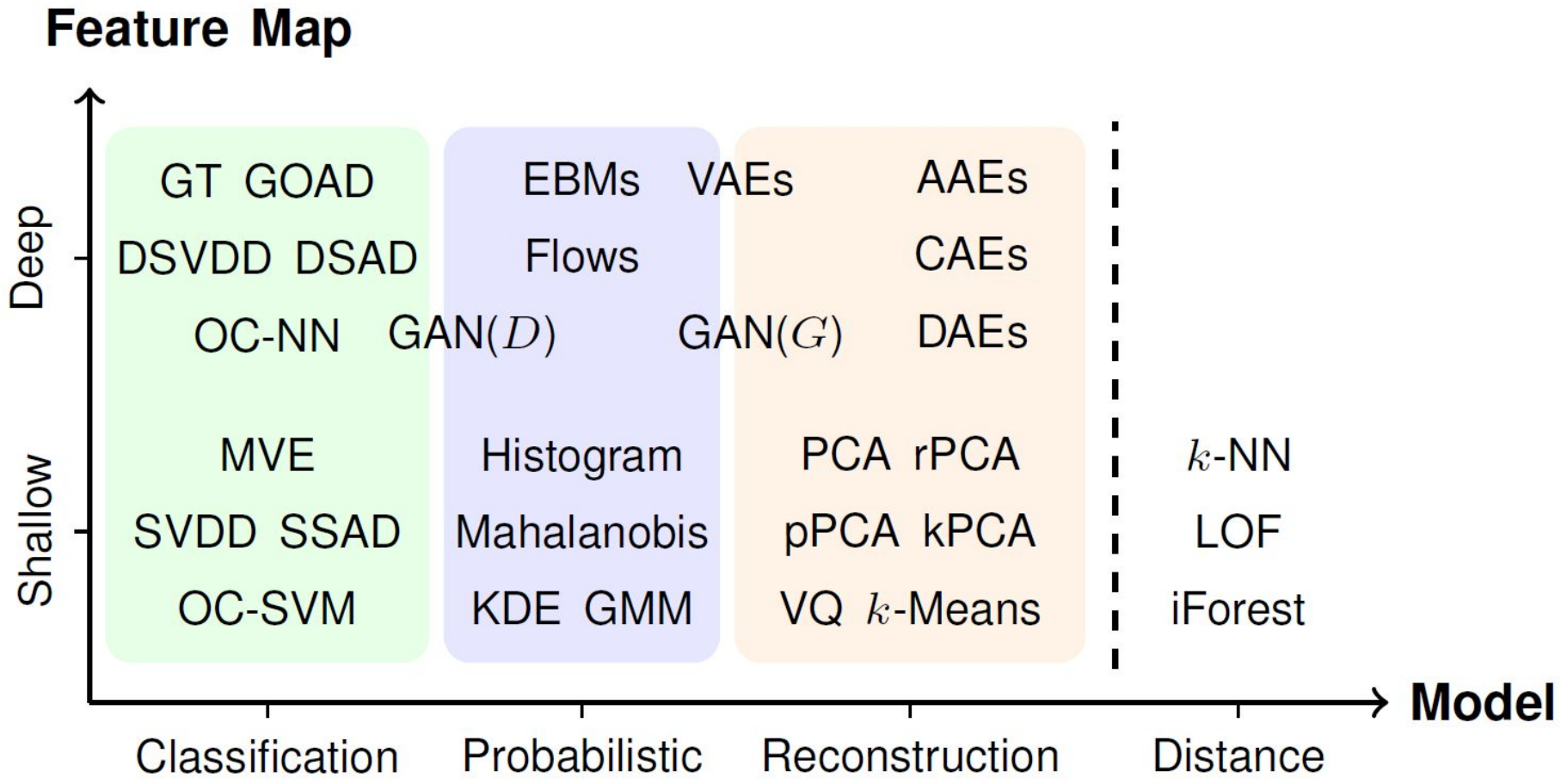
i-TAP
Outlier Detection

홍익대학교
노승문
2021. 3. 26.

A Unifying Review of Deep and Shallow Anomaly Detection

Lukas Ruff, Jacob R. Kauffmann, Robert A. Vandermeulen, Grégoire Montavon, Wojciech Samek, *Member, IEEE*, Marius Kloft*, *Senior Member, IEEE*, Thomas G. Dietterich*, *Member, IEEE*, Klaus-Robert Müller*, *Member, IEEE*.

2021 Feb arxiv


Feature Map
Deep
Shallow
GT GOAD
DSVDD DSAD
OC-NN
GAN(D)
EBMs
Flows
VAEs
GAN(G)
AAEs
CAEs
DAEs
MVE
SVDD SSAD
OC-SVM
Histogram
Mahalanobis
KDE GMM
PCA rPCA
pPCA kPCA
VQ k-Means
k-NN
LOF
iForest
Model
Classification
Probabilistic
Reconstruction
Distance

Definition of Anomaly

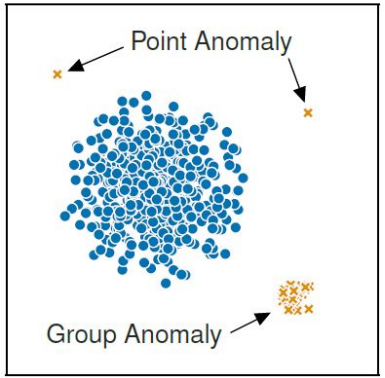
An anomaly is an observation that deviates considerably from some concept of normality.

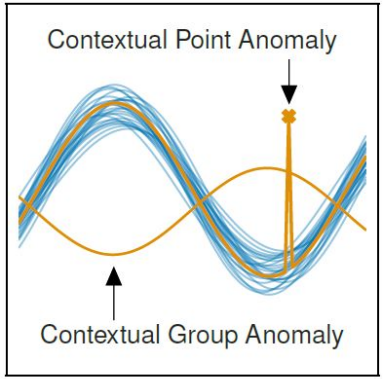
Assuming that $\mathbb{P}^+$ has a corresponding probability density function (pdf) $p^+(\boldsymbol{x})$, we can define a *set of anomalies* as

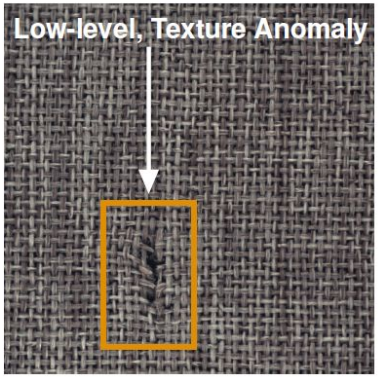
$$\mathcal{A} = \{\boldsymbol{x} \in \mathcal{X} \mid p^+(\boldsymbol{x}) \leq \tau\}, \quad \tau \geq 0, \tag{1}$$

where τ is some threshold such that the probability of $\mathcal{A}$ under $\mathbb{P}^+$ is 'sufficiently small' which we will specify further below.

Types of Anomalies







High-level, Semantic Anomaly

Concentration Assumption

That is, that there exists some threshold $\tau \geq 0$ such that

$$\mathcal{X} \setminus \mathcal{A} = \{\boldsymbol{x} \in \mathcal{X} \mid p^{+}(\boldsymbol{x}) > \tau\} \tag{2}$$

is non-empty and small (typically in the Lebesgue-measure sense, which is the ordinary notion of volume in D-dimensional space). This is known as the so-called *concentration* or *cluster assumption* [211]–[213].

Density Level Set Estimation

We can formally express this objective as the problem of *density level set estimation* [214]–[217] which is equivalent to *minimum volume set estimation* [218]–[220]

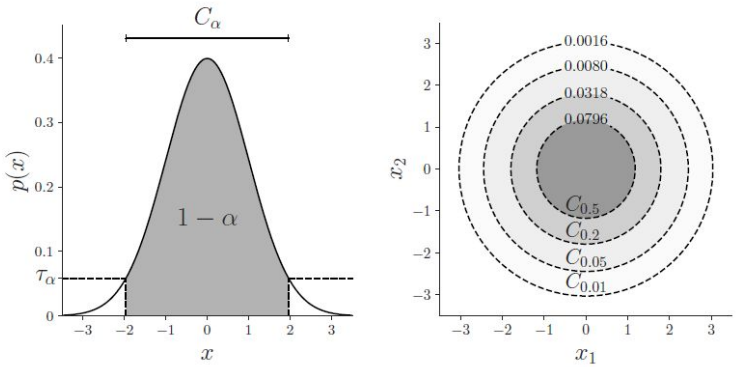


Fig. 3. An illustration of the α-density level sets C_α with threshold τ_α for a univariate (left) and bivariate (right) standard Gaussian distribution.

Density Estimation

Given some estimated density model $\hat{p}(\boldsymbol{x}) = \hat{p}(\boldsymbol{x}; \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \approx p(\boldsymbol{x})$ and some target level $\alpha \in [0, 1]$, one can estimate a corresponding threshold $\hat{\tau}_\alpha$ via the empirical p-value function:

$$\hat{\tau}_\alpha = \inf_{\tau} \left\{ \tau \geq 0 \;\middle|\; \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{[0, \hat{p}(\boldsymbol{x}_i))}(\tau) \geq 1 - \alpha \right\}, \qquad (5)$$

where $\mathbb{1}_A(\cdot)$ denotes the indicator function for some set A.

Anomaly Score

Many anomaly detection methods also target some strictly increasing transformation $T : [0, \infty) \to \mathbb{R}$ of the density for estimating a model (e.g., log-likelihood instead of likelihood). The resulting target $T(p(\boldsymbol{x}))$ is usually no longer a proper density but still preserves the density ranking [225], [226]. An *anomaly score* $s : \mathcal{X} \to \mathbb{R}$ can then be defined by using an additional order-reversing transformation, for example $s(\boldsymbol{x}) = -T(p(\boldsymbol{x}))$ (e.g., negative log-likelihood), so that high scores reflect low density values and vice versa.

Dataset Setting

2) The Unsupervised Setting: The unsupervised anomaly detection setting is the case in which *only unlabeled data*

$$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X} \tag{6}$$

3) The Semi-Supervised Setting: The semi-supervised anomaly detection setting is the case in which both *unlabeled and labeled data*

$$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X} \quad \text{and} \quad (\tilde{\boldsymbol{x}}_1, \tilde{y}_1), \ldots, (\tilde{\boldsymbol{x}}_m, \tilde{y}_m) \in \mathcal{X} \times \mathcal{Y} \tag{7}$$

4) The Supervised Setting: The supervised anomaly detection setting is the case in which *completely labeled data*

$$(\tilde{\boldsymbol{x}}_1, \tilde{y}_1), \ldots, (\tilde{\boldsymbol{x}}_m, \tilde{y}_m) \in \mathcal{X} \times \mathcal{Y} \tag{8}$$

Overview

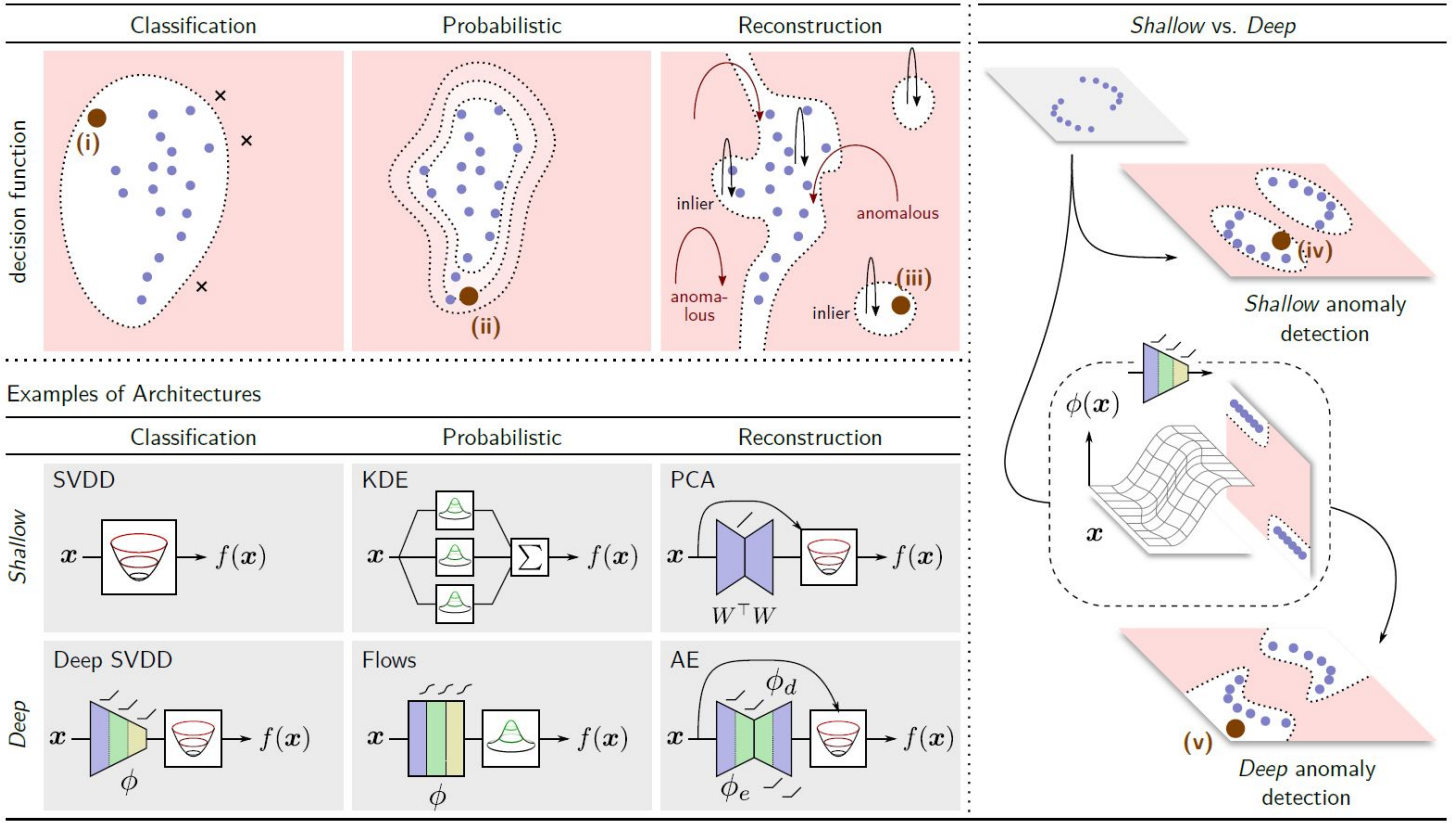


Fig. 5. An overview of the different approaches to anomaly detection. *Top*: Typical decision functions learned by the different anomaly detection approaches, where white corresponds to normal and red to anomalous decision regions. One-class *Classification* models typically learn a discriminative decision boundary, *Probabilistic* models a density, and *Reconstruction* models some underlying geometric structure of the data (e.g., manifold or prototypes). *Right*: Deep feature maps enable to learn more flexible, non-linear decision functions suitable for more complex data. *Bottom:* Diagrams of architectures for a selection of different methods with deep and shallow feature maps. *Points (i)–(v)*: Locations in input space, where we highlight some model-specific phenomena: (i) A too loose, biased one-class boundary may leave anomalies undetected; (ii) Probabilistic models may underfit (or overfit) the tails of a distribution; (iii) Manifold or prototype structure artifacts may result in good reconstruction of anomalies; (iv) Simple shallow models may fail to fit complex, non-linear distributions; (v) Compression artifacts of deep feature maps may create 'blind spots' in input space.

Probabilistic (Density)

A. Classic Density Estimation

One of the most basic approaches to multivariate anomaly detection is to compute the Mahalanobis distance from a test point to the training data mean [253]. This is equivalent to fitting a multivariate Gaussian distribution to the training data and evaluating the log-likelihood of a test point according to that model [254]. Compared to modeling each dimension of the data independently, fitting a multivariate Gaussian captures linear interactions between pairs of dimensions. To model more complex distributions, nonparametric density estimators have been introduced, such as kernel density estimators (KDE) [12], [252], histogram estimators, and Gaussian mixture models (GMMs) [255], [256]. The kernel density estimator is arguably the most widely used nonparametric density estimator due to theoretical advantages over histograms [257] and the practical issues with fitting and parameter selection for GMMs [258]. The standard kernel density estimator, along with a more recent adaptation that can deal with modest levels of outliers in the training data [259], [260], is therefore a popular approach to anomaly detection. A GMM with a finite number of K mixtures can also be viewed as a soft (probabilistic) clustering method that assumes K prototypical modes (cf., section V-A2). This has been used, for example, to represent typical states of a machine in predictive maintenance [261].

While classic nonparametric density estimators perform fairly well for low dimensional problems, they suffer notoriously from the curse of dimensionality: the sample size required to attain a fixed level of accuracy grows exponentially in the dimension of the feature space. One goal of deep statistical models is to overcome this challenge.

Probabilistic (Energy)

B. Energy-Based Models

Some of the earliest deep statistical models are energy based models (EBMs) [262]–[264]. An EBM is a model whose density is characterized by an energy function $E_\theta(\boldsymbol{x})$ with

$$p_\theta(\boldsymbol{x}) = \frac{1}{Z(\theta)} \exp\left(-E_\theta(\boldsymbol{x})\right), \tag{9}$$

where $Z(\theta) = \int \exp\left(-E_\theta(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}$ is the so-called *partition function* that ensures that p_θ integrates to 1. These models are typically trained via gradient descent, and approximating the log-likelihood gradient $\nabla_\theta \log p_\theta(\boldsymbol{x})$ via Markov chain Monte Carlo (MCMC) [265] or Stochastic Gradient Langevin Dynamics (SGLD) [266], [267]. While one typically cannot

Probabilistic (Energy)

B. Energy-Based Models

Some of the earliest deep statistical models are energy based models (EBMs) [262]–[264]. An EBM is a model whose density is characterized by an energy function $E_\theta(\boldsymbol{x})$ with

$$p_\theta(\boldsymbol{x}) = \frac{1}{Z(\theta)} \exp\left(-E_\theta(\boldsymbol{x})\right), \tag{9}$$

where $Z(\theta) = \int \exp\left(-E_\theta(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}$ is the so-called *partition function* that ensures that p_θ integrates to 1. These models are typically trained via gradient descent, and approximating the log-likelihood gradient $\nabla_\theta \log p_\theta(\boldsymbol{x})$ via Markov chain Monte Carlo (MCMC) [265] or Stochastic Gradient Langevin Dynamics (SGLD) [266], [267]. While one typically cannot

Probability (Generative Model)

C. Neural Generative Models (VAEs and GANs)

Neural generative models aim to learn a neural network that maps vectors sampled from a simple predefined source distribution $\mathbb{Q}$, usually a Gaussian or uniform distribution, to the actual input distribution $\mathbb{P}^+$. More formally, the objective is to train the network so that $\phi_\omega(\mathbb{Q}) \approx \mathbb{P}^+$ where $\phi_\omega(\mathbb{Q})$ is the distribution that results from pushing the source distribution $\mathbb{Q}$ through neural network ϕ_ω. The two most established neural generative models are Variational Autoencoders (VAEs) [272]–[274] and Generative Adversarial Networks (GANs) [275].

Probability (Generative Model)

1) VAEs: Variational Autoencoders learn deep latent-variable models where the inputs $\boldsymbol{x}$ are parameterized on latent samples $\boldsymbol{z} \sim \mathbb{Q}$ via some neural network so as to learn a distribution $p_\theta(\boldsymbol{x}\,|\,\boldsymbol{z})$ such that $p_\theta(\boldsymbol{x}) \approx p^*(\boldsymbol{x})$. A common instantiation of this is to let $\mathbb{Q}$ be an isotropic multivariate Gaussian distribution and let the neural network $\phi_{d,\omega} = (\boldsymbol{\mu}_\omega, \boldsymbol{\sigma}_\omega)$ (the *decoder*) with weights ω, parameterize the mean and variance of an isotropic Gaussian distribution, so $p_\theta(\boldsymbol{x}\,|\,\boldsymbol{z}) \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_\omega(\boldsymbol{z}), \boldsymbol{\sigma}^2_\omega(\boldsymbol{z})I)$. Performing maximum likelihood estimation on θ is typically intractable. To remedy this an additional network $\phi_{e,\omega'}$ (the *encoder*) is introduced to parameterize a variational distribution $q_{\theta'}(\boldsymbol{z}\,|\,\boldsymbol{x})$, with θ' encapsulated by the output of $\phi_{e,\omega'}$, to approximate the latent posterior $p(\boldsymbol{z}\,|\,\boldsymbol{x})$. The full model is then optimized via the evidence lower bound (ELBO) in a variational Bayes manner:

$$\max_{\theta,\theta'} \; -D_{\mathrm{KL}}\left(q_{\theta'}(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z})\right) + \mathbb{E}_{q_{\theta'}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right]. \quad (10)$$

Probability (Generative Model)

2) GANs: Generative Adversarial Networks pose the problem of learning the target distribution as a zero-sum-game: a generative model is trained in competition with an adversary that challenges it to generate samples whose distribution is similar to the training distribution. A GAN consists of two neural networks, a *generator* network $\phi_\omega : \mathcal{Z} \to \mathcal{X}$ and a *discriminator* network $\psi_{\omega'} : \mathcal{X} \to (0, 1)$ which are pitted against each other so that the discriminator is trained to discriminate between $\phi_\omega(\boldsymbol{z})$ and $\boldsymbol{x} \sim \mathbb{P}^+$ where $\boldsymbol{z} \sim \mathbb{Q}$. The generator is trained to fool the discriminator network thereby encouraging the generator to produce samples more similar to the target distribution. This is done using the following adversarial objective:

$$\min_{\omega} \max_{\omega'} \quad \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}^+} \left[\log \psi_{\omega'}(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z} \sim \mathbb{Q}} \left[\log(1 - \psi_{\omega'}(\phi_\omega(\boldsymbol{z})))\right]. \tag{11}$$

Probability (Flow)

D. Normalizing Flows

Like neural generative models, normalizing flows [282]–[284] attempt to map data points from a source distribution $\boldsymbol{z} \sim \mathbb{Q}$ (usually called *base distribution* for normalizing flows) so that $\boldsymbol{x} \approx \phi_\omega(\boldsymbol{z})$ is distributed according to p^+. The crucial

of L layers $\phi_{i,\omega_i} : \mathbb{R}^D \to \mathbb{R}^D$ so $\phi_\omega = \phi_{L,\omega_L} \circ \cdots \circ \phi_{1,\omega_1}$ where each ϕ_{i,ω_i} is designed to be invertible for all ω_i, thereby making the entire network invertible. The benefit of this formulation is that the probability density of $\boldsymbol{x}$ can be calculated exactly via a change of variables

$$p_{\boldsymbol{x}}(\boldsymbol{x}) = p_{\boldsymbol{z}}(\phi_\omega^{-1}(\boldsymbol{x})) \prod_{i=1}^{L} \left| \det J\phi_{i,\omega_i}^{-1}(\boldsymbol{x}_i) \right| \tag{13}$$

where $\boldsymbol{x}_L = \boldsymbol{x}$ and $\boldsymbol{x}_i = \phi_{i+1}^{-1} \circ \cdots \circ \phi_L^{-1}(\boldsymbol{x})$ otherwise. Normalizing flow models are typically optimized to maximize the likelihood of the training data. Evaluating each layer's Jacobian and its determinant can be very expensive. Consequently, the layers of flow models are usually designed so that the Jacobian is guaranteed to be upper (or lower) triangular, or have some other nice structure, such that one does not need to compute the full Jacobian to evaluate its determinant [282], [285], [286]. See [287] for an application in physics.

An advantage of these models over other methods is that one can calculate the likelihood of a point directly without any approximation while also being able to sample from it reasonably efficiently. Because the density $p_{\boldsymbol{x}}(\boldsymbol{x})$ can be computed exactly, normalizing flow models can be applied directly for anomaly detection [288], [289].

A drawback of these models is that they do not perform any dimensionality reduction, which argues against applying them to images where the true (effective) dimensionality is much smaller than the image dimensionality. Furthermore, it has been observed that these models often assign high likelihood to anomalous instances [277]. Recent work suggests that one

Probability

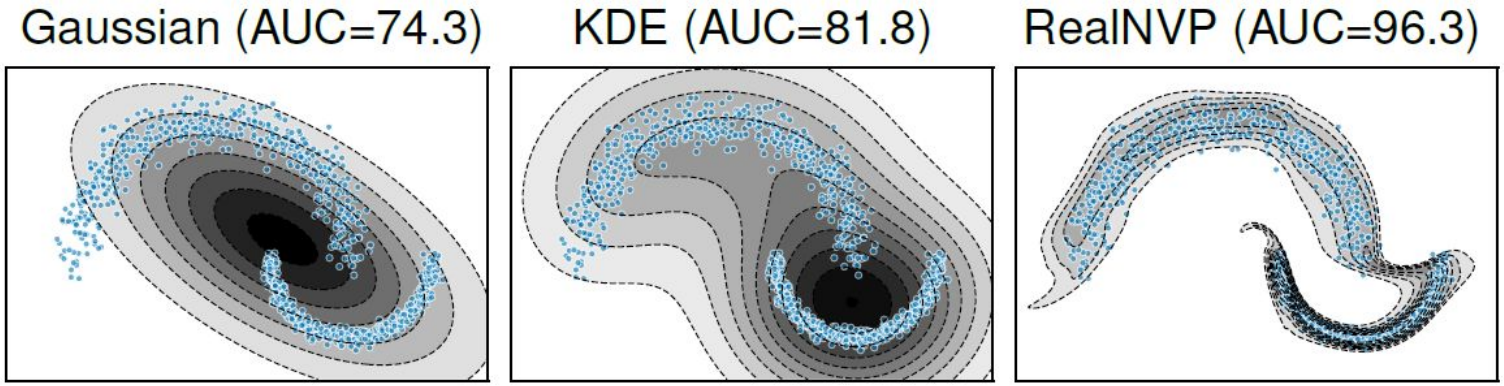


Fig. 6. Density estimation models on the *Big Moon, Small Moon* toy example (see Fig. 4). The parametric Gaussian model is limited to an ellipsoidal (convex, unimodal) density. KDE with an RBF kernel is more flexible, yet tends to underfit the (multi-scale) distribution due a uniform kernel scale. RealNVP is the most flexible model, yet flow architectures induce biases as well, here a connected support caused by affine coupling layers in RealNVP.

One-Class Classification

IV. One-Class Classification

One-class classification [223], [224], [301]–[303], occasionally also called *single-class classification* [304], [305], adopts a discriminative approach to anomaly detection. Methods based on one-class classification try to avoid a full estimation of the density as an intermediate step to anomaly detection. Instead, these methods aim to directly learn a decision boundary that corresponds to a desired density level set of the normal data distribution $\mathbb{P}^+$, or more generally, to produce a decision boundary that yields a low error when applied to unseen data.

One-Class Classification

A. *One-Class Classification Objective*

We can see one-class classification as a particularly tricky classification problem, namely as binary classification where we only have (or almost only have) access to data from one class — the normal class. Given this imbalanced setting, the one-class classification objective is to learn a one-class decision boundary that minimizes (i) falsely raised alarms for true normal instances (i.e., the false alarm rate or type I error), and (ii) undetected or missed true anomalies (i.e., the miss rate or type II error). Achieving a low (or zero) false alarm rate, is conceptually simple: given enough normal data points, one could just draw some boundary that encloses all the points, for example a sufficiently large ball that contains all data instances. The crux here is, of course, to simultaneously keep the miss rate low, that is, to not draw this boundary too loosely. For this reason, one usually *a priori* specifies some target false alarm rate $\alpha \in [0, 1]$ for which the miss rate is then sought to be minimized. Note that this precisely corresponds to the idea of estimating an α-density level set for some a priori fixed level $\alpha \in [0, 1]$. The key question in one-class classification thus is how to minimize the miss rate for some given target false alarm rate with access to no (or only few) anomalies.

One-Class Classification

$\ell : \mathbb{R} \times \{\pm 1\} \to \mathbb{R}$ be a binary classification loss and $f : \mathcal{X} \to \mathbb{R}$ be some real-valued score function. The classification risk of f under loss ℓ is then given by:

$$R(f) = \mathbb{E}_{X \sim \mathbb{P}^+}[\ell(f(X), +1)] + \mathbb{E}_{X \sim \mathbb{P}^-}[\ell(f(X), -1)]. \quad (14)$$

Minimizing the second term — the expected loss of classifying true anomalies as normal — corresponds to minimizing the (expected) miss rate. Given some unlabeled data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X}$, and potentially some additional labeled data $(\tilde{\boldsymbol{x}}_1, \tilde{y}_1), \ldots, (\tilde{\boldsymbol{x}}_m, \tilde{y}_m)$, we can apply the principle of empirical risk minimization to obtain

$$\min_f \quad \frac{1}{n} \sum_{i=1}^{n} \ell(f(\boldsymbol{x}_i), +1) + \frac{1}{m} \sum_{j=1}^{m} \ell(f(\tilde{\boldsymbol{x}}_j), \tilde{y}_j) + \mathcal{R}. \quad (15)$$

This solidifies the empirical one-class classification objective. Note that the second term is an empty sum in the unsupervised setting. Without any additional constraints or regularization, the empirical objective (15) would then be trivial. We add $\mathcal{R}$ as an additional term to denote and capture regularization which may take various forms depending on the assumptions about f, but critically also about $\mathbb{P}^-$. Generally, the regular-

One-Class Classification (SVDD)

Support
Vector
Data
Description

B. One-Class Classification in Input Space

As an illustrative example that conveys useful intuition, consider the simple idea from above of fitting a data-enclosing ball as a one-class model. Given $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X}$, we can define the following objective:

$$\begin{aligned} \min_{R, \boldsymbol{c}, \boldsymbol{\xi}} \quad & R^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & \|\boldsymbol{x}_i - \boldsymbol{c}\|^2 \leq R^2 + \xi_i, \quad \xi_i \geq 0, \quad \forall i. \end{aligned} \tag{16}$$

In words, we aim to find a hypersphere with radius $R > 0$ and center $\boldsymbol{c} \in \mathcal{X}$ that encloses the data ($\|\boldsymbol{x}_i - \boldsymbol{c}\|^2 \leq R^2$). To control the miss rate, we minimize the volume of this hypersphere by minimizing R^2 to achieve a tight spherical boundary. Slack variables $\xi_i \geq 0$ allow some points to fall outside the sphere, thus making the boundary soft, where hyperparameter $\nu \in (0, 1]$ balances this trade-off.

Kernel based SVDD

methods. Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be some positive semi-definite (PSD) kernel with associated reproducing kernel Hilbert space (RKHS) $\mathcal{F}_k$ and corresponding feature map $\phi_k : \mathcal{X} \to \mathcal{F}_k$, so $k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \langle \phi_k(\boldsymbol{x}), \phi_k(\tilde{\boldsymbol{x}}) \rangle$ for all $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{X}$. The objective of (kernel) SVDD is again to find a data-enclosing hypersphere of minimum volume. The SVDD primal problem is the one given in (16), but with the hypersphere model $f_\theta(\boldsymbol{x}) = \|\phi_k(\boldsymbol{x}) - \boldsymbol{c}\|^2 - R^2$ defined in feature space $\mathcal{F}_k$

Kernel based: OC-SVM

instead. In comparison, the OC-SVM objective is to find a hyperplane $\boldsymbol{w} \in \mathcal{F}_k$ that separates the data in feature space $\mathcal{F}_k$ with maximum margin from the origin:

$$\min_{\boldsymbol{w},\rho,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 - \rho + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i \tag{20}$$
$$\text{s.t.} \quad \rho - \langle \phi_k(\boldsymbol{x}_i), \boldsymbol{w}\rangle \leq \xi_i, \quad \xi_i \geq 0, \quad \forall i.$$

So the OC-SVM uses a linear model $f_\theta(\boldsymbol{x}) = \rho - \langle \phi_k(\boldsymbol{x}), \boldsymbol{w}\rangle$ in feature space $\mathcal{F}_k$ with model parameters $\theta = (\boldsymbol{w}, \rho)$. The margin to the origin is given by $\frac{\rho}{\|\boldsymbol{w}\|}$ which is maximized via maximizing ρ, where $\|\boldsymbol{w}\|$ acts as a normalizer.

Deep One Class

D. Deep One-Class Classification

Selecting kernels and hand-crafting relevant features can be challenging and quickly become impractical for complex data. Deep one-class classification methods aim to overcome these challenges by learning useful neural network feature maps $\phi_\omega : \mathcal{X} \to \mathcal{Z}$ from the data or transferring such networks from related tasks. Deep SVDD [137], [144], [145], [327] and deep OC-SVM variants [136], [328] employ a hypersphere model $f_\theta(\boldsymbol{x}) = \|\phi_\omega(\boldsymbol{x}) - \boldsymbol{c}\|^2 - R^2$ and linear model $f_\theta(\boldsymbol{x}) = \rho - \langle \phi_\omega(\boldsymbol{x}), \boldsymbol{w} \rangle$ with explicit neural feature maps $\phi_\omega(\cdot)$ in (16) and (20) respectively. These methods are typically optimized with SGD variants [329]–[331], which, together with GPU parallelization, makes them scale to large datasets.

Deep One Class

The *One-Class Deep SVDD* [137], [332] has been introduced as a simpler variant compared to using a neural hypersphere model in (16), which poses the following objective:

$$\min_{\omega, \boldsymbol{c}} \quad \frac{1}{n} \sum_{i=1}^{n} \|\phi_\omega(\boldsymbol{x}_i) - \boldsymbol{c}\|^2 + \mathcal{R}. \tag{22}$$

Here, the neural network transformation $\phi_\omega(\cdot)$ is learned to minimize the mean squared distance over *all* data points to center $\boldsymbol{c} \in \mathcal{Z}$. Optimizing this simplified objective has been found to converge faster and be effective in many situations [137], [144], [332]. In light of our unifying view, we will see that we may interpret One-Class Deep SVDD also as a single-prototype deep clustering method (cf., sections V-A2 and V-D).

Deep One Class: Regularization

A recurring question in deep one-class classification is how to meaningfully regularize against a feature map collapse $\phi_\omega \equiv \boldsymbol{c}$. Without regularization, minimum volume or maximum margin objectives such as (16), (20), or (22) could be trivially solved with a constant mapping [137], [333]. Possible solutions for this include adding a reconstruction term or architectural constraints [137], [327], freezing the embedding [136], [139], [140], [142], [334], inversely penalizing the embedding variance [335], using true [144], [336], auxiliary [139], [233], [332], [337], or artificial [337] negative examples in training, pseudo-labeling [152], [153], [155], [335], or integrating some manifold assumption [333]. Further variants of deep one-class classification include multimodal [145] or time-series extensions [338] and methods that employ adversarial learning [138], [141], [339] or transfer learning [139], [142].

One Class: Summary

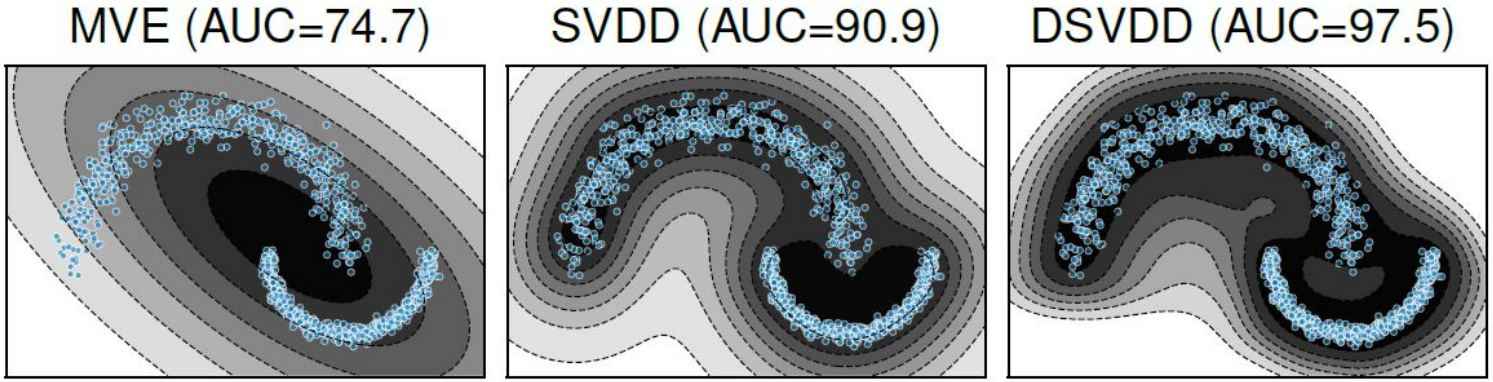


Fig. 7. One-class classification models on the *Big Moon, Small Moon* toy example (see Fig. 4). A Minimum Volume Ellipsoid (MVE) in input space is limited to enclose an ellipsoidal, convex region. By (implicitly) fitting a hypersphere in kernel feature space, SVDD enables non-convex support estimation. Deep SVDD learns an (explicit) neural feature map (here with smooth ELU activations) that extracts multiple data scales to fit a hypersphere model in feature space for support description.

Deep One Class: Negative Example

E. Negative Examples

One-class classifiers can usually incorporate labeled negative examples ($y = -1$) in a direct manner due to their close connection to binary classification as explained above. Such negative examples can facilitate an empirical estimation of the miss rate (cf., (14) and (15)). We here recognize three qualitative types of negative examples that have been studied in the literature, that we distinguish as *artificial*, *auxiliary*, and *true* negative examples which increase in their informativeness in this order.

Deep One Class: Negative Example (artificial)

The idea to approach unsupervised learning problems through generating *artificial* data points has been around for some time (see section 14.2.4 in [340]). If we assume that the anomaly distribution $\mathbb{P}^-$ has some form that we can generate data from, one idea would be to simply train a binary classifier to discern between the normal and the artificial negative examples. For the uniform prior $\mathbb{P}^- \equiv \mathcal{U}(\mathcal{X})$, this approach yields an asymptotically consistent density level set estimator [212]. However, classification against uniformly drawn points from a hypercube quickly becomes ineffective in higher dimensions. To improve over artificial uniform sampling, more informed sampling strategies have been proposed [341] such as resampling schemes [342], manifold sampling [343], and sampling based on local density estimation [344], [345] as well as active

Deep One Class: Negative Example (auxiliary)

learning strategies [346]–[348]. Another recent idea is to treat the enormous quantities of data that are publicly available in some domains as *auxiliary* negative examples [233], for example images from photo sharing sites for computer vision tasks and the English Wikipedia for NLP tasks. Such auxiliary examples provide more informative domain knowledge, for instance about the distribution of natural images or the English language in general, as opposed to sampling random pixels or words. This approach, called *Outlier Exposure* [233], which trains on known anomalies can significantly improve deep anomaly detection performance in some domains [153], [233]. Outlier exposure has also been used with density-based methods by employing a margin loss [233] or temperature annealing [243] on the log-likelihood ratio between positive and negative examples. The most informative labeled negative ex-

Deep One Class: Negative Example (true)

negative examples. The most informative labeled negative examples are ultimately *true* anomalies, for example verified by some domain expert. Access to even a few labeled anomalies has been shown to improve detection performance significantly [144], [224], [229]. There also have been active learning algorithms proposed that include subjective user feedback (e.g., from an expert) to learn about the user-specific informativeness of particular anomalies in an application [349]. Finally, we remark that negative examples have also been incorporated heuristically into reconstruction models via using a bounded reconstruction error [350] since maximizing the unbounded error for negative examples can quickly become unstable. We will turn to reconstruction models next.

Reconstruction

V. Reconstruction Models

Models that are trained on a reconstruction objective are among the earliest [351], [352] and most common [180], [182] neural network approaches to anomaly detection. Reconstruction-based methods learn a model that is optimized to well-reconstruct normal data instances, thereby aiming to detect anomalies by *failing* to accurately reconstruct them under the learned model. Most of these methods have a purely geometric motivation (e.g., PCA or deterministic autoencoders), yet some probabilistic variants reveal a connection to density (level set) estimation. In this section, we define the general reconstruction learning objective, highlight common underlying assumptions, as well as present standard reconstruction-based methods and discuss their variants.

Reconstruction

A. *Reconstruction Objective*

Let $\phi_\theta : \mathcal{X} \to \mathcal{X}, \boldsymbol{x} \mapsto \phi_\theta(\boldsymbol{x})$ be a feature map from the data space $\mathcal{X}$ onto itself that is composed of an *encoding* function $\phi_e : \mathcal{X} \to \mathcal{Z}$ (the *encoder*) and a *decoding* function $\phi_d : \mathcal{Z} \to \mathcal{X}$ (the *decoder*), that is, $\phi_\theta \equiv (\phi_d \circ \phi_e)_\theta$ where θ holds the parameters of both the encoder and decoder. We call $\mathcal{Z}$ the *latent space* and $\phi_e(\boldsymbol{x}) = \boldsymbol{z}$ the *latent representation* (or *embedding* or *code*) of $\boldsymbol{x}$. The reconstruction objective then is to learn ϕ_θ such that $\phi_\theta(\boldsymbol{x}) = \phi_d(\phi_e(\boldsymbol{x})) = \hat{\boldsymbol{x}} \approx \boldsymbol{x}$, that is, to find some encoding and decoding transformation so that $\boldsymbol{x}$ is reconstructed with minimal error, usually measured in Euclidean distance. Given unlabeled data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X}$, the reconstruction objective is given by

$$\min_\theta \quad \frac{1}{n} \sum_{i=1}^{n} \|\boldsymbol{x}_i - (\phi_d \circ \phi_e)_\theta(\boldsymbol{x}_i)\|^2 + \mathcal{R}, \tag{23}$$

Reconstruction: Manifold assumption

1) The Manifold Assumption: The manifold assumption asserts that the data lives (approximately) on some lower-dimensional (possibly non-linear and non-convex) manifold $\mathcal{M}$ that is embedded within the data space $\mathcal{X}$ — that is $\mathcal{M} \subset \mathcal{X}$ with $\dim(\mathcal{M}) < \dim(\mathcal{X})$. In this case $\mathcal{X}$ is sometimes also called the *ambient* or *observation space*. For natural images

Reconstruction: Prototype

2) The Prototype Assumption: The prototype assumption asserts that there exists a finite number of prototypical elements in the data space $\mathcal{X}$ that characterize the data well. We can model this assumption in terms of a data-generating distribution that depends on a discrete latent categorical variable $Z \in \mathcal{Z} = \{1, \dots, K\}$ that captures some K prototypes or modes of the data distribution. This prototype assumption is also common in clustering and classification when we assume a collection of prototypical instances represent clusters or classes well. With the reconstruction objective under the prototype assumption, we aim to learn an encoding function that for $\boldsymbol{x} \in \mathcal{X}$ identifies a $\phi_e(\boldsymbol{x}) = k \in \{1, \dots, K\}$ and a decoding function $k \mapsto \phi_d(k) = \boldsymbol{c}_k$ that maps to some k-th prototype (or some prototypical distribution or mixture of prototypes more generally) such that the reconstruction error $\|\boldsymbol{x} - \boldsymbol{c}_k\|$ becomes minimal. In contrast to the manifold assumption where we aim to describe the data by some continuous mapping, under the (most basic) prototype assumption we characterize the data by a discrete set of vectors $\{\boldsymbol{c}_1, \dots, \boldsymbol{c}_K\} \subseteq \mathcal{X}$. The method of representing a data distribution by a set of prototype vectors is also known as Vector Quantization (VQ) [360], [361].

Reconstruction: anomaly score

3) The Reconstruction Anomaly Score: A model that is trained on the reconstruction objective must extract salient features and characteristic patterns from the data in its encoding — subject to imposed model assumptions — so that its decoding from the compressed latent representation achieves low reconstruction error (e.g., feature correlations and dependencies, recurring patterns, cluster structure, statistical redundancy, etc.). Assuming that the training data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X}$ includes mostly normal points, we therefore expect a reconstruction-based model to produce a *low* reconstruction error for normal instances and a *high* reconstruction error for anomalies. For this reason, the anomaly score is usually also directly defined by the reconstruction error:

$$s(\boldsymbol{x}) = \|\boldsymbol{x} - (\phi_d \circ \phi_e)_\theta(\boldsymbol{x})\|^2. \tag{24}$$

Reconstruction: PCA

B. Principal Component Analysis

A common way to formulate the Principal Component Analysis (PCA) objective is to seek an orthogonal basis W in data space $\mathcal{X} \subseteq \mathbb{R}^D$ that maximizes the empirical variance of the (centered) data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X}$:

$$\max_W \sum_{i=1}^{n} \|W\boldsymbol{x}_i\|^2 \quad \text{s.t. } WW^\top = I. \tag{25}$$

$$\min_W \sum_{i=1}^{n} \|\boldsymbol{x}_i - W^\top W\boldsymbol{x}_i\|^2 \quad \text{s.t. } WW^\top = I, \tag{26}$$

yields exactly the same PCA solution. So PCA optimally solves the reconstruction objective (23) for a linear encoder $\phi_e(\boldsymbol{x}) = W\boldsymbol{x} = \boldsymbol{z}$ and transposed linear decoder $\phi_d(\boldsymbol{z}) = W^\top \boldsymbol{z}$ with constraint $WW^\top = I$. For linear PCA, we can also readily identify its probabilistic interpretation [362], namely that the data distribution follows from the linear transformation $X = W^\top Z + \varepsilon$ of a d-dimensional latent Gaussian distribution $Z \sim \mathcal{N}(\mathbf{0}, I)$, possibly with added noise $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$, so that $\mathbb{P} \equiv \mathcal{N}(\mathbf{0}, W^\top W + \sigma^2 I)$. Maximizing the likelihood of this Gaussian over the encoding and decoding parameter W again yields PCA as the optimal solution [362]. Hence, PCA assumes the data lives on a d-dimensional ellipsoid embedded in data space $\mathcal{X} \subseteq \mathbb{R}^D$. Standard PCA therefore provides an illustrative example for the connections between density estimation and reconstruction.

Reconstruction: kPCA

Linear PCA, of course, is limited to data encodings that can only exploit linear feature correlations. Kernel PCA [3] introduced a non-linear generalization of component analysis by extending the PCA objective to non-linear kernel feature maps and taking advantage of the 'kernel trick'. For a PSD kernel $k(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ with feature map $\phi_k : \mathcal{X} \to \mathcal{F}_k$, kernel PCA solves the reconstruction objective (26) in feature space $\mathcal{F}_k$,

$$\min_{W} \sum_{i=1}^{n} \|\phi_k(\boldsymbol{x}_i) - W^\top W \phi_k(\boldsymbol{x}_i)\|^2 \quad \text{s.t. } WW^\top = I, \quad (27)$$

which results in an eigenvalue problem of the kernel matrix [3]. For kernel PCA, the reconstruction error can again serve as an anomaly score. It can be computed implicitly via the dual [4]. This reconstruction from linear principal components in feature space $\mathcal{F}_k$ corresponds to a reconstruction from some non-linear subspace or manifold in input space $\mathcal{X}$ [371]. Replacing the reconstruction $W^\top W \phi_k(\boldsymbol{x})$ in (27)

Reconstruction: Deep autoencoders

C. Autoencoders

Autoencoders are reconstruction models that use neural networks for the encoding and decoding of data. They were originally introduced during the 80s [376]–[379] primarily as methods to perform non-linear dimensionality reduction [380], [381], yet they have also been studied early on for anomaly detection [351], [352]. Today, deep autoencoders are among the most widely adopted methods for deep anomaly detection in the literature [44], [51], [54], [125]–[135] likely owing to their long history and easy-to-use standard variants. The standard autoencoder objective is given by

$$\min_{\omega} \quad \frac{1}{n}\sum_{i=1}^{n} \|\boldsymbol{x}_i - (\phi_d \circ \phi_e)_\omega(\boldsymbol{x}_i)\|^2 + \mathcal{R}, \tag{28}$$

which is a realization of the general reconstruction objective (23) with $\theta = \omega$, that is, the optimization is carried out over the weights ω of the neural network encoder and decoder. A common way to regularize autoencoders is by mapping to a lower dimensional 'bottleneck' representation $\phi_e(\boldsymbol{x}) = \boldsymbol{z} \in \mathcal{Z}$ through the encoder network, which enforces data compression and effectively limits the dimensionality of the manifold or subspace to be learned. If linear networks

Reconstruction: DAE, CAE

Denoising autoencoders (DAEs) [391], [392] explicitly feed noise-corrupted inputs $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \varepsilon$ into the network which is then trained to reconstruct the original inputs $\boldsymbol{x}$. DAEs thus provide a way to specify a noise model for ε (cf., II-C2), which has been applied for noise-robust acoustic novelty detection [42], for instance. In situations in which the training data is already corrupted with noise or unknown anomalies, robust deep autoencoders [127], which split the data into well-represented and corrupted parts similar to robust PCA [374], have been proposed. Contractive autoencoders (CAEs) [393] propose to penalize the Frobenius norm of the Jacobian of the encoder activations with respect to the inputs to obtain a smoother and more robust latent representation. Such ways of regularization influence the geometry and shape of the subspace or manifold that is learned by the autoencoder, for example by imposing some degree of smoothness or introducing invariances towards certain types of input corruptions or transformations [131]. Hence, these regularization choices should again reflect the specific assumptions of a given anomaly detection task.

Reconstruction: VAE

Besides the deterministic variants above, probabilistic autoencoders have also been proposed, which again establish a connection to density estimation. The most explored class of probabilistic autoencoders are Variational Autoencoders (VAEs) [272]–[274], as introduced in section III-C1 through the lens of neural generative models, which approximately maximize the data likelihood (or evidence) by maximizing the ELBO. From a reconstruction perspective, VAEs adopt a stochastic autoencoding process, which is realized by encoding and decoding the parameters of distributions (e.g., Gaussians) through the encoder and decoder networks, from which the latent code and reconstruction then can be sampled. For a standard Gaussian VAE, for example, where $q(\boldsymbol{z}|\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{x}}, \mathrm{diag}(\boldsymbol{\sigma}_{\boldsymbol{x}}^2))$, $p(\boldsymbol{z}) \sim \mathcal{N}(\mathbf{0}, I)$, and $p(\boldsymbol{x}|\boldsymbol{z}) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{z}}, I)$ with encoder $\phi_{e,\omega'}(\boldsymbol{x}) = (\boldsymbol{\mu}_{\boldsymbol{x}}, \boldsymbol{\sigma}_{\boldsymbol{x}})$ and decoder $\phi_{d,\omega}(\boldsymbol{z}) = \boldsymbol{\mu}_{\boldsymbol{z}}$, the empirical ELBO objective (10) becomes

$$\min_{\omega,\omega'} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{M} \Big[\tfrac{1}{2} \|\boldsymbol{x}_i - \boldsymbol{\mu}_{\boldsymbol{z}_{ij}}\|^2 + D_{\mathrm{KL}} \left(\mathcal{N}(\boldsymbol{z}_{ij}; \boldsymbol{\mu}_{\boldsymbol{x}_i}, \mathrm{diag}(\boldsymbol{\sigma}_{\boldsymbol{x}_i}^2)) \| \mathcal{N}(\boldsymbol{z}_{ij}; \mathbf{0}, I) \right) \Big], \tag{29}$$

Reconstruction: VAE

D. Prototypical Clustering

Clustering methods that make the prototype assumption provide another approach to reconstruction-based anomaly detection. As mentioned above, the reconstruction error here is usually given by the distance of a point to its nearest prototype, which ideally has been learned to represent a distinct mode of the normal data distribution. Prototypical clustering methods [403] include the well-known Vector Quantization (VQ) algorithms k-means, k-medians, and k-medoids, which define a Voronoi partitioning [404], [405] over the metric space where they are applied — typically the input space $\mathcal{X}$. Kernel variants of k-means have also been studied [406] and considered for anomaly detection [316]. GMMs with a finite number of k mixtures (cf., section III-A) have been used for (soft) prototypical clustering as well. Here, the distance to each cluster (or mixture component) is given by the Mahalanobis distance that is defined by the covariance matrix of the respective Gaussian mixture component [261].

TABLE II

ANOMALY DETECTION METHODS IDENTIFIED WITH OUR UNIFYING VIEW (LAST COLUMN CONTAINS REPRESENTATIVE REFERENCES).

Method	Loss $\ell(s, y)$	Model $f_\theta(\boldsymbol{x})$	Feature Map $\phi(\boldsymbol{x})$		Parameter θ	Regularization $\mathcal{R}(f, \phi, \theta)$	Bayes?	References
Parametric Density	$-\log(s)$	$p(\boldsymbol{x}\|\theta)$	$\boldsymbol{x}$	(input)	θ	choice of density class $\{p_\theta \mid \theta \in \Theta\}$	✗	[414], [415]
Gaussian/Mahalanobis	$-\log(s)$	$\mathcal{N}(\boldsymbol{x}\|\boldsymbol{\mu}, \Sigma)$	$\boldsymbol{x}$	(input)	$(\boldsymbol{\mu}, \Sigma)$	–	✗	[414], [415]
GMM	$-\log(s)$	$\sum_k \pi_k \mathcal{N}(\boldsymbol{x}\|\boldsymbol{\mu}_k, \Sigma_k)$	$\boldsymbol{x}$	(input)	$(\pi, \boldsymbol{\mu}, \Sigma)$	number of mixture components K	latent	[416]
KDE	$-\log(s)$	$\exp(-\|\phi_k(\boldsymbol{x}) - \boldsymbol{\mu}\|^2)$	$\phi_k(\boldsymbol{x})$	(kernel)	$\boldsymbol{\mu}$	kernel hyperparameters (e.g., bandwidth h)	✗	[255], [256]
EBMs	$-\log(s)$	$\frac{1}{Z(\theta)} \exp(-E(\phi(\boldsymbol{x}), \boldsymbol{z}; \theta))$	$\phi(\boldsymbol{x})$	(various)	θ	latent prior $p(\boldsymbol{z})$	latent	[264], [271]
Normalizing Flows	$-\log(s)$	$p_{\boldsymbol{z}}(\phi_\omega^{-1}(\boldsymbol{x}))\,\lvert\det J_{\phi_\omega^{-1}}(\boldsymbol{x})\rvert$	$\phi_\omega(\boldsymbol{x})$	(neural)	ω	base distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$; diffeomorphism architecture	✗	[283], [288]
GAN (D-based)	$-\log(s)$	$\sigma(\langle \boldsymbol{w}, \psi_\omega(\boldsymbol{x})\rangle)$	$\psi_\omega(\boldsymbol{x})$	(neural)	$(\boldsymbol{w}, \omega)$	adversarial training	✗	[56], [339]
Min. Vol. Sphere	$\max(0, s)$	$\|\boldsymbol{x} - \boldsymbol{c}\|^2 - R^2$	$\boldsymbol{x}$	(input)	$(\boldsymbol{c}, R)$	νR^2	✗	[224]
Min. Vol. Ellipsoid	$\max(0, s)$	$(\boldsymbol{x} - \boldsymbol{c})^\top \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{c}) - R^2$	$\boldsymbol{x}$	(input)	$(\boldsymbol{c}, R, \Sigma)$	$\nu(\frac{1}{2}\|\Sigma\|_{\text{Fr}}^2 + R^2)$	✗	[307]
SVDD	$\max(0, s)$	$\|\phi_k(\boldsymbol{x}) - \boldsymbol{c}\|^2 - R^2$	$\phi_k(\boldsymbol{x})$	(kernel)	$(\boldsymbol{c}, R)$	νR^2	✗	[7]
Semi-Sup. SVDD	$\max(0, ys)$	$\|\phi_k(\boldsymbol{x}) - \boldsymbol{c}\|^2 - R^2$	$\phi_k(\boldsymbol{x})$	(kernel)	$(\boldsymbol{c}, R)$	νR^2	✗	[7], [229]
Soft Deep SVDD	$\max(0, s)$	$\|\phi_\omega(\boldsymbol{x}) - \boldsymbol{c}\|^2 - R^2$	$\phi_\omega(\boldsymbol{x})$	(neural)	$(\boldsymbol{c}, R, \omega)$	νR^2; weight decay; collapse reg. (various)	✗	[137]
OC Deep SVDD	s	$\|\phi_\omega(\boldsymbol{x}) - \boldsymbol{c}\|^2$	$\phi_\omega(\boldsymbol{x})$	(neural)	$(\boldsymbol{c}, \omega)$	weight decay; collapse reg. (various)	✗	[137]
Deep SAD	s^y	$\|\phi_\omega(\boldsymbol{x}) - \boldsymbol{c}\|^2$	$\phi_\omega(\boldsymbol{x})$	(neural)	$(\boldsymbol{c}, \omega)$	weight decay	✗	[144]
OC-SVM	$\max(0, s)$	$\rho - \langle \boldsymbol{w}, \phi_k(\boldsymbol{x})\rangle$	$\phi_k(\boldsymbol{x})$	(kernel)	$(\boldsymbol{w}, \rho)$	$\nu(\frac{1}{2}\|\boldsymbol{w}\|^2 - \rho)$	✗	[6]
OC-NN	$\max(0, s)$	$\rho - \langle \boldsymbol{w}, \phi_\omega(\boldsymbol{x})\rangle$	$\phi_\omega(\boldsymbol{x})$	(neural)	$(\boldsymbol{w}, \rho, \omega)$	$\nu(\frac{1}{2}\|\boldsymbol{w}\|^2 - \rho)$; weight decay	✗	[328]
Bayesian DD	$\max(0, s)$	$\|\phi_k(\boldsymbol{x}) - \boldsymbol{c}\|^2 - R^2$	$\phi_k(\boldsymbol{x})$	(kernel)	$(\boldsymbol{c}, R)$	$\boldsymbol{c} = \sum_i \alpha_i \phi_k(\boldsymbol{x}_i)$ with prior $\alpha \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$	fully	[323]
GT	$-\log(s)$	$\prod_k \sigma_k(\langle \boldsymbol{w}, \phi_\omega(T_k(\boldsymbol{x}))\rangle)$	$\phi_\omega(\boldsymbol{x})$	(neural)	$(\boldsymbol{w}, \omega)$	transformations $\mathcal{T} = \{T_1, \ldots, T_K\}$ for self-labeling	✗	[152], [153]
GOAD (CE)	$-\log(s)$	$\prod_k \sigma_k(-\|\phi_\omega(T_k(\boldsymbol{x})) - \boldsymbol{c}_k\|^2)$	$\phi_\omega(\boldsymbol{x})$	(neural)	$(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K, \omega)$	transformations $\mathcal{T} = \{T_1, \ldots, T_K\}$ for self-labeling	✗	[155]
BCE (supervised)	$-y\log(s) - \frac{1-y}{2}\log(1-s)$	$\sigma(\langle \boldsymbol{w}, \phi_\omega(\boldsymbol{x})\rangle)$	$\phi_\omega(\boldsymbol{x})$	(neural)	$(\boldsymbol{w}, \omega)$	weight decay	✗	[332]
BNN (supervised)	$-y\log(s) - \frac{1-y}{2}\log(1-s)$	$\sigma(\langle \boldsymbol{w}, \phi_\omega(\boldsymbol{x})\rangle)$	$\phi_\omega(\boldsymbol{x})$	(neural)	$(\boldsymbol{w}, \omega)$	prior $p(\boldsymbol{w}, \omega)$	fully	[417], [418]
PCA	s	$\|\boldsymbol{x} - W^\top W \boldsymbol{x}\|_2^2$	$\boldsymbol{x}$	(input)	W	$WW^\top = I$	✗	[365]
Robust PCA	s	$\|\boldsymbol{x} - W^\top W \boldsymbol{x}\|_1$	$\boldsymbol{x}$	(input)	W	$WW^\top = I$	✗	[372]
Probabilistic PCA	$-\log(s)$	$\mathcal{N}(\boldsymbol{x}\|\boldsymbol{0}, W^\top W + \sigma^2 I)$	$\boldsymbol{x}$	(input)	(W, σ^2)	linear latent Gauss model $\boldsymbol{x} = W^\top \boldsymbol{z} + \varepsilon$	latent	[362]
Bayesian PCA	$-\log(s)$	$\mathcal{N}(\boldsymbol{x}\|\boldsymbol{0}, W^\top W + \sigma^2 I)\, p(W\|\boldsymbol{\alpha})$	$\boldsymbol{x}$	(input)	(W, σ^2)	linear latent Gauss model with prior $p(W\|\boldsymbol{\alpha})$	fully	[363]
Kernel PCA	s	$\|\phi_k(\boldsymbol{x}) - W^\top W \phi_k(\boldsymbol{x})\|^2$	$\phi_k(\boldsymbol{x})$	(kernel)	W	$WW^\top = I$	✗	[3], [4]
Autoencoder	s	$\|\boldsymbol{x} - \phi_\omega(\boldsymbol{x})\|_2^2$	$\phi_\omega(\boldsymbol{x})$	(neural)	ω	advers. (AAE), contract. (CAE), denois. (DAE), etc.	✗	[127], [135]
VAE	$-\log(s)$	$p_{\phi_\omega}(\boldsymbol{x}\|\boldsymbol{z})$	$\phi_\omega(\boldsymbol{x})$	(neural)	ω	latent prior $p(\boldsymbol{z})$	latent	[274], [278]
GAN (G-based)	$-\log(s)$	$p_{\phi_\omega}(\boldsymbol{x}\|\boldsymbol{z})$	$\phi_\omega(\boldsymbol{x})$	(neural)	ω	adversarial training and latent prior $p(\boldsymbol{z})$	latent	[50], [147]
k-means	s	$\|\boldsymbol{x} - \text{argmin}_{\boldsymbol{c}_k} \|\boldsymbol{x} - \boldsymbol{c}_k\|_2\|_2^2$	$\boldsymbol{x}$	(input)	$(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$	number of prototypes K	✗	[403], [416]
k-medians	s	$\|\boldsymbol{x} - \text{argmin}_{\boldsymbol{c}_k} \|\boldsymbol{x} - \boldsymbol{c}_k\|_1\|_1$	$\boldsymbol{x}$	(input)	$(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$	number of prototypes K	✗	[403]
VQ	s	$\|\boldsymbol{x} - \phi_d(\text{argmin}_{\boldsymbol{c}_k} \|\phi_e(\boldsymbol{x}) - \boldsymbol{c}_k\|)\|$	$\phi(\boldsymbol{x})$	(various)	$(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$	number of prototypes K	✗	[360], [361]

Distance based

these methods, there also exists a rich literature on purely 'distance-based' anomaly detection methods and algorithms that have been studied extensively in the data mining community in particular. Many of these algorithms follow a *lazy learning* paradigm, in which there is no a priori training phase of learning a model, but instead new test points are evaluated with respect to the training instances only as they occur. We here group these methods as 'distance-based' without further granularity, but remark that various taxonomies for these types of methods have been proposed [161], [179]. Examples of such methods include nearest-neighbor-based methods [8], [9], [420]–[422] such as LOF [10] and partitioning tree-based methods [423] such as Isolation Forest [424], [425]. These methods usually also aim to capture the high-density regions of the data in some manner, for instance by scaling distances in relation to local neighborhoods [10], and thus are mostly consistent with the formal anomaly detection problem definition presented in section II. The majority of these algorithms have

Benchmarks

TABLE III
EXISTING ANOMALY DETECTION BENCHMARKS.

k-classes-out	(Fashion-)MNIST, CIFAR-10, STL-10, ImageNet
Synthetic	MNIST-C [428], ImageNet-C [429], ImageNet-P [429], ImageNet-O [434]
Real-world	*Industrial:* MVTec-AD [190], PCB [435] *Medical:* CAMELYON16 [60], [436], NIH Chest X-ray [60], [437], MOOD [438], HCP/BRATS [51], Neuropathology [59], [124] *Security:* Credit-card-fraud [439], URL [440], UNSW-NB15 [441] *Time series:* NAB [442], Yahoo [443] *Misc.:* Emmott [433], ELKI [444], ODDS [445], UCI [446], [447]

Timeseries Anomaly Detection using Temporal Hierarchical One-Class Network

Lifeng Shen[1], Zhuocong Li[2], James T. Kwok[1]
[1] Department of Computer Science and Engineering,
Hong Kong University of Science and Technology, Hong Kong
{lshenae,jamesk}@cse.ust.hk
[2] Cloud and Smart Industries Group, Tencent, China
zhuocongli@tencent.com

NIPS 2020

Abstract

Real-world timeseries have complex underlying temporal dynamics and the detection of anomalies is challenging. In this paper, we propose the Temporal Hierarchical One-Class (THOC) network, a temporal one-class classification model for timeseries anomaly detection. It captures temporal dynamics in multiple scales by using a dilated recurrent neural network with skip connections. Using multiple hyperspheres obtained with a hierarchical clustering process, a one-class objective called Multiscale Vector Data Description is defined. This allows the temporal dynamics to be well captured by a set of multi-resolution temporal clusters. To further facilitate representation learning, the hypersphere centers are encouraged to be orthogonal to each other, and a self-supervision task in the temporal domain is added. The whole model can be trained end-to-end. Extensive empirical studies on various real-world timeseries demonstrate that the proposed THOC network outperforms recent strong deep learning baselines on timeseries anomaly detection.

Challenge

An effective timeseries anomaly detection method should be able to model the complex nonlinear temporal dynamics of the underlying system's normal behavior, while being robust and can be generalized to unseen anomalies. However, the development of such a technique is very challenging. First, real-world timeseries have highly nonlinear temporal dependencies and complex interactions among variables. Moreover, anomalies are often rare. The finding and labeling of these anomalies are very time-consuming and expensive in practice. Hence, timeseries anomaly detection is usually formulated in the unsupervised learning setting [28], which is also the focus in this paper.

One-class

One-class classification [19] is a popular paradigm for anomaly detection. The idea is that by assuming that most of the training data are normal, their characteristics are captured and learned by a model. An outlier is then detected when the current observation cannot be well-fitted by the model. Two well-known and closely related one-class classification models are the one-class support vector machine (OC-SVM) [26], which uses a hyperplane to separate the normal data from anomalous data; and the support vector data description (SVDD) [29], which uses a hypersphere to enclose the normal data. While they have been successfully used in many real-world applications [3, 15, 32], they are often limited to data-rich scenarios so that the normal patterns can be sufficiently captured.

One-class

The OC-SVM and SVDD rely on the kernel trick [27] to map input features to a high-dimensional space for data separation. Motivated by the immense success of deep learning in various applications such as computer vision, speech recognition and natural language processing [12, 25], recent efforts try to integrate the powerful representation learning ability of deep networks into the traditional one-class classifiers. For example, deep SVDD [22] replaces the kernel-induced feature space in SVDD by the feature space learned in a deep network. DAGMM [34] is a density-based one-class classifier that integrates a deep autoencoder with the Gaussian mixture model (GMM), such that the normal data can be well-captured by the GMM in a low-dimensional latent space. As in other deep networks, these models can all be conveniently trained via back-propagation in an end-to-end manner.

Time series data

The above-mentioned traditional/deep one-class classifiers are designed for fixed-dimensional input data. It is still an open issue on how to extend them for timeseries anomaly detection. A simple approach is to run a sliding window on the timeseries data. A fixed-dimensional vector containing the history information is then extracted and fed to the one-class classifier. However, this fails to adequately capture the underlying temporal dependency. To alleviate this problem, a number of models based on recurrent networks have been recently proposed for timeseries anomaly detection. An early attempt is a LSTM-based encoder-decoder model [17], and an anomaly score is defined based on the reconstruction error on the timeseries. However, it suffers from error accumulation on decoding a long sequence. Other more powerful deep generative models, such as the recurrent variational autoencoder [28], and variants of the generative adversarial network (GAN) (e.g., MAD-GAN [13] and BeatGAN [33]) have also been proposed. However, training of the GAN is usually difficult, and requires a careful balance between the discriminator and generator [10].

Contribution

Inspired by deep SVDD, we propose in this paper the Temporal Hierarchical One-Class (THOC) network. First, it uses the dilated recurrent neural network (RNN) [2] with skip connections to efficiently extract multi-scale temporal features from the timeseries. Instead of using only the lowest-resolution features obtained at the top layer of the dilated RNN, THOC fuses features from all intermediate layers together by a differentiable hierarchical clustering mechanism. At each resolution, normal behaviors are represented by multiple hyperspheres. This captures the complex characteristics in real-world timeseries data, and is more powerful than the use of a single hypersphere in deep SVDD. A multiscale support vector data description (MVDD), which is a one-class objective defined based on the difference between the fused multiscale features and hypersphere centers, allows the whole model to be trained end-to-end. Finally, a novelty score, which measures how the current observation deviates from the normal behaviors represented by the hyperspheres, is used for anomaly detection on an unseen observation. Experiments performed on a number of real-world timeseries data sets show that the proposed model outperforms the recent state-of-the-arts.

Recap: SVDD and deep SVDD

$$\min_{\mathbf{c},R,\xi} \quad R^2 + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i$$

$$\text{s.t.} \qquad \|\phi(\mathbf{x}_i) - \mathbf{c}\|^2 \leq R^2 + \xi_i,\ \xi_i \geq 0 \quad \forall i = 1, \ldots, N,$$

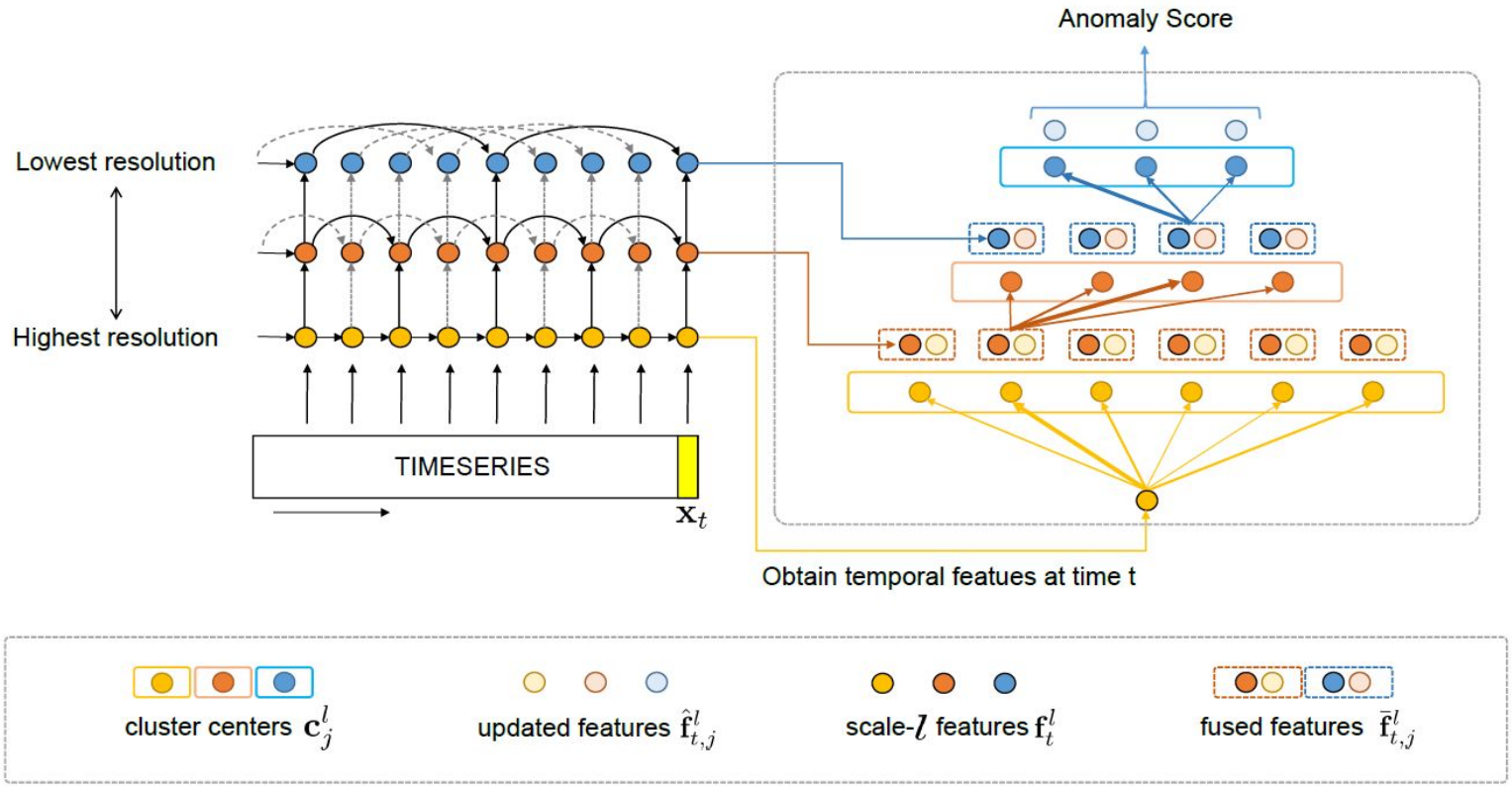
$$\min_{R,\mathcal{W}} \ R^2 + \frac{1}{\nu N}\sum_{i=1}^{N}\max\{0, \|\mathrm{NN}(\mathbf{x}_i;\mathcal{W}) - \mathbf{c}\|^2 - R^2\} + \lambda\Omega(\mathcal{W}),$$

$$\min_{\mathcal{W}} \frac{1}{N}\sum_{i=1}^{N}\|\mathrm{NN}(\mathbf{x}_i;\mathcal{W}) - \mathbf{c}\|^2 + \lambda\Omega(\mathcal{W}).$$

Problem Description

In timeseries anomaly detection, we are given a set of timeseries $\mathcal{D} = \{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$. $\mathbf{X}_s$ is of length T_s, and its observation at time t is $\mathbf{x}_{t,s} \in \mathbb{R}^D$. The task is to determine if $\mathbf{x}_{t,s}$ is anomalous, based on the partial timeseries $\mathbf{x}_{1:t,s}$ that have been observed so far. Following [17, 28, 13, 31], we consider the unsupervised learning setting, and do not use any label information during training. This is more practical as labeled anomalies are rare and often difficult to identify.

Temporal Hierarchical One-Class (THOC)



Multiple Temporal Features

To extract multiscale temporal features from the timeseries, we use an L-layer dilated recurrent neural network (RNN) [2] with multi-resolution recurrent skip connections. Other networks capable of extracting multiscale features (such as the WaveNet [20]) can also be used. For simplicity of notations, we drop the subscript s in this section. At time t, let the (partial) timeseries that has been observed so far be $\mathbf{x}_{1:t-1}$. For a particular layer l, the hidden state $\mathbf{f}_t^l$ of the recurrent cell is:

$$\mathbf{f}_t^l = \begin{cases} \mathcal{F}_{\text{RNN}}(\mathbf{x}_t, \mathbf{f}_{t-s^{(l)}}^l) & \text{if } l = 1, \\ \mathcal{F}_{\text{RNN}}(\mathbf{f}_t^{l-1}, \mathbf{f}_{t-s^{(l)}}^l) & \text{otherwise,} \end{cases} \tag{4}$$

where $\mathcal{F}_{\text{RNN}}$ is any RNN cell (such as the vanilla RNN cell, LSTM or GRU), and $\mathbf{f}_{t-s^{(l)}}^l$ is the input for the skip connection with skip length $s^{(l)}$. The use of skip connections helps model long-term dependencies in the timeseries and alleviates the problem of vanishing gradient. The larger the $s^{(l)}$, the longer the dependency layer l captures. Similar to the WaveNet [20], an exponentially increasing skip dilation is used here: $s^{(l)} = M_0 \prod_{i=1}^{l-1} M$ (in Figure 1, $M_0 = 1$ and $M = 2$). With multiple layers, the dilated-RNN can extract an abundance of multiscale temporal features, with shorter-term information learned at the lower layers and longer-term information at the upper layers.

Fusing the Multiscale Features

Instead of only using features from dilated RNN's last layer, features from intermediate layers may also contain useful information. In this section, we propose a differentiable hierarchical clustering procedure to fuse information from all these different scales.

For a particular scale $l \in \{1, \ldots, L\}$, there is a corresponding clustering layer with K^l clusters whose centers are $\{\mathbf{c}_1^l, \ldots, \mathbf{c}_{K^l}^l\}$. At time t, the inputs to this layer are the outputs from the previous layer $\{\bar{\mathbf{f}}_{t,1}^{l-1}, \ldots, \bar{\mathbf{f}}_{t,K^{l-1}}^{l-1}\}$ (which will be detailed in (8)). When $l = 1$, we use the temporal features $\mathbf{f}_t^1$ from (4) as input (and thus $K^0 = 1$). The hierarchical clustering procedure proceeds by alternating the following two steps.

Fusing the Multiscale Features

Step 1 (Assignment): Based on the similarities between $\bar{\mathbf{f}}_{t,i}^{l-1}$ and each center $\mathbf{c}_j^l$, we assign $\bar{\mathbf{f}}_{t,i}^{l-1}$ to these centers with probabilities:

$$P_{t,i\to j}^l = P(\bar{\mathbf{f}}_{t,i}^{l-1} \to \mathbf{c}_j^l) = \frac{\exp(\text{score}(\bar{\mathbf{f}}_{t,i}^{l-1}, \mathbf{c}_j^l)/\tau)}{\sum_{k=1}^{K^l} \exp(\text{score}(\bar{\mathbf{f}}_{t,i}^{l-1}, \mathbf{c}_k^l)/\tau)}, \tag{5}$$

where $\tau \in (0, \infty)$ is a temperature parameter, and $\text{score}(\cdot, \cdot)$ is a similarity score function. When $\tau \to \infty$, $\bar{\mathbf{f}}_{t,i}^{l-1}$ is assigned to all the centers with equal probabilities. When $\tau \to 0$, the soft assignment becomes hard. As for the score function, we use the simple cosine similarity:

$$\text{score}(\bar{\mathbf{f}}, \mathbf{c}) = \mathbf{f}^\top \mathbf{c}/(\|\mathbf{f}\| \cdot \|\mathbf{c}\|). \tag{6}$$

Other similarity functions can also be used.

Fusing the Multiscale Features

Step 2 (Update): After obtaining the assignment probabilities, all features $\{\bar{\mathbf{f}}_{t,1}^{l-1}, \ldots, \bar{\mathbf{f}}_{t,K^{l-1}}^{l-1}\}$ from the lower level are fused and transformed in each cluster $\mathbf{c}_j^l$ as

$$\hat{\mathbf{f}}_{t,j}^{l} = \sum_{i=1}^{K^{l-1}} P_{t,i\to j}^{l} \mathrm{ReLU}(\mathbf{W}^{l}\bar{\mathbf{f}}_{t,i}^{l-1} + \mathbf{b}^{l}), \quad j = 1, \ldots, K^{l}, \tag{7}$$

where both the weight $\mathbf{W}^l$ and bias $\mathbf{b}^l$ are learned end-to-end with the other model parameters (as will be discussed in Section 3.2). If $l \leq L-1$, $\hat{\mathbf{f}}_{t,j}^{l}$ is then concatenated with the corresponding scale-$(l+1)$ feature $\mathbf{f}_t^{l+1}$ from the dilated RNN, and further transformed by a fully-connected layer $\mathcal{F}_{\texttt{MLP}}$ with output having the same dimension as $\hat{\mathbf{f}}_{t,j}^{l}$. Note that $\hat{\mathbf{f}}_{t,j}^{L}$'s are directly used as output at the last layer.

$$\bar{\mathbf{f}}_{t,j}^{l} = \begin{cases} \mathbf{f}_t^1 & \text{if } l = 0 \\ \mathcal{F}_{\texttt{MLP}}([\hat{\mathbf{f}}_{t,j}^{l}; \mathbf{f}_t^{l+1}]) & \text{if } 1 \leq l \leq L-1 \\ \hat{\mathbf{f}}_{t,j}^{L} & \text{otherwise (i.e., } l = L) \end{cases}. \tag{8}$$

Multiple Support Vector Data Description (MVDD)

As in deep SVDD, we measure the difference between features $\{\bar{\mathbf{f}}_{t,j}^L\}$ and centers $\{\mathbf{c}_1^L, \ldots, \mathbf{c}_{K^L}^L\}$ at the last layer. As we use the cosine similarity in (6), the cosine distance (i.e., $1 -$ cosine similarity) is used as $d(\bar{\mathbf{f}}_{t,s}^L, \mathbf{c}_k^L)$. Let $\mathcal{W}$ be all the learnable weights in the network. Our objective is:

$$\mathcal{L}_{\text{THOC}} = \frac{1}{NK^L} \sum_{s=1}^{N} \frac{1}{T_s} \sum_{t=1}^{T_s} \sum_{j=1}^{K^L} R_{t,j,s}^L d(\bar{\mathbf{f}}_{t,j,s}^L, \mathbf{c}_j^L) + \lambda \Omega(\mathcal{W}), \tag{9}$$

where $\Omega(\mathcal{W})$ is the ℓ_2-regularizer. Note that we have explicitly added back the subscript s for samples.

The following shows that $R_{t,j,s}^L$ can be computed easily in a recursive manner. For simplicity of notations, we drop the subscript s here. At the first layer, the only input is $\mathbf{f}_t^1$, and so $R_{t,j}^1 = P_{t,1\to j}^1$ in (5). For layer l, $R_{t,j}^l$ is obtained by a softmax over all centers in the same layer:

$$R_{t,j}^l = \frac{\exp(\tilde{R}_{t,j}^l)}{\sum_{i=1}^{K^l} \exp(\tilde{R}_{t,i}^l)}, \text{ where } \tilde{R}_{t,j}^l = \begin{cases} P_{t,i\to j}^1, & \text{if } l = 1 \\ \sum_{i=1}^{K^{l-1}} P_{t,i\to j}^l R_{t,i}^{l-1} & \text{if } 1 < l \leq L \end{cases}. \tag{10}$$

Diverse Center

To allow the centers $\mathbf{c}_k^l$'s in each layer to be as diverse as possible, we add the following loss

$$\mathcal{L}_{\text{orth}} = \frac{1}{L}\sum_{l=1}^{L} \|(\mathbf{C}^l)^\top \mathbf{C}^l - \mathbf{I}\|_F^2, \tag{11}$$

where $\mathbf{C}^l = [\mathbf{c}_1^l \cdots \mathbf{c}_{K^l}^l]$, $\mathbf{I}$ is the identity matrix, and $\|\cdot\|_F$ is the Frobenius norm. This encourages $\mathbf{c}_k^l$'s to be orthogonal to each other.

Temporal Self Supervision Loss

Moreover, self-supervision [4], which constructs related auxiliary tasks to aid in the learning of informative features, has recently shown to be an effective unsupervised representation learning method in many real-world applications [4, 11, 7]. For timeseries data, a natural self-supervised learning task is multi-step-ahead prediction. Here, to encourage the learning of useful features at all layers of the dilated-RNN, we use a linear model (with learnable weight $\mathbf{W}^l_{\text{pred}}$ in each layer l) to predict $\mathbf{x}_{t,s}$ from the corresponding layer-l hidden state $\mathbf{f}^l_{t-s^{(l)},s}$ at time $t - s^{(l)}$. This leads to the following temporal self-supervision loss (TSS):

$$\mathcal{L}_{\text{TSS}} = \frac{1}{NL} \sum_{s=1}^{N} \sum_{l=1}^{L} \left(\frac{1}{T_s - s^{(l)}} \sum_{t=s^{(l)}+1}^{T_s} \|\mathbf{W}^l_{\text{pred}} \mathbf{f}^l_{t-s^{(l)},s} - \mathbf{x}_{t,s}\|^2 \right). \tag{12}$$

Final Objective

Combining all three losses, we obtain the following *multiscale support vector data description* (MVDD) objective:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{THOC}} + \lambda_{\text{orth}}\mathcal{L}_{\text{orth}} + \lambda_{\text{TSS}}\mathcal{L}_{\text{TSS}}, \tag{13}$$

where λ_{orth} and λ_{TSS} are tradeoff hyperparameters. All the parameters can be learned in an end-to-end manner. The whole procedure is shown in Algorithm 1.

THOC: recap

Algorithm 1 Temporal hierarchical one-class learning (THOC).

Input: timeseries $\mathbf{X}_s = (\mathbf{x}_{1,s}, \mathbf{x}_{2,s}, \ldots, \mathbf{x}_{T_s,s})$; number of centers $\{K^l\}$; skip lengths $\{s^{(l)}\}$.

repeat
  feed $\mathbf{x}_{t,s}$ into the L-layer dilated-RNN and obtain $\{\mathbf{f}_t^l\}$ from each layer;
  for layer $l = 1, \ldots, L$ **do**
    obtain the lth clustering layer's input $\{\bar{\mathbf{f}}_{t,i}^{l-1}\}_{i=1,\ldots,K^{l-1}}$ by (8) where $K^0 = 1$;
    compute probabilities $\{P_{t,i\to j}^l\}_{i=1,\ldots,K^{l-1},j=1,\ldots,K^l}$ from (5);
    compute $\{R_{t,j}^l\}_{j=1,\ldots,K^l}$ for each cluster center at layer l from (10);
    update and obtain output features $\{\hat{\mathbf{f}}_{t,j}^l\}_{j=1,\ldots,K^l}$ from (7);
  end for
  minimize MVDD objective in (13) by the Adam optimizer;
until convergence.

Experiment: Data

(i) *2D-gesture* [9], which records the X-Y coordinate sequences of hand gestures in a video;

(ii) *Power demand*[9], which contains a year of power demand at a Dutch research facility;

(iii) *KDD-Cup99* data from the DARPA'98 Intrusion Detection Evaluation Program [14]. It contains around seven million network traffic connection records over a 7-week period. A connection is a sequence of TCP packets. Each record is labeled as either normal or attack;

(iv) Secure Water Treatment (*SWaT*) data [18], which is collected from a water treatment testbed over 11 days. 36 attacks were launched during the last 4 days of the collection process. These attacks were launched with different intents and diverse lasting durations (from a few minutes to an hour);[1]

(v) Mars Science Laboratory rover (*MSL*); and

(vi) Soil Moisture Active Passive satellite (*SMAP*) data: Both *MSL* and *SMAP* are public data sets from NASA [8]. They contain telemetry anomaly data derived from the Incident Surprise Anomaly (ISA) reports of spacecraft monitoring systems. Each data set has a training and a testing set. Anomalies in the testing set are labeled, while the training set contains unlabeled anomalies.

Experiment: Data

Table 1: Statistics of the data sets used.

	dim	length	#training	#validation	#testing
2D-gesture	2	80	8,170	420	2,420
power-demand	1	80	18,145	4,786	10,000
KDD-Cup99	34	100	56,139	24,601	24,602
SWaT	51	100	47,420	22,396	22,396
MSL	55	100	40,721	17,396	73,629
SMAP	25	100	94,528	40,455	427,517

Experiment: Results

Table 2: Precision (prec), recall (rec) and F1 score results (as %) on various data sets. The number in brackets after the F1 value is the rank of the method. The smaller the better.

	2D-gesture			power-demand			KDD-Cup99			SWaT			avg
	prec	rec	F1	prec	rec	F1	prec	rec	F1	prec	rec	F1	rank
LOF	27.82	87.21	42.18 (8)	15.29	28.13	19.81 (9)	95.38	99.55	97.42 (11)	76.97	98.36	86.36 (7)	8.75
OC-SVM	65.50	25.57	36.78 (14)	12.40	60.43	20.58 (8)	95.25	99.92	97.53 (10)	99.47	61.47	75.98 (13)	11.25
iso forest	28.54	68.04	40.22 (10)	7.85	89.77	14.44 (13)	96.85	99.38	98.10 (7)	99.00	74.47	85.00 (9)	9.75
deep SVDD	26.26	64.53	37.32 (13)	11.51	64.74	19.54 (10)	89.83	100.0	94.64 (14)	97.68	71.88	82.82 (11)	12
AnoGAN	57.85	46.50	51.55 (4)	20.28	44.41	28.85 (5)	93.11	99.93	96.40 (12)	99.01	77.01	86.64 (5)	6.5
DAGMM	25.66	80.47	38.91 (12)	34.37	41.72	37.69 (4)	96.12	99.70	97.86 (8)	90.60	80.72	85.38 (8)	8.0
EncDec-AD	24.88	100.0	39.85 (11)	13.98	54.20	22.22 (6)	89.74	99.50	94.37 (13)	93.69	63.31	75.56 (14)	11
LSTM-VAE	36.62	67.76	47.54 (6)	8.00	56.66	14.03 (14)	98.84	98.09	98.47 (3)	98.39	77.01	86.39 (6)	7.25
MadGAN	29.41	76.40	42.47 (7)	13.20	60.57	21.67 (7)	96.73	99.55	98.12 (6)	98.72	77.60	86.89 (2)	5.5
BeatGAN	55.11	45.33	49.74 (5)	8.04	76.58	14.56 (12)	97.54	98.94	98.23 (5)	88.37	76.41	81.95 (12)	8.5
OmniAnomaly	27.70	79.67	41.11 (9)	8.55	78.73	15.42 (11)	97.63	99.69	98.65 (2)	99.01	77.06	86.67 (4)	6.5
MSCRED	61.26	59.11	60.17 (2.5)	55.80	34.32	42.50 (3)	97.31	99.43	98.36 (4)	98.43	77.69	86.84 (3)	3.125
CVDD	56.05	64.95	60.17 (2.5)	49.65	38.36	43.30 (2)	96.37	98.75	97.54 (9)	97.33	73.21	83.56 (10)	5.875
THOC	54.78	75.00	**63.31** (1)	61.50	36.34	**45.68** (1)	98.20	99.54	**98.86** (1)	98.08	79.94	**88.09** (1)	1.0

Table 3: Comparison of variants using different flat and hierarchical timeseries representations.

			$\mathcal{L}_{\text{orth}}$	$\mathcal{L}_{\text{TSS}}$	prec	recall	F1
flat	one hypersphere	RNN-top	×	✓	31.67	75.70	44.66
		USRL	×	×	59.49	27.10	37.24
	multiple hyperspheres	RNN-top	✓	✓	41.32	68.93	51.66
		USRL	✓	×	50.80	45.40	47.95
hierarchical	one hypersphere	THOC-variant	×	✓	53.27	60.98	56.86
	multiple hyperspheres	THOC	✓	✓	54.78	75.00	**63.31**

4.5.2 Effectiveness of $\mathcal{L}_{\text{orth}}$ and $\mathcal{L}_{\text{TSS}}$

In this experiment, we consider the effectiveness of $\mathcal{L}_{\text{orth}}$ and $\mathcal{L}_{\text{TSS}}$ by dropping one or both components from the objective in (13). Results are shown in Table 4. As can be seen, both $\mathcal{L}_{\text{orth}}$ and the $\mathcal{L}_{\text{TSS}}$ are indeed important. Without the orthogonal loss, the centers may be very similar or even duplicate; without the self-supervised loss (which is based on timeseries prediction), the model may fail to capture temporal dependencies, which are essential for a proper representation of timeseries data.

Table 4: Effectiveness of $\mathcal{L}_{\text{orth}}$ and $\mathcal{L}_{\text{TSS}}$.

$\mathcal{L}_{\text{orth}}$	$\mathcal{L}_{\text{TSS}}$	prec	recall	F1
×	×	52.22	24.77	33.60
✓	×	34.00	67.29	45.17
×	✓	42.08	57.71	48.67
✓	✓	54.78	75.00	**63.31**

Questions?

EXPLAINABLE DEEP ONE-CLASS CLASSIFICATION

Philipp Liznerski[1]* **Lukas Ruff**[2]* **Robert A. Vandermeulen**[2]*
Billy Joe Franks[1] **Marius Kloft**[1] **Klaus-Robert Müller**[2 3 4 5]
[1]ML group, Technical University of Kaiserslautern, Germany
[2]ML group, Technical University of Berlin, Germany
[3]Google Research, Brain Team, Berlin, Germany
[4]Department of Artificial Intelligence, Korea University, Seoul, Republic of Korea
[5]Max Planck Institute for Informatics, Saarbrücken, Germany
{liznerski, franks, kloft}@cs.uni-kl.de
{lukas.ruff, vandermeulen, klaus-robert.mueller}@tu-berlin.de

ICLR 2021

ABSTRACT

Deep one-class classification variants for anomaly detection learn a mapping that concentrates nominal samples in feature space causing anomalies to be mapped away. Because this transformation is highly non-linear, finding interpretations poses a significant challenge. In this paper we present an explainable deep one-class classification method, *Fully Convolutional Data Description* (FCDD), where the mapped samples are themselves also an explanation heatmap. FCDD yields competitive detection performance and provides reasonable explanations on common anomaly detection benchmarks with CIFAR-10 and ImageNet. On MVTec-AD, a recent manufacturing dataset offering ground-truth anomaly maps, FCDD sets a new state of the art in the unsupervised setting. Our method can incorporate ground-truth anomaly explanations during training and using even a few of these (~ 5) improves performance significantly. Finally, using FCDD's explanations, we demonstrate the vulnerability of deep one-class classification models to spurious image features such as image watermarks.[1]

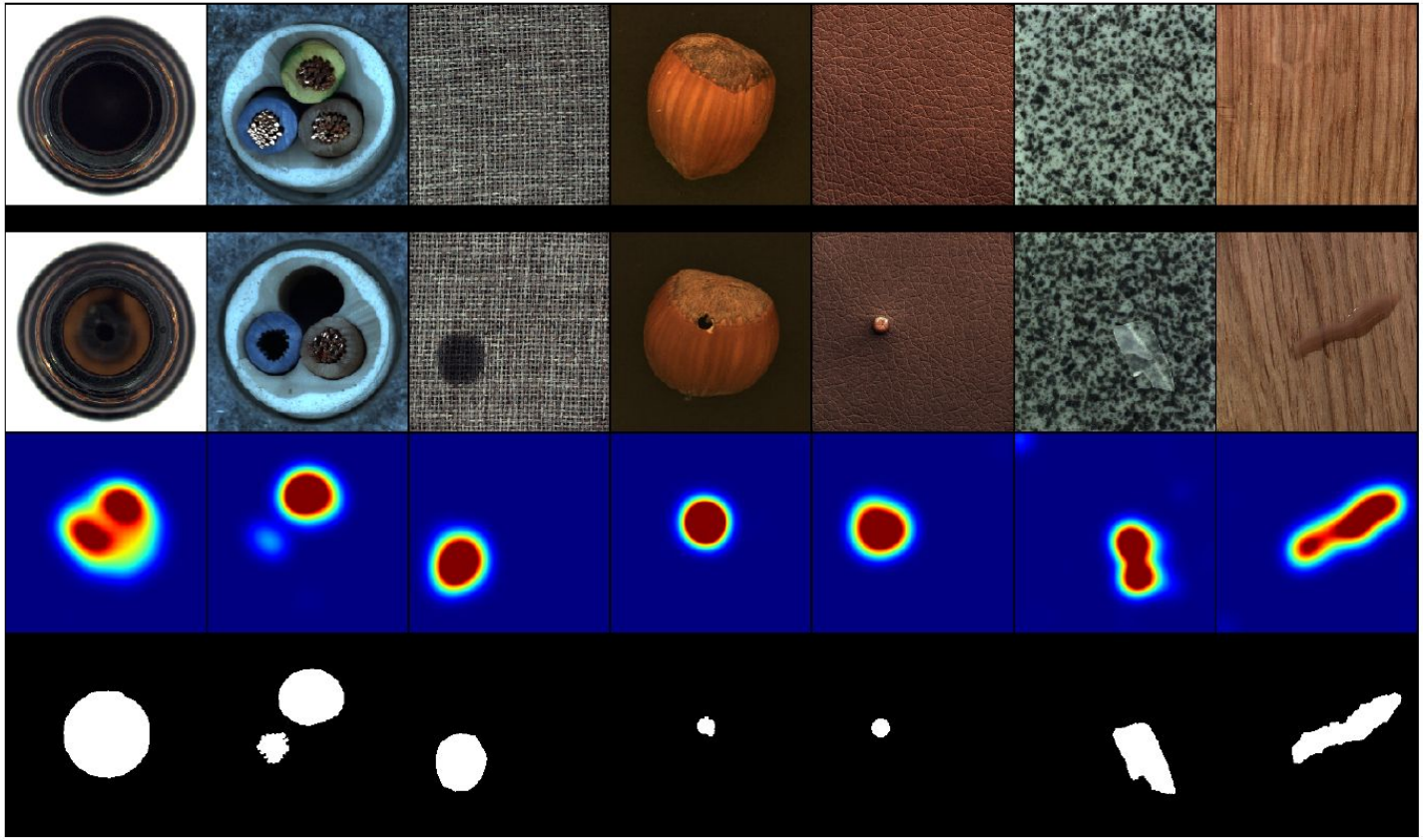


Figure 1: FCDD explanation heatmaps for MVTec-AD (Bergmann et al., 2019). Rows from top to bottom show: (1) nominal samples (2) anomalous samples (3) FCDD anomaly heatmaps (4) ground-truth anomaly maps.

Deep One-Class Classification Deep one-class classification (Ruff et al., 2018; 2020b) performs anomaly detection by learning a neural network to map nominal samples near a center $\mathbf{c}$ in output space, causing anomalies to be mapped away. For our method we use a *Hypersphere Classifier* (HSC) (Ruff et al., 2020a), a recently proposed modification of Deep SAD (Ruff et al., 2020b), a semi-supervised version of DSVDD (Ruff et al., 2018). Let $X_1, \ldots, X_n$ denote a collection of samples and $y_1, \ldots, y_n$ be labels where $y_i = 1$ denotes an anomaly and $y_i = 0$ denotes a nominal sample. Then the HSC objective is

$$\min_{\mathcal{W}, \mathbf{c}} \quad \frac{1}{n} \sum_{i=1}^{n} (1 - y_i) h(\phi(X_i; \mathcal{W}) - \mathbf{c}) - y_i \log \left(1 - \exp\left(-h(\phi(X_i; \mathcal{W}) - \mathbf{c})\right)\right), \tag{1}$$

where $\mathbf{c} \in \mathbb{R}^d$ is the center, and $\phi : \mathbb{R}^{c \times h \times w} \to \mathbb{R}^d$ a neural network with weights $\mathcal{W}$. Here h is the pseudo-Huber loss (Huber et al., 1964), $h(\mathbf{a}) = \sqrt{\|\mathbf{a}\|_2^2 + 1} - 1$, which is a robust loss that interpolates from quadratic to linear penalization. The HSC loss encourages ϕ to map nominal samples near $\mathbf{c}$ and anomalous samples away from the center $\mathbf{c}$. In our implementation, the center $\mathbf{c}$ corresponds to the bias term in the last layer of our networks, i.e. is included in the network ϕ, which is why we omit $\mathbf{c}$ in the FCDD objective below.

Fully Convolutional Architecture Our method uses a *fully convolutional network* (FCN) (Long et al., 2015; Noh et al., 2015) that maps an image to a matrix of features, i.e. $\phi : \mathbb{R}^{c \times h \times w} \rightarrow \mathbb{R}^{1 \times u \times v}$ by using alternating convolutional and pooling layers only, and does not contain any fully connected layers. In this context, pooling can be seen as a special kind of convolution with fixed parameters.

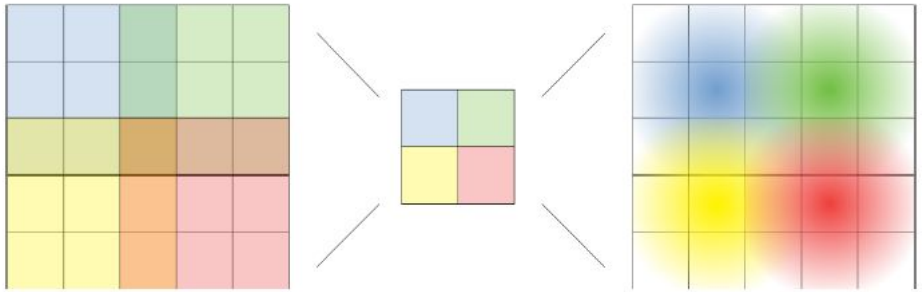


Figure 2: Visualization of a 3×3 convolution followed by a 3×3 transposed convolution with a Gaussian kernel, both using a stride of 2.

A core property of a convolutional layer is that each pixel of its output only depends on a small region of its input, known as the output pixel's *receptive field*. Since the output of a convolution is produced by moving a filter over the input image, each output pixel has the same relative position as its associated receptive field in the input. For instance, the lower-left corner of the output representation has a corresponding receptive field in the lower-left corner of the input image, etc. (see Figure 2 left side). The outcome of several stacked convolutions also has receptive fields of limited size and consistent relative position, though their size grows with the amount of layers. Because of this an FCN preserves spatial information.

Fully Convolutional Data Description

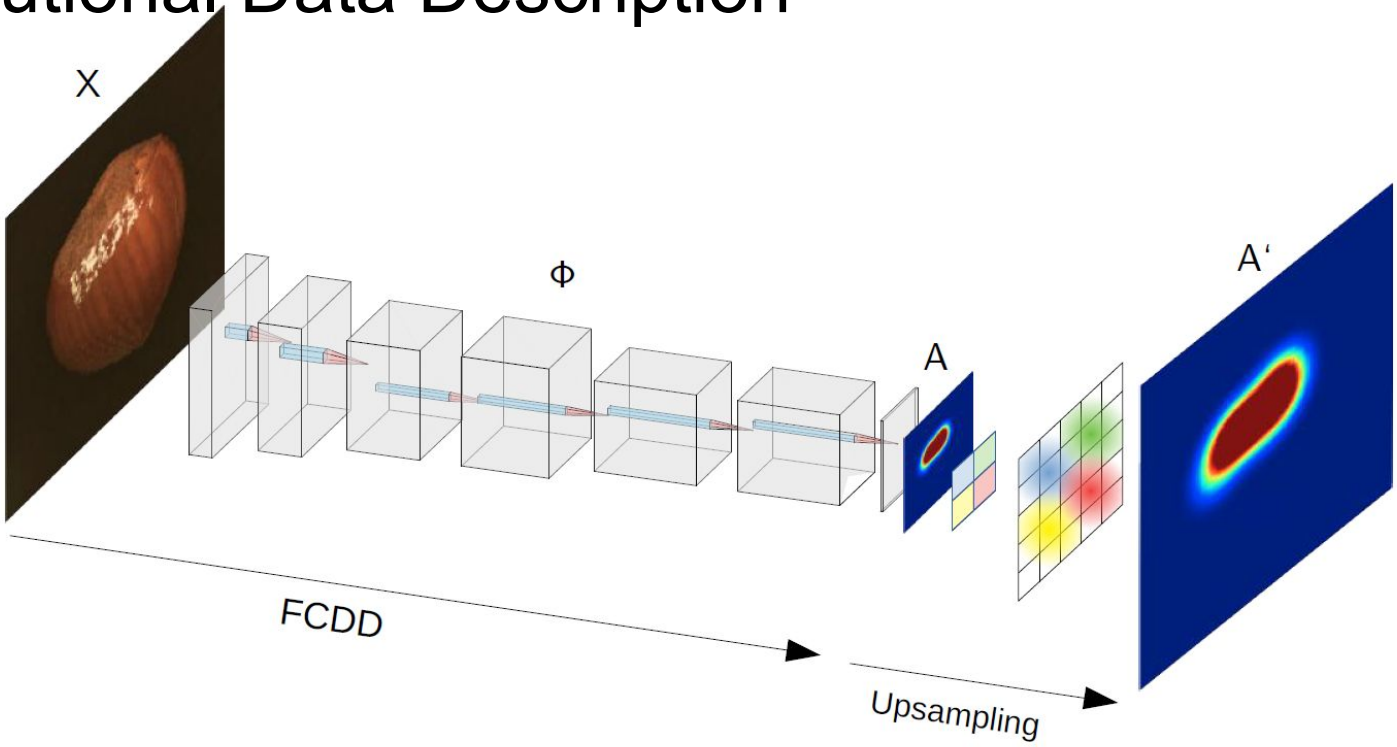


Figure 3: Visualization of the overall procedure to produce full-resolution anomaly heatmaps with FCDD. X denotes the input, ϕ the network, A the produced anomaly heatmap and A' the upsampled version of A using a transposed Gaussian convolution.

Generate Anomalous Samples

FCDD is trained using samples that are labeled as nominal or anomalous. As before, let $X_1, \ldots, X_n$ denote a collection of samples with labels $y_1, \ldots, y_n$ where $y_i = 1$ denotes an anomaly and $y_i = 0$ denotes a nominal sample. Anomalous samples can simply be a collection of random images which are not from the nominal collection, e.g. one of the many large collections of images which are freely available like 80 Million Tiny Images (Torralba et al., 2008) or ImageNet (Deng et al., 2009). The use of such an auxiliary corpus has been recommended in recent works on deep AD, where it is termed *Outlier Exposure* (OE) (Hendrycks et al., 2019a;b). When one has access to "true" examples of the anomalous dataset, i.e. something that is likely to be representative of what will be seen at test time, we find that even using a few examples as the corpus of labeled anomalies performs exceptionally well. Furthermore, in the absence of *any* sort of known anomalies, one can generate synthetic anomalies, which we find is also very effective.

FCDD Objectives

With an FCN $\phi : \mathbb{R}^{c \times h \times w} \to \mathbb{R}^{u \times v}$ the FCDD objective utilizes a pseudo-Huber loss on the FCN output matrix $A(X) = \left(\sqrt{\phi(X; \mathcal{W})^2 + 1} - 1 \right)$, where all operations are applied element-wise. The FCDD objective is then defined as (cf., (1)):

$$\min_{\mathcal{W}} \quad \frac{1}{n} \sum_{i=1}^{n} (1 - y_i) \frac{1}{u \cdot v} \left\| A(X_i) \right\|_1 - y_i \log \left(1 - \exp \left(- \frac{1}{u \cdot v} \left\| A(X_i) \right\|_1 \right) \right). \tag{2}$$

Here $\|A(X)\|_1$ is the sum of all entries in $A(X)$, which are all positive. FCDD is the utilization of an FCN in conjunction with the novel adaptation of the HSC loss we propose in (2). The objective maximizes $\|A(X)\|_1$ for anomalies and minimizes it for nominal samples, thus we use $\|A(X)\|_1$ as the anomaly score. Entries of $A(X)$ that contribute to $\|A(X)\|_1$ correspond to regions of the input

FCDD Objectives

Heatmap Upsampling Since we generally do not have access to ground-truth pixel annotations in anomaly detection during training, we cannot learn how to upsample using a deconvolutional type of structure. We derive a principled way to upsample our lower resolution anomaly heatmap instead. For every output pixel in $A(X)$ there is a unique input pixel which lies at the center of its receptive field. It has been observed before that the effect of the receptive field for an output pixel decays in a Gaussian manner as one moves away from the center of the receptive field (Luo et al., 2016). We use this fact to upsample $A(X)$ by using a strided transposed convolution with a fixed Gaussian kernel (see Figure 2 right side). We describe this operation and procedure in Algorithm 1 which simply corresponds to a strided transposed convolution. The kernel size is set to the receptive field range of FCDD and the stride to the cumulative stride of FCDD. The variance of the distribution can be picked empirically (see Appendix B for details). Figure 3 shows a complete overview of our FCDD method and the process of generating full-resolution anomaly heatmaps.

Algorithm 1 Receptive Field Upsampling

Input: $A \in \mathbb{R}^{u \times v}$ (low-res anomaly heatmap)

Output: $A' \in \mathbb{R}^{h \times w}$ (full-res anomaly heatmap)

Define: $[G_2(\mu, \sigma)]_{x,y} \triangleq \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x-\mu_1)^2+(y-\mu_2)^2}{2\sigma^2}\right)$

$A' \leftarrow 0$
for all output pixels a in A **do**
  $f \leftarrow$ receptive field of a
  $c \leftarrow$ center of field f
  $A' \leftarrow A' + a \cdot G_2(c, \sigma)$
end for
return A'

Experiment

Fashion-MNIST We consider each of the ten Fashion-MNIST (Xiao et al., 2017) classes in a one-vs-rest setup. We train Fashion-MNIST using EMNIST (Cohen et al., 2017) or grayscaled CIFAR-100 (Krizhevsky et al., 2009) as OE. We found that the latter slightly outperforms the former ($\sim$3 AUC percent points). On Fashion-MNIST, we use a network that consists of three convolutional layers with batch normalization, separated by two downsampling pooling layers.

CIFAR-10 We consider each of the ten CIFAR-10 (Krizhevsky et al., 2009) classes in a one-vs-rest setup. As OE we use CIFAR-100, which does not share any classes with CIFAR-10. We use a model similar to LeNet-5 (LeCun et al., 1998), but decrease the kernel size to three, add batch normalization, and replace the fully connected layers and last max-pool layer with two further convolutions.

ImageNet We consider 30 classes from ImageNet1k (Deng et al., 2009) for the one-vs-rest setup following Hendrycks et al. (2019a). For OE we use ImageNet22k with ImageNet1k classes removed (Hendrycks et al., 2019a). We use an adaptation of VGG11 (Simonyan and Zisserman, 2015) with batch normalization, suitable for inputs resized to 224$\times$224 (see Appendix D for model details).

State-of-the-art Methods We report results from state-of-the-art deep anomaly detection methods. Methods that do not incorporate known anomalies are the autoencoder (AE), DSVDD (Ruff et al., 2018), Geometric Transformation based AD (GEO) (Golan and El-Yaniv, 2018), and a variant of GEO by Hendrycks et al. (2019b) (GEO+). Methods that use OE are a Focal loss classifier (Hendrycks et al., 2019b), also GEO+, Deep SAD (Ruff et al., 2020b), and HSC (Ruff et al., 2020a).

Experiment

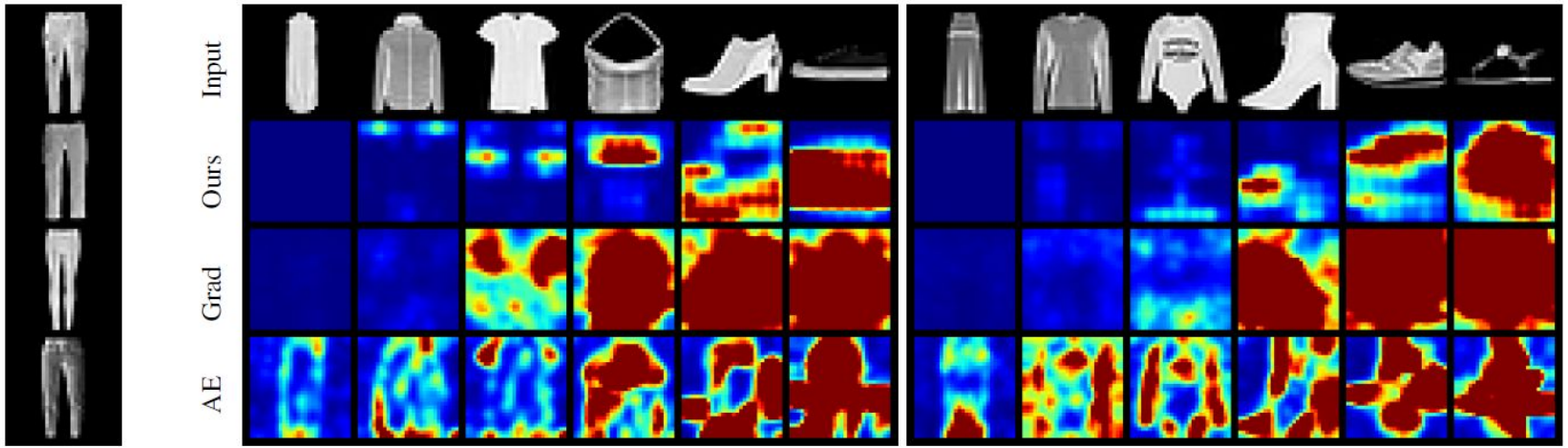


Figure 4: Anomaly heatmaps for *anomalous* test samples of a Fashion-MNIST model trained on nominal class "trousers" (nominal samples are shown in (a)). In (b) CIFAR-100 was used for OE and in (c) EMNIST. Columns are ordered by increasing anomaly score from left to right, i.e. what FCDD finds the most nominal looking anomaly on the left to the most anomalous looking anomaly on the right.

Experiment

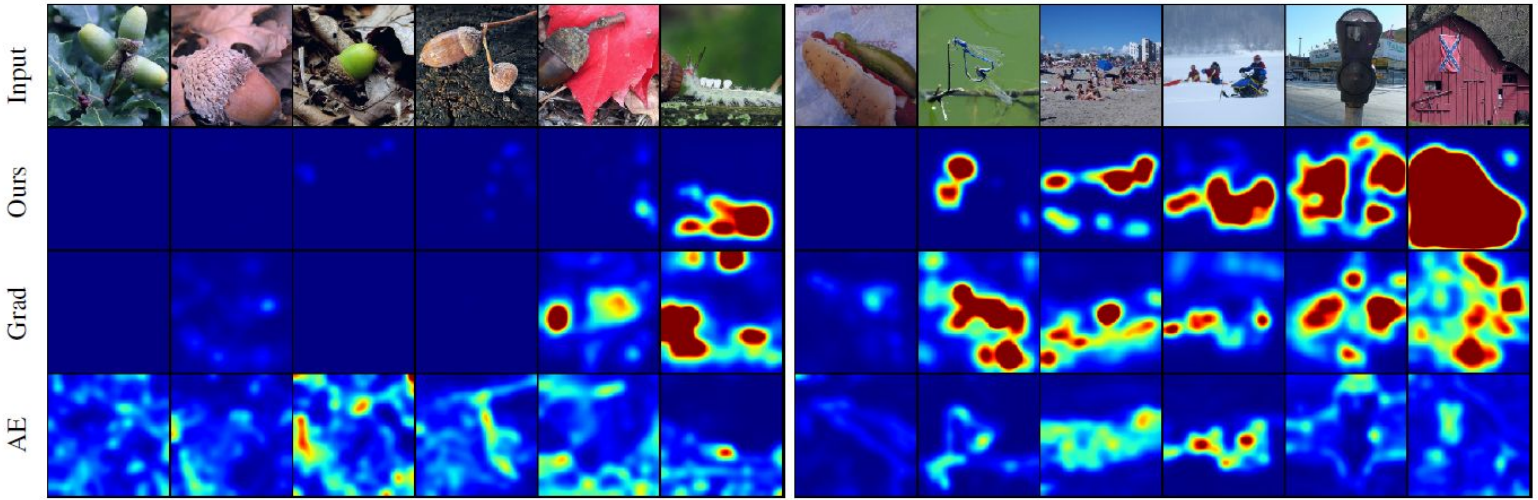


Figure 5: Anomaly heatmaps of an ImageNet model trained on nominal class "acorns." Here (a) are nominal samples and (b) are anomalous samples. Columns are ordered by increasing anomaly score from left to right, i.e. what FCDD finds the most nominal looking on the left to the most anomalous looking on the right for (a) nominal samples and (b) anomalies.

Experiment

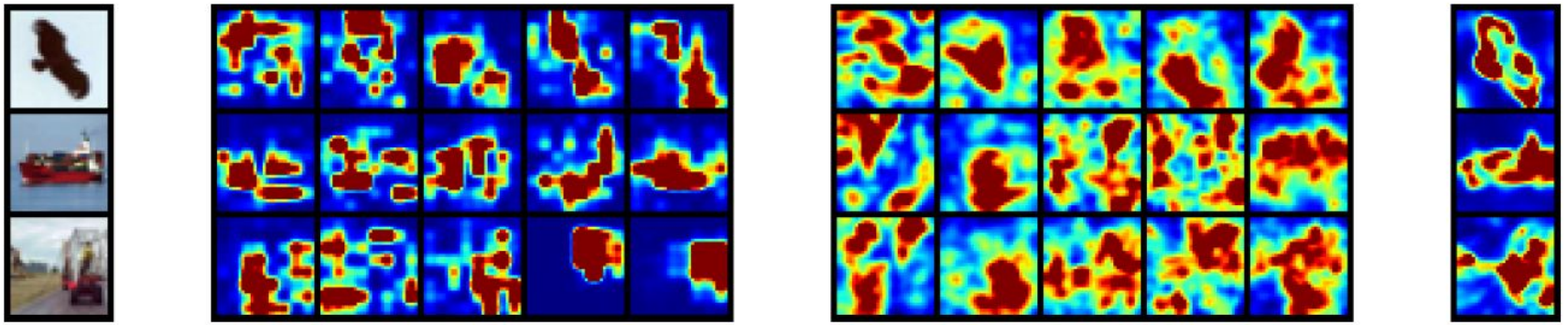


Figure 6: Anomaly heatmaps for three anomalous test samples on a CIFAR-10 model trained on nominal class "airplane." The second, third, and fourth blocks show the heatmaps of FCDD, gradient-based heatmaps of HSC, and AE heatmaps respectively. For Ours and Grad, we grow the number of OE samples from 2, 8, 128, 2048 to full OE. AE is not able to incorporate OE.

4.2 EXPLAINING DEFECTS IN MANUFACTURING

Here we compare the performance of FCDD on the MVTec-AD dataset of defects in manufacturing (Bergmann et al., 2019). This datasets offers annotated ground-truth anomaly segmentation maps for testing, thus allowing a quantitative evaluation of model explanations. MVTec-AD contains 15 object classes of high-resolution RGB images with up to 1024×1024 pixels, where anomalous test samples are further categorized in up to 8 defect types, depending on the class. We follow Bergmann et al. (2019) and compute an AUC from the heatmap pixel scores, using the given (binary) anomaly segmentation maps as ground-truth pixel labels. We then report the mean over all samples of this "explanation" AUC for a quantitative evaluation. For FCDD, we use a network that is based on a VGG11 network pre-trained on ImageNet, where we freeze the first ten layers, followed by additional fully convolutional layers that we train.

Synthetic Anomalies OE with a natural image dataset like ImageNet is not informative for MVTec-AD since anomalies here are subtle defects of the nominal class, rather than being out of class (see Figure 1). For this reason, we generate synthetic anomalies using a sort of "confetti noise," a simple noise model that inserts colored blobs into images and reflects the local nature of anomalies. See Figure 7 for an example.

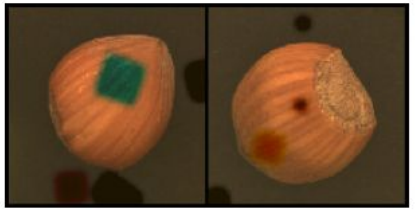


Figure 7: Confetti noise.

Semi-Supervised FCDD A major advantage of FCDD in comparison to reconstruction-based methods is that it can be readily used in a semi-supervised AD setting (Ruff et al., 2020b). To see the effect of having even only a few labeled anomalies and their corresponding ground-truth anomaly maps available for training, we pick for each MVTec-AD class just *one* true anomalous sample per defect type at random and add it to the training set. This results in only 3–8 anomalous training samples. To also take advantage of the ground-truth heatmaps, we train a model on a pixel level. Let $X_1, \ldots, X_n$ again denote a batch of inputs with corresponding ground-truth heatmaps $Y_1, \ldots, Y_n$, each having $m = h \cdot w$ number of pixels. Let $A(X)$ also again denote the corresponding output anomaly heatmap of X. Then, we can formulate a pixel-wise objective by the following:

$$\min_{\mathcal{W}} \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{m} \sum_{j=1}^{m} (1 - (Y_i)_j) A'(X_i)_j \right) - \log \left(1 - \exp \left(-\frac{1}{m} \sum_{j=1}^{m} (Y_i)_j A'(X_i)_j \right) \right). \quad (3)$$

Table 2: Pixel-wise mean AUC scores for all classes of the MVTec-AD dataset (Bergmann et al., 2019). For competitors we include the baselines presented in the original MVTec-AD paper and previously published works from peer-reviewed venues that include the MVTec-AD benchmark. The competitors are Self-Similarity and L2 Autoencoder (Bergmann et al., 2019), AnoGAN (Schlegl et al., 2017; Bergmann et al., 2019), CNN Feature Dictionaries (Napoletano et al., 2018; Bergmann et al., 2019), Visually Explained Variational Autoencoder (Liu et al., 2020), Superpixel Masking and Inpainting (Li et al., 2020), Gradient Descent Reconstruction with VAEs (Dehaene et al., 2020), and Encoding Structure-Texture Relation with P-Net for AD (Zhou et al., 2020).

	unsupervised									semi-supervised
	AE-SS*	AE-L2*	AnoGAN*	CNNFD*	VEVAE*	SMAI*	GDR*	P-NET*	FCDD	FCDD
Bottle	0.93	0.86	0.86	0.78	0.87	0.86	0.92	**0.99**	0.97	0.96
Cable	0.82	0.86	0.78	0.79	0.90	**0.92**	0.91	0.70	0.90	**0.93**
Capsule	**0.94**	0.88	0.84	0.84	0.74	0.93	0.92	0.84	0.93	**0.95**
Carpet	0.87	0.59	0.54	0.72	0.78	0.88	0.74	0.57	**0.96**	**0.99**
Grid	0.94	0.90	0.58	0.59	0.73	0.97	0.96	**0.98**	0.91	0.95
Hazelnut	0.97	0.95	0.87	0.72	**0.98**	0.97	**0.98**	0.97	0.95	0.97
Leather	0.78	0.75	0.64	0.87	0.95	0.86	0.93	0.89	**0.98**	**0.99**
Metal Nut	0.89	0.86	0.76	0.82	**0.94**	0.92	0.91	0.79	**0.94**	**0.98**
Pill	0.91	0.85	0.87	0.68	0.83	0.92	**0.93**	0.91	0.81	**0.97**
Screw	0.96	0.96	0.80	0.87	0.97	0.96	0.95	**1.00**	0.86	0.93
Tile	0.59	0.51	0.50	0.93	0.80	0.62	0.65	**0.97**	0.91	**0.98**
Toothbrush	0.92	0.93	0.90	0.77	0.94	0.96	**0.99**	**0.99**	0.94	0.95
Transistor	0.90	0.86	0.80	0.66	**0.93**	0.85	0.92	0.82	0.88	0.90
Wood	0.73	0.73	0.62	0.91	0.77	0.80	0.84	**0.98**	0.88	0.94
Zipper	0.88	0.77	0.78	0.76	0.78	0.90	0.87	0.90	**0.92**	**0.98**
Mean	0.86	0.82	0.74	0.78	0.86	0.89	0.89	0.89	**0.92**	**0.96**
σ	0.10	0.13	0.13	0.10	0.09	0.09	0.09	0.12	**0.04**	**0.02**

Questions?