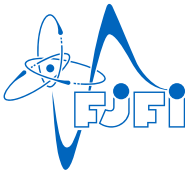
Czech Technical University in Prague
Faculty of Nuclear Sciences and Physical Engineering

Neural network-based generative models for anomaly detection

Dissertation



Author: Ing. Vít škvára
Academic year: Year

Poděkování:

Thanks

Čestné prohlášení:

Prohlašuji na tomto místě, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze dne Date

.....
Ing. Vít škvára

Název:	**Generativní neuronové sítě pro detekci anomálií**
Autor:	Ing. Vít škvára
Obor:	Matematické inženýrství
Druh práce:	Disertační práce
Školitel:	doc. Ing. Václav Šmídl, Ph.D.
Školitel specialista:	Ing. Jakub Seidl, Ph.D.
Abstrakt:	Tato práce se zabývá
Klíčová slova:	neuronové sítě, generativní modely, detekce anomálií

Title: **Neural-network-based generative models for anomaly detection**

Abstract: Abstract abstract

Keywords: neural networks, generative models, anomaly detection

Contents

1

Introduction

Anomaly detection is an important task in environments where we have a good knowledge of what is the normal behaviour, but we know very little about the behaviour of yet unseen or otherwise abnormal events – anomalies. The reasons for this can be numerous: either there is no common generating principle behind anomalies and each new anomaly may be very different from those that we have yet seen, or the acquisition of anomalous data is too expensive or downright impossible (an example of this might be an industrial process). Usually, there is a disbalance in the number of labeled normal and anomalous data samples that are available to us, and sometimes no anomalies are available at all. While it might be tempting to solve anomaly detection as supervised binary classification, for the reasons listed above, a supervised classifier is likely to be unrobust to actual anomalies that it will encounter in a production environment. Together with an ever-increasing volume of collected data and available computing power, this motivates the development of specialized methods for automatic anomaly detection. What these methods have in common is that they learn a model of normal data in an unsupervised manner, and detect anomalies as deviations from this model.

Anomaly detection is important for many industries, where it is typically difficult to obtain a training set containing representative samples of anomalous data. The actions taken after an anomaly is detected might be varied. Sometimes, the anomaly might be considered to be an erroneous measurement and as such is ignored, which is the case of some of the earliest scientific essays [1, 2] on the topic of anomaly detection. In other cases, a preventive measure must be taken in order to mitigate unwanted behaviour, such as the case of cybersecurity [3, 4, 5], fraud detection [6, 7, 8], medical diagnosis [9, 10, 11, 12] or industrial process monitoring [13, 14, 15]. Finally, detected anomalies might drive forward scientific discovery in astronomy [16], plasma physics [17], chemistry [18] or particle physics [19].

There are countless models and algorithms for anomaly detection, tackling the problem from different angles based on the basic principle of the algorithm, the expected nature of the data, and the application domain. There are methods based on random forests [20], the k–nearest neighbors algorithm [21], Gaussian mixture models [22], clustering neural networks [23], histogram estimation [24], kernel density estimates [25] or support vector machines [26]. A comprehensive overview of anomaly detection methods is presented in studies such as [27, 28, 29, 30, 31] where the authors compare several existing methods on benchmark datasets. Most of the comparative studies however do not include methods based on (deep) neural networks and especially not generative models. The probably most recent and complete overview of deep generative models can be found at [32].

Deep generative models have recently attracted a lot of attention due to their ability to produce (generate) very high-quality artificial images that resemble those from the training dataset. Since the seminal papers [33, 34, 35] on the main types of generative models have been published, a myriad of improvements and tweaks have been proposed. While the original purpose of generative models was not aimed towards anomaly detection, some of them were redesigned for it. This text intends to collect some (but definitely not all) of the relevant information on deep generative models in one place and assess the potential suitability of the different generative models to the task of anomaly detection.

This chapter is organized in the following fashion: first, the basic principles of anomaly detection are introduced. A short section on the evaluation of anomaly detectors is followed by a section describing the classification of anomaly detectors into main categories based on their principles, as some of them will be useful in the later chapters.

1.1 What is anomaly detection?

Anomaly detection has been extensively studied under many different names: outlier detection [36, 37], novelty detection [27], one-class classification [38] or out-of-distribution detection [39]. There is a small distinction between these terms based on the application domain, but the methods used to solve the problems they present are in principle the same. In the same vein, the terms outlier, novelty, and anomaly may have a slightly different meaning in some publications, but at the same time are often used interchangeably. In this text, we will resort to the use of the last term. An often cited definition of what constitutes an anomaly is "an observation which deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism" [40]. This broad statement highlights the fact that anomalies may have very different sources of origin and their anomality depends on the context in which they are considered.

The probabilistic definition assumes a probability distribution P^+ of normal data, operating on a data space $\mathcal{X}$, which is defined by a given problem, and which is most of the time known only through a set of normal samples. We can call a sample $\boldsymbol{x} \in \mathcal{X}$ to be an **anomaly** if it lies in a region where P^+ has very low density. In other words, we can define a **set of anomalies** [32] as

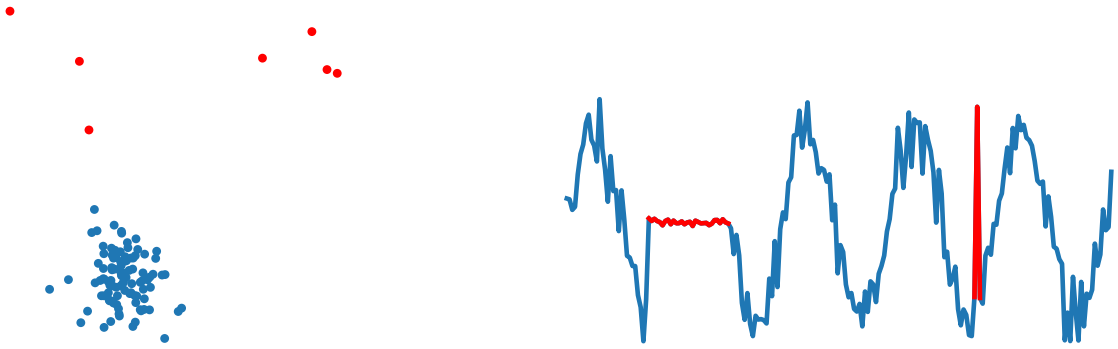
$$\mathcal{A} = \{\boldsymbol{x} \in \mathcal{X} | p^+(\boldsymbol{x}) \leq \tau\}, \tau \geq 0, \tag{1.1}$$

where $p^+(\boldsymbol{x})$ is the probability density function corresponding to P^+ and τ is a **threshold** which defines the line between normal and anomalous samples.

It is also often assumed that the region of data space that is occupied by normal data is concentrated, that is, there exists a threshold $\tau \geq 0$ such that

$$\mathcal{X}/\mathcal{A} = \{\boldsymbol{x} \in \mathcal{X} | p^+(\boldsymbol{x}) > \tau\} \tag{1.2}$$

is not empty, which however does not imply that the support of p^+ is bounded. On the other hand, $\mathcal{A}$ is not required to be concentrated and can be unbounded. Notice that we do not explicitly define any sort of anomalous distribution P^-. This is because most anomaly detection methods only model P^+. When P^- is considered, such as in KDE [41] or OCSVM [26] detectors, it is assumed that it is uniform over $\mathcal{X}$.



(a) point anomalies

(b) group (left) and contextual (right) anomaly

(c) semantic anomaly

Figure 1.1: Examples of different types of anomalies.

Different types of anomalies which require different approaches have been identified in literature [30, 32]. Examples are presented in Fig. 1.1.

- **Point anomaly** is a single datapoint of $\mathcal{A}$, for example, an outlying measurement or a photograph of a cat among other images of dogs. This is the most often studied type of anomaly in the research literature. Note that a point anomaly can become an anomaly of the two following types if the datapoints in a dataset are somehow dependent (e.g. through time) or if some additional context about the data can be extracted.

- **Group anomalies** are a collection of correlated datapoints that are only anomalous together. Only a large number of malicious requests is enough to shut down a server in a DDoS atack [42]. Other research [43, 44] focuses on finding anomalies under the multiple-instance learning (MIL) [45] paradigm, where individual datapoints (called bags) are comprised of a variable number of observations or measurements (called instances). This calls for an aggregation method, on top of which an anomaly detector can operate.

- **Contextual** anomaly is a kind of anomaly that is only anomalous in a certain context. A person measuring over 195 cm is an outlier in almost any place except a locker room of a basketball team. If a target dataset consists of pictures of birds photographed mid-flight – is a bird sitting on grass an anomaly? Or a different flying object, such as an airplane? The answers to those questions depend on what problem is actually being solved. Contextual anomalies often arise in time series [46] or in spatial data [47].

true label/estimated label	normal	anomalous
normal	tn	fp
anomalous	fn	tp

Table 1.1: A confusion matrix of a model which captures the perfromance of a model by showing the total number of correctly (tp = true positives and tn = true negative) and incorrectly (fp = false positives and fn = false negatives) identified samples. In context of anomaly detection, positive samples are anomalies, while normal samples are considered negative.

- **Semantic** anomalies arise in image data and are opposed to **sensory** anomalies. While sensory anomalies appear in low-level image features such as edges or textures (e.g. breaks or defects), semantic anomalies can be detected in the high-level information of an image (e.g. an object of a different category than what appears in the training dataset). Semantic anomalies can be hard to detect, as they can be very similar to normal data [48]. We will cover their detection in chapters 3 and 4.

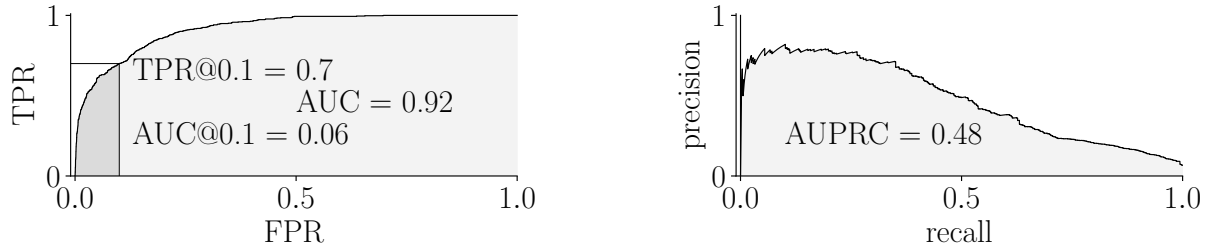
We define the **contamination rate** of a dataset $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subset \mathcal{X}$, which is a finite collection of samples from data space $\mathcal{X}$ as

$$C(\mathcal{X}) = \frac{|\{\boldsymbol{x}_i | \boldsymbol{x}_i \in \mathcal{A} \wedge \boldsymbol{x}_i \in X\}_{i=1}^{n}|}{|X|}, \tag{1.3}$$

i.e. the ratio of the number of anomalies to the total size of the dataset. We can think of any anomaly detection model as providing a function that produces a ranking of the individual data points with respect to their anomalousness. This is called an **anomaly score** $s : \mathcal{X} \to \mathbb{R}$ of a model. In certain contexts [49], anomaly score might be called *decision* or *scoring function*. In this text, we will assume that a higher anomaly score is attributed to a point more likely to be anomalous. To be able to use an anomaly score for decision-making, one must choose the threshold $\tau \in \mathbb{R}$. From (1.1), a sample $\boldsymbol{x}$ is considered to be an anomaly if $s(\boldsymbol{x}) >= \tau$ and normal otherwise. The selection of τ can sometimes be a process more complicated than the fitting of the actual model [50]. Its value is typically determined on the basis of the tolerated false positive rate and an estimate of the true contamination rate of a dataset (1.3).

1.2 Comparing anomaly detectors

Comparing different models on the same set of data is a basic requirement in practical and research problems. As already mentioned at the beginning of this chapter, anomaly detection has some common ground with binary classification tasks. Therefore, we can readily apply the evaluation metrics that are used to evaluate these tasks in comparisons of anomaly detectors. However, there are specifics of anomaly detection problems, mainly the often encountered large imbalance of labeled normal and anomalous samples, that we have to keep in mind. Also, with one exception, all the metrics that will be described here require at least some labeled anomalous samples, no matter how difficult it might be to obtain them. A more complete overview can be found in [51].



(a) ROC curve

(b) PR curve

Figure 1.2: An example of an ROC curve and the derived measures based on FPR=0.1. AUC is the whole shaded area under the ROC curve. The darker shading corresponds to AUC@0.1. On the left, the PR curve of the same detector is shown.

Table 1.1 displays a confusion table that introduces the basic concepts and notation needed below. It summarizes the performance of an algorithm with a particular threshold.

AUC

The most widely used measure in the field of anomaly detection is the area under the ROC (receiver operating characteristic) curve. The acronym AUC will be used in this text for the sake of brevity). The ROC curve is a parametric curve describing the trade-off between **true positive rate** (sometimes also called **recall**) $\mathrm{TPR}(\tau) = \frac{\mathrm{tp}}{\mathrm{tp+fn}}(\tau)$ and **false positive rate** $\mathrm{FPR}(\tau) = \frac{\mathrm{fp}}{\mathrm{fp+tn}}(\tau)$ for different values of the decision threshold τ.

Then, the area under the curve is calculated as the following integral

$$\mathrm{AUC} = \int_{\mathbb{R}} \mathrm{TPR}(\tau) d\mathrm{FPR}'(\tau) d\tau = \int_0^1 \mathrm{TPR}(\mathrm{FPR}) d\mathrm{FPR}. \tag{1.4}$$

The last integral that uses $\mathrm{TPR}(\cdot)$ as a function of the corresponding FPR shows the simple concept behind the AUC that can be easily discerned from an ROC curve drawn in a graph. An AUC value of 1.0 is equal to perfect classification, while a value of 0.5 is equal to classifying by random guessing. An example of an ROC curve and the corresponding AUC is in Figure 1.2. In practice, the corresponding AUC is estimated from an empirical ROC curve using some numerical integration scheme, e.g. the trapezoidal rule.

As mentioned above, the main advantage of AUC is that it does not depend on the choice of a particular decision threshold. Also, the measure has a straightforward interpretation – it is an estimate of the probability that a randomly chosen positive sample is ranked higher than a randomly chosen negative sample [52]. However, a lot of information is lost when the whole ROC curve is summarized into a single number. This is especially concerning for the case of anomaly detection, where usually the region of low false positive rates is of interest since anomalies are sparse with respect to normal data and we strive to achieve a low false positive rate. It is frequent in security applications to draw an ROC curve in logarithmic scale on the x-axis.

AUPRC

The Area Under the Precision–Recall Curve is very similar to AUC, as it is given by computing the **precision** $\mathrm{PREC}(\tau) = \frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fp}}(\tau)$ and recall for different values of classification threshold τ and then integrating the area under the resulting curve. A PR curve has at most as many unique recall values as positive samples in the dataset. This is problematic for anomaly detection, where the number of anomalies is low, which leads to a very sparse estimate of the true PR curve. In fact, using the same trained anomaly detector and changing the contamination rate of a testing dataset produces different AUPRC results, which then makes any analysis based on AUPRC useless when the true contamination rate is unknown. Furthermore, a correct PR curve lacks a universal starting point, unlike an ROC curve, because precision is undefined for zero recall, making the computation and normalization of the area under the PR curve and the comparison between datasets even more complicated. It is still a metric that is reported very often besides the AUC.

Areas of low TPR

The two already mentioned metrics put the same weight on all areas on the x-axis. This might not be always ideal for the purposes of anomaly detection, as low FPR areas might be more interesting – after all, when reporting detected anomalies for a manual check, there is always a limit on how many samples can be realistically processed. A performance measure popular among practitioners is **TPR@α**, which is simply the true positive rate evaluated at a given false positive rate $\alpha \in [0,1]$. This measure can be easily read from an ROC curve. In practice, it is necessary to interpolate the ROC curve since FPR has discrete values, especially for datasets with a small number of samples.

Another alternative to AUC is **AUC@α**, which is the area under the ROC curve calculated only up to some value of false positive rate α. Numerically, it is again important to interpolate the ROC curve for a given α before computing the integral. In Fig. 1.2, AUC@0.1 corresponds to the darker gray region. AUC@α can be easily normalized by dividing by the chosen α, in which case the best detector has AUC@$\alpha = 1$ similarly to AUC.

Volume of decision region

All of the previous measures originate from the evaluation of the performance of binary classifiers. Since labeled anomalies are often difficult to obtain, a measure that does not require labels for its evaluation might be more useful and better describe behaviour on unknown samples. If the goal is to compare two models supposed to characterize the normal class, and with no other assumptions about the distribution of the anomalous class, it makes sense to choose the model enclosing the training data more tightly. This corresponds to calculating the volume inside the model's decision boundary in a similar fashion to [53], where a theoretical justification is given. This decision boundary can be chosen to correspond to a certain level of false positive rate. We define the **volume of decision region** as

$$\mathrm{VOL}(\alpha) = \int_{\mathcal{X}} \mathbb{1}_{\{\boldsymbol{x}\in\mathcal{X} | s(\boldsymbol{x}) <= \tau\}}(\boldsymbol{x}) d\boldsymbol{x} \text{ s.t. } \mathrm{FPR}(\tau) = \alpha, \tag{1.5}$$

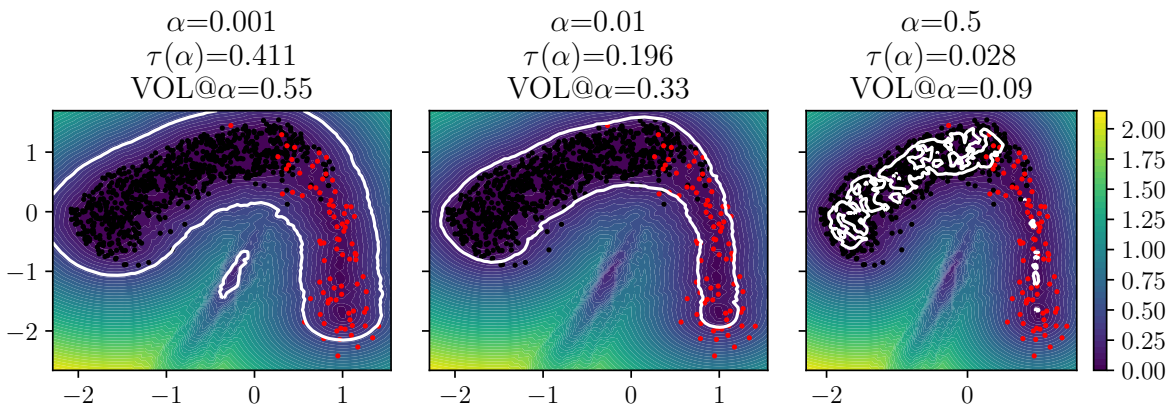


Figure 1.3: An example of a detector and the decision region for differing values of FPR α. The decision boundary is drawn as a white isoline at level $\tau(\alpha)$ with the estimated enclosed volume VOL@α, black and red dots represent normal and anomalous samples in the training set. Clearly, smaller tolerance of false positives forces us to set a higher threshold which results in higher decision region volume.

where $\mathcal{X}$ is the input space, $s(\boldsymbol{x})$ is an anomaly score, and α is a given false positive rate. In other words, VOL(α) is the volume of the subspace where the classifier returns "normal" answer with a false positive rate α. An example of a model and its decision region for different values of α is shown in Fig. 1.3. It should be noted that the idea of minizing the enclosed volume is native to some models, e.g. the SVDD model described in Sec. 1.3.3.

Computing the empirical VOL(α) in data space $\mathcal{X}$ requires first finding the threshold τ corresponding to the chosen FPR α and then integrating the volume which corresponds to the space enclosed by the isosurface where the anomaly score is equal to τ. It is numerically estimated by Monte Carlo sampling [54]. This comes with its downsides. Mainly, the number of samples required to cover d–dimensional sample space grows exponentially with d. This issue has been addressed in [55], where the volume is computed multiple times for different subsampling of input features, however this does not seem to be optimal as it requires training a new model for each subset of features.

1.3 Anomaly detectors taxonomy

Anomaly detection methods are based on a wide range of paradigms. In this section we will follow the taxonomy outlined in publications [27, 32], which divide shallow (not based on neural networks) and deep (based on neural networks) detectors into groups based on their common traits. Examples of methods will be given since their knowledge will be useful in the later chapters of this text, but the list is far from complete. For a complete overview of the landscape of anomaly detection methods, see the surveys [27, 31, 28, 56, 57, 58, 59, 60, 32].

1.3.1 Probabilistic methods

Since we have defined anomaly detection as detecting samples that deviate from the normal distribution P^+, it is straightforward to try to model the distribution by modeling its probability distribution function (pdf). One of the simplest such methods is the Grubbs' test [61], which assumes a Gaussian distribution of normal one-dimensional data and declares a sample to be anomalous if its distance from the mean is larger than some threshold (e.g. three standard deviations). Many such methods based on statistical tests have been proposed [62], but we will focus on advanced modeling probabilistic techniques.

Shallow methods

A **Mahalabonis distance** anomaly detector [63] builds a parametric estimate of a multivariate normal distribution by computing the mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d,d}$ of the training data. The anomaly score of a test point $\boldsymbol{x} \in \mathbb{R}^d$ is then

$$s(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}), \tag{1.6}$$

which is equivalent to the evaluation of the negative log-likelihood under the Gaussian distribution. Even though this is one of the simplest and most non-robust methods, terms similar to (1.6) will be encountered a lot in the remainder of this text.

Instead of limiting the model to a single-mode distribution, a mixture of K distributions

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} p(k) p(\boldsymbol{x}|k), p(k) \in [0,1], \sum_{k=1}^{K} p(k) = 1, \tag{1.7}$$

can be estimated instead, where $p(k)$ is the prior probability of the k-th component. **Gaussian Mixture Models** (GMMs) have been used for anomaly detection in [64, 22]. There, we assume that $p(\boldsymbol{x}|k)$ are Gaussian distribution densities and their parameters are estimated using the EM algorithm [65] or via Variational Bayes [66]. The term $-\log p(\boldsymbol{x})$ is usually used as the anomaly score, although sometimes the maximum of the Mahalabonis distance (1.6) over the K components is used instead. However, the viable usecases for a GMM anomaly detector are limited, mainly due to the need to compute and invert the covariance matrix, which is $O(d^3)$ in the size of the input space d (when the full covariance matrix is estimated). See Fig. 1.4, where the same anomaly detection problem is solved by GMMs with different number of components. Even though the AUC score is already perfect for 2 components, the difficult shape of the normal data needs more components than that to be fully covered.

Time series data anomaly detection was approached with the use of **Autoregressive Integrated Moving Average** (ARIMA) models in [64, 67], or using a **Hidden Markov Model** (HMM) in [68, 69]. Both approaches offer predictions for future states of an observed system, and the anomaly score is the difference between the prediction and the actual state of the system. The problems to which these methods can be applied are encountered in medicine or computer network intrusion detection.

All the previous were examples of parametric models, where a set of parameters $\boldsymbol{\theta}$ is directly estimated. Opposed to these are non-parametric models, which are less restricting – e.g. they do not assume any (normal, Poisson, ...) distribution – instead, they build a parameter-free model of the normal data. **Kernel density estimation** (KDE)

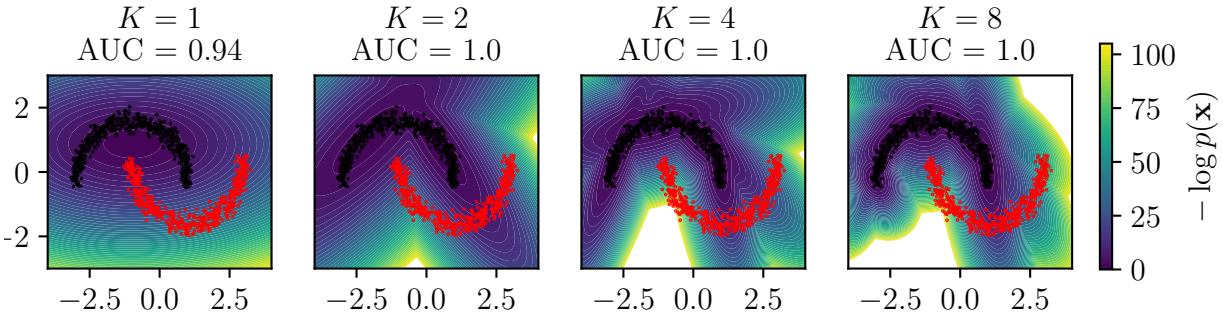


Figure 1.4: The *two bananas* dataset consists of two clusters of points. The black points are considered to be normal and the red are considered to be anomalies. The dataset is useful for introducing behaviour of anomaly detectors, because as an anomaly detection problem, it cannot be solved by a linear separation of the two groups of points. Here, GMM models with a varying number of components K are fitted to the normal datapoints. The contours of the anomaly score based on (1.7) are shown together with the AUC computed with respect to the anomalous datapoints.

method [41], which estimates an unknown univariate probability density function by an empirical estimate

$$\hat{p}(x) = \frac{1}{hn} \sum_{i=1}^{n} k_f\left(\frac{x - x_i}{h}\right), x_i \in X \tag{1.8}$$

where $X = \{x_1, x_2, \ldots, x_n\}$ is the training dataset of n samples, $h \in \mathbb{R}^+$ is a bandwidth parameter, and $k_f : \mathbb{R} \to \mathbb{R}^+$ is a kernel function (uniform, triangular, normal, etc.). KDE has been used under the name Parzen window estimate e.g. in [9, 70], where $-\ln \hat{p}(x)$ is used to score anomalies. **Histogram-based Outlier Score** (HBOS) is a method [71] for anomaly detection on multivariate data $\boldsymbol{x} \in \mathbb{R}^d$. It computes normalized histograms for each feature x_j independently. Then the anomaly score for an unlabeled sample $\boldsymbol{x}$ is

$$s(\boldsymbol{x}) = -\sum_{j=1}^{d} \ln \tilde{p}_j(\boldsymbol{x}) \tag{1.9}$$

where $\tilde{p}_j(\boldsymbol{x})$ is the height of the bin in which x_j is located. The main advantage of this method in comparison with e.g. distance-based is the relative computational simplicity. In the **Lightweight Online Detector of Anomalies** (LODA) [24], an ensemble of one-dimensional histograms is used in a space of diversified projection vectors. The anomaly score is then an average of the logarithm of probabilities estimated from the histograms on individual projection vectors. Due to its simplicity and computational efficiency, it is popular in settings with high volumes of data and potentially missing input values.

Deep methods

The recent advent of neural networks has given rise to many novel methods that are more capable of handling high-dimensional datasets that would be otherwise extremely difficult to handle. A mixture model was used in the **DAGMM** method [72],

where a neural network creates a low-dimensional latent representation $\boldsymbol{z}$ from an input $\boldsymbol{x}$. The GMM and its parameters are then estimated in the latent space, again via a neural network, which enables online learning of the whole model together.

Energy based models (EBMs) use the energy function $E_{\boldsymbol{\theta}}(\boldsymbol{x})$ to approximate the density as

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{\theta})} \exp(-E_{\boldsymbol{\theta}}(\boldsymbol{x})), \tag{1.10}$$

where $Z(\boldsymbol{\theta})$ is the partition function parametrized by $\boldsymbol{\theta}$ to ensure proper normalization of $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. Although the partition function usually cannot be directly computed, an EBM can still be trained via gradient descent coupled with Markov Chain Monte Carlo estimation [73], which is however also the reason for its ineffectiveness relative to more novel approaches, as the MCMC is computationally costly. The anomaly score is then the energy function $E_{\boldsymbol{\theta}}(\boldsymbol{x})$. Although the direct use of the early examples of EBMs such as **Deep Belief Networks** [74] and **Deep Boltzmann Machines** [75] was impractical for anomaly detection, they were eventually refined and successfully used on anomaly detection benchmarks in [76].

Finally, the most recent advancements in anomaly detection were achieved with the use of deep generative models that model the distribution of the data via a generative process, where it is assumed that the data is generated from a hidden latent variable. **Flow models** [35], **Variational autoencoders** [34] and **Generative Adversarial Networks** [33] will be described in greater detail in Chapter 2.

1.3.2 Distance-based methods

Instead of modeling the probability distribution of the normal data, distance-based anomaly detectors use heuristics that compute a well-defined similarity measure between two data points. One of the simplest such models is the **k-Nearest Neighbor** (kNN) [77] model where the anomaly score of a sample is based on its proximity to its k–nearest neighbours from the training dataset. Different proximity measures are described in [21], where the Euclidean distance is used as the similarity measure and the set $X_k(\boldsymbol{x})$ is defined as the set of k-nearest neighbors of $\boldsymbol{x}$ from a training dataset $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$.

- $\kappa(\boldsymbol{x})$: the anomaly score is the distance between $\boldsymbol{x}$ and its kth-nearest neighbor,

$$s_\kappa(\boldsymbol{x}) = \max_{\boldsymbol{x}_i \in X_k(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2, \tag{1.11}$$

- $\gamma(\boldsymbol{x})$: the anomaly score is the average distance of $\boldsymbol{x}$ to its k-nearest neighbors,

$$s_\gamma(\boldsymbol{x}) = \frac{1}{k} \sum_{\boldsymbol{x}_i \in X_k(x)} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2, \tag{1.12}$$

- $\delta(\boldsymbol{x})$: the anomaly score is the length of the mean of the vectors pointing from $\boldsymbol{x}$ to its k-nearest neighbours,

$$s_\delta(\boldsymbol{x}) = \left\| \frac{1}{k} \sum_{\boldsymbol{x}_i \in X_k(x)} (\boldsymbol{x}_i - \boldsymbol{x}) \right\|_2. \tag{1.13}$$

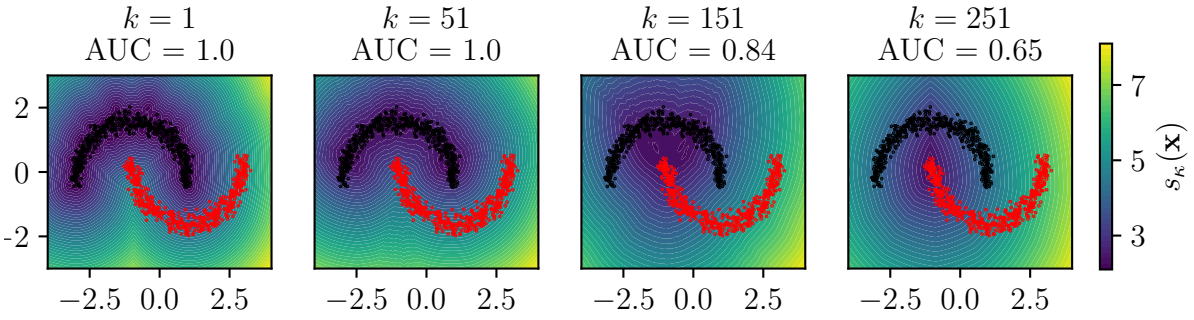


Figure 1.5: K-nearest neighbor detectors on the *two bananas dataset*. The contours of the score (1.11) are shown for different values of k together with the resulting AUC.

All these definitions presume that normal data lie in concentrated regions of the data space $\mathcal{X}$, according to (1.2), while anomalies lie outside of them. The kNN anomaly detector is very popular thanks to its simplicity and good performance [31]. See Fig. 1.5, where low values of k result in a very local behaviour of the model, which is suitable for this problem. The biggest disadvantage of the model is the computational cost. Even though there is no actual training procedure, the prediction complexity is $O(ndk)$. This can be ameliorated by the construction of a k-d tree [78], which partitions the space of the training data to speed-up prediction, or using GPU-based similarity search techniques such as [79]. Other methods, such as **REPEN** [80], use the kNN detector trained on low-dimensional representations of otherwise high-dimensional data produced by a deep neural network.

The **Local Outlier Factor** (LOF) algorithm [81] is based on comparing the local density of a sample $\boldsymbol{x}$ with the local density of its k–nearest neighbours. To correctly describe the way in which the density is defined and the anomaly score is computed, we use the formula (1.11). Then we define the *local reachability density* as

$$\mathrm{LRD}_k(\boldsymbol{x}) = \frac{|X_k(\boldsymbol{x})|}{\sum_{\boldsymbol{x}_i \in X_k(x)} \max\{s_\kappa(\boldsymbol{x}), \|\boldsymbol{x} - \boldsymbol{x}_i\|_2\}}. \tag{1.14}$$

Since this is basically the inverse of an average distance between $\boldsymbol{x}$ and its neighbours, the closer the neighbours are to $\boldsymbol{x}$, the higher this approximation of local density is. Finally, anomaly score of $\boldsymbol{x}$ is given by comparing the LRD_k of $\boldsymbol{x}$ and its neighbours

$$s_{\mathrm{LOF}}(\boldsymbol{x}) = \frac{\sum_{\boldsymbol{x}_i \in X_k(\boldsymbol{x})} \mathrm{LRD}_k(\boldsymbol{x}_i)}{\mathrm{LRD}_k(\boldsymbol{x})|X_k(\boldsymbol{x})|}. \tag{1.15}$$

The rationale behind the formula is that if the local density of the neighbours is higher or the local density of $\boldsymbol{x}$ is lower then it is more likely for $\boldsymbol{x}$ to be an anomaly. This method however does not scale well in number of training samples, as its complexity is even greater than that of a simple kNN detector. Similar methods to LOF are the **connectivity–based outlier factor** (COF) [82] or **clustering–based local outlier factor** (CBLOF) [83].

A different approach is taken by the **isolation forest** (IF) model [20] where an ensemble of N_t isolation trees is constructed for the whole dataset. Isolation trees are constructed in such a way as to isolate each individual datapoint from the rest of the

dataset using consecutive splits on different features. It is presumed that an anomaly can be isolated using a smaller number of splits and therefore it lies on a branch closer to the root of the tree. The anomaly score is then based on the number of splits of a sample averaged over multiple randomly initialized trees in the ensemble. A method similar to IF is the **Partial Identification Forest** (PIDForest) [84], which uses a more informed way of choosing the data features for split, favouring the more informative features.

In **Angle-Based Outlier Detection** (ABOD) [85] presumes that outliers lie far from clusters of normal data, therefore the viewing angle that covers a cluster of normal datapoints when "looking" at it from a sample x is smaller when x is anomalous. More concretely, the method computes the angles between x and all pairs of points in the training dataset, and the anomaly score is inversely proportional to the variance of these angles – the more varied the angles are, the more likely x is close to some cluster of normal data and the anomaly score is thus lower. In the original paper, the method is lauded for the lack of hyperparameters that need to be tuned and the ability to operate in high-dimensional data spaces. However, in the experiments in Chapter 3, it proves to be the method with one of the slowest prediction times.

1.3.3 Domain-based methods

Shallow methods

In domain-based models, the data space is partitioned into subspaces by a decision boundary. Instead of estimating the density of the whole training dataset, they only consider a few samples from it, which are called *support vectors*, and which are used to define the decision boundary. A very simple example is an anomaly detector which, for a training set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$, constructs a hypersphere with center $\boldsymbol{c} \in \mathbb{R}^d$ and radius $R > 0$ that encloses the training data. It is found by solving the objective

$$\begin{aligned} &\min_{R,\boldsymbol{c},\boldsymbol{\xi}} R^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i \\ &\text{s.t.} \|\boldsymbol{x}_i - \boldsymbol{c}\|_2^2 \leq R^2 + \xi_i, \xi_i \geq 0, \forall i \end{aligned} \tag{1.16}$$

where ξ_i are slack variables that allow some data points to lie outside of the hypersphere. The ratio of the maximum number of outliers is controlled by the variable $\nu \in (0, 1]$, which is at the same time a lower bound on the number of support vectors, which are samples $\boldsymbol{x}_i$ that lie exactly on the boundary of the hypersphere. The solution to (1.16) is given by solving the dual problem. Notice that a simple criterion $\|\boldsymbol{x} - \boldsymbol{c}\|_2^2 \leq R^2$ already gives a decision whether the point $\boldsymbol{x}$ is already inside the sphere. To convert this to a continuous anomaly score, we can compute the distance of $\boldsymbol{x}$ from the boundary

$$s(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{c}\|_2^2 - R^2 \tag{1.17}$$

which is negative for points inside and positive for points outside the hypersphere.

Abstracting the above, one can use *kernel functions* [86] to move the problem (1.16) from the original input space to a transformed kernel space. The kernel is a function $k_f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, with which we associate a *feature map* $\Phi : \mathbb{R}^d \to \mathcal{F}_k$ such that the relation defined by the inner product $k_f(\boldsymbol{x}, \boldsymbol{y}) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y}) \rangle$ is true for all $\boldsymbol{x}, \boldsymbol{y}$. The space $\mathcal{F}_k$ is a reproducing kernel Hilbert space and we choose the kernel in such a way that

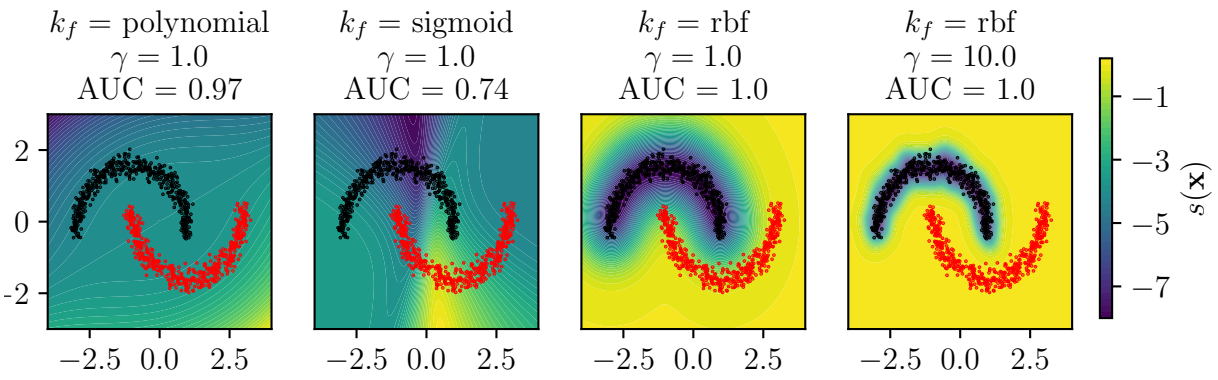


Figure 1.6: OCSVM detectors with different kernel functions and kernel width parameter γ values on the *two bananas dataset*. The contours of the score (1.17) together with the resulting AUC. Although the use of both the polynomial and sigmoid kernels results in a relatively good AUC value, the anomaly score contours reveal that the model would fail if the anomalies would be placed slightly differently. On the other hand, the tight fit of the training data with the *radial basis function* (rbf) is proportional to the kernel width.

its dimensionality is higher than d. This is the basis for the **Support Vector Data Descriptor** (SVDD) anomaly detector, where the inequality condition in (1.17) is replaced by $\|\Phi(\boldsymbol{x}_i) - \boldsymbol{c}\|_2^2 \leq R^2 + \xi_i$. By solving the problem in a space of higher dimensionality, it is possible to find a decision boundary (hypersphere) for datapoints that would otherwise not be separable in the original space. In comparison, the **One-Class Support Vector Machine** (OCSVM) [26] does not construct a hypersphere, but instead aims to find a separating hyperplane in the kernel space. See Fig. 1.6 for a demonstration of the OCSVM model on our toy dataset. Unlike in traditional SVM [87], which is used to separate two classes in a binary classification problem, OCSVM instead aims to find a hyperplane that maximizes the separation of the majority of the training data from the origin in the kernel space. The anomaly score of an OCSVM detector is, similarly to (1.17), the distance from the separating hyperplane. Apart from ν, a very important hyperparameter of the model is the kernel scale parameter γ_{OCSVM}. Many variants of both of the presented approaches were introduced over the years, such as **Minimum Volume Ellipsoid** [88], **Multi-sphere SVDD** [89] or **Bayesian Data Description** [90].

Deep methods

Instead of choosing a kernel and its parameters manually, one can instead parametrize the feature maps Φ using neural networks and train them using standard gradient descent techniques, or use pretrained networks from related tasks. This is the basis for deep OCSVM methods such as **One-class Neural Network** (OCNN) [91] or [92], or methods based on the SVDD formulation [93]. In **Deep SVDD** (DSVDD) [38], the objective (1.16) is reformulated without the slack variables ξ_i, which means that the hypersphere is expected to enclose all datapoints in the training dataset. This simplification leads to faster convergence and yet still proves to be an effective anomaly detector. However, all the deep methods have a basic flaw. Without any further restrictions,

the solution $\Phi(\boldsymbol{x}) = \boldsymbol{c}$ is valid but does not provide a useful detector. This behaviour is called the *feature map collapse*. In order to prevent this, many techniques such as freezing the neural networks that provide the feature map, architectural choices or using adversarial learning are used (for an extensive list of possibilities, see [32]).

Training with negative examples

Although we have stated in Sec. 1.1 that anomaly detection methods do not usually consider the distribution of anomalies P^-, there are some approaches that do. These still fall under the domain-based category of anomaly detectors, as they are mostly constructed as binary classifiers that divide the input space into subspaces where either the anomalous or normal class is expected. In the simplest case [94], P^- is assumed to be uniform and a supervised classifier is trained using anomalies sampled from a hypercube centered around the normal data. This approach however suffers from the curse of dimensionality and saturating the hypercube in high-dimensional spaces is computationally infeasible, especially for image data. Some attempts for a more efficient sampling have been proposed, such as sampling based on local density esimation [95]. In **outlier exposure** [96] techniques, a large auxiliary dataset, that is somehow related to normal data, is used to improve the generalization properties of a deep anomaly detector. For example, if the normal class consists of images of birds, it might be useful to train the detector to recognize normal data from images containing other animals, even though this auxiliary dataset may contain animal classes that will not be encountered as anomalies in test/production environment. It has been shown to be an effective technique in [97].

Outlier exposure is a form of **weak supervision** [98], which is a more general term covering the approach to learning with imperfectly labeled anomaly samples. In [99, 100] it is shown that using even a few labeled examples of anomalies can dramatically improve the detection performance. It is however important to robustify the model in order to be able to generalize to the types of anomalies not present in the labeled training dataset. These techniques can help in **active learning** in anomaly detection [101], where the anomaly detector sends queries to its operator, asking for the most informative/relevant samples to be manually labeled. It is stated in [32] that weak supervision is essential for potential breakthroughs in anomaly detection, which is the motivation for the method proposed in Chapter 4.

Another paradigm where the model is trained in the presence of additional data is **self-supervision** [102], where the model solves an auxiliary task, such as prediction of transformations applied to the image [103]. The important distinction between this approach and outlier exposure is that the transformed data is obtained from the normal training dataset by applying random translations, scalings, rotations, etc. Transforming the data in a controlled manner, the anomaly detector is then a multiclass classifier that predicts the correct transformation applied to the data. The anomaly score of such a model is then based on the softmax activations in the output layer – if the prediction uncertainty is high, the scored sample is more likely to be an anomaly. This was very successfully used in the **Geometric Transformations** (GT) [104] anomaly detector. Furthermore, the **GOAD** method [105] extends this approach to non-image data.

1.3.4 Reconstruction-based methods

One of the most popular approaches to anomaly detection is to build a model that learns to reconstruct input samples. When this model is trained on normal data and learns to reconstruct them well, it is expected that it will be able to identify anomalies by failing to reconstruct them properly, since they have properties unseen by the model at training time. To formalize this, a reconstruction-based anomaly detector consists of an *encoder*, which is a mapping $e_{\boldsymbol{\phi}} : \mathcal{X} \to \mathcal{Z}$, and a *decoder* $g_{\boldsymbol{\theta}} : \mathcal{Z} \to \mathcal{X}$. Here, $\mathcal{X} \subset \mathbb{R}^d$ is the input space, $\mathcal{Z} \subset \mathbb{R}^h$ is a so called *latent* space and $\boldsymbol{\phi}, \boldsymbol{\theta}$ are parameters. A *latent representation* or *encoding* of a sample $\boldsymbol{x}$ is $z = e_{\boldsymbol{\phi}}(\boldsymbol{x})$. Reconstruction-based methods then try to match the original sample $\boldsymbol{x}$ with its reconstruction $\boldsymbol{x}' = g_{\boldsymbol{\theta}}(\boldsymbol{z}) = g_{\boldsymbol{\theta}}(e_{\boldsymbol{\phi}}(\boldsymbol{x}))$. This is done by finding such parameters $\boldsymbol{\phi}, \boldsymbol{\theta}$ as to minimize the reconstruction objective

$$\min_{\boldsymbol{\phi},\boldsymbol{\theta}} \|\boldsymbol{x} - g_{\boldsymbol{\theta}}(e_{\boldsymbol{\phi}}(\boldsymbol{x}))\|_2^2 + \mathcal{R}, \tag{1.18}$$

where $\mathcal{R}$ is a regularization term. Some sort of regularization is needed in order to prevent the decoding function $(g_{\boldsymbol{\theta}} \circ e_{\boldsymbol{\phi}})(\boldsymbol{x})$ to become an identity function, in which case the detector would be useless. The anomaly score of reconstruction-based methods is the *reconstruction error*

$$s(\boldsymbol{x}) = \|x - g_{\boldsymbol{\theta}}(e_{\boldsymbol{\phi}}(\boldsymbol{x}))\|_2^2. \tag{1.19}$$

The **Principal Component Analysis** (PCA) [106, 107], although not originally developed for this purpose, is an example of a reconstruction anomaly detection method. It is assumed that the normal data lie on a lower-dimensional manifold in the data space, which is an assumption also used in other methods. This means that theoretically, they can be represented by a transformation into a lower dimensional latent space, and the eventual differences between the reconstructions from the latent back to the data space and the original samples are only due to noise that is present in the data, and therefore the latent representation contains all the relevant information of a sample. PCA seeks to represent the data by finding an orthonormal basis $W \in \mathbb{R}^{h,d}$ that maximizes the empirical variance of the data $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$. The original objective of PCA can be reformulated with (1.18) in mind, which yields

$$\max_W \sum_{i=1}^{n} \|\boldsymbol{x}_i - W^T W \boldsymbol{x}_i\|_2^2, \text{s.t.} WW^T = I. \tag{1.20}$$

Thus, this means $\boldsymbol{z} = e_{\boldsymbol{\phi}}(\boldsymbol{x}) = W\boldsymbol{x} \in \mathbb{R}^h$ is the encoding represented by the first h *principal components*, while the decoder is $g_{\boldsymbol{\theta}}(\boldsymbol{z}) = W^T \boldsymbol{z}$. The solution to (1.20) is obtained by collecting the first h eigenvectors of the covariance matrix of the training data, e.g. through its eigendecomposition, or by computing the *singular value decomposition* of the data matrix. Like in the domain-based methods in Sec. 1.3.3, some of the restrictions imposed by the linear formulation of classical PCA are circumvented by using a **kernel PCA** (kPCA) [108], where x is replaced by its nonlinear transformation $\Phi(\boldsymbol{x})$. This was used for anomaly detection e.g. in [109].

An **autoencoder** (AE) [110] is in its basic principle a nonlinear PCA, where the encoder $e_{\boldsymbol{\phi}}$ and decoder $g_{\boldsymbol{\theta}}$ are neural networks and the parameters $\boldsymbol{\phi}, \boldsymbol{\theta}$ are the trainable weights of these networks. The nonlinearity comes from the use of nonlinear activation functions between the individual layers of the neural networks. An example of an AE

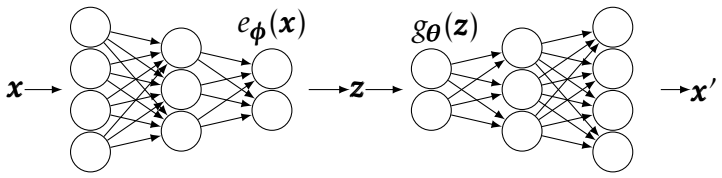


Figure 1.7: An example of an autoencoder consisting of fully connected layers. The latent code $\boldsymbol{z} \in \mathbb{R}^2$ is computed by propagating the input $\boldsymbol{x} \in \mathbb{R}^4$ through the encoder $e_{\boldsymbol{\phi}}(\boldsymbol{x})$ and then used to produce the reconstruction $\boldsymbol{x}' \in \mathbb{R}^4$ via the decoder $g_{\boldsymbol{\theta}}(\boldsymbol{z})$.

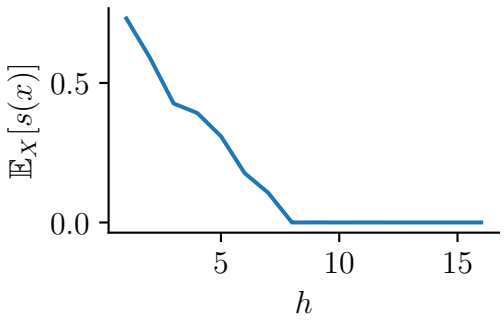


Figure 1.8: The figure demonstrates the ability of an autoencoder to reconstruct data. The dimension h of the latent space $\mathcal{Z}$ is on the x–axis, while the average reconstruction error (1.19) over the whole dataset is on the y–axis. Note that although the artificial dataset is 16–dimensional, it only contains 8 non–correlated dimensions while the remaining are a linear combination of them, which makes the data lie on an 8-dimensional manifold. This results in the error dropping to zero for $h >= 8$ where the model is able to disentangle the correlations and learn the identity function.

network is in Fig. 2.10. To find the weights of the neural networks, the reconstruction error (1.19) is minimized

$$\mathcal{L}_{\mathrm{AE}}(\boldsymbol{x},\boldsymbol{\phi},\boldsymbol{\theta}) = \|\boldsymbol{x} - g_{\boldsymbol{\theta}}(e_{\boldsymbol{\phi}}(\boldsymbol{x}))\|_2^2 \tag{1.21}$$

with respect to $\boldsymbol{\phi},\boldsymbol{\theta}$ using a gradient descent technique, such as ADAM [111] or AMSGrad [112], which use backpropagation [113]. Note that the explicit regularization term $\mathcal{R}$ from (1.18) here is omitted and instead, the regularization is enforced by creating a *bottleneck* $h < d$, which again forces the model to find the optimal representation of data on a manifold of the data space, as is demonstrated in Fig. 2.10. Other types of regularizations include sparse autoencoders [114], where sparsity of encodings is enforced, or denoising autoencoders [115], which reconstruct samples with added artificial noise. The process of training an autoencoder is described in Alg. 1. Note that the space if inputs $\mathcal{X}$ might be an euclidean space $\mathbb{R}^d$ for tabular data, while RGB images are usually represented by three-dimensional tensors of width W and height H, therefore $\mathcal{X} = \mathbb{R}^{H\times W\times 3}$. Autoencoders were used for anomaly detection e.g. in [116, 117], where the reconstruction error (1.19) is used as the anomaly score, and also as a powerful nonlinear dimensionality reduction technique coupled with traditional method in a two-stage approach, as in [92, 118]. They are also the basis for the Variational autoencoder, which will be discussed in depth in the next chapter.

Algorithm 1 Autoencoder training procedure.

Require: Autoencoder $(g_{\boldsymbol{\theta}}, e_{\boldsymbol{\phi}})$, a training set $X = \{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_n\} \subset \mathcal{X}$, maximum number of iterations $I \in \mathbb{N}$, batchsize $B \in \mathbb{N}$.

$\boldsymbol{\phi},\boldsymbol{\theta} \leftarrow$ Initialize weights
$i \leftarrow$ Iteration counter
while $i < I$ or $\boldsymbol{\phi},\boldsymbol{\theta}$ are not converged **do**
    $X_B \leftarrow$ A random batch of B samples from X
    $l \leftarrow \frac{1}{B}\sum_{i=1}^{B} \mathcal{L}_{\mathrm{AE}}(\boldsymbol{x}_i,\boldsymbol{\phi},\boldsymbol{\theta}), \boldsymbol{x}_i \in X_B$
    $\boldsymbol{\phi} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\phi}} l$ update of encoder weights
    $\boldsymbol{\theta} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}} l$ update of decoder weights
    $i \leftarrow i + 1$
end while
return encoder $e_{\boldsymbol{\phi}}(\boldsymbol{x})$, decoder $g_{\boldsymbol{\theta}}(\boldsymbol{z})$

2

Generative models in anomaly detection

Suppose that a given problem (e.g. classification) is requires us to model some directly observable data, denoted by $\boldsymbol{x}$, which is somehow connected with a second variable $\boldsymbol{z}$, which might not be directly observable, or we only have a finite set of observed pairs $(\boldsymbol{x}, \boldsymbol{z})$. When $\boldsymbol{z}$ is discrete, it has the interpretation of a **label** which denotes an affiliation with one of a finite number of classes (in that case, it is often denoted by $\boldsymbol{y}$ instead). When it is continuous, we use the term **hidden** or **latent** variable, see Sec. 1.3.1, where it was already discussed. In that case, we assume that there is some mechanism that connects $\boldsymbol{x}$ and $\boldsymbol{z}$ that can be modeled e.g. by a decoder $g_{\boldsymbol{\theta}}(z)$ from Sec. 1.3.4. The term **generative model** denotes an approach when the joint probability distribution of $p(\boldsymbol{x}, \boldsymbol{z})$ is modeled, as opposed to a **discriminative model**, which tries to estimate $p(\boldsymbol{z}|\boldsymbol{x})$. An example of a discriminative model is a logistic classifier, which in fact estimates $p(\boldsymbol{z}|\boldsymbol{x})$ as a function $f : \mathcal{X} \to \mathcal{Z}$ where $\mathcal{Z} = [0, 1]$, i.e. it produces a guess for binary label based on input data. While it seems that a generative model is more flexible, as it can provide $p(\boldsymbol{z}|\boldsymbol{x}) = p(\boldsymbol{x}, \boldsymbol{z})/p(\boldsymbol{x})$, it is usually subpar in classification tasks [119, 120]. The advantage of using a generative model is that it can generate artificial samples $\boldsymbol{x}$, and most importantly, it can be trained in an unsupervised manner (without observed labels/latent variables) and provide an estimate of the data distribution $p(x)$. This is the reason generative models are interesting for anomaly detection, together with the advent of deep generative models that can model large quantities of high-dimensional data.

Some generative models were actually already introduced in the previous chapter in Sec. 1.3.1, such as the Gaussian Mixture Model, autoregressive models or various energy based models. Recently, very large deep generative models with billions of parameters were introduced. They are pushing the boundaries in many domains and produce human-like outputs, such as the BigGAN [121] for images, GPT-3 [122] for text, the Jukebox model [123] for music generation, or diffusion models [124] for text-to-image translation [125]. In the previous chapter, we have already mentioned the three main types of deep generative models that will be discussed in greater depth in this chapter – the **Generative Adversarial Network** (GAN) [33], the **Variational Autoencoder** (VAE) [34] and various **flow models** [35]. They present the basic paradigms on which most of novel anomaly detectors are built [126, 127, 128, 129, 130, 131].

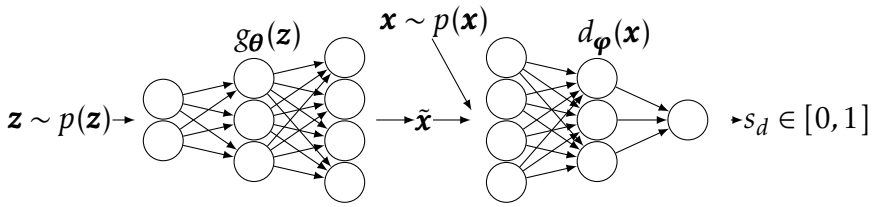


Figure 2.1: A schematic of a GAN consisting of fully connected layers. A latent noise sample $\boldsymbol{z} \sim p(\boldsymbol{z})$ is fed to the generator $g_{\boldsymbol{\theta}}(\boldsymbol{z})$ which then produces an artificial sample $\tilde{\boldsymbol{x}}$. Alternatively, a sample $\boldsymbol{x}$ is sampled from the data distribution $p(\boldsymbol{x})$. Both are passed to the discriminator $d_{\boldsymbol{\varphi}}(\boldsymbol{x})$ that produces a score s_d – the probability that $\tilde{\boldsymbol{x}}$ or $\boldsymbol{x}$ come from the true data distribution.

2.1 GAN–based models

The **Generative Adversarial Network** (GAN) was introduced in [33] where it was successfully used to generate MNIST digits, faces and CIFAR-10 images. Since then, GAN-based models were used in a multitude of different areas, such as next frame prediction in videos [132], semi–supervised learning [133], image–to–image translation [134], semantic manipulation of high resolution images [135] or for generating realistic artificial image [136] and audio data [137]. In comparison with VAE-based generative models, it is believed that GAN-based models produce pictures that are more realistic (less blurry), at the cost of difficult and highly unstable training [133]. In the following text, the basics of GANs will be introduced together with their applications to anomaly detection.

2.1.1 Basic GAN model

Suppose that we have samples from the true data distribution $p(\boldsymbol{x}),\boldsymbol{x} \in \mathcal{X}$, which we are trying to imitate. We don't know the true form of $p(\boldsymbol{x})$, since it is usually high-dimensional and not representable directly by a function, but instead it is available to us through a finite set of samples that comprise the training dataset $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subset \mathcal{X}$. The goal is to build a proxy for $p(\boldsymbol{x})$ so that we can draw new, yet unseen samples from it. The GAN tackles this problem by a model with two principal parts. First is the **generator**, which is a neural network that represents a mapping $g_{\boldsymbol{\theta}}(\boldsymbol{z}) : \mathcal{Z} \to \mathcal{X}$, where $\boldsymbol{\theta}$ are its weights, and $\mathcal{Z}$ is the latent space. Note that we use the same notation for the generator that was already used for a decoder in the AE model – this is because they fulfill the same role in both models. We will denote a generated sample by $\tilde{\boldsymbol{x}} = g_{\boldsymbol{\theta}}(\boldsymbol{z})$. Since the generator as we have defined it so far is deterministic and we need to cover a random distribution, the inputs to the generator come from a **prior** noise distribution $p(\boldsymbol{z})$. The prior is usually chosen such that it it easy to generate samples from it, e.g. $p(\boldsymbol{z}) = \mathcal{N}(0, \boldsymbol{I})$, in which case $\mathcal{Z} = \mathbb{R}^h$ with h being the dimension of the latent space. Then, the task is to train the generator in such a fashion that it learns the potentially highly non–linear mapping from $p(\boldsymbol{z})$ to $p(\boldsymbol{x})$. This is stimulated by the adversary of the generator – the **discriminator**. We define it as a neural network with weights $\boldsymbol{\theta}$, i.e. a mapping $d_{\boldsymbol{\varphi}}(\boldsymbol{x}) : \mathcal{X} \to [0,1]$. The output of the discriminator has the interpretation of the probability that its input comes from $p(\boldsymbol{x})$ rather than being

generated by the generator, in other words, true samples $\boldsymbol{x}$ should be given higher value than the generated samples $\tilde{\boldsymbol{x}}$.

Both parts of a GAN are trained (their weights are updated) in tandem, so each of them iteratively improves in its task. The discriminator is trained both with true and generated samples to maximize the probability of assigning the correct label to them, while the generator is minimizing the probability of the discriminator recognizing the generated sample. This can be written down as a two player minimax [138] game

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\varphi}} \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\Big[\ln d_{\boldsymbol{\varphi}}(\boldsymbol{x})\Big] + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\Big[\ln\Big(1 - d_{\boldsymbol{\varphi}}(g_{\boldsymbol{\theta}}(\boldsymbol{z}))\Big)\Big]. \tag{2.1}$$

It can be shown [33] that this objective has a saddle point at at $-\ln 4$. In practice, (2.1) is not used directly. That is because the term $\ln\Big(1 - d_{\boldsymbol{\varphi}}(g_{\boldsymbol{\theta}}(\boldsymbol{z}))\Big)$ suffers from vanishing gradients – when the generated samples do not resemble the real data in the beginning of the training, than this terms is almost zero and the generator is not trained. Therefore, instead of minimizing this, we can maximize $\ln d_{\boldsymbol{\varphi}}(g_{\boldsymbol{\theta}}(\boldsymbol{z}))$ to train the generator, which has much stronger gradients [33]. During training, we sample $\boldsymbol{x}$ from the training dataset and $\boldsymbol{z}$ from the prior distribution and update the generator and discriminator weights by minimizing

$$\mathcal{L}_g(\boldsymbol{z},\boldsymbol{\theta}) = -\ln d_{\boldsymbol{\varphi}}(g_{\boldsymbol{\theta}}(\boldsymbol{z})), \tag{2.2}$$

$$\mathcal{L}_d(\boldsymbol{x},\boldsymbol{z},\boldsymbol{\varphi}) = -\ln d_{\boldsymbol{\varphi}}(\boldsymbol{x}) - \ln\Big(1 - d_{\boldsymbol{\varphi}}(g_{\boldsymbol{\theta}}(\boldsymbol{z}))\Big). \tag{2.3}$$

Note that we explicitly remove the dependency of the loss functions on the weights that are not being optimized through them. A detailed training procedure of a GAN is described in Alg. 2. Interestingly, during training, the generator never encounters any sample coming from $p(\boldsymbol{x})$ but is still able to eventually learn the shape of $p(\boldsymbol{x})$. The choice of $p(\boldsymbol{z})$ can be rather arbitrary as far as sampling from it is possible and the generator and discriminator have sufficient capacity to process it. In practice, uniform or normal distribution is usually used, as already mentioned.

Achieving convergence such that the generated samples $\tilde{\boldsymbol{x}}$ resemble the training data might be difficult and require multiple random initializations of the model weights. Details on stable training of GANs can be found e.g. in [139]. A phenomenon called **mode collapse** has been described [140], which happens when $p(\boldsymbol{x})$ is multimodal, but the generator distribution collapses to a single mode. A generator that has collapsed to a single mode of a MNIST dataset will produce only a single digit, e.g. "1", no matter where the code is sampled from. To mitigate this issue, several practices have been proposed [133, 141]. One of them is the use of an enhanced generator loss

$$\mathcal{L}_{\text{gfm}}(\boldsymbol{z},\boldsymbol{x},\boldsymbol{\theta}) = \alpha\mathcal{L}_g(\boldsymbol{z},\boldsymbol{\theta}) + \mathcal{L}_h(\boldsymbol{z},\boldsymbol{x},\boldsymbol{\theta}), \tag{2.4}$$

where the second term is the so-called **feature-matching** loss

$$\mathcal{L}_{fm}(\boldsymbol{z},\boldsymbol{x},\boldsymbol{\theta}) = \|d_{l,\boldsymbol{\varphi}}(\boldsymbol{x}) - d_{l,\boldsymbol{\varphi}}(g_{\boldsymbol{\theta}}(\boldsymbol{z}))\|_2^2 \tag{2.5}$$

where $\alpha > 0$ is a tunable weight and $d_{l,\boldsymbol{\varphi}}(\boldsymbol{x})$ is the intermediate representation of $\boldsymbol{x}$ after propagation through $l \in \mathbb{N}$ layers of the discriminator. This loss is supposed to provide improved gradients for the generator to stabilize the training. In the following text, a GAN model with the loss (2.4) will be referred to as the **feature-matching GAN** (fmGAN).

Algorithm 2 GAN training procedure.

Require: Generator $g_{\boldsymbol{\theta}}$, discriminator $e_{\boldsymbol{\phi}}$, a training set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subset \mathcal{X}$, maximum number of iterations $I \in \mathbb{N}$, batchsize $B \in \mathbb{N}$.

$\boldsymbol{\theta}, \boldsymbol{\varphi} \leftarrow$ Initialize weights
$i \leftarrow$ Iteration counter
while $i < I$ or $\boldsymbol{\theta}, \boldsymbol{\varphi}$ are not converged **do**
    $X_B \leftarrow$ A random batch of B samples from X
    $Z_B \leftarrow$ A random batch of B samples from $p(\boldsymbol{z})$
    $l_d \leftarrow \frac{1}{B} \sum_{j=1}^{N} \mathcal{L}_d(\boldsymbol{x}_j, \boldsymbol{z}_j, \boldsymbol{\varphi}), \boldsymbol{x}_j \in X_B, \boldsymbol{z}_j \in Z_B$
    $\boldsymbol{\varphi} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\varphi}} l_d$ update of discriminator weights
    $l_g \leftarrow \frac{1}{L} \sum_{j=1}^{L} \mathcal{L}_g(\boldsymbol{z}_j, \boldsymbol{\theta}), \boldsymbol{z}_j \in Z_L$
    $\boldsymbol{\theta} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}} l_g$ update of generator weights
    $i \leftarrow i + 1$
end while
return generator $g_{\boldsymbol{\theta}}(\boldsymbol{z})$, discriminator $d_{\boldsymbol{\varphi}}(\boldsymbol{x})$

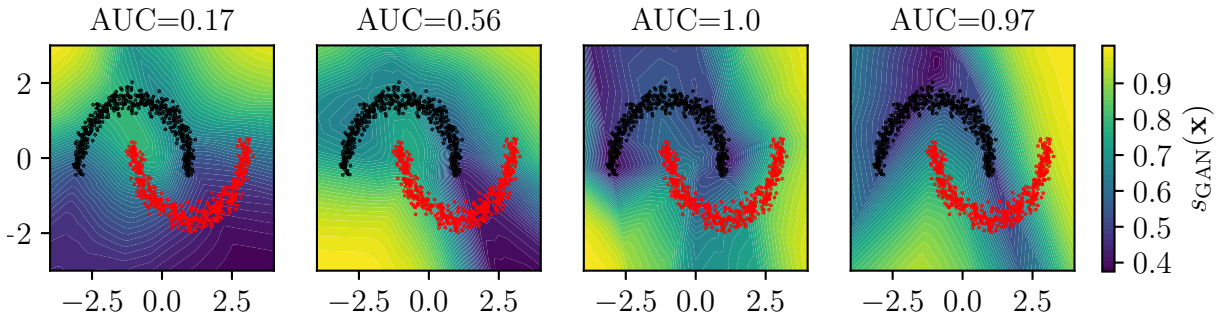


Figure 2.2: Four different GAN models trained on the normal (black) data of the *two bananas* dataset, the contours of the respective anomaly scores (2.6) and the AUC of the model with respect to the anomalous (red) data. All the models have the same hyperparameters and yet they converged to very different states.

2.1.2 GANs in anomaly detection

The idea of using a GAN for anomaly detection comes from the the ability of the generator to learn the true data distribution $p(\boldsymbol{x})$ and especially the ability of the discriminator to recognize samples coming from $p(\boldsymbol{x})$. When the training dataset consists of normal samples, the discriminator output can be converted into an anomaly score

$$s_{\text{GAN}}(\boldsymbol{x}) = 1 - d_{\boldsymbol{\varphi}}(\boldsymbol{x}), \tag{2.6}$$

which is higher for suspected anomalies and lower for normal data. The common critique is that the discriminator was not trained to recognize an arbitrary distribution of the anomalies but only that of the latent transformed by the generator. Thus it may fail to recognize anomalous samples of interest. Also, the training procedure of a GAN model has very high variance in the sense that two models with the same architecture and hyperparameters and trained on the same data might converge to very different states, see Fig. 2.2. This implies that more models need to be crated and trained in order to find a good anomaly detector. This is further discussed in Chapter 3.

The authors of the **AnoGAN** model [23] recognize some of these flaws. Their convolutional GAN model is trained with the feature-matching loss (2.4). For identification of anomalies in medical images, instead of using (2.6), they propose an iterative procedure that searches for the latent code $\boldsymbol{z}$ most likely to generate the tested sample to identify anomalous images. However, this procedure is computationally expensive. Therefore, an update to this model was published, the **f(ast)AnoGAN** [142]. It uses a Wasserstein GAN [143, 144] (more on Wasserstein optimization objective in the next section) with gradient penalization [143] to improve training stability and adds an encoder distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to find $\boldsymbol{z}$ closest to given $\boldsymbol{x}$ faster. The anomaly score of fAnoGAN is a combination of the discriminator score (2.6) and the feature-matching loss (2.5).

The fmGAN model is used the publication [145], where it is tested on benchmark datasets such as MNIST and CIFAR-10. In [128], a GAN model was used to detect anomalies in time series data coming from industrial processes, whereas in [146], network intrusions were detected. In the **Multiple-Objective Generative Adversarial Active Learning** (MOGAAL) [147], $k \in \mathbb{N}$ generators are trained against a single discriminator on input data divided into k subsets. The discriminator score Eq. (2.6) is used to test new samples. The authors of the OCGAN model [129] claim to have achieved state-of-the-art results in one-class classification through severely restricting the latent space of the GAN combined with an autoencoder and employing an adversarial data augmentation strategy.

The use of GAN with an encoder [148] or an autoencoder with a discriminator [149] for anomaly detection is often and somehow blurs the line between GAN and VAE-based models, but we believe that presenting both concepts separately is useful. True GAN-like models for anomaly detection are however far less prevalent than models that use some autoencoding structure, which will be the focus of the next section. Experimental comparison of both approaches is presented in Chapter 3.

2.2 VAE-based models

The **Variational Autoencoder** (VAE) is a generative model that has enjoyed a great success in a number of fields since its introduction in [34]. Its basic architecture [150]

is very similar to that of an AE model described in Sec. 1.3.4, but that is where the similarities end, as VAE is more of a probabilistic anomaly detector, since it models the probability distribution of the normal data. Unlike in AE, where the encoding to latent space $\mathcal{Z}$ is not constrained from taking any shape or form as far as the learning objective is minimized, in VAE a desired distribution of the encodings is explicitly prescribed in the form of a prior distribution $p(\boldsymbol{z})$. If the network is trained properly and the encodings follow the prior, we can feed samples from $p(\boldsymbol{z})$ to the decoder and expect to obtain random samples in the $\mathcal{X}$ space that will resemble those from the training dataset. This is the simple principle of how VAE works — more details will be given in the following text.

The VAE has been mainly used for generation of artificial images such as faces [151] or sentences [152], but also for other tasks such as semi–supervised learning [153], segmentation [154], static image forecasting [155] and of course for anomaly detection [126, 127, 156]. Since its popularity, a multitude of approaches enhancing the original VAE has been published, approaching the paradigm from different angles, with some of the more prominent examples published in [157, 158, 159, 160, 161]. In the following text, we will go through the basic theory, its implications for the basic VAE, and through some of the extensions.

2.2.1 Basic VAE

Let's begin by defining the VAE from a probabilistic perspective. Assume that there is a dataset X consisting of i.i.d samples. We want to obtain a tractable estimate of the true data distribution $p(\boldsymbol{x})$ in order to be able to sample from it. For that purpose, suppose that there is a hidden random process that generates the data and which involves a latent variable $\boldsymbol{z}$. Then, like in the case of the GAN model, we can redirect the sampling from the data space $\mathcal{X}$ to the latent space $\mathcal{Z}$, in which it might be easier. Specifically, we want to sample from the latent prior distribution specified by a density $p(\boldsymbol{z})$, then pass this sample to the generative distribution with density $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, where $\boldsymbol{\theta}$ are its parameters, and obtain a sample $\boldsymbol{x}$ that will be very similar to the samples coming from the true data distribution $p(\boldsymbol{x})$. In other words, we want to maximize the probability of each sample obtained through the generative process

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int_{\mathcal{Z}} p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}. \tag{2.7}$$

Unfortunately, there are several issues with this. Firstly, we do not know the optimal value of parameters $\boldsymbol{\theta}$. Secondly, the integral (2.7) is usually intractable, e.g. in the case where $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is represented by a neural network. Finally, we want to avoid expensive sampling methods such as Monte Carlo Expectation Maximization [162]. A sampling procedure is eventually used, but we only want to pass such samples $\boldsymbol{z}$ to the generative model that will already be very likely under $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$. To this end, we introduce a discriminative distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ which is an approximation of the true intractable posterior $p(\boldsymbol{z}|\boldsymbol{x})$, with parameters $\boldsymbol{\phi}$.

The ELBO objective

Now, we would like to relate the generative and discriminative distributions together in way that would enable us to optimize the model with respect to $\boldsymbol{\phi}$,$\boldsymbol{\theta}$. Continuing

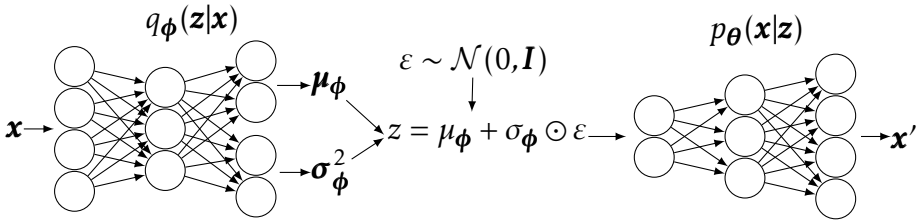


Figure 2.3: A schematic of a Variational Autoencoder consisting of fully connected layers with a Gaussian encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ with mean $\boldsymbol{\mu}_{\boldsymbol{\phi}}$ and variance $\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}$, which are extracted from the last layer of the encoder. They are used to sample an encoding $\boldsymbol{z}$ through the reparametrization trick with a noise variable ε. The encoding is then passed forward and the reconstruction $\boldsymbol{x}'$ is sampled from the decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$.

from (2.7),

$$\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x})] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + \ln p(\boldsymbol{z}) - \ln p(\boldsymbol{z}|\boldsymbol{x})], \tag{2.8}$$

where we have used the Bayes' rule and the fact that $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ does not depend on $\boldsymbol{z}$. Now we use the KL divergence (A.5)

$$\begin{aligned} \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}) - D_{\mathrm{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z}|\boldsymbol{x})\Big) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\Big[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + \ln p(\boldsymbol{z}) - \ln q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\Big] && (2.9)\\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - D_{\mathrm{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z})\Big) && (2.10)\\ &= -\mathcal{L}_{\mathrm{VAE}}(\boldsymbol{x}, \boldsymbol{\phi}, \boldsymbol{\theta}) && (2.11) \end{aligned}$$

This is the variational lower bound of the VAE model, sometimes called **ELBO** (evidence lower boundary) through which we can optimize the marginal likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. This is due to the fact that the analytically unsolvable term $D_{\mathrm{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z}|x)\Big)$ is always nonnegative, thus by maximization of ELBO (minimization of $\mathcal{L}_{\mathrm{VAE}}(\boldsymbol{x}, \boldsymbol{\phi}, \boldsymbol{\theta})$) we also maximize $p_{\boldsymbol{\theta}}(\boldsymbol{x})$.

By looking at the individual parts of Eq. (2.10), we can see that by maximizing the ELBO, we simultaneously maximize the likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ and minimize the distance between $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{z})$. While looking at the left–hand side of (2.9) we can see that at the same time, the marginal likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is maximized and the error term $D_{\mathrm{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z}|\boldsymbol{x})\Big)$ is minimized, forcing the shape of $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ to the true posterior. Also, (2.10) captures the autoencoding nature of the VAE model. We pass $\boldsymbol{x}$ to the discriminative distribution, which acts as an encoder, sample latent encoding $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and pass this back to generative distribution, which acts as a decoder, to obtain a reconstructed sample $\boldsymbol{x}' \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$. From this point on, we will use the term encoder and decoder to describe the discriminative and generative distributions.

Vanilla VAE

To be able to optimize (2.11), we must make some additional assumptions about the model. Here, we will describe those that were made by the authors of the original "vanilla" VAE model, but note that different choices are also possible. First, the **prior** $p(\boldsymbol{z})$ is chosen to be an h-dimensional unit normal distribution $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|0, \mathbf{I})$.

The encoded training data are expected to follow this distribution after optimization and it is used for generation of new samples. In tandem with this, the encoder is assumed to model a normal distribution with diagonal covariance matrix $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\Big(\boldsymbol{z}|\mu_{\boldsymbol{\phi}}(x), \mathrm{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(x))\Big), \boldsymbol{\mu}_{\boldsymbol{\phi}}, \boldsymbol{\sigma}^2_{\boldsymbol{\phi}} : \mathbb{R}^d \to \mathbb{R}^h$, where the mean and variance estimates are computed by neural networks with shared weights $\boldsymbol{\phi}$. These choices lead to an analytically solvable expression for the KL divergence in (2.10), see (A.10) in the Appendix

$$D_{\mathrm{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z})\Big) = \frac{1}{2}\sum_{i=1}^{d} 1 - \sigma^2_{\boldsymbol{\phi},i}(\boldsymbol{x}) + \ln \sigma^2_{\boldsymbol{\phi},i}(\boldsymbol{x}) - \mu^2_{\boldsymbol{\phi},i}(\boldsymbol{x}). \tag{2.12}$$

Second, the optimization of the ELBO (2.11) requires sampling from the encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. This is however problematic for optimization by backpropagation, because sampling is not a differentiable operation. Therefore, a **reparametrization trick** was introduced in [34]. Instead of directly drawing samples from $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, we first take a sample from an h-dimensional noise distribution $\varepsilon \sim p(\varepsilon) = \mathcal{N}(0, \mathbf{I})$ and then compute the encoding as $\boldsymbol{z} = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}) \odot \varepsilon$, where $\odot$ denotes element-wise multiplication. This changes the first term of the ELBO, which is called the **log-likelihood** to

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] = \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,\mathbf{I})}\Big[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}) \odot \varepsilon)\Big]. \tag{2.13}$$

In the following text, whenever it can apply, we use the notation $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ as a shorthand for the reparametrization trick.

Finally, the decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is also assumed to model a normal distribution $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}), \sigma^2 I), \boldsymbol{\mu}_{\boldsymbol{\theta}} : \mathbb{R}^h \to \mathbb{R}^d, \sigma^2 \in \mathbb{R}^+$, although a Bernoulli distribution is sometimes used for data scaled to the interval [0,1]. The mean of the decoder distribution is again represented by a neural network with weights $\boldsymbol{\theta}$. The variance parameter σ^2 is either fixed, or it can be estimated from data during training. Therefore, the log-likelihood takes on the form

$$\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = -\frac{1}{2\sigma^2}\|\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z})\|_2^2 - \frac{d}{2}\ln 2\pi - d\ln\sigma. \tag{2.14}$$

Note that the last two terms can be left out of the optimization, since they are not dependent on any inputs or weights. Again, we see the connection with an autoencoding model, where the log-likelihood (2.14) has very similar form to the objective (1.21). Here, the reconstruction takes the form of $\boldsymbol{x}' = \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z})$. A notable property of the VAE model is that the reconstruction is stochastic, which is due to the sampling used in the reparametrization trick (2.13).

Now, we combine the assumptions and equations (2.12)–(2.14) to derive the final analytic form of the ELBO objective. The expectation in (2.13) is replaced by a mean of L sample of z through the reparametrization trick. The VAE loss function, that is minimized with respect to $\boldsymbol{\phi}, \boldsymbol{\theta}$, and which is in fact equal to negative ELBO (2.11), has the form

$$\mathcal{L}_{\mathrm{VAE}}(\boldsymbol{x}, \boldsymbol{\phi}, \boldsymbol{\theta}) = \frac{1}{2\sigma^2 L}\sum_{l=1}^{L}\|\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}^l)\|_2^2 - \frac{1}{2}\sum_{i=1}^{d} 1 - \sigma^2_{\boldsymbol{\phi},i}(x) + \ln\sigma^2_{\boldsymbol{\phi},i}(x) - \mu^2_{\boldsymbol{\phi},i}(x), \tag{2.15}$$

$$\boldsymbol{z}^l = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}) \odot \varepsilon^l, \varepsilon^l \sim \mathcal{N}(0, \mathbf{I}).$$

This can be directly optimized via gradient descent techniques. We set $L = 1$ in accordance with [34]. See Fig. 2.3 for a schematic example of a VAE model. The training procedure of a VAE is described in Alg. 3. An example of the outputs of a VAE model trained on the MNIST hand–written digits dataset [163] is in Fig. 2.4.

Algorithm 3 Variational Autoencoder training procedure.

Require: A VAE model with encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, training set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subset \mathcal{X}$, maximum number of iterations $I \in \mathbb{N}$, batchsize $B \in \mathbb{N}$.

$\boldsymbol{\phi}, \boldsymbol{\theta} \leftarrow$ Initialize parameters
$i \leftarrow$ Iteration counter
while $i < I$ or $\boldsymbol{\phi}, \boldsymbol{\theta}$ are not converged **do**
    $X_B \leftarrow$ A random batch of B samples from X
    $l \leftarrow \frac{1}{B} \sum_{j=1}^{L} \mathcal{L}_{\text{VAE}}(\boldsymbol{x}_j, \boldsymbol{\phi}, \boldsymbol{\theta}), \boldsymbol{x}_j \in X_B$
    $\boldsymbol{\phi} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\phi}} l$ update of encoder weights
    $\boldsymbol{\theta} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}} l$ update of decoder weights
    $i \leftarrow i + 1$
end while
return encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$

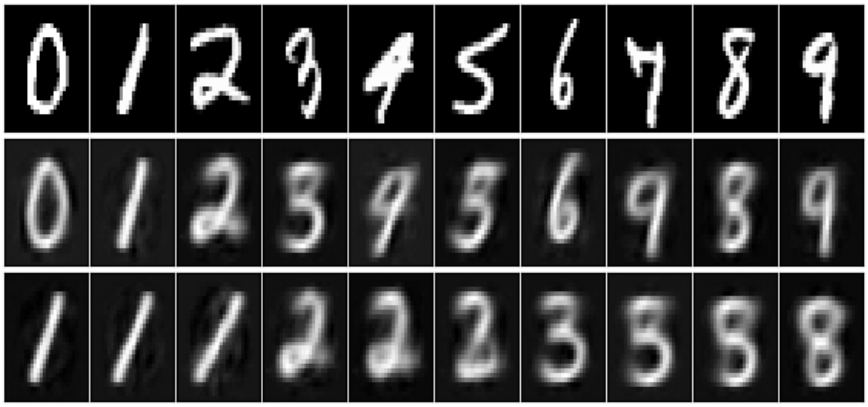


Figure 2.4: Example of a simple VAE trained on the MNIST dataset. Here, the neural networks modelling the encoder and decoder parameters contained two levels of convolutional blocks. Ground truth examples are in the top row, reconstructed samples are in the middle row, artificially generated digits are in the bottom row. The reconstructions are blurry, which is a typical VAE behaviour. Also, the reconstruction is imperfect for digits which resemble each other, such as "9", "4" and "7", or "3" and "8". The artificial digits were created by linearly interpolating between two coordinates in the latent space and using this as an input to the decoder. The VAE then produces a smooth interpolation between digits "1" and "8" that contains the related digits "2" and "3".

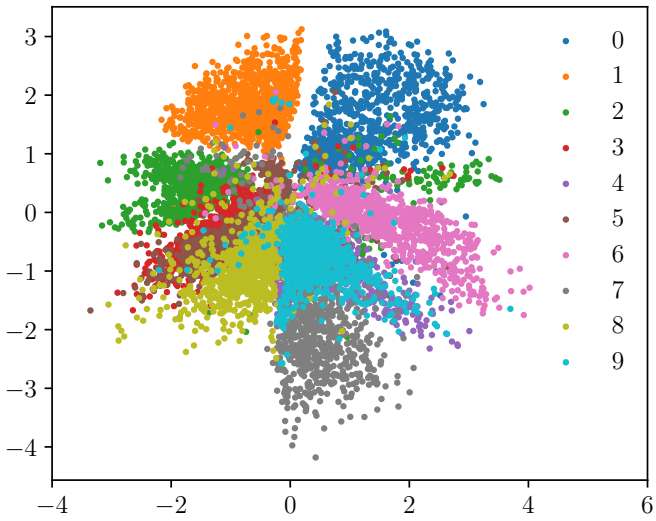


Figure 2.5: The latent space of the MNIST dataset produced by a VAE with a two-dimensional latent space. Note the overlapping of some digit encodings, e.g. "7" and "9".

Some VAE properties

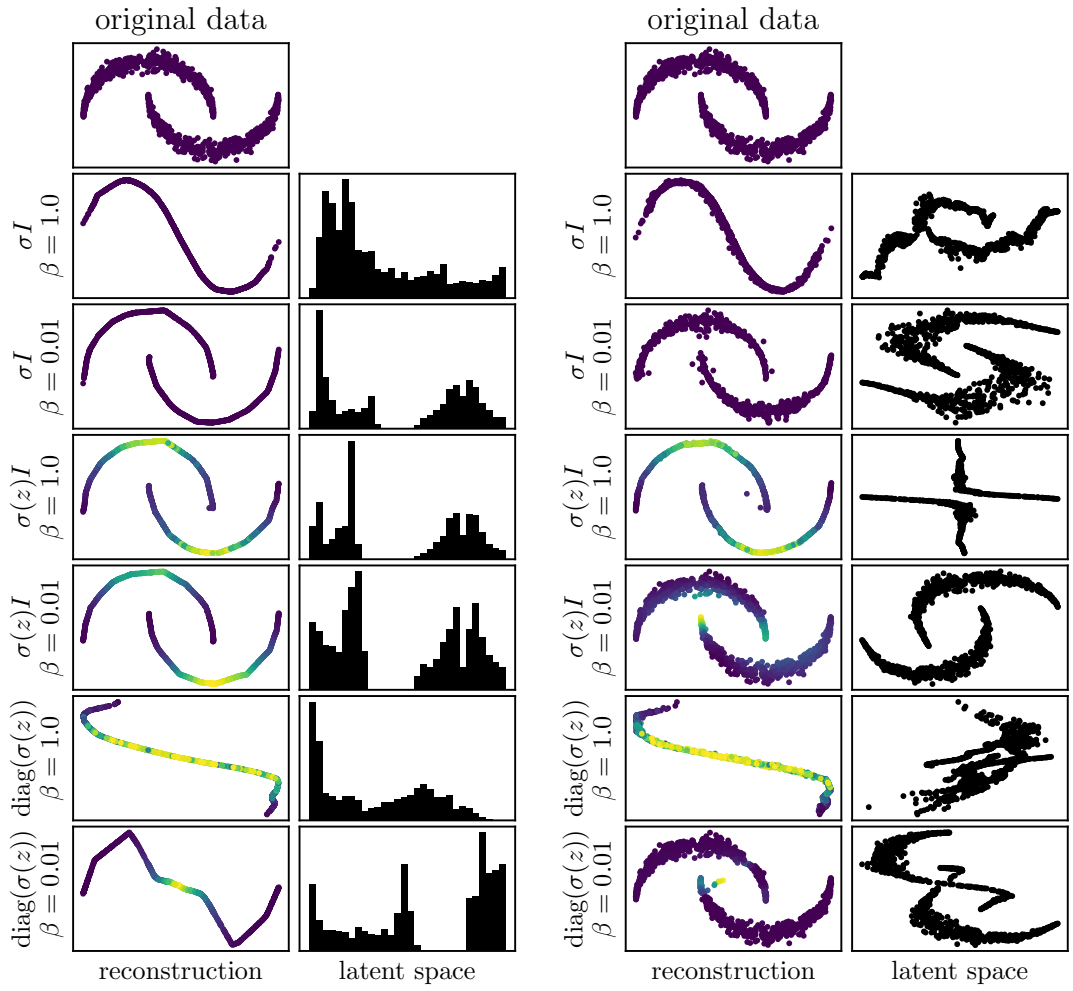
If a VAE model is correctly trained, we can assume that the encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and prior $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|0,\mathbf{I})$ are close to each other. Then, samples from the prior can be fed to the decoder and one can expect artificial samples that resemble the training data. A **generated sample** can therefore be described as

$$\tilde{\boldsymbol{x}} = \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}), \boldsymbol{z} \sim p(\boldsymbol{z}). \tag{2.16}$$

Note that a **reconstructed sample** is instead obtained by sampling the encoding $\boldsymbol{z}$ from the encoder through the reparametrization trick (2.13). This is also a place to note why we are using neural network representation of the encoder and decoder. Since neural networks have be proven to be universal function approximators, we know that given enough capacity, data and training time, the decoder can learn a mapping from the prior to an arbitrary function.

Fig. 2.5 shows the latent space of an example VAE model. Note that although the overall distribution of the encodings resembles the normal prior, in order to be able to reconstruct the samples from the encodings, the model has to learn to encode the samples from different digit classes to different parts of the latent space. It was shown [157] that the reconstruction and regularization parts of the VAE loss (2.11) actually work against each other. A model with encoder perfectly copying the randomness of the prior would not be able to reconstruct the inputs. On the other hand, a model without the latent regularization would not be able to generate new samples, as it would be practically identical with an AE model from Sec. 1.3.4, which is free to encode the different classes to arbitrary parts of the latent space. The basic loss (2.15) leads to a model that is usually in an equilibrium between both of these states. It is however possible to push the model in one of these directions by using a scaling parameter

$$\mathcal{L}_{\beta\text{VAE}}(\boldsymbol{x},\boldsymbol{\phi},\boldsymbol{\theta},\beta) = -\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] + \beta D_{\text{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\Big), \beta > 0. \tag{2.17}$$



(a) $h = 1$ (b) $h = 2$

Figure 2.6: An overview of VAE behaviour with respect to the scaling parameter β of the objective (2.17) and to the way the covariance of the decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is estimated. The VAE model was trained on the two–moons data, plotted in the plots at the very top. Variants with 1D (a) and 2D (b) latent spaces are compared, means of the decoder $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z})$ are plotted on the left and the latent representations on the right. Clearly, smaller values of β lead to better sample reconstruction, especially in the case of a two-dimensional latent space, as well as leading to a better separation of the encodings. This is understandable, since the prior $p(\boldsymbol{z}) = \mathcal{N}(0, \mathbf{I})$ is unimodal. From the top: the covariance of the decoder is given either by a fixed scalar ($\sigma^2 I, \sigma = 1$), by a scalar estimated from the data ($\sigma^2(\boldsymbol{z})I$), or by an estimate of its diagonal terms ($\text{diag}(\sigma^2(\boldsymbol{z}))$). The magnitude of the estimated variance in the two latter cases is denoted by color, where a brighter color corresponds to a higher value of variance. It is interesting that the second case ($\sigma^2(\boldsymbol{z})I$) seems to alleviate the reconstruction difficulties with higher β, while the estimation of the full covariance diagonal does not exhibit such property. Also, the third case seems to "exploit" the estimation of variance. Instead of pushing and optimizing the mean, it can instead simply put higher variance in the direction in which the reconstruction is worse and still incur only a small loss. Due to this behaviour, the second case seems to be the most robust and stable way of estimation of the reconstruction variance. Not surprisingly, the 2D case provides better reconstructions since it was provided with one more dimension to encode data to.

A VAE model trained with this loss is known as **BetaVAE** [157] and is one of the first VAE-based models that attempt some sort of **unsupervised disentanglement**. A disentangled model captures the possible factors of variation of a dataset in orthogonal dimensions of the latent space. Imagine a colored MNIST dataset, on which a disentangled model with a two dimensional latent space is trained. If properly disentangled, one latent dimension would capture the identity of the digit, while the other one would capture its color. This concept will be again revisited in Chapter 4. The influence of the β parameter is discussed in Fig. 2.6.

Instead of setting a fixed variance parameter σ^2 in the decoder, one can optimize and extract it instead from the last layer of the decoder, either as a scalar $\sigma^2_{\boldsymbol{\theta}}(\boldsymbol{z}) \in \mathbb{R}^+$ or even a full diagonal of the covariance $\text{diag}\left(\boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(\boldsymbol{z})\right), \boldsymbol{\sigma}^2_{\boldsymbol{\theta}}(\boldsymbol{z}) \in \mathbb{R}^{d+}$. From the experiment in Fig. 2.6, it seems (at least for tabular data) that the best results were surprisingly obtained not by the most complex variant, but the one with a scalar value $\sigma^2_{\boldsymbol{\theta}}$ optimized during training.

2.2.2 Wasserstein and adversarial autoencoders

The asymmetry of the KL divergence motivated search for a more accurate metric measuring the distance between the prior and the encoder. An alternative approach to VAE has been published in [164] and improved in [159], which proposes a general form of a generative autoencoder that uses a Wasserstein metric [165]. Unlike the KL divergence term in the VAE loss, which forces all the input data samples to zero (the mean of the standard prior), in **Wasserstein autoencoders** (WAE) the encoding is loosened, which reportedly leads to improved reconstruction [159]. A general form of the loss function of a WAE model is

$$\mathcal{L}_{\text{DW}}(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\phi}) = -\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] + \lambda D_{\text{W}}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) || p(\boldsymbol{z})\right), \tag{2.18}$$

where $\lambda > 0$ is a scalar hyperparameter, and D_{W} is a Wasserstein metric. The most commonly used form of the Wasserstein metric is the kernelized **maximum-mean-discrepancy** (MMD) with a kernel function $k_f : \mathbb{R}^h \times \mathbb{R}^h \to \mathbb{R}$, which was reported to perform well in matching high dimensional distributions [158]. From the theoretical point of view, KLD only matches the first and the second moment of the two distributions, while MMD can potentially match an infinite amount of moments with the right kernel. Some authors [159] argue that by minimizing KLD, the latent representation might become uninformative for the decoder to reconstruct the code. On the other hand, MMD maximizes the mutual information between $\boldsymbol{x}$ and $\boldsymbol{z}$ [158].

Under some mild assumptions about the kernel function k_f, which needs to be characteristic and positive-definite (see details in [159]), the MMD can be expressed in such a way that enables optimization of the model by backpropagation. Then, the MMD of the prior and the encoder can be approximated solely by comparing sets of samples from these distributions $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n\}, \tilde{Z} = \{\tilde{\boldsymbol{z}}_1, \tilde{\boldsymbol{z}}_2, \ldots, \tilde{\boldsymbol{z}}_n\}, \boldsymbol{z}_i \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}), \tilde{\boldsymbol{z}}_i \sim p(\boldsymbol{z}), \forall i \in \hat{n}$ in a closed expression

$$\text{MMD}_k(Z, \tilde{Z}) = \frac{1}{n(n-1)} \sum_{i \neq j} k_f(\boldsymbol{z}_i, \boldsymbol{z}_j) + \frac{1}{n(n-1)} \sum_{i \neq j} k_f(\tilde{\boldsymbol{z}}_i, \tilde{\boldsymbol{z}}_j) - \frac{2}{n^2} \sum_{i,j} k_f(\boldsymbol{z}_i, \tilde{\boldsymbol{z}}_j). \tag{2.19}$$

The most notable characteristic of the MMD is that in practice, it only requires samples from the distributions in question, and it is therefore less restricting than the

Algorithm 4 Wasserstein autoencoder training procedure.

Require: A WAE model with encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ and a prior $p(\boldsymbol{z})$, training set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subset \mathcal{X}$, maximum number of iterations $I \in \mathbb{N}$, batchsize $B \in \mathbb{N}$, regularization coefficient $\lambda > 0$, characteristic positive definite kernel k_f, standard deviation parameter $\sigma > 0$.

$\boldsymbol{\phi}, \boldsymbol{\theta} \leftarrow$ Initialize parameters
$i \leftarrow$ Iteration counter
while $i < I$ or $\boldsymbol{\phi}, \boldsymbol{\theta}$ are not converged **do**
    $X_B \leftarrow$ A random batch of B samples from X
    $Z \leftarrow \{\boldsymbol{z}_j \sim q_\phi(\boldsymbol{z}|\boldsymbol{x}_j), \boldsymbol{x}_j \in X_B\}$ samples from the encoder
    $\tilde{Z} \leftarrow$ A random batch of B samples from prior $p(\boldsymbol{z})$
    $l \leftarrow \frac{1}{B}\sum_{j=1}^{B} \|\boldsymbol{x}_j - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}_j)\|_2^2 + \lambda \mathrm{MMD}_k(Z, \tilde{Z}), \boldsymbol{x}_j \in X_B, \boldsymbol{z}_j \in Z$
    $\boldsymbol{\phi} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\phi}} l$ update of encoder weights
    $\boldsymbol{\theta} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}} l$ update of decoder weights
    $i \leftarrow i + 1$
end while
return encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$

KLD, which required normal prior and encoder in order to obtain the analytic expression (2.12). This offers the potential for the use of a variety of prior and encoder distributions, however, the reparametrization trick (2.13) or a similar technique must be used in order to keep the sampling from $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ a differentiable operation. The two most common choices of k_f are the RBF $k_f(\boldsymbol{z}, \tilde{\boldsymbol{z}}) = \exp(-\gamma\|\boldsymbol{z} - \tilde{\boldsymbol{z}}\|_2^2), \gamma > 0$ and inverse multiquadratics (IMQ) $k_f(\boldsymbol{z}, \tilde{\boldsymbol{z}}) = (c^2 + \|\boldsymbol{z} - \tilde{\boldsymbol{z}}\|_2^2)^\beta, c > 0, \beta < 0$ kernels. The training algorithm for a WAE with the MMD is in Alg. 4. Note that although the parameters $\boldsymbol{\phi}$ do not appear explicitly in the overall loss function, they are present through the samples in Z.

A different architecture arises when the **Jensen–Shannon divergence** D_{JS} [166] is used in place of D_{W}. The JSD is a symmetrical (unlike KL divergence) measure of distance between two probability distributions. The use of JSD leads to a model that is called the **adversarial autoencoder** (AAE) and was originally proposed in [160]. The connection between an AAE model and the general formulation (2.18) was shown in [159]. To regularize the encoder, we add a discriminator $d_{\boldsymbol{\eta}}(\boldsymbol{z}) : \mathcal{Z} \to [0, 1]$ represented by a neural network with parameters $\boldsymbol{\eta}$. The discriminator has a similar function to the one in the GAN model from Sec. 2.1. It tries to recognize latent space samples produced by the encoder and those sampled from the prior $p(\boldsymbol{z})$. The difference is that here, the discriminator operates on the latent space $\mathcal{Z}$ instead of the data space $\mathcal{X}$.

A modified loss (2.3) is used to train the discriminator, while the loss function (2.2) is used in place of the general Wasserstein distance in (2.18) for training of the encoder and decoder. Explicitly,

$$\mathcal{L}_{\mathrm{AAE}_d}(\boldsymbol{z}, \tilde{\boldsymbol{z}}, \boldsymbol{\eta}) = -\ln d_\eta(\boldsymbol{z}) - \ln(1 - d_{\boldsymbol{\eta}}(\tilde{\boldsymbol{z}})), \tag{2.20}$$

$$\mathcal{L}_{\mathrm{AAE}}(\boldsymbol{x}, \boldsymbol{z}, \tilde{\boldsymbol{z}}, \boldsymbol{\theta}, \boldsymbol{\phi}) = \frac{1}{\sigma^2}\|\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z})\|_2^2 - \lambda \ln d_{\boldsymbol{\eta}}(\tilde{\boldsymbol{z}}), \tag{2.21}$$

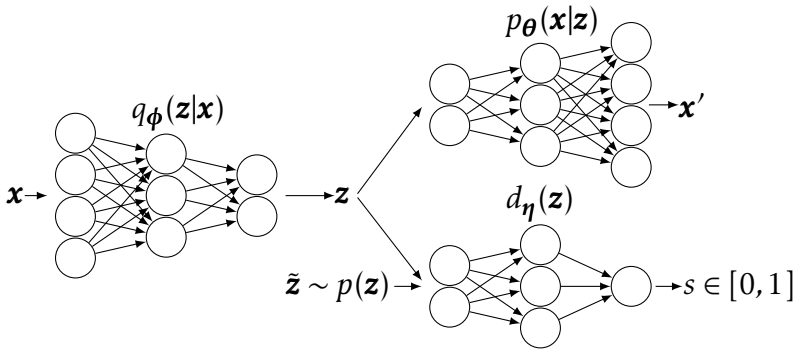


Figure 2.7: A schematic of a AAE model consisting of fully connected layers. A data sample x is mapped to latent space representation $\tilde{z}$ via the encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|x)$. Also, a sample z is sampled from the latent prior $p(\boldsymbol{z})$. Both z and $\tilde{z}$ are passed to the discriminator $d_{\eta}(\boldsymbol{z})$ that produces a score s – the probability that the input sample comes from the prior. At the same time, the latent representation is passed to the decoder $p_{\boldsymbol{\theta}}(x|z)$ which maps it to a reconstruction $\tilde{x}$.

Algorithm 5 AAE training procedure.

Require: An AAE model with encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, a prior $p(\boldsymbol{z})$ and a discriminator $d_{\boldsymbol{\eta}}(\boldsymbol{z})$, training set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subset \mathcal{X}$, maximum number of iterations $I \in \mathbb{N}$, batchsize $B \in \mathbb{N}$, regularization coefficient $\lambda > 0$, standard deviation parameter $\sigma > 0$.

$\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{\eta} \leftarrow$ Initialize parameters
$i \leftarrow$ Iteration counter
while $i < I$ or $\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{\eta}$ are not converged **do**
    $X_B \leftarrow$ A random batch of B samples from X
    $Z \leftarrow \{\boldsymbol{z}_j \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_j), \boldsymbol{x}_j \in X_B\}$ samples from the encoder
    $\tilde{Z} \leftarrow$ A random batch of B samples from prior $p(\boldsymbol{z})$
    $\tilde{X} \leftarrow \{\tilde{\boldsymbol{x}}_j \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|\tilde{\boldsymbol{z}}_j), \tilde{\boldsymbol{z}}_j \in \tilde{Z}_B\}$ generated samples
    $l_{ae} \leftarrow \frac{1}{B} \sum_{j=1}^{B} \mathcal{L}_{\text{AAE}}(\boldsymbol{x}_j, \boldsymbol{z}_j, \tilde{\boldsymbol{z}}_j, , \boldsymbol{\theta}, \boldsymbol{\phi}), x_j \in X_B, \boldsymbol{z}_j \in Z, \tilde{\boldsymbol{z}}_j \in \tilde{Z}$
    $l_d \leftarrow \frac{1}{B} \sum_{j=1}^{B} \mathcal{L}_{\text{AAE}_d}(\boldsymbol{z}_j, \tilde{\boldsymbol{z}}_j, \boldsymbol{\eta}), \boldsymbol{z}_j \in Z, \tilde{\boldsymbol{z}}_j \in \tilde{Z}$
    $\boldsymbol{\phi} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\phi}} l_{ae}$ update of encoder weights
    $\boldsymbol{\theta} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}} l_{ae}$ update of decoder weights
    $\boldsymbol{\eta} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\eta}} l_d$ update of discriminator weights
    $i \leftarrow i + 1$
end while
return encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$, discriminator $d_{\boldsymbol{\eta}}(\boldsymbol{z})$

where $\lambda > 0$, $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, $\tilde{\boldsymbol{z}} \sim p(\boldsymbol{z})$. Not that the loss (2.21) has implicit functional dependence on $\boldsymbol{\phi}$ through the samples $\boldsymbol{z}$. A schematic of the AAE model is in Fig. 2.7 and the AAE training procedure is described in Alg. 5. Again, any prior $p(\boldsymbol{z})$ that we can sample from is suitable for the regularization of AAE, even a multimodal one. In practice, an AAE model compared to a WAE behaves similarly as GAN compared to VAE. The adversarial loss leads to less blurry reconstructions and generated samples at the cost of higher training instability [159]. One way to gain advantages of both is to use an AAE architecture as depicted in Fig. 2.7 and add the MMD regularization term (2.19) to the loss (2.21).

The single-mode prior $p(\boldsymbol{z})$ of the VAE model stimulates the distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ to have a single mode as well, and therefore it is hard to fit data with a multi-modal latent distribution. The publication [167] proposes a learnable multimodal **Vamp** prior realized as a mixture of $K \in \mathbb{N}$ independent Gaussian components. However, the Vamp prior is not compatible with the basic VAE model since its use does not lead to an analytical expression of the KLD (2.12). Fortunately, we have just described two alternatives that only require samples from the prior and which are viable with Vamp. The parameters of the components of the mixture can be then learned together with the parameters of the model. We will present some experiments with this prior in Chapter 3.

2.2.3 Generative autoencoders in anomaly detection

Anomaly scores

The likelihood function (2.7) constitutes the ideal anomaly score. Some training losses such as ELBO (2.11) were designed as approximations of the likelihood and can thus be used as anomaly scores. However, this interpretation is not so clear for other training losses, i.e. (2.18) or (2.21), hence these models were proposed for anomaly detection with separately defined anomaly scores. Nevertheless, many scores are interchangeable, giving rise to another degree of freedom (hyperparameter) for the use of autoencoders in anomaly detection. A common score is based on the first term in the loss i.e. a Monte Carlo estimate of the expectation of conditional log-likelihood over the encoder $-\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$, which is in the case of a isotropic Gaussian decoder equal to

$$s_{\mathrm{rs}}(\boldsymbol{x}) = \frac{1}{\sigma^2 L} \sum_{l=1}^{L} \|x - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}^l)\|_2^2. \tag{2.22}$$

This score, called the **sampled reconstruction error**, was shown in [127] to be more accurate than evaluating (2.7) by sampling $\boldsymbol{z}$ from the prior $p(\boldsymbol{z})$, which is equal to computing $-\mathbb{E}_{p(\boldsymbol{z})}[\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$. Further simplification is based on replacing samples from the encoder by its mean $-\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}))$ and therefore avoiding sampling, yielding the common **reconstruction error** score

$$s_{\mathrm{rm}}(\boldsymbol{x}) = \|x - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}))\|_2^2. \tag{2.23}$$

The usage of (2.23) is justified by the assumption that taking the mean at the encoder should approximate (2.22) while having lower computational demands. However, as we show in some of the experiments in Chapter 3, the score (2.22) performs generally better.

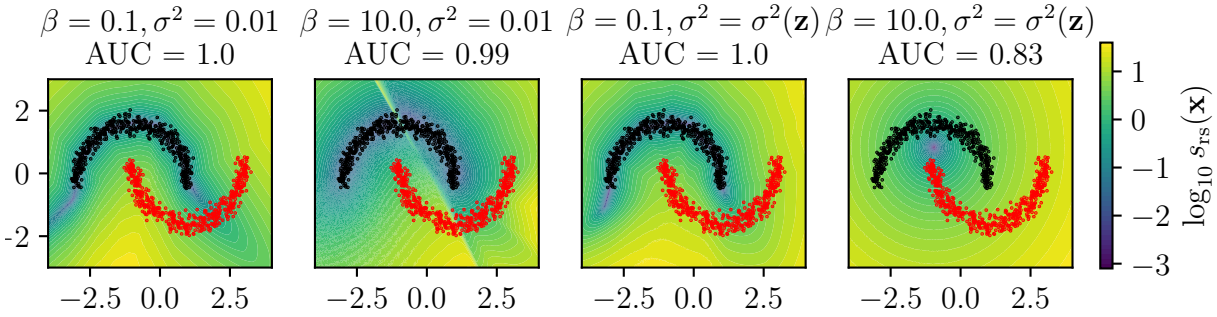


Figure 2.8: Anomaly detection with a VAE trained with the (2.17) objective on the *two bananas* dataset with varying value of β and different way of treating the decoder variance σ^2, where it is either considered fixed, or it is subject to optimization and funcionally dependent on the latent encoding $\boldsymbol{z}$. The contours of the anomaly score (2.22) are shown.

For adversarial autoencoders, these simplifications can be combined with the discriminator score [23, 146],

$$s_{\text{ad}}(\boldsymbol{x}) = \alpha s_{\text{rm}}(\boldsymbol{x}) + (1-\alpha) d_{\boldsymbol{\eta}}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x})), \alpha \in [0,1]. \tag{2.24}$$

The reconstruction error-based anomaly scores were criticized in [168] for not capturing the true data density $p(\boldsymbol{x})$. The proposed replacement is based on the orthogonal decomposition of the data into $\boldsymbol{x} = \boldsymbol{x}^{\perp} + \boldsymbol{x}^{||}$ where the $\boldsymbol{x}^{||}$ lies in the tangent space of to the manifold defined by the decoder. This allows to decompose the marginal likelihood into a product of two orthogonal parts

$$p(\boldsymbol{x}) \approx p(\boldsymbol{x}^{||}) p(\boldsymbol{x}^{\perp}), \tag{2.25}$$

where $p(\boldsymbol{x}^{\perp})$ is the reconstruction error term, e.g. (2.22), and $p(\boldsymbol{x}^{||})$ is obtained by transformation of variables (**??**). The calculation of (2.25) is expensive, as it needs to compute the singular value decomposition of a Jacobian. For more details, see [168] or [169] and Sec. 4.2.2.

The use of the score (2.22) is demonstrated in Fig. 2.8. In the plot, the difference between fixed and estimated decoder variance are explored again, similarly to Fig. 2.6. It is again shown that over-parametrizing the model might lead to undesired behaviour, as in the rightmost subplot, where the VAE collapsed into a single-modal Gaussian distribution.

Practical examples

The reason for using the VAE model instead of a reconstruction error of AE is the improved generalization that was described in the previous section. It is discussed in [170] that a VAE model is equivalent to a non–linear robust PCA model and is proficient at dismissing sparse outliers. The authors also make note of the fact that VAE is very efficient in pruning of unnecessary latent dimensions in case when the real latent structure has lower dimension than the chosen VAE latent space.

In [127] the authors present a so-called **DonutVAE** model with an enhanced loss function to detect anomalies in times series data. The architecture is similar to that of a vanilla VAE, but the training loss (2.15) is modified to consider the nature of the time series data. The authors then show that usage of this loss function improves overall results. This however has a caveat – known anomalous samples must be available, otherwise the proposed loss is the same as (2.15). Furthermore, the authors claim that using reconstruction probability (2.22) can be seen as a weighted kernel density estimate. A similar problem is solved in [171], where a VAE model coupled with an LSTM recurrent neural network with attention mechanism is used for detecting anomalies in time series. Again, the model operates in the semisupervised setting, and it is able to capture the temporal structure in the data.

In the self-adversarial Variational Autoencoder (adVAE) [172], an encoder-decoder pair is augmented with a transformer [173], whose goal is to simulate anomalies during training. A seeming flaw of the model is that it is trained only on normal data, and there is no link between the real and the simulated anomalies, although in the original paper, the authors claim its uperiority over a number of well-known competing methods. The sampled reconstruction error 2.22 is used as an anomaly score.

A variant of AAE is used in [149] where it is benchmarked on the MNIST problem. Standard normal distribution prior is compared to a Gaussian mixture model (GMM) latent prior and a special rejection component is introduced for representation of anomalies. In [174] the AAE model is compared to VAE on the task of detection of brain abnormalities in MRI images. The loss function of the autoencoding part is enhanced by a term $\gamma\|\boldsymbol{z}-\boldsymbol{z}'\|, \gamma > 0$, where $\boldsymbol{z}$ is a latent representation of a sample $\boldsymbol{x}$ and $\boldsymbol{z}'$ is the latent representation of the reconstructed sample $\boldsymbol{x}'$, which is supposed to improve consistency of the representation. The thesis [175] also uses AAEs for detection of abnormalities in videos. The model presented in [168] uses an additional discriminator on top of the decoder in AAE to improve its reconstruction and generative properties. The model is then tested for anomaly detection on standard benchmark datasets.

Despite its name, GANomaly [176, 177] is more related to adversarial autoencoders than to GANs. It consists of an encoder-decoder-encoder architecture with a discriminator, similar to an AAE. The anomaly score is the difference between latent representations of a sample after the first and second encoding. An upgrade to this model, skip-GANomaly [178], uses skip connections in a U-Net type architecture. Here, the anomaly score is a combination of reconstruction error and feature-matching loss.

Another way of using generative autoencoders for anomaly detection is to combat the **curse of dimensionality** by employing their ability to produce a low–dimensional representation of high–dimensional data that preserves the important relations between individual datapoints. Then, an anomaly detection model (be it a generative or a classical one) can be trained on the data encoded in the latent space. This **two-stage** approach is especially useful when the target domain is image data and some kind of vectorization has to be used anyway. Usage of this technique will be demonstrated an experiment in the next chapter and it has been used in many publications [179, 180, 38, 17]. It is also used in [181], where both stages are a VAE and the second stage has the same input and latent space dimensionality (therefore it does not compress data at all). Although this paper does not present an application in anomaly detection, it shows an improvement in learning of the latent space prior. The authors of [72] couple an ordinary autoencoder with a Gaussian mixture model represented by

a neural network. The AE reduces the problem dimension to help overcome the curse of dimensionality, while the GMM model serves as a density estimate in the latent space. Both the autoencoder and the GMM model are learnt jointly which improves the performance of the model. Secondly, the input of the GMM model is not only the latent representation, but also the reconstruction error of the sample. Although this model is not based on VAE, the proposed loss function can be viewed as being similar to the general form (2.18), as it imposes some kind of structure on the distribution of encodings.

In [38], the model optimizes the projection of data (by virtue of NNs) to a new space, where they can be easily enclosed in a sphere of minimum radius. The approach presented in [17, 180] explicitly splits the creation of the detector into two parts. It first trains a VAE (and its variants), and then it fixes the encoder. The anomaly score is calculated by a kNN [17] or by OC-SVM [180] detectors in the latent space, obtained by projecting the sample by the fixed encoder. The two-stage models can also be viewed as a kNN with a trained metric or OC-SVM with a trained kernel. The embedding can be optimized differently, for example, by enforcing the margin between anomaly candidates and normal data as done in the REPEN [80] method, which uses an ensemble of 1NN detectors as the second stage.

2.3 Normalizing flows

Generative models that are based on **normalizing flows** aim to model the data distribution $p(\boldsymbol{x})$ in a general and expressive manner. While GANs do not explicitly consider a model for $p(\boldsymbol{x})$ and only build a proxy (generator) that enables sampling, VAEs instead optimize a posterior $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ through a lower bound on the data log-likelihood. In this regard, normalizing flows are the most exact in handling of the data distribution. This is at the cost of some requirements that need to be followed when building and training a normalizing flow model that will be described in the following text. Normalizing flows have been used to model complex probability distributions in multiple publications [35, 182, 183, 184]. Since the exact log-likelihood of a new sample is readily available when using a normalizing flow model, they have been applied to anomaly detection tasks in [130, 131, 185, 186, 187].

A normalizing flow is a transformation $f_{\boldsymbol{\nu}} : \mathcal{Z} \to \mathcal{X}$ that can be used to express a data sample $\boldsymbol{x} \sim p(\boldsymbol{x})$ as

$$\boldsymbol{x} = f_{\boldsymbol{\nu}}(\boldsymbol{z}), \boldsymbol{z} \sim p(\boldsymbol{z}), \tag{2.26}$$

where $p(\boldsymbol{z})$ is the prior which has some desirable properties. The defining property of normalizing flow models, that differentiates them from VAEs and GANs, is that the transformation $f_{\boldsymbol{\nu}}$ must be invertible and both $f_{\boldsymbol{\nu}}$ and $f_{\boldsymbol{\nu}}^{-1}$ must be differentiable. This means that the dimension of $\boldsymbol{x}$ and $\boldsymbol{z}$ must be the same, so here we assume $\mathcal{Z} = \mathcal{X} = \mathbb{R}^d$. Therefore, from (2.26) and the formula for change of variables, we have

$$p(\boldsymbol{x}) = p(\boldsymbol{z})\left|\frac{\partial f_{\boldsymbol{\nu}}(\boldsymbol{z})}{\partial \boldsymbol{z}}\right|^{-1} = p(f_{\boldsymbol{\nu}}^{-1}(\boldsymbol{x}))\left|\frac{\partial f_{\boldsymbol{\nu}}^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right| \tag{2.27}$$

which contains the determinant of the square Jacobian matrix. In practice, the normalizing flow $f_{\boldsymbol{\nu}}$ is implemented as a neural network with weights $\boldsymbol{\nu}$. Notice that since it is fully invertible, we do not need an encoder-decoder pair like in a VAE. Again, like in the case of the vanilla VAE and GAN models, the prior might be e.g. $\mathcal{N}(0, \mathbf{I})$. Thanks

to (2.27), the model can be directly optimized using the log-likelihood $-\ln p(\boldsymbol{x})$. When properly trained, it is possible to use a normalizing flow model to generate new samples using (2.26) or evaluate its density using (2.27), which is the basis for anomaly detection.

One can see the transformation $f_{\boldsymbol{\nu}}$ as a process of reshaping the space $\mathbb{R}^d$ in order for $p(\boldsymbol{z})$ to closely fit $p(\boldsymbol{x})$. This process is commonly broken down into incremental steps described as a series of $K \in \mathbb{N}$ elementary invertible operations (flows)

$$f_{\boldsymbol{\nu}} = f_{\boldsymbol{\nu},K} \circ f_{\boldsymbol{\nu},K-1} \circ \ldots \circ f_{\boldsymbol{\nu},1} \tag{2.28}$$

which are composed together to form the full normalizing flow. This gives the model its name, as we can observe the "flow" of transformed variables

$$\boldsymbol{z}_k = f_{\boldsymbol{\nu},k}(\boldsymbol{z}_{k-1}), k \in \hat{K}, \boldsymbol{z}_0 = \boldsymbol{z}, \boldsymbol{z}_K = \boldsymbol{x}. \tag{2.29}$$

From (2.27), we can express the normalizing flow objective as

$$\mathcal{L}_f(\boldsymbol{x}, \boldsymbol{\nu}) = -\ln p(\boldsymbol{x}) = -\ln p(\boldsymbol{z}) + \sum_{k=1}^{K} \ln \left| \frac{\partial f_{\boldsymbol{\nu},k}(\boldsymbol{z}_{k-1})}{\partial \boldsymbol{z}_{k-1}} \right|, \tag{2.30}$$

where the inverse evaluation in the direction from $\boldsymbol{x}$ to $\boldsymbol{z}$ is given by $\boldsymbol{z}_{k-1} = f^{-1}_{\boldsymbol{\nu},k}(\boldsymbol{z}_k)$. As mentioned earlier, this is used to train a normalizing flow model and is also a well-suited anomaly score.

The different flow models vary in the way the individual tranformations $f_{\boldsymbol{\nu},k}$ are constructed and trained, which is largely dependent on their application. In the **Nonlinear Independent Component Estimator** (NICE) [35], each individual flow is a called **aditive coupling layer**. It is an operation that splits the input $\boldsymbol{z}_{k-1}$ into two halves along its dimensions. The first half $\boldsymbol{z}_{k-1,1}$ remains unchanged and the second $\boldsymbol{z}_{k-1,2}$ undergoes a transformation using a parametrizable function m, which is a simple fully connected neural network with ReLU activations. Therefore

$$\boldsymbol{z}_{k,1} = \boldsymbol{z}_{k-1,1} \tag{2.31}$$

$$\boldsymbol{z}_{k,2} = \boldsymbol{z}_{k-1,2} + m(\boldsymbol{z}_{k-1,1}), \tag{2.32}$$

and the reverse operation is

$$\boldsymbol{z}_{k-1,1} = \boldsymbol{z}_{k,1} \tag{2.33}$$

$$\boldsymbol{z}_{k-1,2} = \boldsymbol{z}_{k,2} - m(\boldsymbol{z}_{k,1}), \tag{2.34}$$

which is differentiable since m is differentiable and its jacobian is easily computed. The **Real-valued Non-Volume Preserving** (RealNVP) [188] is based on the NICE model and uses an **affine coupling layer**, where the second half of input dimesions undergo a scale-and-shift transformation

$$\boldsymbol{z}_{k,1} = \boldsymbol{z}_{k-1,1} \tag{2.35}$$

$$\boldsymbol{z}_{k,2} = \boldsymbol{z}_{k-1,2} \odot \exp(s(\boldsymbol{z}_{k-1,1})) + t(\boldsymbol{z}_{k-1,1}), \tag{2.36}$$

where $s, t : \mathbb{R}^d \to \mathbb{R}^{d/2}$ are again represented by neural networks. The reverse operation for the second input is

$$\boldsymbol{z}_{k-1,2} = (\boldsymbol{z}_{k,2} - t(\boldsymbol{z}_{k,1})) \odot \exp(-s(\boldsymbol{z}_{k,1})). \tag{2.37}$$

The **Glow** [189] demonstrates applicability of normalizing flows to image data, which is otherwise a very difficult domain. The flow used in this model consists of invertible 1x1 convolutions and it is demonstrated that it is able to generate realistic images of celebrity faces. Some normalizing flow models use an autoregressive constraint [183] to represent the flow of information from the prior to the data distribution. The most prominent is the **Masked Autoregressive Flow** (MAF) [190]. A promising class of **Sum-product-Transform Networks** (SPTN) [186] combines normalizing flows with a graphical model. For a more complex overview and introduction to generative models based on normalizing flows, see [191, 192].

Although there are some applications of normalizing flows to anomaly detection [193, 194, 195, 196], they are less popular than different autoencoding generative models. This might seem surprising, as they are at the same time praised for the theoretical exactness of the log-likelihood computation. However, it seems that maatching high-dimensional, complex distributions this way is very difficult and requires normalizing flow models with hundreds of flow levels. Thus, they are very slow to train on complex data, which is also due to the requirement $\dim(\mathcal{Z}) = \dim(\mathcal{X})$. The Glow model requires 200 million parameters optimized over 13,000 GPU-hours to achieve simillar performance to computationally much less expensive alternatives [189]. Still, normalizing flows present a very interesting branch of generative modeling.

2.4 Anomaly detection with generative models: practical example

In this section, an application of generative modelling on a practical example is presented. It was previously published in [17]. Some of different generative autoencoders presented in Sec. **??** will be used here to model data measured in a complex scientific experiment. Apart from the comparison of the previously presented anomaly scores, consider this to be a preliminary exploration of the already mentioned two-stage modeling principle, where an autoencoding model is used to create meaningful low-dimensional representation of complex data, and which is then coupled with some classical detector from Sec. 1.3. This concept will also be further explored in Chapter 4.

2.4.1 The application problem

As already mentioned in the introductory chapter, physics has recently enjoyed an influx of very large amounts of data [197, 198] that needs to be processed and most importantly, from which new scientific discoveries may be extracted. This is also true for the field of plasma fusion, which strives to ignite and control a fusion reaction as a clean and basically inexhaustible source of energy. The ITER project [199] is expected to produce up to 2 petabytes of data every day. This naturally calls for automatic processing of the data for a multitude of tasks, including anomaly detection. In this section, an anomaly detection problem that appears during the operation of the COMPASS [200] tokamak will be dissected.

During the operation of COMPASS, **Alfvén eigenmodes** [201, 202, 203] were observed. Alfvén eigenmodes are magnetic instabilities that degrade the performance of the tokamak and possibly endanger and the plasma-facing components of the magnetic chamber [204]. For this reason, their automatic detection is very important. Also,

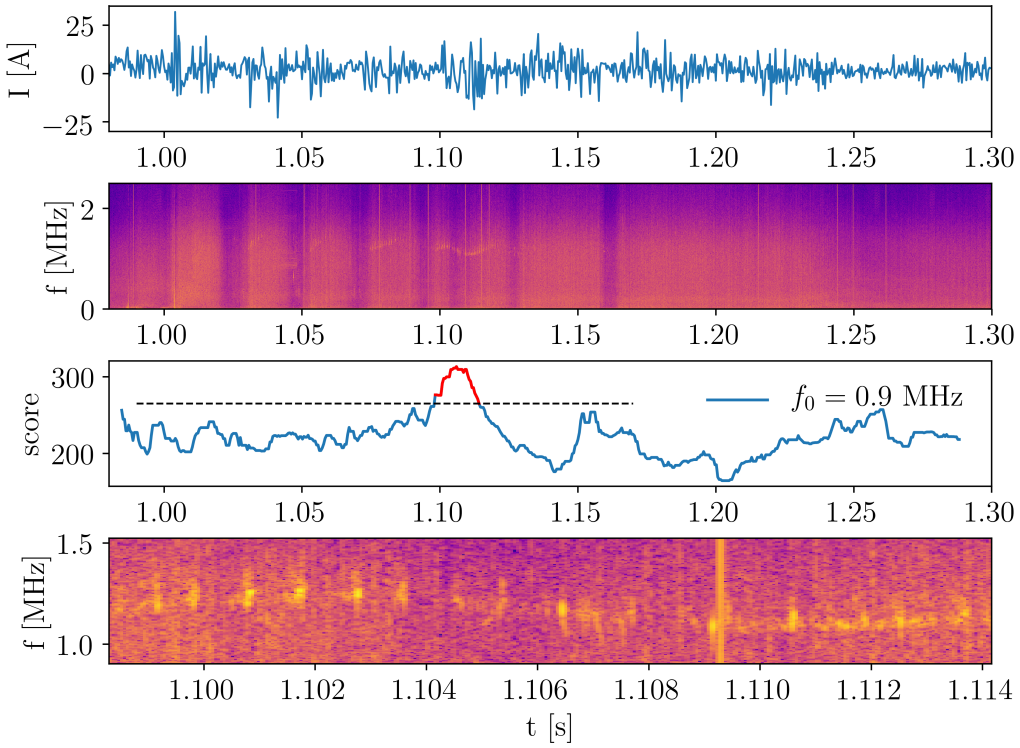


Figure 2.9: COMPASS shot 10870. Raw U-probe signal is in the upper plot. The corresponding spectrogram is below that. The score is the output of one of tested models over the spectrogram at $f_0 = 0.9$ MHz and it is plotted third from the top. The highest peak around 1.1s corresponds to a detected chirping mode. A close-up of the spectrogram part containing the chirping mode as detected from the red part of the score plot is at the bottom. The size of the close-up is 128×311 pixels.

it may offer an opportunity for the study of their interactions with high-energy particles present in the plasma during an experiment. On COMPASS, chirping Alfvén eigenmodes are estimated to appear in about 0.1% of all experiments. The primary means of their identification is manual inspection of spectrograms drawn from the signal of certain magnetic probes, which is very time-consuming. See Fig. 2.9 for an example of the measured signal, a spectrogram that is derived from it and a detected chirping Alfvén eigenmode. The spectrograms such as the one in Fig. 2.9 are large, so they are divided into patches of 128x128 pixels which is a feasible input size for current convolutional neural network architectures. It is also enough to capture most of a typical chirping mode as can be seen in the bottom plot in Fig. 2.9. The patches that contain a chirping Alfvén eigenmodes are considered to be anomalies. There are 370 labeled examples of patches with a chirping Alfvén eigenmode and a large database containing 33000 unlabeled (but considered normal) patches. This is a typical anomaly detection problem, where labeled anomalies are only used for evaluation and comparison of different models and the training dataset is considered to be anomaly-free.

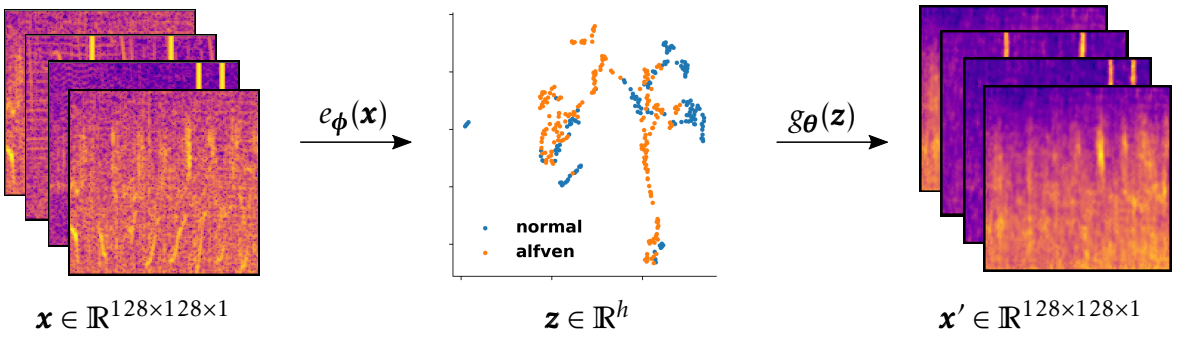


Figure 2.10: A schematic diagram of the convolutional autoencoder used for our experiments. Spectrogram patches are encoded through several convolutional[205], maxpooling[206] and dense (fully connected) layers into h-dimensional vectors (here $h = 2$) and then decoded back with transposed convolutions and upscaling layers.

2.4.2 The experimental setup

Two basic experimental setups are tested in this section. Both are be based on generative autoencoders from Sec. 2.2. The basic models have similar architectures, but they differ in the probability divergences used to regularize the latent space. The KLD (2.12) of a VAE model is compared against the MMD (2.19). A WAE model with a discriminator is regularized by JSD which results in the training losses (2.20) and (2.21). Finally, a plain AE from Sec. 1.3.4 is included to verify that a regularized latent space is useful. The individual components of the models (encoders, decoders, discriminators) are represented by neural networks with convolutional architecture. This is the most often used architecture for image data, as several levels of convolution operations are designed to capture shift invariant features at different scales of an image. For more technical details on the construction of the models, see [17]. A schematic of a prototypical model is in Fig. 2.10. The models have Gaussian encoders and decoders. The prior is either $\mathcal{N}(0, \mathbf{I})$ or Vamp, as described in Sec. 2.2.2, and the choice is treated as a hyperparameter. Note that Vamp is only possible to use with the MMD or JSD metrics or their combination.

In the first setup, the described models are compared against each other as primary anomaly detectors in the **one class** setting. This means that they are trained on the (assumed) normal spectrogram patches and the anomaly score is the sampled reconstruction error (2.22) in case of probabilistic autoencoders, and the reconstruction error (1.21) in case of the plain autoencoder model. This will test the proposed robustness of generative autoencoders.

In the second setup, the encoding capabilities of generative encoders are leveraged to create uncorrelated low-dimensional representations of spectrogram patches. This is combined with a classifier in a combined **two-stage** model. The first stage is a convolutional generative autoencoder trained with unlabeled data. Through the use of MMD or JS_D measures and Vamp, a separation of the encoded data into clusters that contain similar inputs can be enforced, which makes the task of the classifier easier. The second stage is a classifier that is trained on encoded labeled data. Two different classifiers were tested. The kNN classifier, which is similar to the kNN anomaly detector described in Sec. 1.3.2, and where the score of a sample is the average label of its

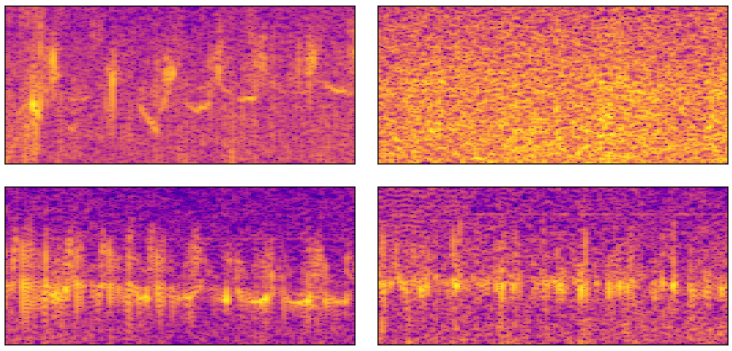


Figure 2.11: Examples of spectrogram patches identified as containing a chirping mode.

k-nearest neighbors, assuming that the label $y \in \{0, 1\}$ is zero for normal samples. The GMM classifier with M components was fitted on the latent representations of both labeled and unlabeled training data. Afterward, we determine one or more components of the mixture into which the positively labeled training samples are most likely to be projected via the encoder. Then, for a new sample, the score is the (average) log-likelihood of the sample in the anomalous components.

For more details on model architecture and hyperparameters used in the experiments, see [17]. For both experimental setups, 10-fold crossvalidation over different splits of training and testing data was done. In the experimental results below, the models are selected using average performance on the test set. This is not ideal, as will be shown in Chapter 3, but here the very low number of labeled anomalies would lead to nonrobust results if the proper train/validation/test split was done, which is even more pronounced by the splitting procedure described in Sec. **??**.

2.4.3 Results

During the use of one of the proposed models in the production environment of the tokamak, the following workflow would be observed. A set of experiments to be analyzed would be selected. Then, the needed signals would be extracted, spectrograms computed and divided into patches of appropriate size. These would be fed to a trained model that would produce scores to enable ranking of the patches. Since this would produce thousands of patches and scores for each tokamak experiment, the operator would ideally only want to go through a few with the highest score. The problem is illustrated in Fig. 2.11, where the output of such procedure using one of the best performing models is shown. It contains 4 patches with the highest score, out of which 3 contain a chirping mode. It illustrates that even though the neural network encoding might be powerful, it is still basically a black box model and we need to be very careful in its evaluation. Because of this, we evaluate the model performance not only by AUC (1.4), and also by the precision@n score, which is the precision at the n-highest scoring samples, and which is useful because it can be tuned to a certain n given by the operating conditions of the tokamak. Here, we use $n = 50$, which is a realistic number of samples that an operator can go through for each tokamak experiment.

Tab. 2.1 compares the performance of models in the one class setup. The results are split by the diverence used to regularize the latent space. The difference between

divergence	AUC	precision@50
–	0.82 ± 0.03	0.86 ± 0.06
KLD	0.46 ± 0.05	0.50 ± 0.14
MMD	0.84 ± 0.03	0.90 ± 0.06
JSD	0.84 ± 0.05	0.83 ± 0.10
MMD + JSD	0.84 ± 0.01	0.87 ± 0.01

Table 2.1: Results of optimization of the one class model by the divergence used in latent space regularization. The top three values are highlighted with shading. No divergence is used in a plain autoencoder with the training objective (1.21).

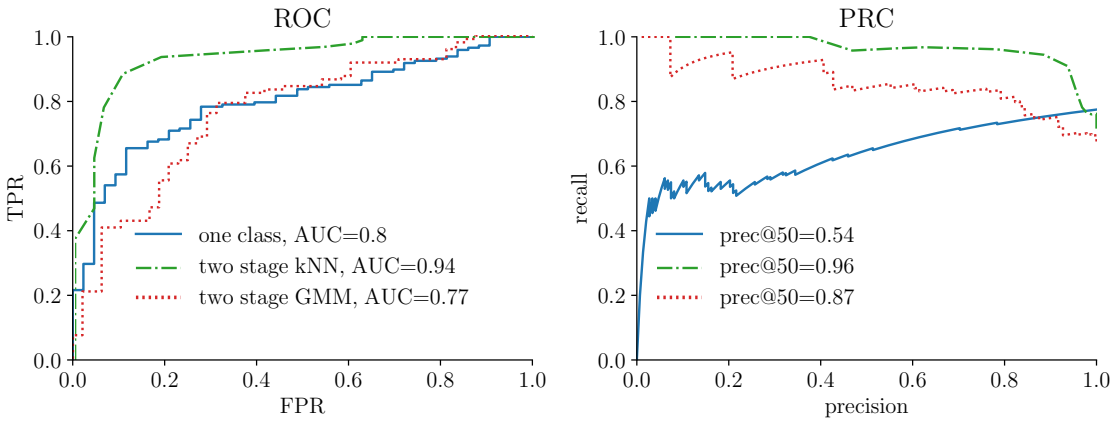


Figure 2.12: ROC and PR curves of selected models. For brevity, we include the best one class and best two-stage models in the same plot, although they are not directly comparable.

MMD, JSD and their combination in terms of AUC is neglibible, but MMD is slightly better than the rest in terms of precision@50. Surprisingly, vanilla VAE with KLD fails in this task altogether, which indicates that the distribution of the patches with Alfvén eigenmodes is difficult to model through a latent space with a unimodal prior.

The performance of the two-stage models is summarized in Tab. 2.2. What is immediately obvious is that the simple kNN model is superior to the GMM approach with any encoding. Also, MMD regularization produces the best results. We might speculate that this might be due to the improved ability to produce a well-separated encoding enforced by the used prior. Fig. 2.12 captures the ROC and PR curves for the best one class and two-stage models.

A question one might ask is whether the autoencoding is truly necessary. In the end, we are doing a projection from $d = 128 \times 128 = 16384$ dimensional picture space into at most $h = 64$ dimensional latent space which must naturally lead to a loss of information. As shown in Fig. 2.13, where $h = 8$, the autoencoder is able to identify the difficult nonlinear correlations and improve the performance of a subsequent second stage kNN model. The compression is clearly necessary for overcoming the curse of dimensionality which means that L_2 distance degenerates in large dimensions. An alternative approach to overcoming the issue of large input dimension might be to train a classification convolutional neural network, which does the compression by its

divergence	classifier	AUC	precision@50
–	kNN	0.80±0.07	0.88±0.10
KLD	kNN	0.80±0.08	0.85±0.11
MMD	kNN	0.91±0.06	0.94±0.05
JSD	kNN	0.83±0.07	0.87±0.10
MMD + JSD	kNN	0.86 ± 0.07	0.91±0.10
–	GMM	0.75±0.06	0.80±0.10
KLD	GMM	0.74±0.06	0.83±0.11
MMD	GMM	0.66±0.12	0.72±0.12
JSD	GMM	0.74±0.06	0.82±0.11
MMD + JSD	GMM	0.76±0.06	0.84±0.10

Table 2.2: Results of hyperparameter tuning of the two-stage model across 10 cross-validation splits.

nature. However, such a network would be highly susceptible to overfitting since it requires a lot of labeled data which is not available.

Influence of the train/test splitting methodology

At first, the splitting of testing and training labeled patches was done on the level of patches, without any regard for the spectrogram/experiment that the patch came from. It was assumed that the labeled chirping modes are homogeneous across the spectrograms. However, this turned out not to be true. Therefore, the train/test splits were done on the level of spectrograms which were then subsequently divided into patches. See Fig. 2.13 where on the left side, the AUC curves for different values of k of the kNN model are for the case when the data split was done on the level of patches. The blue line that is the result of kNN fit peaks at $k = 3$. On the other hand, there is no such peak on the right side of the figure, where splitting was done on the level of spectrograms. This indicates that the positively labeled patches in a single spectrogram are much more similar to each other than to those in different spectrograms, as only a relatively low number of neighbors is sufficient for optimal performance. Also, the variance of the right side plots is much higher, again indicating larger differences across spectrograms. If we continued with the splitting on the level of patches, we would have a biased and too optimistic estimate of performance before using the framework in a production environment.

Final remarks

To sum up the findings from this section: generative autoencoders are a viable tool for unsupervised and semi-supervised anomaly detection. The information contained in the latent spaces is useful for anomaly detection, provided we have at least some examples of labeled anomalies. And finally, the kNN classifier proved to work very well in this simple experiment. All of these findings are going to be useful in the building of the model that is presented in Chapter 4.

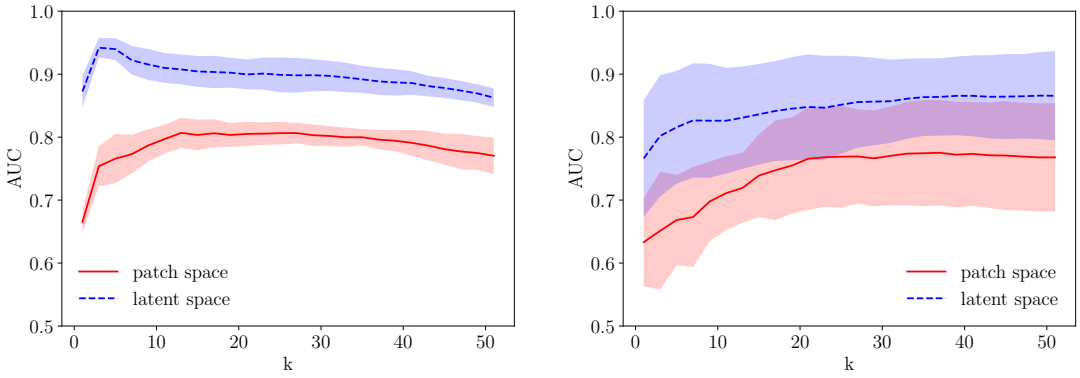


Figure 2.13: kNN fits for different values of k. The red line and band show the mean and one standard deviation bands of the resulting AUC values when kNN is fitted to the original vectorized images. The input space dimensionality is $h = 16384$. The blue dashed line and band are the same quantities for a $h = 8$ dimensional representation by a first stage model. On the left, the training and testing splits were done on the level of individual patches, leading to a better performance and less variance. On the right, the split was done on the level of the original spectrograms, which is a more realistic scenario. The standard deviation and mean were computed from 10 random splits.

3
Empirical comparison of anomaly detectors

Practical use of anomaly detectors (as well as other machine learning models) consists of finding the right choice of algorithm and its hyperparameters that is the most suitable for the problem at hand. This usually requires a time consuming effort of training many models and evaluating them fairly on some test data. It is not clear however, if there is a method or a class of methods that is generally more suitable for solving certain anomaly detection problems than any other. The objective of this chapter is twofold: to provide an empirical comparison of anomaly detection methods of various paradigms with a focus on deep generative models, and to identifify the sources of variability in data that have the largest influence on the suitability of a method. The methods are compared on popular tabular and image datasets and under varying number of anomalies available for hyperparameter optimization. This serves in establishing a direction for the novel anomaly detector described in Chapter 4.

As far as anomaly detectors based on deep generative models are concerned, the number of modifications and extensions of VAE or GANs is sharply increasing (as is documented in Chapter 2), each claiming superiority over the prior art. Although almost every newly published method provides evidence of outperforming its predecessors, sometimes there are contradictory results when same methods are included in different comparisons. This raises a suspicion that some of the methods are overspecialized or poorly tested. This chapter, inspired by the paper "Do we need hundreds of classifiers to solve real-world classification problems?" [207], strives to compare anomaly detectors under fair conditions to observe how the field has evolved in the last twenty years — the oldest compared detector (kNN) was published in 2000. Specifically, it investigates if methods based on **deep** generative models offer a benefit over methods based on alternative paradigms, either the **shallow** methods that were introduced in Chapter **??** or deep architectures without the capability of generating samples.

There is a number of works that try to achieve a simillar goal, but we have found some deficiencies (with respect to the previously stated goals) in most of them. Earlier surveys [27, 31, 28, 24] do not compare to deep generative methods because they were not developed or sufficiently popular at that time. Contrary to that, the study in [208] contains a detailed description of deep models but provides experiments only with the basic VAE and only on specialized video datasets. Ref. [60] introduces a taxonomy of deep anomaly detection models but does not compare them experimentally. Other recent surveys [56, 57, 58, 59, 209] either ignore deep generative models altogether or describe them only theoretically, without making any experimental comparison.

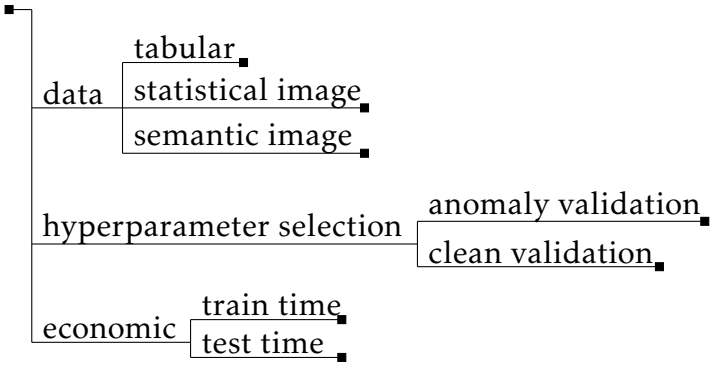


Figure 3.1: Various aspects of anomaly detection comparison forming the context of an experiment.

The most relevant prior art is [32], which tries to theoretically link deep and shallow techniques. But again, an extensive experimental comparison of different generative models is missing. One would also expect papers introducing new methods to contain such a comparison. Some of them do [24], but generally, we have found comparisons limited (e.g. using a small number of datasets or methods) or flawed, which is elaborated below.

How do we avoid the aforementioned deficiencies? First, eight shallow methods from Chapter **??** serve as a baseline, with which we compare the state-of-the-art deep methods from Chapter **??**. Second, the comparison uses a large number of tabular (40) and image (6) datasets popular in the evaluation of deep models, which are described in Appendix B. Third, all methods have been given the same conditions, which primarily means the budget for optimization of hyperparameters, as [210] has shown this to have a significant impact.

This chapter is organized as follows. In Sec. 3.1, the anomaly detection contexts that have the greatest influence on the outcome of our experiments are defined. Sec. 3.2 details the datasets, different approaches to the selection of hyperparameters, and other design decisions in the experimental setup. Sec. 3.3 discusses the experimental results and lessons we have learned. We summarize the chapter with a recommendation to practitioners and our suggestions for future work. The original paper [211], which this chapter summarizes, contains more, mainly technical details and results.

3.1 Anomaly Detection Contexts

While many practitioners are eager to see which method is the best for their application, the specifics of the application may differ. In this chapter, a large number of experiments is conducted in order to identify the main sources of variability influencing the performance of anomaly detection methods. The number of combinations of these aspects is huge. Therefore, we have chosen the key axes of variability: datasets, hyperparameter selection strategy, and economic point of view. From these axes, we select a few discrete points, on which we will provide a comparison. The particular combination of the selected aspects will be called **context**, see Fig. 3.1 for illustration.

The first axis is the target data domain. Our experiments use two types of datasets: **tabular** and **image**. This is the most obvious split, and indeed most authors of prior art test their methods on either choice of data. Another possible way to look at data

is whether they contain **statistical** or **semantic** anomalies, see Sec. 1.1. Statistical anomalies should be located in areas of a low likelihood of the normal class, while semantic [48] anomalies cannot be differentiated from normal data statistically. This is because they appear in datasets with multiple sources of variations, where only some of them are considered anomalous. Such types of anomalies are most common in image datasets. The suitability of the compared methods for the dataset context axis is studied in Sec. 3.3.1.

The second axis of variability is the hyperparameter selection strategy. It should be a gold standard that the experiments are repeated on different splits of data to training, validation, and testing subsets, especially if the datasets are small. However, in most of the reviewed recent papers [147, 172, 23, 176, 129], this procedure was not mentioned with the exception of [38]. Therefore, our comparison fills this gap. Also, it is important to define the nature of information available for the selection of the hyperparameters: it is indeed a very different task if there is some (often small) number of known anomalies in the validation dataset that can be used to choose hyperparameters by cross-validation or if the validation dataset is clean (i.e. it contains no anomalies). In our experience, the former case is more common. Our observations are summarised in Sec. 3.3.2.

The third axis is the economic aspect of a problem. There might be serious computational restrictions present in solving real-life problems. One might then not opt for a method that promises state-of-the-art performance, but for another that reaches slightly worse performance but can be trained economically, and its performance is robust with regards to hyperparameter optimization. More details on this can be found in Sec. 3.3.3.

Finally, Sec. 3.3.4 contains other influences that have been originally considered to be important but eventually did not prove to make a significant difference in comparison of multiple methods. These include the use of performance measures other than traditional AUC, the use of Bayesian optimization, and others.

3.2 Experimental setup

In order to achieve a fair and robust comparison, a strict procedure for testing each model was followed, which is shortly described in this section. For more details, see the original publication [211] for details.

3.2.1 Data

Two criteria guided the choice of datasets (mainly the tabular ones): first, they ought to be publicly available, and second, they should appear in surveys or articles presenting new methods. In total, we have collected 40 tabular datasets, the majority of which came from the UCI repository [212]. The complete listing of number of samples and dimensions in each dataset is recorded in Appendix B and Tab. B.1. Except for the ANNThyroid, Arrhytmia, HAR, HTRU2, KDD Cup 99 (small), Spambase, Mammography, and Seismic, where the anomaly class has a clear meaning (security incident or disease), we have followed the technique of [213] for creating artificial datasets for anomaly detection tasks from classification datasets. More precisely, we have used only "easy" and "medium" anomalies, as "hard" and "very hard" are not truly anomalous in the sense of being statistically distinct from the normal class.

The number of image datasets used for the evaluation of deep models is limited, as there are very few publicly available image datasets designed purely for anomaly detection - MNIST-C [214] and MVTec-AD [215]. Only three subsets *wood*, *grid*, *transistor* of the MVTec-AD dataset were used due to our computational constraints. These were chosen since they represent problems of various degrees of difficulty. Furthermore, we have extended these with artificially created anomaly datasets based on common image datasets MNIST [216], FashionMNIST [217], CIFAR10 [218], and SVHN2 [219]. These are also used for anomaly detection tasks in the prior art [129, 168, 38]. Again, basic statistics on image datasets are shown in the appendix Tab. B.2. In the setting of this whole work, a single class (of digits/objects) is considered normal and the rest anomalous. Therefore, one classification image dataset is transformed into ten different anomaly detection subdatasets. Since the MNIST, FashionMNIST, MVTec-AD, and MNISTC datasets have a rich and consistent number of samples in the normal class and clear anomalies, we consider them to be statistical anomalies. On the other hand, images in the majority of classes in CIFAR10 and SVHN2 have a strong background and are thus considered to contain semantic anomalies. This prior division is also supported by a different behavior of different methods as reported in the following text.

The normal data samples in each dataset were randomly split in 60%/20%/20% ratios to train/validation/test subsets, respectively. Anomalous data samples were split such that 50% were in the validation part and 50% in the testing part, which means the training subset has not contained anomalous samples. [1] The proportion of anomalies that were used in the validation phase varied from zero to the selected 50%. This was done five times to produce different folds in all tabular and small (MvTec-AD and MNIST-C) image datasets, but only once for other image datasets, as the folds would be very homogenous, see the original paper.

3.2.2 Models and their hyperparameters

Since the number of tested models is quite large, Tab. 3.1 offers their overview together with acronyms used in graphs and tables and also an orientational division of the models into separate categories. Most of the deep models were coded from scratch, since the original implementations were either missing or non-functional. Properly exploring the space of hyperparameters of all models is paramount to achieving fair and comparable experimental comparison, yet this is often superficially treated. Researchers often use default or recommended values ignoring that they are sub-optimal on datasets they use in their comparison. The conflicting results of the MOGAAL method in the original publication [147] and in [172] demonstrate our argument. Another prototypical example is OC-SVM, which is typically used with Gaussian kernel and with ν set to some default value, e.g. 0.05 [24], but can achieve better results with different kernels. The choice of hyperparameters in anomaly detection is everything but easy. But this means that the experimental settings should be set up such that all methods have been optimized equally. We conjecture that recommended and default values of hyperparameters are strongly correlated with the choice of evaluation datasets in the publications that recommend them.

In order to explore the hyperparameter space of each method properly, we have employed a random search over a predefined grid for each method. This allows the

[1] A training set without any anomalies is in practice very optimistic, but this decision removes another degree of freedom from the evaluation for the sake of clarity of results.

class	model	acronym	class	model	acronym
flows	MAF	maf	two-stage	DAGMM	dgmm
	RealNVP	rnvp		DeepSVDD	dsvd
	SPTN	sptn		REPEN	rpn
autoencoders	AAE	aae		VAE-kNN	vaek
	adVAE	avae		VAE-OC-SVM	vaeo
	GANomaly	gano	shallow	ABOD	abod
	skipGANomaly	skip		HBOS	hbos
	VAE	vae		IsolationForest	if
	WAE	wae		kNN	knn
gans	fAnoGAN	fano		LODA	loda
	fmGAN	fmgn		LOF	lof
	GAN	gan		OC-SVM	osvm
	MOGAAL	mgal		PidForest	pidf

Table 3.1: Overview of the main classes of compared methods and the acronyms used in the text.

construction of sections through the space for sensitivity studies. Moreover, it is frequently more efficient than grid search [220] and more flexible. For each model, dataset, and repetition, we have sampled 100 configurations from corresponding sets and trained the models with them. A fixed time budget was also given for training of each model, which automatically penalizes complicated models, which might theoretically achieve great performance, but are too computationally demanding.

A thorough exploration of the hyperparameter selection context also requires changing the criteria of model selection. When anomalies are available for validation, we select hyperparameters maximizing the AUC on the validation set. For experiments with no available anomalies, we have decided on the following hyperparameter selection mechanism. For shallow methods, we have used default hyperparameter values from literature - either authors of the method recommended them, they were used in a survey, or are default in a given implementation. Their overview is found in [211]. For deep methods, this is unfortunately impossible since their hyperparameter space is much larger, and the values are usually tuned to a specific dataset. Therefore, to have a universal solution, we have selected the already trained and evaluated models based on the lowest average anomaly score on the clean validation data, i.e. valdiation data without anomalies. This approach is theoretically justified for models with proper likelihood.

3.3 Experimental results

Unless specified otherwise, the performance results are estimates of the AUC on the testing set and averaged over all folds of the random cross-validation repetitions. When ranks are reported, they are calculated by ordering methods on each dataset and cal-

culating the average across them (as recommended in [221]). Hyperparameters are selected using the best average performance over the folds of the validation dataset.

3.3.1 Dataset context

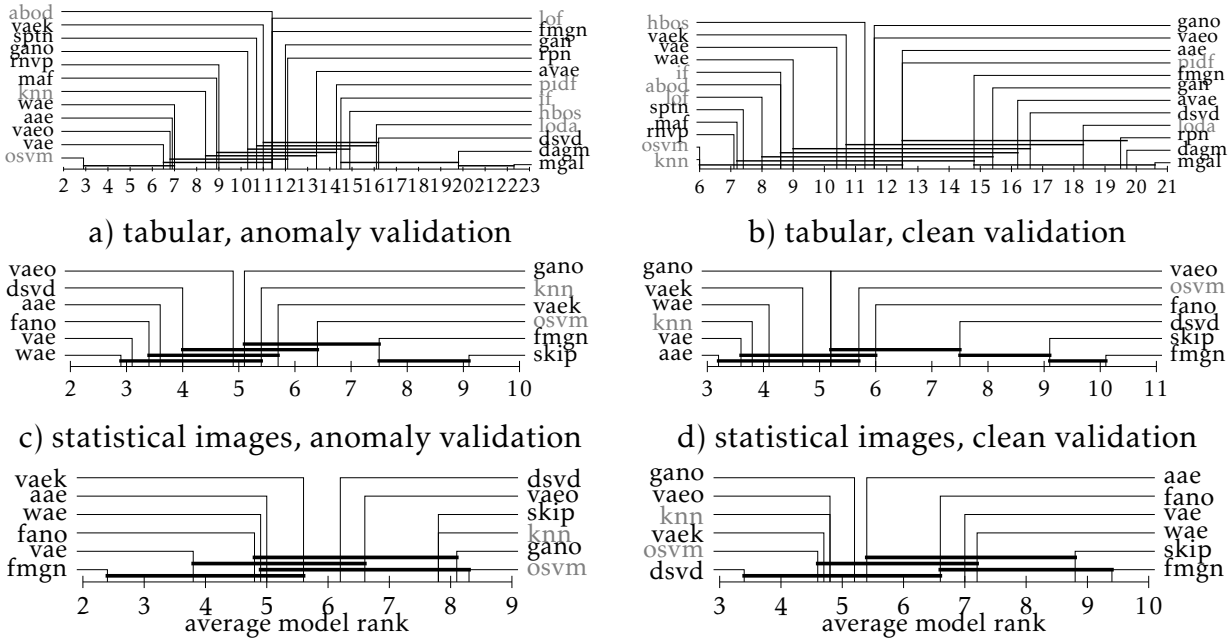


Figure 3.2: Critical difference diagram of models ranked via the test AUC. Models whose performance is statistically indistinguishable have a difference of ranks under the critical value of the Nemenyi test $CD_{0.1}$ and are joined by a horizontal band. Results are presented for different types of datasets: tabular (Top row), image datasets with statistical anomalies (Middle row), and image datasets with semantic anomalies (Bottom row); and two different hyperparameter selection cases: using anomalies in validation (left) and using clean validation (right).

The results of the experimental comparison on all dataset types are presented in the form of critical difference diagrams (CDD) as recommended by Demšar [221], see Fig. 3.2. These diagrams show the average rank of detectors across the datasets together with a confidence band that indicates that a statistical test cannot reject the hypothesis that two detectors perform the same. What follows is a commentary on the influence of the datatype with respect to two types of hyperparameter selection strategies differing in the number of anomalies in the validation set as defined in Section 3.1: i) anomaly validation context, and ii) clean validation context.

Tabular data: OC-SVM performs the best and it is *statistically better* than almost all detectors except autoencoder-based generative models and VAE combined with OC-SVM in the case of anomaly validation context. The first 11 places (roughly one half) belong to models that can be divided into three groups: (i) OC-SVM and its variants, which estimate a density level of a distribution; (ii) flow models and kNN, which estimate the pdf (un-normalized in case of kNN); (iii) and variants of autoencoders, where the reconstruction error is related to pdf as explained in Sec. 2.2. The same types of methods occupy the top positions in the clean validation context, Fig. 3.2b), however

in a different order. The best is the kNN (due to the simplicity of its hyperparameters), and all other pdf-modeling methods (flows) have improved relative to the anomaly validation context. The autoencoder-based methods moved beyond shallow methods (LOF, ABOD, IF). We believe that models in the lower half of the scale in both validation contexts are not suitable for detecting statistical anomalies. We cannot explain the poor performance of MOGAAL, DAGMM, and adVAE, and we attribute it to different experimental environment. DeepSVDD was primarily implemented for image problems, where it performs relatively well.

Moreover, differences in mean ranks of many models in Fig. 3.2 are statistically insignificant at level $p = 0.1$, which is disappointing. Assuming the ranks remain the same, another 51 datasets would be needed to make the difference between OC-SVM and VAE statistically significant on tabular data with 50% anomalies. This indicates that the results are relatively noisy and can be easily changed for a different choice of datasets.

Statistical image data: WAE and VAE models have the best average rank when evaluated on statistical image data, although their lead is not statistically significant over most of the other models as is evident from Fig. 3.2c). The autoencoder-based methods (AAE,VAE,WAE) perform well also in the clean validation context, complemented by the kNN, Fig. 3.2d).

Semantic image data: A different story is told by Fig. 3.2e) where the ranking of methods on image datasets with semantic anomalies is dominated by fmGAN by a large margin in the anomaly validation context. However, it is also the worst method in the clean validation context. In an opposite manner, OC-SVM and kNN perform very poorly in the anomaly validation context, but they are among the best in the clean validation context. The best performing method in the clean validation context is DeepSVDD [38]. We conjecture that the performance of the fmGAN is related to the variability of its training. With a sufficient number of anomalies in the validation set, it is possible to find one trained model that fits the problem.

3.3.2 Hyperparameter selection context

The influence of the hyperparameter selection procedure on the results in the previous section is now studied in detail for few selected methods. We choose only those that scored among the best in the previous section. First, we analyze the sensitivity of these methods to the number of anomalies in the validation set. Second, we study hyperparameter selection for two individual methods, variational autoencoder family and OC-SVM.

Impact of the number of anomalies in the validation set

The process of hyperparameter selection described in Sec. 3.2.2 depends on the availability of examples of anomalies in the validation set (recall that it is assumed that the validation dataset does not contain unknown anomalous samples, i.e. is not contaminated). Fig. 3.3 displays the influence of the number of anomalous samples in the validation set on a finer grid between the two contexts reported before. Note that for the first point on the x-axis where the clean valdiation dataset was used, the mechanism of model selection was different from the rest of the graph. This is the reason for the significant difference between the clean context and the remaining points.

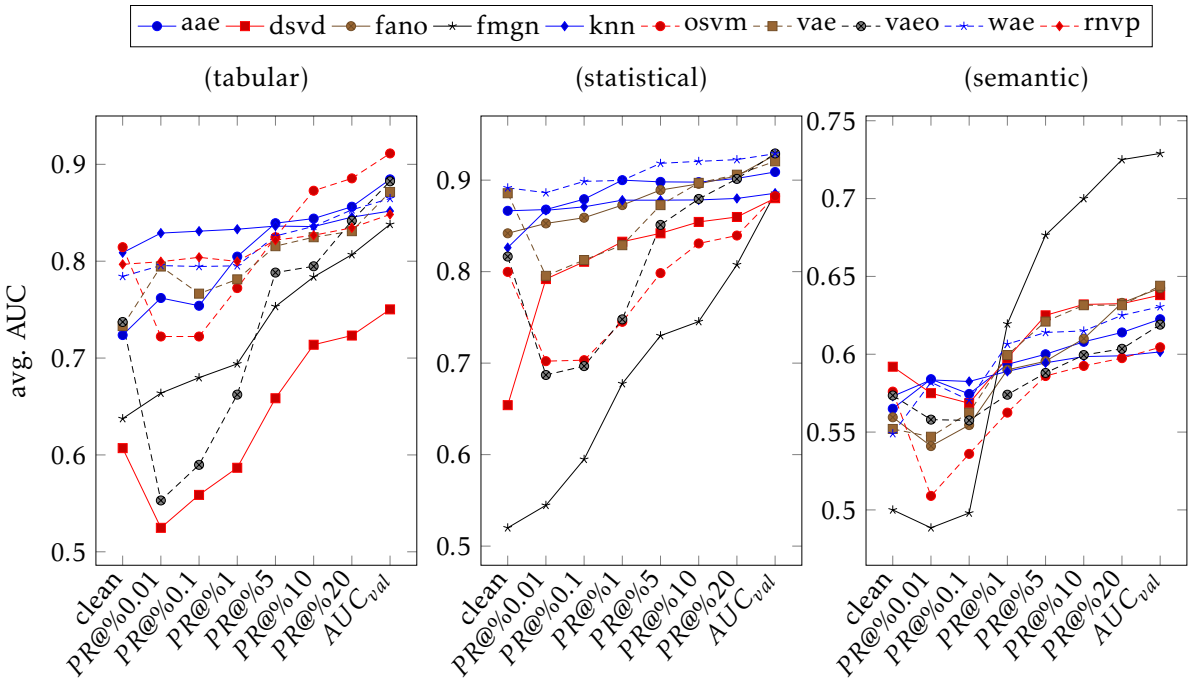


Figure 3.3: Sensitivity of methods to the number of anomalies available in the validation set for hyperparameter selection visualized in terms of the achieved AUC aggregated over all datasets in each category (columns). The clean validation context is the left-most point on the x-axis, and the anomaly validation context (50% of available anomalies) is the right-most point. The points in-between were obtained by selecting models with the highest precision on the reported portion (e.g. 5%) of validation samples with the highest anomaly scores.

First, we observe that the quality of the models selected using anomalies improves with an increasing number of anomalous samples, which is expected. However, for a low number of anomalies, many methods perform significantly worse than in the case of the clean validation context. This behavior is notable across dataset types, especially for OC-SVM, and to some extent VAE. We conjecture that the hyperparameter selection procedure of those methods has a tendency to overfit, and its hyperparameters are not robust. In contrast to this, the performance of kNN, WAE, and RealNVP degrades slowly with the declining number of anomalies, which suggests that they are quite robust in difficult operating conditions. We attribute it to the fact that these methods are more exact in their estimation of data likelihood than the rest.

Second, we notice that the experimental results on the semantic image datasets are generally poor, as the AUC of the best model (fmGAN) on CIFAR10 is 0.72 and similarly on SVHN2, where the best model achieved 0.74. On the other hand, anomaly detection methods perform well on statistical image datasets. This indicates, contrary to popular belief that the models fail to learn or identify the important semantic information, or they consider different semantic information anomalous, and they should be told which semantic aspect of an image should be considered as an anomaly, as for example blurred images might be anomalous as well. Again, more detailed results, e.g. non-aggregated AUC values for individual datasets, or illustrative samples of detected anomalies can be found in the original publication [211].

A practitioner might also desire a method robust with respect to a poor choice of

hyperparameters. In general, deep methods in our experiments have demonstrated higher variance, probably due to the large number of hyperparameters and stochasticity involved in their initialization and training via batched gradient optimization. In this respect, GAN-based models seem to be the least robust, which is in line with [222] stating that GANs are not directly optimized for anomaly detection. This hints at the potential cost of hyperparameter optimization — with higher performance variance, one is less likely to train a well-performing model in a given number of attempts. On the other hand, given enough labeled anomalies for hyperparameter selection, the fm-GAN model gained a noticeable edge on semantic image anomalies.

Sensitivity Analysis of the VAE family

Autoencoder-based methods form a whole family with multiple sources of variability, as identified in Sec. **??** to be: i) approximation of the likelihood in training (loss function), ii) the richness of latent prior, and iii) the anomaly score. We will analyze the sensitivity of the results to these choices on tabular data in the anomaly validation context. We focus on this family since most of the novel deep generative models for anomaly detection are based on the autoencoder architecture. Additional degrees of freedom include the parametrization of variance of $p_{\vec{\theta}}(\vec{x}|\vec{z})$, which could be either fixed (called VAE-constant), used in [180, 172, 177], scalar (called VAE-scalar), or full diagonal (called VAE-diagonal), used in [126, 127, 146]. In the experiments, all three variations were tested on tabular data; however on image data, the full diagonal was skipped due to computational constraints (and in line with the prior art, where only fixed variance is used).

The overall comparison in Fig. 3.2 revealed that WAE and vanilla VAE variants perform best. The other degrees of freedom, namely richness of prior, used anomaly score, and parametrization of the variance, were treated as hyperparameters. Fig. 3.4 extends the study by showing the distribution of ranks over tabular datasets for different variants of VAE, including GANomaly and adVAE.

First, notice that the spread of the method's ranks over various datasets is significant, as even ranks of the best methods vary from 3 to 15. This means that the conclusions below need to be taken with a grain of salt, as the experimental results are extremely noisy.

The ELBO-based score, -el, together with the orthogonal decomposition of the likelihood [168], -jc, does not perform well. The sampled reconstruction error (an MC estimate of (2.22), -rs, almost always performs better than the usual reconstruction error, -rm, calculated according to (2.23). This demonstrates that the common approach of replacing the mean of the decoder with that of the encoder is inferior but computationally cheaper (see Tab. 3.2 with prediction times). The discriminator score (2.6), -di, of AAE (an autoencoder combined with GAN) seems to be also on par with the MC estimate (2.22).

From the same figure, we also conclude that the models modelling full diagonal in $p_{\vec{\theta}}(\vec{x}|\vec{z})$, -d-, seem to be better than the scalar, -s-, or constant, -c-, variants. This result is important, as many comparisons in the prior art use the VAE-constant, despite the version with full diagonal being discussed in the original publication [223].

The rich prior distribution on the latent space proposed in [167], VAMP, -v-, does not seem to give an advantage in the anomaly detection except in the AAE. Similarly, recent variants adVAE and GANomaly do not seem to work well on the tabular data,

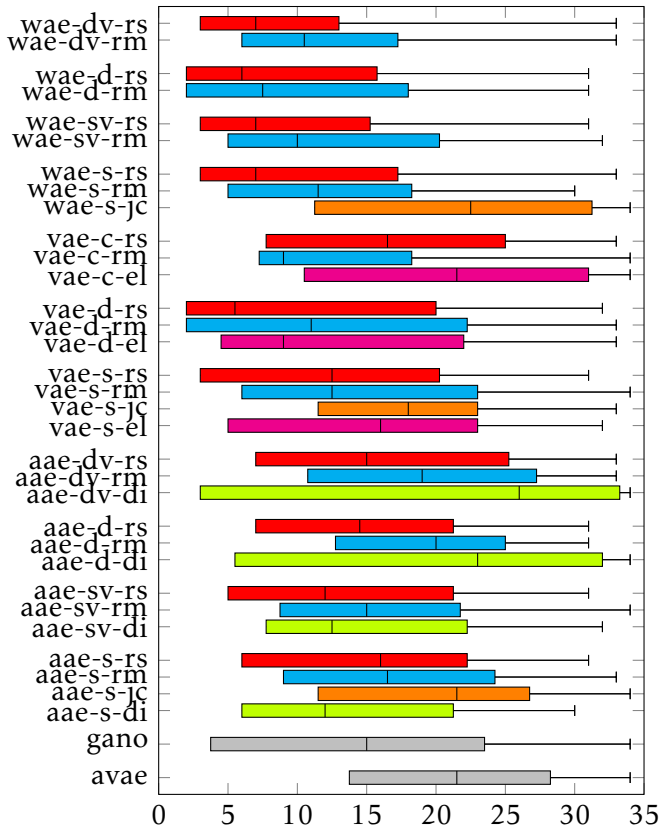


Figure 3.4: Sensitivity study of various variants of autoencoder-based methods displayed in the form of boxplots of their ranks in the AUC metric achieved on the tabular datasets. The first three letters of the method's name denote the training loss. Models with the -d- middle part estimate full diagonal of the decoder variance, -s- estimate only a scalar, and -c- use a fixed scalar variance as a hyperparameter. All variants are using the standard Gaussian latent model. Models using the VampPrior are denoted by extending the decoder variance symbol by the letter v-, i.e. -dv-, -sv-, -cv-. The last part of the name denotes score, -rs stands for the sampled rec. probability (2.22) with $L = 100$, -rm for (2.23), -el for the ELBO (2.15) composed of -rs and KLD, -jc for (4.5), -di for (2.6).

but they were not evaluated on them in the original publications.

Sensitivity study of OC-SVM

The domination of OC-SVM on tabular data in anomaly validation context contrasts to many prior experimental comparisons [28, 224, 222, 84, 225, 172]. The search for the culprit found it to be the hyperparameter selection. This study has varied the ν parameter, kernels, and their parameters, which is much more than the most prior art does, which is fixing the kernel to RBF and tests few values of its width γ and ν. Inclusion of other kernels into the search for hyperparameters seems to be the major source of improvement in this case. Replacing the OC-SVM with one restricted to use only the RBF kernel and $\nu = 0.5$ yields an increase in average rank from 2.9 to 8.1 with an average decrease in performance by 0.06, measured in AUC in the anomaly validation context. This version of OC-SVM is then easily surpassed by variational autoencoders and kNN, as demonstrated in Supplementary, Tab. **??**. The importance

	vae-s-rs	vae-d-rs	vae-s-rm	vae-d-rm	vae-d-jc
$\bar{t}_{pred}$ [s]	12.10	18.51	0.11	0.15	57.31

Table 3.2: Average prediction times on the tabular datasets for different combinations of VAE scores and decoder variance estimations. The -d- part stands for model with an estimate of the full diagonal of the decoder variance, -s- is a scalar estimate. Sampled reconstruction error (2.22) is denoted as -rs, -rm is the anomaly score (2.23) and -jc is (4.5).

of the choice of the kernel is furthermore illustrated by the fact that the sigmoid kernel was the optimal choice for 23 datasets, while the RBF kernel was optimal only for 13. Ref. [28] mentions that setting $\nu = 0.5$ provides universally good results, which may be the reason why many authors do not tune it. In theory, it should be set to much lower values ($\nu = 0.05$) corresponding to the presumed low ratio of anomalies in data, but with Bayesian optimization, we found that the best estimate of ν was in some cases even higher, such as ~ 0.75 on the statlog-vehicle dataset.

3.3.3 Economic context

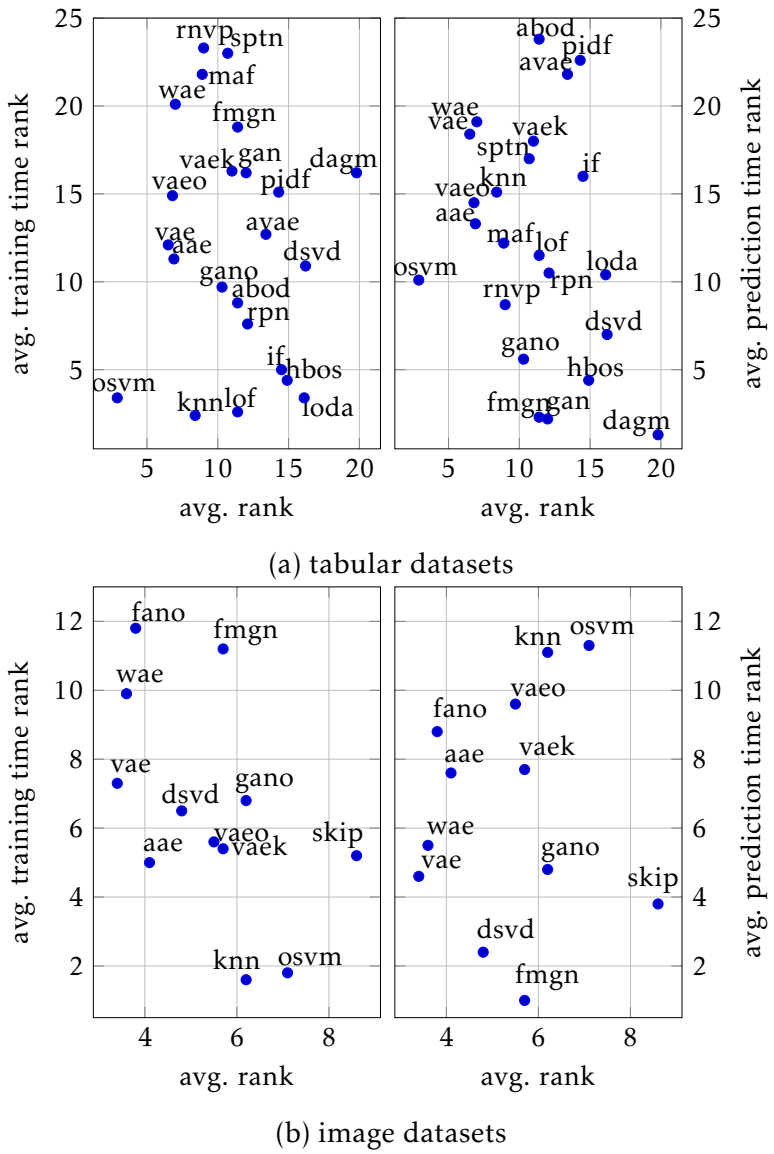
Practitioners ask for fast and accurate algorithms, but these two features rarely go hand in hand, and a decision on a trade-off has to be made. Interesting methods lie on the Pareto frontier, as in the absence of an external factor, rationally behaving practitioner does not have a motivation to choose a different model.

Fig. 3.5a-left shows the trade-off between accuracy and training time for tabular data, where the absolute numbers were replaced by average ranks for robustness. The Pareto frontier contains two methods, which are OC-SVM and kNN. The position of OC-SVM is rather surprising, as its training time is known to scale poorly (quadratically) with respect to the number of samples, but it is caused by most of the tabular datasets being small. Different results may arise for a dataset with many data records. Fig. 3.5a-right shows a similar trade-off between accuracy and testing (inference) time. OC-SVM is still on the Pareto frontier, but it is expensive, as the complexity grows linearly in the number of samples. fmGAN, GAN and DAGMM methods are there as well – these methods have fast inference but lower accuracy.

We provide results averaged over the studied contexts on image data. Due to the variability of the results in each context, the x-axis will vary. In the averaged ranks, VAE is on the Pareto front in both fit and prediction times, see Fig. 3.5b. Its prediction complexity is given mainly by choice of the number of samples taken in the computation of the sampled reconstruction score (2.22). The kNN detector has negligible training time, given only by the construction of the tree structure representing data, but seems to be mostly unusable on image data due to slow prediction times on datasets that are large in dimension and number of samples. The fmGAN finds itself in a completely reversed scenario.

3.3.4 Other influences

In this section, we report the results of three sources of variability of performance of AD methods that were found to have minimal impact.



(a) tabular datasets

(b) image datasets

Figure 3.5: Scatter-plots of the average rank in the AUC metric on the tabular (a) and image (b) data versus average rank of the computational complexity of the displayed methods measured via training time (left) and prediction time (right). MO-GAAL has been omitted from the tabular figures, as its performance positioned it too far to the right with the training time rank of 19.4 and the prediction time rank of 10.0.

Ensembles/Hyperparameter averaging The benefits of ensembles in prior art seem to be mixed. While [226] claims that a combination of VAE or GAN ensembles using WAIC might be useful, Ref. [227] claims a negligible effect. In our experiments, we have used ensembles as a way to reduce uncertainty in hyperparameters [228], meaning that unlike in [226], models in ensembles were of a single type differing only in architecture. The effect of such an ensemble on average AUC was overall zero, sometimes even negative, see Supplementary **??**, Tab. **??**. Exceptions are methods based on GANs featuring improvements by 0.02 in average AUC. These findings are on par with those in [227].

Bayesian optimization Bayesian optimization was introduced as an alternative to a random search of hyperparameters. It has the potential to find better hyperparameters with a low number of trials. Comparison of the random search and Bayesian optimization under the same conditions is reported in the Supplementary, Tab. **??**. We can conclude that Bayesian optimization was able to find hyperparameters with better performance for the vast majority of the methods. However, this improvement is often quite small and insufficient to change the ranks of the methods. Notable exceptions are the GANomaly and GAN methods which improved by two ranks.

Performance metric AUC/TPR The hypothesis that the methods may perform differently when chosen for different optimization criteria than the usual AUC has not been proved. The results are summarized in the Supplementary, Tab. **??**. While small changes in performance have been observed, the ranks of the methods remain the same in both criteria, AUC and TPR@5%.

3.4 Conclusion

The presented extensive comparison of anomaly detection methods based on deep generative methods, namely variants of flows, variational autoencoders, and generative adversarial networks with methods based on alternative paradigms (Support vector machines, random forests, histogram, and distance-based methods) revealed that the performance of anomaly detection methods strongly depends on experimental conditions. We have identified the main sources of variation to be the choice of data and hyperparameter tuning. We presented detailed results under a combination of experimental conditions, called contexts, specifically the data context, hyperparameter context, and economic context.

In the dataset context, a clear distinction in performance was found between tabular data, image data containing statistical and semantic anomalies. The main distinction in hyperparameter selection was how many anomalies were present in the validation set. Different order of performance of the tested method was observed for a different combination of dataset type and hyperparameter selection context, explaining various outcomes of comparison in the prior art. We strongly recommend that authors provide more details on the context of their experimental studies in the future.

The comparison is not only aimed at researchers but also at practitioners desiring accurate methods with low computational complexity. Therefore, we visualize these trade-offs between performance and computational cost. Also, we present a list of some of the most important practical observations and recommendations.

- Methods with more exact likelihood modeling, such as kNN, flow, and autoencoder-

based models, perform better in scenarios with a limited number of anomalies available for hyperparameter tuning.

- Majority of the methods fail to detect semantic anomalies. The exception is the fmGAN, but only if given enough computational resources and many anomalous samples for cross-validation.
- OC-SVM, when properly tuned, can defeat most of the state-of-the-art on tabular data, although it suffers from overfitting when hyperparameters are selected using too few anomalies in the validation set.
- The method of the first choice appears to be the VAE/WAE due to its relatively cheap, precise, and consistent performance in most of the experiments. However, it possesses so many degrees of freedom that it forms a full family of methods. It was found that the best performance is obtained when estimating the full variance of samples on the output of the decoder and evaluating anomalies using sampled reconstruction score.

Many aspects have not been covered and remain a topic of future work, such as the identification of the relevant kind of anomalies in the semantic datasets or the design of ensembles of methods of various types. Although we have shown that the number of anomalies available for validation is an important context, there was no further budget for a comparison of anomaly detection methods with active learning. We have also completely left out the comparison of models on temporal data, such as time series or video, which is a very challenging task. Finally, an interesting area to study further is the influence of the presence of unlabeled anomalies in the training data.

We identified that the main sources of variability are the experimental conditions: i) the type of dataset (tabular or image) and the nature of anomalies (statistical or semantic), and ii) strategy of selection of hyperparameters, especially the number of available anomalies in the validation set. Methods perform differently in different contexts, i.e. under a different combination of experimental conditions together with computational time. This explains the variability of the previous results and highlights the importance of careful specification of the context in the publication of a new method. All our code and results are available for download.

4
Anomaly detection in multi-factor data

So far, we have been gathering information about the state-of-the-art methods, available datasets and potential unsolved problems in anomaly detection. The ultimate goal of this endeavour was to find an intriguing anomaly detection problem and propose a novel method for its solution. This will be done in this chapter. Here is a list of the findings that have been gathered in the previous text that guided us towards our goal.

1. In Section 2.4, the problem of identyfying anomalies in high resolution image data was solved and the two-stage model approach proved to be the most succesful. The best models were a combination of probabilistic autoencoder, that produced informative latent encoding, and a classical anomaly detector, that operated on the encoded data of highly reduced dimensionality. The dimensionality reduction alleviated computational burden from the second stage model which could be more deeply fine-tuned. Eventually, a simple kNN detector was the best choice of a second stage model.

2. In Chapter 3, a thorough comparison of existing anomaly detectors was done. We have seen what are the most important contexts of an anomaly detection problem (data type, number of labeled anomalies, and budget available for hyperparameter tuning) that one should consider when choosing a method for solving a practical problem. It was shown that deep generative models gain an edge on other methods when solving anomaly detection problems on image data, and especially on problems where semantic anomalies occur.

3. In the same chapter, it was shown that although probabilistic autoencoders are generally well-suited for image anomaly detection problems, adversarial training might be beneficial for detection of semantic anomalies with enough budget for hyperparameter tuning.

Since detection of semantic anomalies on images is a relatively novel field, with some publications TODO fill the gaps from the recent years, we have decided to cover this problem in this chapter using the findings described in the list above. We will describe the construction of a deep generative adversarial autoencoder that will be combined with a kNN detector operating on its latent space.

We propose a novel form of anomaly detection that considers several potential sources of anomalies. The approach is demonstrated on detection of semantic anomalies in image data, for which we have proposed the SGVAEGAN model. The model decomposes images into three independent components—the shape of an object and its

foreground and background textures—and provides anomaly scores for each of those components separately. The source of anomality in the image can be identified by comparing the component scores. This novel concept allows an explanation of the anomaly type in a completely unsupervised case. The overall anomaly score of an image is obtained by a weighted combination of the individual component scores. When a few labeled samples of anomalies are available, the model is able to learn the weights in the scores as a hyper-parameter. We have shown that this overall score is useful in the detection of anomalies in the semantic sense, where their anomalousness depends on the context in which they are evaluated. On classical anomaly detection benchmarks, the model was proven to outperform all baseline models. This was shown in a rigorous experimental study that covered the behavior of the model under a varying range of conditions.

sectionIntroduction this was already covered in 1 Anomaly detection [30, 229] detects samples, which are in some sense different from those considered normal. The probabilistic definition assumes some probability distribution function P, which is most of the time known only through a set of normal samples, and calls a sample anomalous if it lies in a region where P has very low density. Anoumalous samples therefore raise a suspicion that they were generated by a different distribution than P. Anomaly detection is important for many industries, where it is typically difficult to obtain representative training set to use supervised learning, for example network security [4], medical imagining [142], video surveillance [12], and industrial process monitoring [13, 14, 15]. In all these industries, users have some examples of anomalies, but there is a danger of a new type of anomaly to emerge, which contradicts the iid assumption of supervised learning.

Anomaly detection has been already extensively studied under many different names: outlier detection [36, 37], novelty detection [27], one-class classification [38] or out-of-distribution detection [39]. All these approaches have in common is the assumption that only samples from the *normal* class are available at training time. They model only the normal data and produce an *anomaly score*, which is a measure of how much a sample deviates from the normality model. A recent large-scale study [211] has shown that classical methods [36, 26, 20, 24] developed for tabular datasets of small dimension and number of samples work very well on them. But they do not work well on large high-dimensional datasets (typically containing image data), where modern methods built on top of deep neural networks [176, 129, 99, 230] work slightly better (we call these detectors deep anomaly detectors).

The very same study also showed an inability of most deep anomaly detectors to detect *semantic* anomalies. For example, an image (used in [211]) can be anomalous because it is blurred, or because it has a different object in the foreground (e.g. plane in the sky instead of a bird in the sky), or different background (e.g. the plane is on a runway, whereas it is usually on the sky). From the point of view of the above definition, they are all anomalies, but the user might be interested in anomalies of a certain type, or they might want the anomalies to be automatically sorted into different types. The detection of semantic anomalies has been previously studied under the name *subspace* anomaly detection [231, 232, 233]. The idea, as the name suggests, is that the anomaly is not visible in the full input space, where it might be shadowed by noise, but only in the subspace. The subspace is most of the time "axis parallel" or defined in a linear transformation of the input space.

It seems unlikely in practice that the subspace in which anomalies would be de-

tectable would be aligned with the input space, or with its linear transformation. Contrary, independent components emerge after non-linear projections as shown in [234, 235, 236, 237, 238, 239, 240], which motivates this work. We suggest to disentangle the input into independent factors, each providing an anomaly score. Ideally, a semantic anomaly would be then better detectable by one (few) of these scores rather than only a single complete score.

This is the list of our contributions.

- We propose a new generative model for multi-factor image datasets based on contextual generative networks.
- We propose a multi-factor anomaly score: a weighted combination of anomaly scores corresponding to individual factors of the latent space.
- We demonstrate that the proposed model is capable of detection of the most likely anomaly factor in the unsupervised setting, and quickly improves with very few labeled samples,
- We perform a large-scale study of the method on standard image benchmarks for anomaly detection together with a sensitivity analysis of the hyperparameter tuning on the number of known anomalies available for validation. With an increasing number of known anomalies, the problem is approaching a two-class problem, therefore, we also compare the method with a supervised classifier.

In the next section, we theoretically justify disentangled models and the proposed composition of their scores. In Sec. 4.2, the model used in experiments is described together with details of its construction, training, and evaluation. Sect. 4.3 discusses the related work, and in the experimental Sec. 4.4, the proposed model is extensively compared to baselines on several image benchmarks.

4.1 Decomposing the anomaly score

Variational autoencoder (VAE) [34] fits a generative model with data $x \in \mathbb{R}^n$

$$p(x) = \int p_\theta(x|z)p(z)dz \qquad\qquad p_\theta(x|z) = \mathcal{N}(x; g_\theta(z), \sigma^2\mathbf{I}), \tag{4.1}$$

where the prior on the latent k-dimensional variable z is either fixed $p(z) = \mathcal{N}(0, \mathbf{I})$, or further parametrized $p_\theta(z)$ as in [241]. Distribution $p_\theta(x|z)$ is frequently called *decoder* with $g_\theta(z)$ being realized by a neural network parametrizing the mean of the normal distribution. To fit the model, VAE introduces an *encoder*, which is a conditional probability distribution $q_\phi(z|x)$ parametrized similarly to the decoder as $q_\phi(z|x) = \mathcal{N}(z; \mu_\phi(x), \mathrm{diag}(\sigma_\phi(x))$ with $mu_\phi(x), \sigma_\phi(x)$ being neural networks. During training, parameters (θ, ϕ) of both neural networks are optimized to maximize the evidence lower bound (ELBO) on given training samples.

Estimated parameters (θ, ϕ) uniquely define marginal likelihood in (4.1), but since the expression contains a complex integral, in practice it is crudely approximated by the reconstruction error as

$$p(x) \approx \int p_\theta(x|z)\delta(z - \mu_\phi(x))dz \;\propto\; \exp\Big(-|x - g_\theta(\mu\phi(x))|^2\Big), \tag{4.2}$$

where $\propto$ denotes equality up to a multiplicative constant. Notice that in this expression, the integration over z is replaced by an evaluation of the likelihood at the "most probable" point given by the encoder $q_\phi(z|x)$.

4.1.1 Orthogonal generative model

The above generative model (4.1) can be rewritten as $x = g_\theta(z) + e$ where e is (usually isotropic) Gaussian noise. This formulation emphasizes the need for marginalization since $g_\theta(z)$ is a random variable of dimension k, and noise e is a random variable of dimension n, which is the same as the data. Equation (4.1) therefore marginalizes away random variables corresponding to latent z. An alternative formulation [168, 169] assumes orthogonal decomposition of the noise as

$$x = g_\theta(z) + e \qquad \rightarrow \qquad x = g_\theta(z') + e^\perp, \tag{4.3}$$

where z' is a point in the latent space, and $e^\perp$ is the observation noise that lies in the normal space perpendicular to the manifold defined by the decoder $g_\theta(z)$. Denoting $x' = g_\theta(z')$, any point x can be decomposed into

$$x = x' + e^\perp,$$

where x' lies on a k-dimensional manifold, and $e^\perp$ in its normal space (of $n-k$ dimensions).

Due to the (local) orthogonality, x' and $e^\perp$ can be considered independent and therefore probability distribution $p(x)$ to be well approximated as $p'(x)p(e^\perp)$. Since the probability of $p'(x)$ is given by the change of coordinate formula from $p_z(z)$, and $p(e^\perp) \approx p(e)$, for the likelihood $p(x)$ now holds

$$p(x) \approx p'(x)p(e) = p_z(g_\theta^{-1}(x))\left|\frac{\partial g_\theta^{-1}(x)}{\partial x}\right|p(e). \tag{4.4}$$

A gaussian noise model might be assumed, e.g. $p(e) = \mathcal{N}(x - x', \sigma^2\mathbf{I})$. Note that the assignment $p(e^\perp) = p(e)$ is correct up to a normalizing constant if the z' point is correctly estimated (more on this in Sec. 4.2.1).

In practice, an anomaly score $s(x) = -\log p(x)$ is computed as

$$s(x) = -\log p(e) - \log p_z(z) - \log\left|\frac{\partial g_\theta^{-1}(x)}{\partial x}\right|, \tag{4.5}$$

which is essentially the reconstruction error with additional terms. We will use s to denote various unnormalized anomaly scores throughout this text.

4.1.2 Anomaly in the latent space

Let's now observe how the above generative model changes when we assume the distribution on latent z to be multi-modal conditioned by hidden label y

$$p(z|y) = p_n(z)^y p_a(z)^{1-y}, y \in [0,1], \tag{4.6}$$

where $y = 1$ for normal data, $y = 0$ for anomalies, and $p_n(z), p_a(z)$ are the latent distributions of normal and anomalous data, respectively. Then from (4.5) we have

$$s(x|y) = -\log p(e) - y \log p_n(z) - \log \left| \frac{\partial g_\theta^{-1}(x)}{\partial x} \right| - (1-y) \log p_a(z),$$

Since y is usually unknown, we will integrate it away by expectation over all possible y,

$$\begin{aligned} \mathbb{E}_{p(y)}[s(x|y)] &= -\log p(e) - \log \left| \frac{\partial g_\theta^{-1}(x)}{\partial x} \right| \\ &\quad - \int_{\mathcal{Y}} y \log p_n(z) p(y) dy - \int_{\mathcal{Y}} (1-y) \log p_a(z) p(y) dy \\ &\propto -\log p(e) - \log \left| \frac{\partial g_\theta^{-1}(x)}{\partial x} \right| - \alpha \log p_n(z) = s(x). \end{aligned} \tag{4.7}$$

where we assume $\mathbb{E}_{p(y)}[y] = \alpha$, and $p_a(z) \propto 1$ (because the probability of an anomaly is assumed to be the same everywhere).

Now consider a *disentagled* model, where data are generated from multiple independent latent spaces, $z_i = z_1, \ldots, z_l$,

$$x = f(z_1, \ldots, z_l) + e, \tag{4.8}$$

where each of the latents has distribution $p(z_i) = \mathcal{N}(0, \mathbf{I})$. Each of the latent spaces can be a potential source of various types of anomalies, with hidden labels $y = (y_1, \ldots, y_l)$. Therefore, instead of (4.6), we have

$$p(z|y) = \prod_{i=1}^{l} p_{n_i}(z_i)^{y_i} p_{a_i}(z_i)^{1-y_i}, y_i \in [0, 1]. \tag{4.9}$$

Repeating derivation of (4.7) for multiple latent variables, the anomaly score becomes

$$s(x) = -\log p(e) - \log \left| \frac{\partial g_\theta^{-1}(x)}{\partial x} \right| - \sum_{i=1}^{l} \alpha_i \log p_{n_i}(z_i), \tag{4.10}$$

where α_i is denoting probability that the latent variable of the ith factor is generated from the normal class. In this work, we assume that α_i is not known during training but has to be estimated in the validation stage from examples of anomalous samples.[1]

The interpretation of the values of α_i will be important later, in Sec. 4.2, where we construct the latent spaces in order to give them a specific meaning, and in Sec. 4.2.2, where we describe their fitting. Fitting values of all α_i on a set of labeled data is equal to estimating the mean of $p(y_i)$ for the given data. This can be interpreted as estimating how likely it is for an anomaly to appear in the i-th latent in the context of the current dataset.

In contrast, for a single data sample, we are more interested in the actual values of y_i, which are binary and which determine whether the sample is anomalous in the context of the latent space i. Their estimation is equal to the estimation of the distributions $p(y_i|x)$ and will be discussed in Sec. 4.4.5.

[1] Based on the industrial experience of authors, this assumption is safe, i.e. there are usually examples of anomalies, though their diversity might be low.

4.2 Shape-guided decomposition

Figure 4.1: Examples of decomposition on the wildlife MNIST dataset by the SGVAE-GAN model. The first row contains the input samples, below that are the decoded masks, backgrounds, foregrounds, and finally the whole reconstructions. The model can learn appropriate masks in an unsupervised fashion.

We demonstrate the advantage of disentanglement in detecting anomalies on image data, because (i) the disentanglement has been researched mostly in this domain [235, 239, 242], (ii) most public datasets are in this domain, (iii) the anomaly detection community is the most active in this field.

We assume an image x to be composed of three main components: a mask (shape of the object), a background texture, and a foreground texture (the object). The mask together with the foreground texture defines the semantic meaning of the object, as it is what defines the class of the image in datasets. According to the above assumptions, each component (mask, background texture, and foreground texture) is generated from an independent variable. Therefore, for the latent distribution $p(z)$ it holds that

$$p(z) = p_{z_m}(z_m)p_{z_f}(z_f)p_{z_b}(z_b), \tag{4.11}$$

where subscripts m, f, b denote the mask, foreground, and background latent variable, respectively. A representative example of a dataset where each image is composed of three independent components is the wildlife MNIST dataset [243]. It is constructed using masks from MNIST [216] combined with foreground and background textures from [244] (see more details on its construction in Sec. B). For examples from individual classes see the top row of Fig. 4.1.

The independent decomposition (4.11) is akin to (4.8), but here we ascribe meaning to the individual latent spaces. This means that the model built on top of the assumption (4.11) has an inductive bias, which enables the unsupervised disentangled representation of the individual components of an image. The independency is further reinforced by the fact that the parts of the model responsible for the representation

of the components will not share parameters. Also, since we want to apriori ascribe specific meaning to the disentangled latents, we cannot use automatic disentanglement of individual dimensions as is the case in literature [234, 239, 240], since there, the meaning can be given to the individual dimensions only using a manual post-hoc analysis with labeled data. Another argument against disentanglement on the level of individual latent dimensions is the fact that eventually the model is used to compute anomaly score (4.10), and using too many α_i values would likely lead to overfitting, since labeled anomalies are scarce.

In this chapter, the structure of the proposed generative model with independent latent spaces and its training procedure are described first. Then, we combine the model with the anomaly score derived in the previous chapter in order to be able to detect and describe semantic anomalies.

4.2.1 Shape-guided VAEGAN model

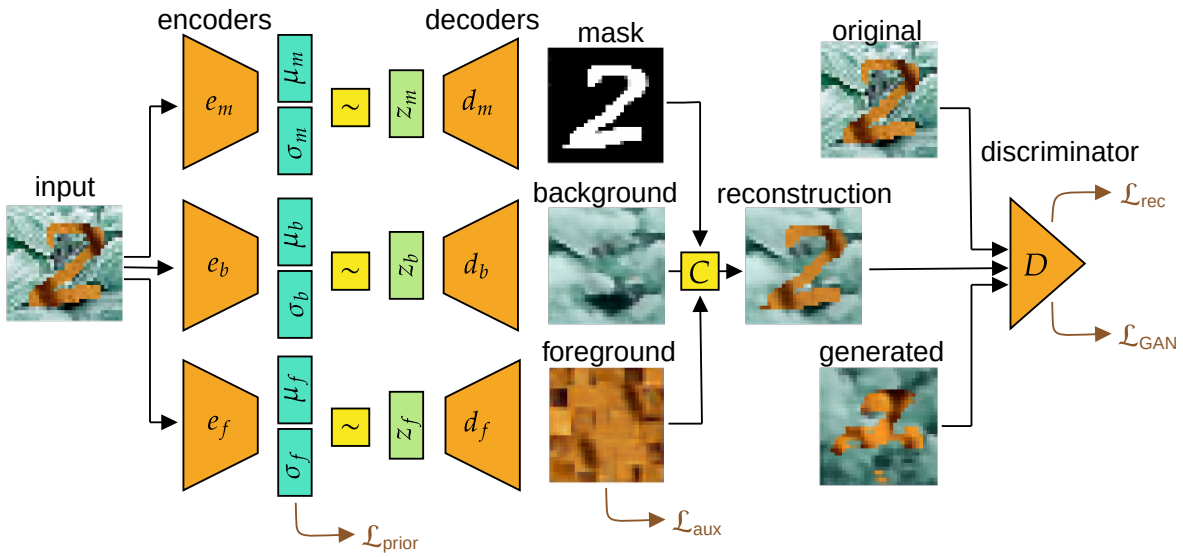


Figure 4.2: Schema of the proposed model. Convolutional blocks are denoted in orange, fully connected in cyan, and intermediate representations in green. The yellow squares represent special operations - c is the composition (4.14) and the reparametrization trick is denoted by $\sim$. The generated sample is obtained by feeding samples from priors $\mathcal{N}(0, \mathbf{I})$ to the decoders.

The model disentangling the input image into its three components is our synthesis of a VAEGAN [245] and Counterfactual generative networks (CGN) [243].[2] Building blocks of this model, further denoted SGVAEGAN, are outlined in Fig. 4.2. The model uses three separate independent autoencoders for three components of an image, a block combining them to the reconstructed image, and a discriminator for a tighter fit. We emphasize that it is this strict separation of autoencoders that motivates the desired disentanglement.

[2]Counterfactual generative networks [243] were selected because based on our preliminary experiments, the result offered superior disentanglement likely due to highly inductive bias.

Autoencoders responsible for individual components are vanilla VAE consisting of an encoder

$$q_i(z_i|x,\theta) = \mathcal{N}(z_i; \mu_i(x|\theta), \mathrm{diag}(\sigma_i(x|\theta))), \quad i \in \{m, f, b\}, \tag{4.12}$$

and a decoder

$$x_i = g_i(z_i|\theta), \quad i \in \{m, f, b\}. \tag{4.13}$$

Latent variables z_i are sampled through the usual reparametrization trick [34] from a normal distribution (4.12) with a diagonal covariance matrix.

All three autoencoders output a tensor of the size of the input image, x_m, x_f, and x_b, which are composed by a compositor $c(x_m, x_f, x_b)$ as

$$x' = c(x_m, x_f, x_b) = x_m \odot x_f + (\mathbf{1} - x_m) \odot x_b, \tag{4.14}$$

where $\odot$ denotes Hadamard (element-wise) product and $\mathbf{1}$ is a matrix of ones with the same dimension as x_m. The elements of the mask x_m lie in the interval $[0,1]$ and the training procedure ensures that they are pushed to the extremes of the closed interval, such that elements with nonzero values represent the pixels that contain the most prominent object in the image.

The *loss function* optimized during training consists of a reconstruction error, GAN-loss, regularization of the latent space, and auxiliary part,

$$\mathcal{L} = \lambda_{\mathrm{rec}}\mathcal{L}_{\mathrm{rec}} + \mathcal{L}_{\mathrm{GAN}} + \mathcal{L}_{\mathrm{prior}} + \mathcal{L}_{\mathrm{aux}}. \tag{4.15}$$

Their contributions are controlled by a scalar weight λ_{rec} and by weights contained in $\mathcal{L}_{\mathrm{aux}}$. The first three parts in Eq. (4.15) are adopted from VAEGAN while the last part from CGN. The rest of this section describes them in detail.

The *reconstruction loss* uses a "feature-matching" construction [133], which was chosen because of its good results in [211]. It generalizes standard reconstruction loss by comparing x and x' at a certain depth of the discriminator as

$$\mathcal{L}_{\mathrm{rec}} = L_{\mathrm{fm}}(x,k) = \|d_k(x) - d_k(x')\|_2, \tag{4.16}$$

where $d_k(x)$ is the intermediate representation of x at the k-th layer of the discriminator. When $k = 0$, the loss coincides with the classical log-likehood $-\mathbb{E}_{q(z|x)}[\log p(x|z)]$ for VAE with gaussian output distribution. In the experiments in Sec. 4.4, k is treated as a hyperparameter subjected to tuning.[3] The authors of the VAEGAN model claim that incorporating the discriminator for image reconstruction leads to better overall reconstruction/generation quality as it pushes the model to be able to abstract beyond the capabilities of pixel-wise reconstruction loss (4.16).

The *GAN loss* $\mathcal{L}_{\mathrm{GAN}}$ was adopted from the VAEGAN model. It optimizes the discriminator d and the decoders $g_{m,\theta}$, $g_{f,\theta}$, and $g_{b,\theta}$, of the model via

$$\mathcal{L}_{\mathrm{GAN}} = -\log d(x) - \log(1 - d(x')) - \log(1 - d(\tilde{x})), \tag{4.17}$$

where $\tilde{x}$ is a sample that was generated by decoding latent codes $\tilde{z}_m$, $\tilde{z}_f$, and $\tilde{z}_b$ sampled from $\mathcal{N}(0, \mathbf{I})$ instead from the learned latents (4.12) and composing them via (4.14). The discriminator outputs a scalar in the range $[0,1]$, where a higher value is meant

[3] Based on our experimental results, we cannot recommend a single good value, as values selected on the validation set ranged from 0 to 7.

to be assigned to a sample coming from the training dataset and a lower to the reconstructed/generated image. This is achieved by minimizing the loss (4.17) with respect to the discriminator parameters and maximizing it with respect to the decoders' parameters. This is similar to discriminator/generator training in a classical GAN [33].

The *regularization* of the latent space is done by the usual KL divergence [34] between the learned latent distribution $q_\phi(z|x)$ and the prior $p(z)$ as

$$\mathcal{L}_{\text{prior}} = D_{\text{KL}}(q_\phi(z|x)||p(z)) = \sum_{i\in\{m,f,b\}} D_{\text{KL}}(q_{i,\phi}(z_i||x)|p(z_i)), \tag{4.18}$$

where the sum is possible because it is assumed z_i to be independent. Furthermore, since it is assumed $p(z_i) = \mathcal{N}(0, \mathbf{I})$, $\mathcal{L}_{\text{prior}}$ can be computed analytically [34].

The *auxiliary loss*, adopted from [243], is a weighted combination of three parts

$$\mathcal{L}_{\text{aux}} = \lambda_{\text{bin}}\mathcal{L}_{\text{bin}}(x_m) + \lambda_{\text{mask}}\mathcal{L}_{\text{mask}}(x_m) + \lambda_{\text{text}}\mathcal{L}_{\text{text}}(x_m, x_f, x). \tag{4.19}$$

The texture loss $\mathcal{L}_{\text{text}}(x_m, x_f, x)$ ensures that the parts of the network assigned to reconstruct the background and foreground are not switched and that no shape information is stored in the mask. It is computed in the following way: 36 patches are sampled from the image x in regions where the mask x_m is nonzero. These are then composed together to form a tensor x_g that is as large as the image. Then, a perceptual loss [246] between x_g and x_f is minimized. The perceptual loss is computed as the L1 distance between the activation maps of x_f and x_g in the first four convolutional layers of a VGG16 [247] network.

The term $\mathcal{L}_{\text{bin}}$ forces elements of the mask to be close to either 0 or 1. The objective to be minimized is

$$\mathcal{L}_{\text{bin}}(x_m) = -\frac{1}{N}\sum_{i}^{N} x_{m,i}\log_2(x_{m,i}) + (1 - x_{m,i})\log_2(1 - x_{m,i}), \tag{4.20}$$

where the index i goes over all the elements of the tensor x_m and N is the number of its elements.

Finally, $\mathcal{L}_{\text{mask}}$ prevents degeneration of the mask to be all zeroes or all ones, which would lead to failure in the identification of background / foreground. This is achieved by computing

$$\mathcal{L}_{\text{mask}}(x_m) = \left[\max\left(0, \tau - \frac{1}{N}\sum_{i}^{N} x_{m,i}\right) + \max\left(0, \frac{1}{N}\sum_{i}^{N} x_{m,i} - \tau\right)\right] \tag{4.21}$$

where $\tau \in [0, 1]$ is a parameter that forces the mask to occupy a total area of the image that is in the interval $[\tau, 1 - \tau]$.

The complete training procedure of the SGVAEGAN model is described in detail in Alg. 6.

4.2.2 Detecting anomalies with SGVAEGAN

If the SGVAEGAN is trained well, particularly if decomposition into latent spaces is (at least approximately) right, different types of anomalies should be in low-density regions of different latent spaces, as is illustrated in Fig. 4.3 on the wildlife MNIST

Algorithm 6 Training of the SGVAEGAN model. The budget is either a time limit or a fixed maximum number of iterations. Capital letters denote a batched variable.

Require: Training set, (ϕ, θ, φ) = (encoder, decoder, discriminator) parameters.

$(\phi, \theta, \varphi) \leftarrow$ initialize parameters
while (ϕ, θ, φ) not converged or budget exhausted **do**
  $X \leftarrow$ batch of L samples from the dataset
  // Computation of prior, reconstruction and auxilliary losses
  $(Z_m, Z_f, Z_b) \leftarrow$ encodings of X
  $\mathcal{L}_{\text{prior}} \leftarrow \sum_{i \in \{m,f,b\}} D_{\text{KL}}(q_{i,\phi}(Z_i \| X) | p(Z_i))$
  $(X_m, X_f, X_b) \leftarrow (g_{m,\theta}(Z_m), g_{f,\theta}(Z_f), g_{b,\theta}(Z_b))$
  $X' \leftarrow c(X_m, X_f, X_b)$
  $\mathcal{L}_{\text{rec}} \leftarrow \|d_l(X) - d_l(X')\|_2$
  $\mathcal{L}_{\text{aux}} \leftarrow \lambda_{\text{bin}} \mathcal{L}_{\text{bin}}(M) + \lambda_{\text{mask}} \mathcal{L}_{\text{mask}}(M) + \lambda_{\text{text}} \mathcal{L}_{\text{text}}(M, F, X).$
  // Adversarial loss
  $(\tilde{Z}_m, \tilde{Z}_f, \tilde{Z}_b) \leftarrow$ samples from the prior
  $(\tilde{X}_m, \tilde{X}_f, \tilde{X}_b) \leftarrow (g_{m,\theta}(\tilde{Z}_m), g_{f,\theta}(\tilde{Z}_f), g_{b,\theta}(\tilde{Z}_b))$
  $\tilde{X} \leftarrow C(\tilde{X}_m, \tilde{X}_f, \tilde{X}_b)$
  $\mathcal{L}_{\text{GAN}} \leftarrow -\log d(X) - \log(1 - d(X')) - \log(1 - d(\tilde{X}))$
  // Update of the parameters
  $\phi \overset{+}{\leftarrow} -\nabla_\phi (\lambda_{\text{rec}} \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{aux}} + \mathcal{L}_{\text{prior}})$
  $\theta \overset{+}{\leftarrow} -\nabla_\theta (\lambda_{\text{rec}} \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{aux}} - \mathcal{L}_{\text{GAN}})$
  $\varphi \overset{+}{\leftarrow} -\nabla_\varphi \mathcal{L}_{\text{GAN}}$
end while

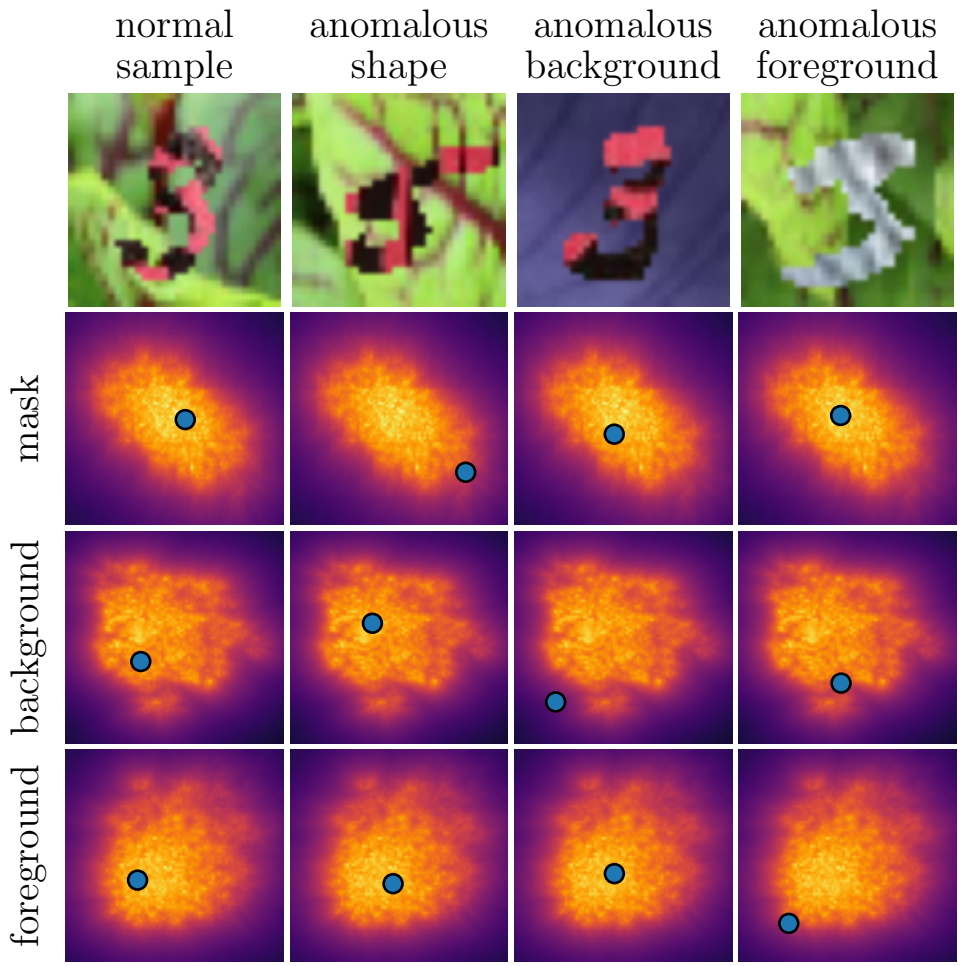


Figure 4.3: Position of different images encoded to individual latent spaces using SG-VAEGAN model with two-dimensional latent spaces. The training set consisted of images with factors fixed to those of the digit in the first column. Also depicted are the densities of encodings of normal (training) data, estimated by a kNN detector (4.28).

dataset, where we see that anomalous shape, background texture, and foreground texture is anomalous in the corresponding latent space. We can use the score (4.10) and assign $p_{n_i}(z_i) = p_{z_i}(z_i), i \in \{m, f, b\}$, since the model is trained on normal data. Then we have

$$s(x) = -\log p(e) - \log \left| \frac{\partial g^{-1}(x)}{\partial x} \right| - \sum_{i \in \{m,f,b\}} \alpha_i \log p_{z_i}(z_i), \tag{4.22}$$

where

$$g(z) = g(z_m, z_f, z_b) = g_m(z_m) \odot g_f(z_f) + (1 - g_m(z_m)) \odot g_b(z_b). \tag{4.23}$$

Following derivations in [169], we use the fact that $\frac{\partial g^{-1}(x)}{\partial x} = \left(\frac{\partial g(z)}{\partial z}\right)^{-1}$, and assume the independency of latent spaces z_m, z_f, z_b, then Eq. (4.22) can be rewritten to

$$s(x) = -\log p(e) + \sum_{i\in\{m,f,b\}} \log\left|\frac{\partial g(z_m, z_f, z_b)}{\partial z_i}\right| - \sum_{i\in\{m,f,b\}} \alpha_i \log p_{z_i}(z_i) \tag{4.24}$$

$$= -\log p(e) + J_d(x) - \sum_{i\in\{m,f,b\}} \alpha_i \log p_{z_i}(z_i), \tag{4.25}$$

where $J_d(x)$ denotes the sum of jacobian terms functionally dependent on the input image x through (4.12). A reader recognizes that $\frac{\partial g(z_m,z_f,z_b)}{\partial z_i}$ in (4.24) is not square, hence its determinant is zero. Ref. [169] suggest estimating the determinant from a diagonal matrix after SVD decomposition, which is valid if one assumes the orthogonality of the data and noise (see Alg. 7).

Algorithm 7 Calculation of $J_d(x)$

Require: image x, encoder means $\mu_{i,\phi}$, decoders $g_{i,\theta}$, $i \in \{m, b, f\}$

$\left(z_m, z_b, z_f\right) \leftarrow \left(\mu_{m,\phi}(x), \mu_{b,\phi}(x), \mu_{f,\phi}(x)\right)$
$J \leftarrow 0$
for $i \in \{m, b, f\}$ **do**
    $J_i \leftarrow \frac{\partial g(z_m,z_b,z_f)}{\partial z_i}$
    $(U_i, S_i, V_i) \leftarrow \mathrm{SVD}(J_i)$
    $J \overset{+}{\leftarrow} \sum_{j,S_{i,j}\neq 0} 2\log S_{i,j}$
end for
$J_d(x) \leftarrow J$

The proposed score, based on (4.24), reads as a weighted sum of the individual components

$$s(x) = \alpha_r s_r(x) + \alpha_j s_j(x) + \alpha_m s_m(x) + \alpha_f s_f(x) + \alpha_b s_b(x), \tag{4.26}$$

where $s_i, i \in \{r, j, m, f, b\}$ are individual anomaly score components which will be described in the following text. The jacobian term is relabeled for future purposes as $s_j(x) = J_d(x)$. The α_r, α_j weights were added in order to tune the total score to the modalities of anomalous data, which were not seen during the training, and therefore the base model is not fitted to them. Also, setting the values of $\alpha_\cdot$ can be seen as a sort of normalization of the otherwise unnormalized anomaly score. The reweighing of the scores via $\alpha_\cdot$ is also important in order to set an appropriate aggregated threshold, which is otherwise different for each subscore. The values can be set manually, but an automatic fitting procedure that requires a small number of labeled anomalies is described in the following text.

Reconstruction error Reconstructed samples are needed for the computation of the reconstruction term $-\log p(e)$. However, since the reconstruction steps (4.12)-(4.14)

contain sampling through the reparametrization trick, the reconstructions are stochastic. To stabilize the estimate, a sampled reconstruction error is computed as

$$s_{\mathrm{r}}(x) = -\frac{1}{\sigma^2 L}\sum_{l=1}^{L}\|x - x_l'\|, \tag{4.27}$$

where the scalar variance $\sigma^2 \in \mathbb{R}$ is estimated from the data during training of the model. The number of samples was set to $L = 10$ during our experiments. Note that we have decided not to use (4.2), where sampling is replaced by propagating the mean of the encoder distribution. This is motivated by the survey [211], where the sampled version (4.27) performed slightly better.

Latent scores A correct estimate of the likelihood of latent representations $p_{z_i}(z_i)$, $i \in \{m, f, b\}$ is important for the score (4.26). Even though latent representations are regularized during the fitting of the model to have normal distribution $\mathcal{N}(0, 1)$, it was shown [181] that the fit is usually not very good, as can also be seen from Fig. 4.3. We therefore approximate $p_{z_i}(z_i)$ by the k-nearest-neighbor (kNN) density estimator [248] trained on latent representations of normal data. The score of a sample in the i-th latent space, $i \in \{m, f, b\}$, has the form

$$s_i(x) = \frac{1}{k}\sum_{z_j \in \mathcal{Z}_{k,i}} \|z_i - z_j\|_2, z = \mu_{i,\phi}(x), \tag{4.28}$$

which is the average distance between the projection z of the tested sample x into the latent space, and the set $\mathcal{Z}_{k,i}$ of the k-nearest projections of the normal data.

Optimization of α The proposed score is effectively a weighted sum of individual parts. In theory, one can set the weights $\alpha_\cdot$ by himself if one knows in which latent to expect the anomaly. But in practice and in our experiments, they will be estimated from data by regularized logistic regression. Their value is the solution of

$$\alpha^* = \arg\min_{\alpha} -\sum_j y_j \log\sigma(s(x_j|\alpha)) + (1 - y_j)\log(1 - \sigma(s(x_j|\alpha))) + \beta\|\alpha - \alpha_0\|_2, \tag{4.29}$$

$$s(x_j|\alpha) = \sum_i \alpha_i \hat{s}_i(x_j), i \in \{r, j, m, f, b\}, \tag{4.30}$$

where $\sigma(.)$ is the sigmoid function, $y_j \in \{0, 1\}$ are labels, α_0 is a prior value and the index j goes over the samples in the labeled dataset. Since the scores $s_i(x)$ can have very different scales (e.g. $s_r(x) \sim 10^4$ while $s_f(x) \sim 10^0$), we normalize the values available for fitting to have zero mean and unit variance. The rescaled scores are denoted as $\hat{s}_i(x)$. We set $\beta = \frac{\beta_0}{n_1}, \beta_0 \in \mathbb{R}$, where n_1 is the number of positive (anomalous) samples in the dataset, and β_0 is a hyperparameter. This ensures that the prior has a large influence over the final value of α when there is a small number of known anomalies, thus ensuring the robustness of the final α estimate. The value of α_0 is set such that $\alpha_{0,k} = 1$ for such k where the AUC computed from $\{s_{i,k}, y_i\}_i$ is maximal and zero everywhere else. The criterion (4.29) is optimized by an LBFGS optimizer [249].

Alternative score While the score (4.24) is theoretically correct under the assumption that anomalies are located in areas of low density, Ref. [169] shows that it does not work well when the model is fit on data without anomalies. Our experiments shown below arrived at the same conclusion. We suspect the cause to be that the decoders $g_.$ can be arbitrary (with arbitrary jacobian) in parts of the space not supported by the data, where anomalous samples are located, which destroys the likelihood. Moreover, the computation of the determinant is so expensive, that the score is effectively useless for state-of-the-art image models. Therefore, we propose dropping the jacobian term $J_d(x)$ from (4.26) and adding the discriminator score

$$s_d(x) = 1 - d(x), \tag{4.31}$$

which works well for anomaly detection according to [245] and our own experience (see the top baseline models in our experiments, which all use anomaly score based on the discriminator). The alternative score then reads

$$s(x) = \alpha_r s_r(x) + \alpha_d s_d(x) + \alpha_m s_m(x) + \alpha_f s_f(x) + \alpha_b s_b(x). \tag{4.32}$$

The values of $\alpha_.$ are optimized similarly to (4.29).

4.3 Related work

Anomaly detectors with neural network backends have become popular recently, mainly for their ability to efficiently process high–dimensional image data. Variants of (variational) autoencoders are a popular choice for novel anomaly detectors. The works [126, 127, 250, 172, 251] consider only the reconstruction error / probability of a sample as the anomaly score. In [252, 72, 99], the anomaly score is computed in the latent space. A combination of two scores is used to detect anomalies in [253, 254], but without any reweighing of those with respect to the expected type of anomalies. Furthermore, it has been shown that anomaly detection via reconstruction score has its limitations [245, 211] on real-world images and it was speculated that models that use the discriminator of a GAN-like model [23, 142, 176, 129, 255, 256, 257] may prove to be better, although more difficult to properly train and tune. The proposed method combines the aforementioned approaches, as it considers a weighted combination of scores computed in the latent and image space, and provides a scheme that chooses their optimal combination. Moreover, we offer a possible explanation of the origin of the anomaly, which is not standard in any of the listed techniques.

Another line of work recommends using some kind of weak supervision [258, 240], which means that the model is a classifier that is trained to recognize between the normal data (one class of CIFAR10) and an auxiliary dataset (all data from CIFAR100). These methods work extremely well, however, depend on the availability of an auxiliary dataset (which is not guaranteed for a real-world problem) and require a completely different evaluation methodology, which is not in line with our objectives, i.e. identification of semantic anomalies.

An inspiration for the proposed model is the Counterfactual Generative Network (CGN) [243], where the authors introduce a GAN model that generates individual components of the image (shape, background, and foreground). However, the CGN model does not provide any scores for the individual latent spaces, which is needed

Figure 4.4: Examples of the datasets used in the experiments. Datasets from top row to bottom: COCOPlaces, CIFAR10, SVHN2 and MVTec-AD. MVTec-AD examples of normal and anomalous samples from the *bottle*, *capsule*, *metal nut*, *pill*, and *transistor* classes are shown.

in (4.11), since it only has one image-level discriminator. On the other hand, the proposed method replaces generators with autoencoders, thus providing a mapping of a sample to independent latent spaces, where an anomaly score can be computed.

The field of unsupervised disentanglement is also related to the problem we are solving, as the methods try to encode individual factors of the data into independent dimensions of a single latent space. Although a lot of work in this area has been done over the last years [234, 235, 236, 237, 238, 239], it has been proven both theoretically and empirically that complete unsupervised disentanglement is impossible [259, 260, 261]. Our approach relies on the introduction of an inductive bias into the generating network which is sufficiently general to allow unsupervised training. Also, it allows us to assign meaning to the disentangled latent spaces, which is not the case with general disentanglement approaches.

4.4 Experiments

4.4.1 Datasets

4.4.2 Baseline methods

We have selected unsupervised anomaly detectors mainly based on the review paper [211], specifically those that were amongst the top performers on colored images. These models include:

Variational Autoencoder (VAE) - a convolutional VAE [126, 34] that uses the sampled reconstruction error (4.27). The decoder variance σ^2 is estimated from the data.

Feature-matching GAN (fmGAN) - a convolutional GAN model trained using the feature-matching loss [133], similar to (4.16). The anomaly score is based on the discriminator (4.31).

VAEGAN - a convolutional VAE where reconstruction is enforced through a discriminator [245]. The anomaly score is (4.31).

Deep Support Vector Data Description (DSVDD) - is a model [38] that learns a transformation of data via a neural network to a subspace where the anomalies lie outside of a hypersphere composed of transformed normal data. The anomaly score is then the distance of a point from the center of the hypersphere.

fast Anomaly GAN (fAnoGAN) - a GAN model [142] trained via Wasserstein loss and gradient penalization that identifies anomalies by backward-searching the latent code z that is the most likely to generate the given test samples.

Counterfactual Generative Network (CGN) - this is a baseline model [243] for the decomposition of data into three components. Although not originally intended as an anomaly detector, it can be used as one as it provides the discriminator score (4.31) and proved itself to be competitive in our experiments.

Shape Guided VAE (SGVAE) - this is a modification of the proposed model that is trained without a discriminator and the reconstruction loss is $L_{\text{fm}}(x, 0) = -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$, similar to a classical VAE. The anomaly score for this model is the sampled reconstruction error (4.27).

Shape Guided VAEGAN (SGVAEGAN) - this is the basic proposed model that is trained in a completely unsupervised fashion without considering the full anomaly scores (4.26),(4.32). Instead, the default anomaly score is the discriminator one (4.31).

SGVAE$_\alpha$ - this is the SGVAE model where the score (4.26) is considered. To compute the anomaly scores, SGVAE models pre-trained in an unsupervised fashion were used and only the weights α were computed on a validation dataset.

SGVAEGAN$_\alpha$ - this is the full proposed model. In the first experiments, the score (4.26) with the jacobian term was used, but later the jacobian score is dropped and the score (4.32) was computed instead, see the discussion in Sec. 4.4.3.

The comparison of the baselines and the proposed models was conducted in a rigorous manner in accord with the extensive overview [211]. The datasets were split into training, validation, and test subsets for each of their classes (or subproblems in the case of the MVTec-AD dataset). Details on the splits for individual experiments can be found in the respective sections below. Then, for each such split and each model, 50 hyperparameter settings were randomly sampled from a set of possible values. The use of Bayesian optimization to select hyperparameters was considered but eventually dropped as it did not have an impact on the relative rank of the methods [211]. The individual models were trained using these hyperparameters. The validation set was used to select the best hyperparameter values for a given model on a specific subproblem. Unless mentioned otherwise, the experiments below report (ROC)AUC values of the selected models computed on the test set. The models were trained for 50 epochs each. All the models were implemented either in PyTorch or in Julia. The code for the proposed model can be found at `github.com/vitskvara/sgad` and the evaluation framework is at `github.com/vitskvara/GenerativeAD.jl`.

class	AUC - no $J_d(x)$	AUC - with $J_d(x)$	α_r	α_j
0	0.70 ±0.04	0.70 ±0.05	1.00	0.00
1	0.82 ±0.01	0.82 ±0.02	1.09	-0.03
2	0.71 ±0.02	0.72 ±0.02	0.93	-0.01
3	0.64 ±0.02	0.64 ±0.02	0.94	0.01
4	0.72 ±0.03	0.72 ±0.03	1.00	0.00
5	0.67 ±0.01	0.66 ±0.01	0.98	0.00
6	0.68 ±0.02	0.68 ±0.02	0.60	0.00
7	0.73 ±0.05	0.73 ±0.05	1.00	0.00
8	0.69 ±0.04	0.71 ±0.02	0.98	-0.04
9	0.63 ±0.05	0.63 ±0.04	0.80	0.00

Table 4.1: Experiment with $J_d(x)$ on a subset of the SVHN2 dataset. For each normal class, training and testing sets containing 750 normal and 150 anomalous samples were used. To obtain the presented statistic, the subsets were sampled 5 times. The mean values of estimated *alpha* weights are also presented and show that the weight of the jacobian term is suppressed during their computation.

4.4.3 The contribution of the jacobian

We start the dissection of experimental results by an analysis of the jacobian term $J_d(x)$ in (4.26). See Tab. 4.1 where the contribution of the term $J_d(x)$ to the anomaly score was measured by comparison of AUC performance of a model that used it and one that did not on a subset of the SVHN2 dataset. The difference in performance is almost negligible, but the computational costs of computing the jacobian matrix for image data are very high, making it extremely impractical. Therefore, the term is omitted from all further experiments, and the score (4.32) is used for the SGVAEGAN$_\alpha$ instead, while for SGVAE$_\alpha$, the term is dropped from (4.26).

4.4.4 Detection of semantic anomalies

In this experiment, we observe how the proposed detector behaves as it gradually incorporates more knowledge about labeled anomalies. To simulate a semantic anomaly scenario, the following training and testing protocol is observed on the Wildlife MNIST and COCOPlaces datasets. The training set consists of samples of one class (the whole experiment is repeated for all 10 classes) from the non-mixed version of the datasets (see Sec. B for details of how this is generated). Then, for each factor of variation, validation and test sets are drawn from the mixed version of the dataset, where the anomality of a sample is based on whether the individual factor is the same or different as in the training dataset. We believe that this simulates a semantic anomaly problem, as for each factor, we have a validation/test dataset where there is data unseen by the model during training but not considered to be anomalous.

See Fig. 4.5 for a comparison of the proposed models and the baselines. On wildlife MNIST, both of the proposed alternatives where the weights α are gradually improved with added anomalies outperform the remaining baselines convincingly in detection of all types of anomalous factors. On COCOPlaces, the difference is smaller and only the SGVAGEAN$_\alpha$ is clearly better. This stems from the dataset being much harder to

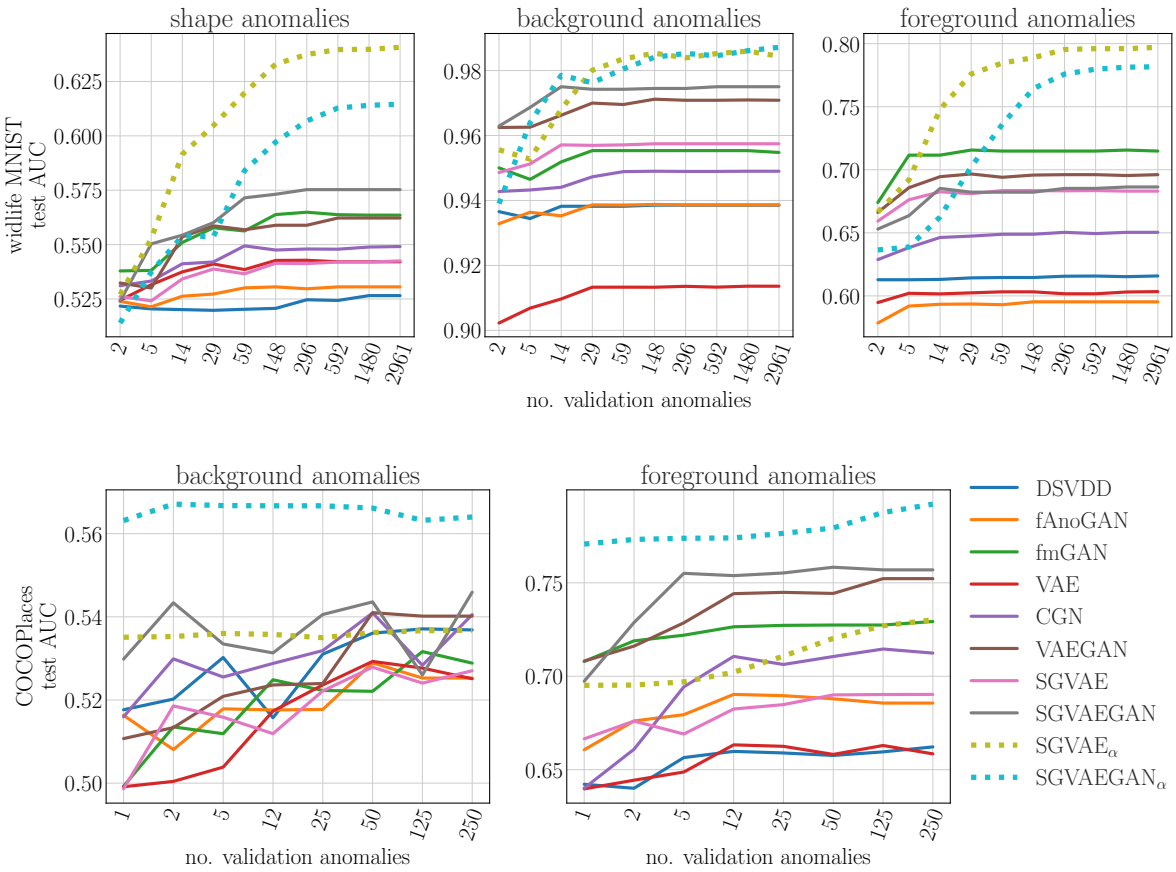


Figure 4.5: Semantic anomaly detection experiment on the wildlife MNIST (top) and COCOPlaces (bottom) datasets. The x-axis covers a changing number of anomalies present in the validation dataset. The number of normal samples remains the same and is equal to the highest anomaly count presented in the figures. That means that in the most-informed scenario the validation set consists of the same number of anomalies and normal samples. The models are trained on the same training non-mixed data, their hyperparameters are selected on the validation dataset and the resulting test set AUC is reported on the y-axis. The AUC values are also averaged over 10 classes and a 5-fold random selection of the validation anomalies.

correctly decompose. Unlike on wildlife MNIST, in some classes, there is no universal shape that all the objects share and the influence of the background is such that even for a human it is sometimes difficult to discern the principal object in an image.

4.4.5 Anomaly factor identification

It is possible to leverage the structure of the model presented above to identify the source of the anomality of an image through the information that is encoded in the individual latent spaces. The scenario here is that a potential anomaly has been identified, and now we want to know what is the cause of the anomaly. Under the formalism (4.6), estimation of the anomaly factor $p(y|z) \propto p(z|y)$ is possible only for proper distributions. Since we have opted for improper distribution to minimize the number of unknown parameters, we have to design an approximate method that does not rely on the availability of proper normalization. We propose two different approaches to it.

Ranked detection: we assume that the anomalous factor can be detected through the latent space kNN scores (4.28). If only one factor is anomalous, then there should be a difference in the encodings of normal samples and the encodings of the anomalous one, and this difference should be visible in one of the latent spaces. In order to achieve this, the encodings of a sample x for each latent space are computed. Then, kNN scores of these encodings with respect to encodings of normal data are computed, resulting in a vector of kNN scores $s_{\text{kNN}}(x) = (s_m(x), s_f(x), s_b(x))$. If the distances in the latent spaces were normalized, one could select the source of anomality (shape, background, or foreground texture) to have the highest kNN score. This is, however, not the case, even though all the latent spaces are regularized to resemble the same prior, see Fig. 4.3. Standardizing the scores post-hoc to zero mean and unit variance did also not prove to work. Instead, for each individual latent space, we estimate the quantile of the sample using the rank of the kNN score of the tested sample among the kNN scores of normal data and compute its percentile – the higher it is, the more anomalous the sample is in this latent space (at least with regard to the normal data). The latent space with the highest percentile should then be the source of the anomaly. The results for this method for models trained on the mixed version of the wildlife MNIST dataset (for a detailed description of the data see Sec. B) are shown in Tab. 4.2.

	ranked			masked	
normal class	shape	background	foreground	background	foreground
0	0.82	0.19	0.62	0.88	0.95
1	0.84	0.92	0.19	0.85	1.0
2	0.92	0.37	0.7	0.95	0.6
3	0.55	0.86	0.24	0.63	0.96
4	0.79	0.46	0.85	0.62	1.0
5	0.57	0.89	0.31	0.72	1.0
6	0.74	0.95	0.52	0.96	0.96
7	0.47	0.84	0.75	0.87	0.98
8	0.64	0.64	0.62	0.98	0.97
9	0.88	0.27	0.1	0.77	0.94

Table 4.2: Accuracy of factor detection on the wildlife MNIST dataset for *ranked* and *masked* methods.

Masked detection: it is evident from the results of the ranked method that in some cases, it is difficult to find a model that would identify all three factors satisfactorily (better than random chance, which would give an accuracy score of 0.33). We believe it is because of the innate dependence of the result on the correctness of the mask, which might be nonsensical if one of the factors was not seen in training data. Therefore, we have devised a second factor detection technique, which only distinguishes between an anomaly in the background and foreground texture. For an input x, the reconstructed image x' and a mask $x_m$is computed. Using these, we compute a normalized

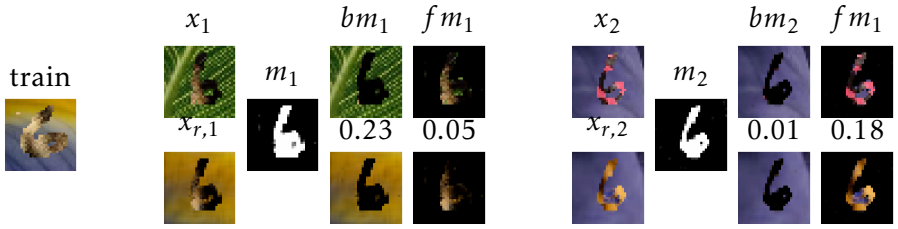


Figure 4.6: Masked detection technique for identifying the source of anomality. The model was originally trained on samples similar to the one on the very left, brown-striped sixes on purple-yellow background. To distinguish between anomalies in the background and foreground texture, a reconstruction x_r and a mask m are computed for a test sample x_1, which is anomalous in the background, and x_2, which is anomalous in the foreground texture. The scores S_b (4.33), S_f (4.34) are computed as scaled differences between the original and reconstructed masked–out backgrounds, $b_m = x \odot (1 - x_m)$ and foregrounds $f_m = x \odot x_m$ and their values are displayed between the image of the original and reconstructed foregrounds and backgrounds.

reconstruction error on the background and foreground separately

$$S_b = \frac{\sum_i \left((x_i - x'_i)(1 - x_{m,i}) \right)^2}{\sum_i 1 - x_{m,i}} \tag{4.33}$$

$$S_f = \frac{\sum_i \left((x_i - x'_i) x_{m,i} \right)^2}{\sum_i x_{m,i}} \tag{4.34}$$

where index i goes over elements in all dimensions of an array $x \in \mathbb{R}^{h,w,3}$ representing an RGB image. See Fig. 4.6 for an illustration of the principle of this method. The normalization factor is important since the object in the foreground usually covers fewer pixels than the background. If S_b is higher, the prediction for the anomaly factor is "background" and vice versa. Results from the factor identification experiment using this method are in the second half of Tab. 4.2. The background anomaly detection accuracies are not completely satisfactory on all digit classes. This is caused by some classes having very similar backgrounds, e.g. a very common misclassification for class "4" is that with anomalous background from classes "1" or "7", see Fig. 4.1, where the background is reconstructed rather well, while even a small imprecision in the mask leads to a high reconstruction error on the foreground. Still, this method of detection of the source of anomality might be useful in some real-world problems, given we can train the model to produce correct masks. The main benefit of these methods is that they are completely unsupervised and at this moment, we are not aware of any other unsupervised anomaly detection method that would offer some kind of explainability of the source of anomality in image data.

4.4.6 Anomaly detection on benchmark datasets

This experiment compares the proposed model in a traditional anomaly detection scenario on the datasets described in Sec. B. We assume the leave-one-in scenario, which is that the training dataset contains samples from one class and the rest is considered

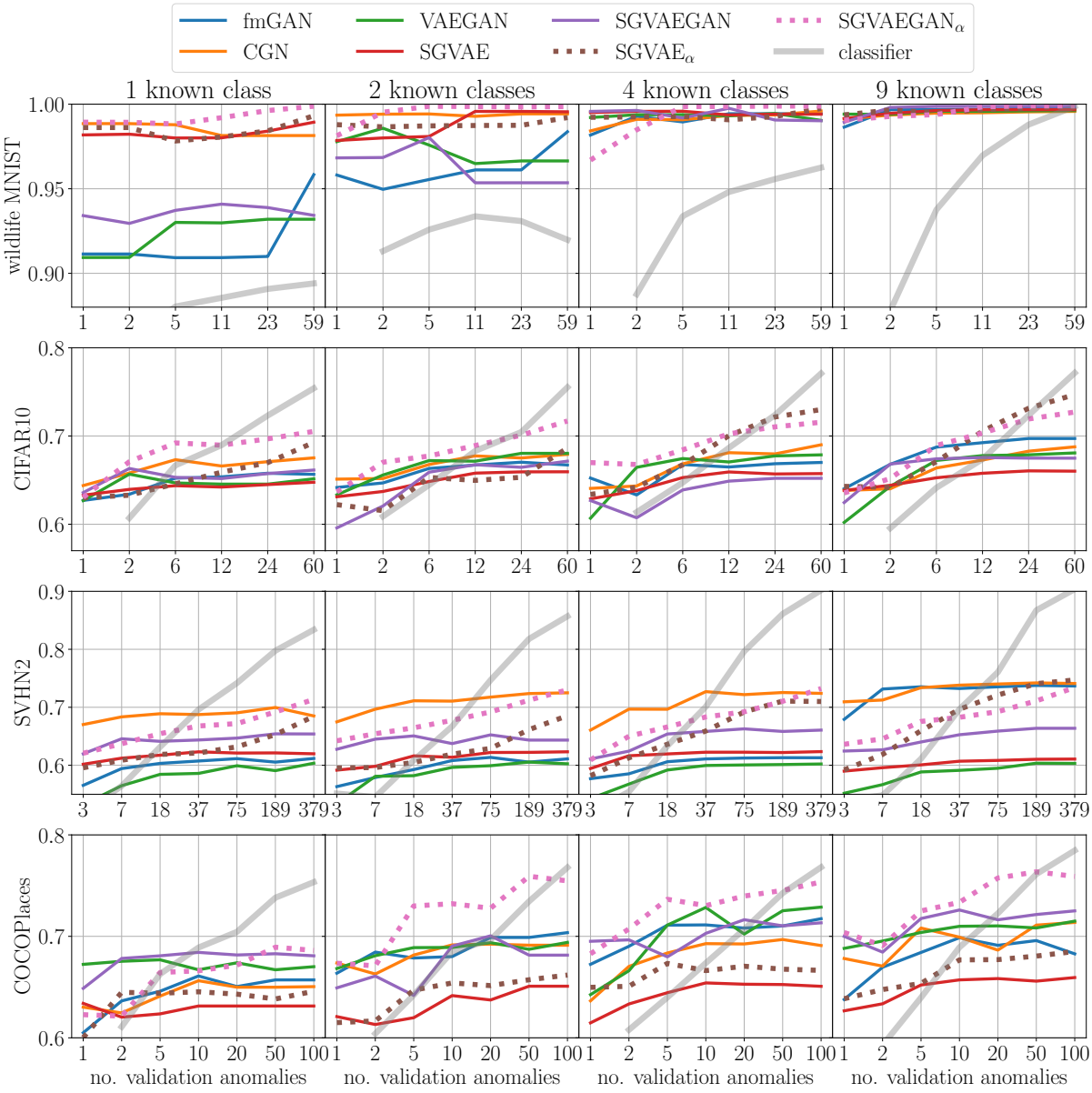


Figure 4.7: Comparison of models on selected image datasets for the leave-one-in experiment. The x-axis covers a changing number of anomalies in the validation dataset, which is used for hyperparameter selection and α computation. The y-axis reports the average test AUC over 10 normal classes. The columns capture experiments with varying availability of validation samples from different anomalous classes, while anomalies of all classes are present in the test set. The number of normal samples in the validation set is the following: wildlife MNIST: 1184, CIFAR10: 1200, SVHN2: 3792, COCOPlaces: 100.

to be anomalous in validation and test datasets. We believe this is a good representation of a semantic anomaly detection problem as well as being a more realistic option since in real-world problems, anomalies may come from many varying distributions. In this regard, we disagree with the authors of [48] which propose the alternative of leave-one-out, stating that in most anomaly detection problems, we want to detect only small perturbations from the target class. However, traditional image benchmarks don't allow for this anyway as the classes are very distinct even in the latter

case. To really test models under this assumption, we have included the MVTec-AD dataset where the normal and anomalous data differ only in detail.

problem	DSVDD	fAnoGAN	fmGAN	VAE	CGN	VAEGAN	SGVAE	SGVAEGAN	SGVAE$_\alpha$	SGVAEGAN$_\alpha$
bottle	0.81	0.97	0.95	0.98	0.90	0.85	0.98	0.83	0.97	0.92
capsule	0.65	0.69	0.67	0.74	0.69	0.58	0.76	0.66	0.80	0.67
nut	0.78	0.72	0.88	0.71	0.82	0.84	0.69	0.78	0.81	0.86
pill	0.64	0.71	0.73	0.73	0.59	0.70	0.77	0.72	0.78	0.73
transistor	0.69	0.77	0.90	0.81	0.88	0.75	0.78	0.79	0.81	0.79
mean rank	8.90	6.20	3.50	4.20	5.50	7.60	4.50	7.00	2.80	4.80

Table 4.3: Aggregated performance in test AUC of models on MVTec-AD problems.

To simulate a broader range of operating conditions, experiments with different anomalous classes in the validation set were conducted. For each normal class, samples from only a limited number of anomalous classes were sampled to the validation set, while the test set contained anomalies from all the classes left out from training. An example with 4 anomalous classes known in validation: training of models was done using class "1" of the SVHN2 dataset. The validation dataset contained normal data from class "1" and anomalies sampled from classes "2", "3", "4" and "5". The testing dataset contained normal samples from class "1" and anomalies sampled from all the remaining classes. This scenario is meant to test the robustness of methods to validation/test discrepancy which might occur in some real-world scenarios. The normal data split is 60/20/20%, 50% of available anomalies are in the test set and the number of validation anomalies is varied to obtain the x-axis in Fig. 4.7. This figure contains the overall comparison of a selection of the best-performing models to improve readability, a complete comparison of all baselines is e.g. in Tab. 4.4.

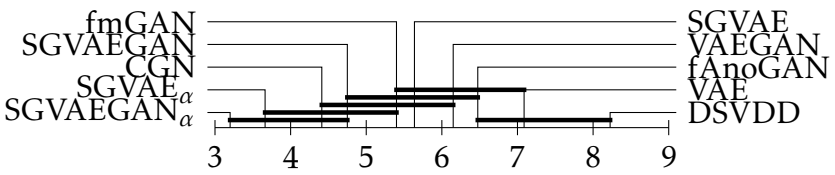


Figure 4.8: A critical difference diagram that shows mean ranks of models from Tab. 4.4. The difference in the performance of 2 models compared on 40 datasets to be statistically significant on level 10% must be greater than the value of the Nemenyi test $CD_{0.1}$=1.95. The thick horizontal lines connect the models with performance differences less than this.

Apart from the SVHN2 dataset, the proposed model outperforms the baselines after already seeing some 10 examples of anomalies, in some cases even less. This is also documented in Tab. 4.4 which contains a more detailed comparison of all baselines

in the case of 2% of labeled anomalies from 4 classes in the validation dataset. The proposed method is also relatively robust to the validation/test anomaly distribution discrepancy. This means that the tuning of the α coefficients focuses on identifying the normal class as well as on the detection of seen anomalies. See Tab. 4.3 for the results on the MVTec-AD subproblems. Here the alternative SGVAE$_\alpha$ trained without a discriminator performs better. A summary comparison of baselines is also presented in the critical difference diagram in Fig. 4.8, where mean ranks of models are compared using the methodology presented in [262]. Of the top four models that are statistically indistinguishable, there are three that were proposed in this paper.

If we assume that authors of other publications use default test/train splits and perform model hyperparameter selection on the same test set on which they report their results, the amount of labeled data that enters the model selection process is significant. Using the default splits on the SVHN2 dataset in the leave-one-in paradigm, the test set contains on average 2603 normal samples and 23429 anomalies (since the class distributions are unequal), while on CIFAR10 there are 900 normal and 9100 anomalous samples. Even if we consider the leave-one-out case, where the numbers of anomalies and normal data are switched, this is still a lot more labeled data than what we report e.g. in Tab. 4.4. Also, by selecting the optimal model on the validation set and reporting on the test set, our experiments provide a fairer comparison of baselines.

A fully supervised classifier trained on the validation dataset is included in the comparison in Fig. 4.7 as well to try to answer a question that is very pertinent for practitioners – if you need at least some labeled anomalies to tune your unsupervised models anyway, what amount of labeled anomalies means that you can train a fully supervised classifier instead? Our comparison shows that this amount is surprisingly low, apart from the (relatively easy) wildlife MNIST dataset. This result is, of course, closely tied to the specific setting of our experiment and should not be extrapolated to other problems without further research.

4.5 Conclusion

We have defined a deep generative model for anomaly detection that uses several independent latent spaces to generate the data sample. The generative model is assumed to generate the normal class, and the flexibility of the latent spaces allows us to question which component of the test sample is anomalous. This concept has been applied to semantic anomaly detection of images for which we developed the SGVAEGAN model with independent latent spaces for shape, foreground, and background textures. The proposed model was tested on synthetic as well as real-world image datasets.

We have shown that the proposed model can detect the source of an anomaly in the sample on synthetic data. Another use of the separate scores is fine-tuning the anomaly score to the type of anomalies that are of interest. This has been achieved by learning the weights of the independent scores for known anomalies in the validation. The weights have been tuned to detect the known anomalies after seeing a few of their examples. Naturally, the performance of the proposed method improves with a growing number of available anomalies in the validation. However, the number of known anomalies is often low, since datasets with a high number of anomalies can be treated by supervised training. A comparison of the proposed method with a supervised classifier trained on the same amount of anomalies reveals that the supervised classifier outperforms any anomaly detector for a relatively low number of anomalous samples.

This sets an upper bound on the meaningful range of problems suitable for anomaly detection methods. We recommend performing such an experiment for every anomaly detection method.

The proposed SGVAEGAN model is a demonstration of the general approach, which can be used with any type of decomposition/disentanglement of the latent space. The requirement for the application of another generative model is that it has to be capable of learning the disentanglement in an unsupervised manner. Further research is required to find such a reliable model.

	class	DSVDD	fAnoGAN	fmGAN	VAE	CGN	VAEGAN	SGVAE	SGVAEGAN	SGVAE$_\alpha$	SGVAEGAN$_\alpha$
wildlife MNIST	0	0.82	0.98	1.00	0.94	1.00	1.00	1.00	1.00	0.99	1.00
	1	0.88	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	2	0.63	0.97	1.00	0.85	0.99	1.00	0.96	0.93	0.99	1.00
	3	0.83	0.99	0.99	0.99	0.99	0.99	1.00	1.00	1.00	1.00
	4	0.91	1.00	0.99	0.99	1.00	1.00	1.00	0.99	1.00	1.00
	5	0.39	0.93	0.99	0.85	0.99	0.99	1.00	1.00	1.00	1.00
	6	0.69	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	7	0.92	0.99	0.99	1.00	0.99	0.99	1.00	1.00	1.00	0.97
	8	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	9	0.71	0.94	0.99	0.94	0.99	1.00	0.99	1.00	0.99	1.00
CIFAR10	airplane	0.72	0.72	0.68	0.68	0.68	0.78	0.78	0.76	0.81	0.87
	automobile	0.63	0.56	0.69	0.62	0.78	0.75	0.46	0.76	0.74	0.78
	bird	0.67	0.67	0.60	0.66	0.60	0.56	0.70	0.47	0.68	0.72
	cat	0.61	0.58	0.59	0.57	0.62	0.60	0.59	0.56	0.69	0.65
	deer	0.71	0.75	0.65	0.74	0.66	0.63	0.74	0.66	0.78	0.77
	dog	0.60	0.59	0.65	0.58	0.64	0.64	0.59	0.63	0.67	0.69
	frog	0.71	0.76	0.71	0.75	0.67	0.69	0.73	0.60	0.78	0.82
	horse	0.61	0.56	0.60	0.54	0.75	0.72	0.67	0.65	0.70	0.68
	ship	0.74	0.80	0.77	0.71	0.74	0.71	0.82	0.72	0.84	0.79
	truck	0.67	0.65	0.79	0.63	0.76	0.67	0.54	0.70	0.73	0.78
SVHN2	0	0.64	0.64	0.68	0.64	0.78	0.64	0.67	0.65	0.71	0.77
	1	0.63	0.60	0.61	0.67	0.76	0.61	0.69	0.70	0.82	0.84
	2	0.61	0.58	0.58	0.62	0.76	0.62	0.62	0.63	0.75	0.74
	3	0.55	0.55	0.59	0.58	0.71	0.54	0.59	0.64	0.64	0.68
	4	0.58	0.58	0.63	0.63	0.80	0.66	0.62	0.69	0.74	0.77
	5	0.56	0.57	0.61	0.58	0.72	0.57	0.60	0.65	0.67	0.69
	6	0.57	0.60	0.58	0.59	0.75	0.62	0.61	0.66	0.73	0.72
	7	0.59	0.58	0.66	0.64	0.82	0.61	0.65	0.69	0.77	0.78
	8	0.58	0.60	0.57	0.58	0.70	0.57	0.59	0.65	0.65	0.71
	9	0.56	0.59	0.62	0.59	0.74	0.56	0.60	0.65	0.68	0.71
COCOPlaces	airplane	0.72	0.74	0.77	0.74	0.68	0.79	0.64	0.81	0.77	0.81
	bird	0.48	0.48	0.64	0.53	0.64	0.61	0.47	0.69	0.66	0.68
	boat	0.65	0.70	0.81	0.76	0.76	0.77	0.77	0.71	0.77	0.81
	bus	0.58	0.82	0.85	0.67	0.83	0.87	0.74	0.70	0.78	0.89
	dog	0.72	0.71	0.71	0.72	0.63	0.64	0.71	0.66	0.71	0.75
	horse	0.58	0.57	0.69	0.65	0.59	0.71	0.65	0.69	0.64	0.74
	motorcycle	0.63	0.65	0.75	0.60	0.77	0.77	0.69	0.73	0.66	0.78
	train	0.64	0.71	0.68	0.65	0.55	0.62	0.69	0.74	0.75	0.77
	truck	0.48	0.65	0.61	0.63	0.63	0.69	0.63	0.70	0.68	0.64
	zebra	0.68	0.63	0.59	0.66	0.84	0.82	0.52	0.79	0.86	0.84
	mean rank	8.23	6.48	5.40	7.09	4.41	6.15	5.64	4.75	3.66	3.20

Table 4.4: Test AUC of models trained on the normal class marked in the first column of the table. The shading highlights the top 3 models. In this experiment, the val-

Apendices

Put the codes here.

A

Mathematical formulations

A.1 Multivariate normal distribution

A random vector $\boldsymbol{x} \in \mathbb{R}^d$ follows the multivariate normal (Gaussian) distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^d$, symmetric and positive–definite covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$, if its probability density is

$$p(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma) = (2\pi)^{-\frac{d}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right), \tag{A.1}$$

where $|\Sigma|$ is the determinant of the covariance matrix. Sometimes, instead of covariance the precision matrix is used $\Lambda = \Sigma^{-1}$ which makes some manipulations easier. Some moments of interest are

$$\mathbb{E}[\boldsymbol{x}] = \boldsymbol{\mu}, \tag{A.2}$$

$$\mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^T\right] = \Sigma + \boldsymbol{\mu}\boldsymbol{\mu}^T, \tag{A.3}$$

$$\mathbb{E}\left[\boldsymbol{x}^T\boldsymbol{x}\right] = \boldsymbol{\mu}^T\boldsymbol{\mu} + \mathrm{Tr}(\Sigma), \tag{A.4}$$

where Tr(.) is the trace operator.

A.2 The Kullback-Leibler divergence

The Kullback-Leibler (KL) divergence is a measure of distance of two probability distributions. For two continuous probability distributions with probability densities $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ defined on the same probability space we define it as

$$D_{\mathrm{KL}}(p(\boldsymbol{x})||q(\boldsymbol{x})) = \int_{\mathcal{X}} p(\boldsymbol{x}) \ln \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x}. \tag{A.5}$$

It is an asymmetric measure, therefore it is not a proper metric. These relations holds

$$p(\boldsymbol{x}) = q(\boldsymbol{x}) \text{ almost everywhere } \iff D_{\mathrm{KL}}(p(\boldsymbol{x})||q(\boldsymbol{x})) = 0, \tag{A.6}$$

$$D_{\mathrm{KL}}(p(\boldsymbol{x})||q(\boldsymbol{x})) \geq 0, \tag{A.7}$$

$$\text{generally } D_{\mathrm{KL}}(p(\boldsymbol{x})||q(\boldsymbol{x})) \neq D_{\mathrm{KL}}(q(\boldsymbol{x})||p(\boldsymbol{x})). \tag{A.8}$$

A.3 Kullback–Leibler divergence of two normal distributions

For the case of two multivariate normal distributions $p_0(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_0, \Sigma_0)$ and $p_1(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_1, \Sigma_1)$ where $\boldsymbol{x} \in \mathbb{R}^d$ the KL divergence is

$$
\begin{aligned}
D_{\mathrm{KL}}(p_0(\boldsymbol{x})\|p_1(\boldsymbol{x})) &= \mathbb{E}_{p_0}[\ln p_0(\boldsymbol{x}) - \ln p_1(\boldsymbol{x})] \\
&= \frac{1}{2}\mathbb{E}_{p_0}\Big[-\ln|\Sigma_0| - (\boldsymbol{x}-\boldsymbol{\mu}_0)^T\Sigma_0^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_0) + \ln|\Sigma_1|\Big] + \\
&\quad \frac{1}{2}\mathbb{E}_{p_0}\Big[(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\Sigma_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)\Big] \\
&= \frac{1}{2}\ln\frac{|\Sigma_1|}{|\Sigma_0|} + \frac{1}{2}\mathbb{E}_{p_0}\Big[-\mathrm{Tr}\Big(\Sigma_0^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_0)(\boldsymbol{x}-\boldsymbol{\mu_0})^T\Big)\Big] + \\
&\quad \frac{1}{2}\mathbb{E}_{p_0}\Big[\mathrm{Tr}\Big(\Sigma_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\Big)\Big] \\
&= \frac{1}{2}\ln\frac{|\Sigma_1|}{|\Sigma_0|} - \frac{1}{2}\mathrm{Tr}\Big(\Sigma_0^{-1}\mathbb{E}_{p_0}\Big[(\boldsymbol{x}-\boldsymbol{\mu}_0)(\boldsymbol{x}-\boldsymbol{\mu}_0)^T\Big]\Big) + \\
&\quad \frac{1}{2}\mathrm{Tr}\Big(\mathbb{E}_{p_0}\Big[\Sigma_1^{-1}(\boldsymbol{x}\boldsymbol{x}^T - 2\boldsymbol{x}\boldsymbol{\mu}_1^T + \boldsymbol{\mu}_1\boldsymbol{\mu}_1^T)\Big]\Big) \\
&= \frac{1}{2}\ln\frac{|\Sigma_1|}{|\Sigma_0|} - \frac{1}{2}\mathrm{Tr}\Big(\Sigma_0^{-1}\Sigma_0\Big) + \frac{1}{2}\mathrm{Tr}\Big(\Sigma_1^{-1}(\Sigma_0 + \boldsymbol{\mu}_0\boldsymbol{\mu}_0^T - 2\boldsymbol{\mu}_0\boldsymbol{\mu}_1^T - \boldsymbol{\mu}_1\boldsymbol{\mu}_1^T)\Big) \\
&= \frac{1}{2}\ln\frac{|\Sigma_1|}{|\Sigma_0|} - \frac{1}{2}d + \frac{1}{2}\mathrm{Tr}\Big(\Sigma_1^{-1}\Sigma_0\Big) + \frac{1}{2}\mathrm{Tr}\Big(\Sigma_1^{-1}(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)^T(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)\Big) \\
&= \frac{1}{2}\left(\ln\frac{|\Sigma_1|}{|\Sigma_0|} - d + \mathrm{Tr}\Big(\Sigma_1^{-1}\Sigma_0\Big) + (\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)^T\Sigma_1^{-1}(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)\right). \qquad \text{(A.9)}
\end{aligned}
$$

where we have used the common trick using the trace of a scalar, the relation $\mathrm{Tr}(ABC) = \mathrm{Tr}(BCA) = \mathrm{Tr}(CAB)$ and the linearity of the trace operator. For us, a special case is of high interest – one where we have a normal distribution with diagonal covariance and a standard normal distribution, that is $p_0(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}), \mathrm{diag}(\boldsymbol{\sigma}^2), \boldsymbol{\sigma}^2 \in \mathbb{R}^d$ and $p_1(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|0, \boldsymbol{I})$. In that case

$$
D_{\mathrm{KL}}(p_0(\boldsymbol{x})\|\mathcal{N}(\boldsymbol{x}|0,1)) = \frac{1}{2}\left(\ln\frac{1}{\prod_{i=1}^d \boldsymbol{\sigma}_i^2} - d + \sum_{i=1}^d \boldsymbol{\sigma}_i^2 + \boldsymbol{\mu}^T\boldsymbol{\mu}\right) = \frac{1}{2}\sum_{i=1}^d \boldsymbol{\sigma}_i^2 - 1 - \ln\boldsymbol{\sigma}_i^2 + \boldsymbol{\mu}_i^2. \qquad \text{(A.10)}
$$

B

Datasets

Wildlife MNIST is an artificial dataset based on standard MNIST [216] which is also used in [243]. For each MNIST digit, a background and foreground texture is sampled from a texture dataset described in [244] to create a colored version. Two versions of this dataset were created - *mixed* and *non-mixed*. In the mixed version, the background and foreground for each digit are sampled randomly from 10 background and foreground texture classes to create a very diverse set of images. In the non-mixed version, the background and foreground class is kept the same for all basic MNIST digits from the same class, resulting in a less diverse dataset. For examples of the non-mixed version, see the top row of Fig. 4.1. Each version contains 60000 samples of 32x32 pixel RGB images with three factors of variation.

COCOPlaces is a dataset created similarly to wildlife MNIST. 10 classes of objects from the COCO dataset [263] were combined with 10 background classes from the Places dataset [264]. Again, mixed and non-mixed variants were created. Because the object shape and texture cannot be separated in this case, it represents a problem with těo factors of variation that is slightly more realistic than the previous one. It is also much harder, as the object shapes are very distinct, sometimes the foreground object is only partially visible and the background itself may sometimes contain some other objects as well. A total of 5000 64x64 pixel RGB images were generated for each variant. For sample images of all classes, see Fig. 4.4.

SVHN2 is a well-known benchmark dataset [219] containing images of house numbers. In this cropped version of the dataset, the target digit is centered in the middle of the picture. However, there may be partial digits adjacent to it, which makes the problem harder. Also, there are no available annotations for the background/foreground factor and we use it mainly to compare to other baseline methods. There are 10 classes for a total of 99288 32x32 pixel RGB images. The distribution of classes is uneven and ranges from 6 to 19 thousand samples.

CIFAR10 is another classical dataset of images of 10 classes of different objects or animals often used for validation of anomaly detectors [38, 99, 91]. For an overview of the dataset see [218]. There is a total of 60000 32x32 pixel RGB images.

MVTec-AD is an industrial dataset[215] that captures several different problems of identifying faulty or damaged objects. We have considered only the problems where there is a distinct object in the foreground, such as *bottle* or *metal nut*. The

dataset	alias	dim	anom	normal
ANNthyroid	ann	21	534	6665
Arrhythmia	arr	275	206	245
HAR	har	561	1944	8355
HTRU2	htr	8	1638	16257
KDD99 (10%)	kdd	118	396742	97276
Mammography	mam	6	260	10921
Seismic	sei	24	170	2412
Spambase	spm	57	1812	2786
Abalone	aba	10	50	2151
Blood Transfusion	blt	4	16	382
Breast Cancer Wisconsin	bcw	30	206	356
Breast Tissue	bts	9	22	65
Cardiotocography	crd	27	228	1830
Ecoli	eco	7	108	205
Glass	gls	10	94	112
Haberman	hab	3	14	225
Ionosphere	ion	33	122	225
Iris	irs	4	46	100
Isolet	iso	617	3300	4496
Letter Recognition	ltr	617	3600	4196
Libras	lbr	90	142	215
Magic Telescope	mgc	10	3882	12331
Miniboone	mnb	50	23922	93565
Multiple Features	mlt	649	800	1200
PageBlocks	pgb	10	384	4911
Parkinsons	prk	22	44	146
Pendigits	pen	16	5384	5537
Pima Indians	pim	8	176	500
Sonar	snr	60	96	110
Spect Heart	sph	44	52	211
Statlog Satimage	sat	36	2630	3592
Statlog Segment	seg	18	938	1320
Statlog Shuttle	sht	8	28	57767
Statlog Vehicle	vhc	18	132	627
Synthetic Control Chart	scc	60	200	400
Wall Following Robot	wrb	24	2220	2921
Waveform-1	wf1	21	1482	3302
Waveform-2	wf2	21	1472	3302
Wine	wne	13	70	106
Yeast	yst	8	390	751

Table B.1: Basic statistics of the tabular dataset designed for anomaly detection (above split) and multi-class datasets (bellow split).

dataset	alias	dim	anom	normal
MNIST-C	mnistc	28x28x1	70000	70000
MVTec-AD - wood	wood	1024x1024x3	60	266
MVTec-AD - grid	grid	1024x1024x3	57	285
MVTec-AD - transistor	transistor	1024x1024x3	40	273
CIFAR10	cifar10	32x32x3	54000	6000
FashionMNIST	fmnist	28x28x1	63000	7000
MNIST	mnist	28x28x1	63686	6312
SVHN2	svhn2	32x32x3	80327	18960

Table B.2: Basic statistics of image datasets designed for anomaly detection (above split) and multi-class datasets (below split).

individual problems are smaller - on average about 200 normal and 100 anomalous samples. The images used in our experiments are downscaled to 128x128 pixels.

Bibliography

[1] JWL Glaisher. On the rejection of discordant observations. *Monthly Notices of the Royal Astronomical Society*, 33:391–402, 1873.

[2] Francis Ysidro Edgeworth. On discordant observations. *The london, edinburgh, and dublin philosophical magazine and journal of science*, 23(143):364–375, 1887.

[3] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013.

[4] Juan Vanerio and Pedro Casas. Ensemble-learning approaches for network security and anomaly detection. In *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, pages 1–6, 2017.

[5] Yang Xin, Lingshuang Kong, Zhi Liu, Yuling Chen, Yanmiao Li, Hongliang Zhu, Mingcheng Gao, Haixia Hou, and Chunhua Wang. Machine learning and deep learning methods for cybersecurity. *Ieee access*, 6:35365–35381, 2018.

[6] Richard J Bolton and David J Hand. Statistical fraud detection: A review. *Statistical science*, 17(3):235–255, 2002.

[7] Johan Perols. Financial statement fraud detection: An analysis of statistical and machine learning algorithms. *Auditing: A Journal of Practice & Theory*, 30(2):19–50, 2011.

[8] Mohiuddin Ahmed, Abdun Naser Mahmood, and Md Rafiqul Islam. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, 55:278–288, 2016.

[9] Lionel Tarassenko, Paul Hayton, Nicholas Cerneaz, and Michael Brady. Novelty detection for the identification of masses in mammograms. *International Conference on Artificial Neural Networks*, 1995.

[10] Weng-Keen Wong, Andrew W Moore, Gregory F Cooper, and Michael M Wagner. Bayesian network anomaly pattern detection for disease outbreaks. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 808–815, 2003.

[11] Dimitris K Iakovidis, Spiros V Georgakopoulos, Michael Vasilakakis, Anastasios Koulaouzidis, and Vassilis P Plagianakos. Detecting and locating gastrointestinal anomalies using deep learning and iterative cluster unification. *IEEE transactions on medical imaging*, 37(10):2196–2210, 2018.

[12] Joey Tianyi Zhou, Jiawei Du, Hongyuan Zhu, Xi Peng, Yong Liu, and Rick Siow Mong Goh. Anomalynet: An anomaly detection network for video surveillance. *IEEE Transactions on Information Forensics and Security*, 14(10):2537–2550, 2019.

[13] Mohamad Mahmoudi, Ahmed Aziz Ezzat, and Alaa Elwany. Layerwise anomaly detection in laser powder-bed fusion metal additive manufacturing. *Journal of Manufacturing Science and Engineering*, 141(3), 2019.

[14] Mingliang Bai, Jinfu Liu, Jinhua Chai, Xinyu Zhao, and Daren Yu. Anomaly detection of gas turbines based on normal pattern extraction. *Applied Thermal Engineering*, 166:114664, 2020.

[15] Yeji Choi, Hyunki Lim, Heeseung Choi, and Ig-Jae Kim. Gan-based anomaly detection and localization of multivariate time series data for power plant. In *2020 IEEE international conference on big data and smart computing (BigComp)*, pages 71–74. IEEE, 2020.

[16] Pavlos Protopapas, JM Giammarco, L Faccioli, MF Struble, Rahul Dave, and Charles Alcock. Finding outlier light curves in catalogues of periodic variable stars. *Monthly Notices of the Royal Astronomical Society*, 369(2):677–696, 2006.

[17] Vít Škvára, Tomáš Pevný, Václav Šmídl, Jakub Seidl, Aleš Havránek, and David Tskhakaya. Detection of alfvén eigenmodes on COMPASS with generative neural networks. *Fusion Science and Technology*, 76(8):962–971, 2020.

[18] Tudor I Oprea. Chemical space navigation in lead discovery. *Current opinion in chemical biology*, 6(3):384–389, 2002.

[19] Katherine Fraser, Samuel Homiller, Rashmish K Mishra, Bryan Ostdiek, and Matthew D Schwartz. Challenges for unsupervised anomaly detection in particle physics. *Journal of High Energy Physics*, 2022(3):1–31, 2022.

[20] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[21] Stefan Harmeling, Guido Dornhege, David Tax, Frank Meinecke, and Klaus-Robert Müller. From outliers to prototypes: ordering data. *Neurocomputing*, 69(13-15):1608–1618, 2006.

[22] Vijay Mahadevan, Weixin Li, Viral Bhalodia, and Nuno Vasconcelos. Anomaly detection in crowded scenes. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1975–1981. IEEE, 2010.

[23] 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.

[24] Tomáš Pevný. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102(2):275–304, 2016.

[25] Longin Jan Latecki, Aleksandar Lazarevic, and Dragoljub Pokrajac. Outlier detection with kernel density functions. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 61–75. Springer, 2007.

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

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

[28] Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.

[29] Aleksandar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 25–36. SIAM, 2003.

[30] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[31] Guilherme O Campos, Arthur Zimek, Jörg Sander, Ricardo JGB Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E Houle. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 30(4):891–927, 2016.

[32] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *arXiv:2009.11732 [cs]*, 2020.

[33] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[34] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[35] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[36] Edwin M. Knorr and Raymond T. Ng. Algorithms for mining distance-based outliers in large datasets. In Ashish Gupta, Oded Shmueli, and Jennifer Widom, editors, *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 392–403. Morgan Kaufmann, 1998.

[37] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004.

[38] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402, 2018.

[39] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[40] Vic Barnett and Toby Lewis. *Outliers in statistical data*. Wiley, 1974.

[41] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[42] Mohiuddin Ahmed. Collective anomaly detection techniques for network traffic analysis. *Annals of data science*, 5(4):497–512, 2018.

[43] Gwenolé Quellec, Mathieu Lamard, Michel Cozic, Gouenou Coatrieux, and Guy Cazuguel. Multiple-instance learning for anomaly detection in digital mammography. *IEEE transactions on medical imaging*, 35(7):1604–1614, 2016.

[44] Boyang Wan, Yuming Fang, Xue Xia, and Jiajie Mei. Weakly supervised video anomaly detection via center-guided discriminative learning. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2020.

[45] Marc-André Carbonneau, Veronika Cheplygina, Eric Granger, and Ghyslain Gagnon. Multiple instance learning: A survey of problem characteristics and applications. *Pattern Recognition*, 77:329–353, 2018.

[46] Ruey S Tsay, Daniel Pena, and Alan E Pankratz. Outliers in multivariate time series. *Biometrika*, 87(4):789–804, 2000.

[47] Sanjay Chawla and Pei Sun. Slom: a new measure for local spatial outliers. *Knowledge and Information Systems*, 9(4):412–429, 2006.

[48] Faruk Ahmed and Aaron Courville. Detecting semantic anomalies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3154–3162, 2020.

[49] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[50] Muhammad Qasim Ali, Ehab Al-Shaer, Hassan Khan, and Syed Ali Khayam. Automated anomaly detector adaptation using adaptive threshold tuning. *ACM Transactions on Information and System Security (TISSEC)*, 15(4):1–30, 2013.

[51] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer, 2006.

[52] David J Hand and Robert J Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, 45(2):171–186, 2001.

[53] Stéphan Clémençon and Jérémie Jakubowicz. Scoring anomalies: a m-estimation formulation. In *Artificial Intelligence and Statistics*, pages 659–667, 2013.

[54] Christian P Robert, George Casella, Christian P Robert, and George Casella. Monte carlo integration. *Monte Carlo statistical methods*, pages 71–138, 1999.

[55] Nicolas Goix. How to evaluate the quality of unsupervised anomaly detection algorithms? *arXiv preprint arXiv:1607.01152*, 2016.

[56] Nour Moustafa, Jiankun Hu, and Jill Slay. A holistic review of network anomaly detection systems: A comprehensive survey. *Journal of Network and Computer Applications*, 128:33–55, 2019.

[57] Donghwoon Kwon, Hyunjoo Kim, Jinoh Kim, Sang C Suh, Ikkyun Kim, and Kuinam J Kim. A survey of deep learning-based network anomaly detection. *Cluster Computing*, pages 1–13, 2019.

[58] Gilberto Fernandes, Joel JPC Rodrigues, Luiz Fernando Carvalho, Jalal F Al-Muhtadi, and Mario Lemes Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, 70(3):447–489, 2019.

[59] Hongzhi Wang, Mohamed Jaward Bah, and Mohamed Hammad. Progress in outlier detection techniques: A survey. *IEEE Access*, 7:107964–108000, 2019.

[60] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv:1901.03407 [cs]*, 2019.

[61] Frank E Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

[62] Vic Barnett, Toby Lewis, et al. *Outliers in statistical data*, volume 3. Wiley New York, 1994.

[63] Jorma Laurikkala, Martti Juhola, Erna Kentala, N Lavrac, S Miksch, and B Kavsek. Informal identification of outliers in medical data. In *Fifth international workshop on intelligent data analysis in medicine and pharmacology*, volume 1, pages 20–24. Citeseer, 2000.

[64] Stephen Roberts and Lionel Tarassenko. A probabilistic resource allocating network for novelty detection. *Neural Computation*, 6(2):270–284, 1994.

[65] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[66] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[67] Stephen W Hoare, David Asbridge, and Paul CW Beatty. On-line novelty detection for artefact identification in automatic anaesthesia record keeping. *Medical engineering & physics*, 24(10):673–681, 2002.

[68] Dit-Yan Yeung and Yuxin Ding. Host-based intrusion detection using dynamic and static behavioral models. *Pattern recognition*, 36(1):229–243, 2003.

[69] Xiaoqiang Zhang, Pingzhi Fan, and Zhongliang Zhu. A new anomaly detection method based on hierarchical hmm. In *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 249–252. IEEE, 2003.

[70] Dit-Yan Yeung and Calvin Chow. Parzen-window network intrusion detectors. In *2002 International Conference on Pattern Recognition*, volume 4, pages 385–388. IEEE, 2002.

[71] Markus Goldstein and Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: Poster and Demo Track*, pages 59–63, 2012.

[72] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.

[73] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[74] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[75] Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 693–700. JMLR Workshop and Conference Proceedings, 2010.

[76] Shuangfei Zhai, Yu Cheng, Weining Lu, and Zhongfei Zhang. Deep structured energy based models for anomaly detection. *arXiv preprint arXiv:1605.07717*, 2016.

[77] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *ACM Sigmod Record*, volume 29, pages 427–438. ACM, 2000.

[78] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[79] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[80] Guansong Pang, Longbing Cao, Ling Chen, and Huan Liu. Learning Representations of Ultrahigh-dimensional Data for Random Distance-based Outlier Detection. *arXiv:1806.04808 [cs, stat]*, June 2018.

[81] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[82] Jian Tang, Zhixiang Chen, Ada Wai-Chee Fu, and David W Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 535–548. Springer, 2002.

[83] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003.

[84] Parikshit Gopalan, Vatsal Sharan, and Udi Wieder. PIDForest: Anomaly Detection via Partial Identification. *arXiv:1912.03582 [cs]*, 2019.

[85] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 444–452, 2008.

[86] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[87] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

[88] Stefan Van Aelst and Peter Rousseeuw. Minimum volume ellipsoid. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1):71–82, 2009.

[89] Nico Görnitz, Luiz Alberto Lima, Klaus-Robert Müller, Marius Kloft, and Shinichi Nakajima. Support vector data descriptions and k-means clustering: one class? *IEEE transactions on neural networks and learning systems*, 29(9):3994–4006, 2017.

[90] Alireza Ghasemi, Hamid R Rabiee, Mohammad Taghi Manzuri, and Mohammad Hossein Rohban. A bayesian approach to the data description problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 907–913, 2012.

[91] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Anomaly detection using one-class neural networks. *arXiv:1802.06360 [cs]*, 2019.

[92] 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 Recognition*, 58:121–134, 2016.

[93] Zahra Ghafoori and Christopher Leckie. Deep multi-sphere support vector data description. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 109–117. SIAM, 2020.

[94] Ingo Steinwart, Don Hush, and Clint Scovel. A classification framework for anomaly detection. *Journal of Machine Learning Research*, 6(Feb):211–232, 2005.

[95] Wei Fan, Matthew Miller, Sal Stolfo, Wenke Lee, and Phil Chan. Using artificial anomalies to detect unknown and known network intrusions. *Knowledge and Information Systems*, 6:507–527, 2004.

[96] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep Anomaly Detection with Outlier Exposure. *arXiv:1812.04606 [cs, stat]*, January 2019.

[97] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. *Advances in neural information processing systems*, 32, 2019.

[98] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1):44–53, 2018.

[99] Lukas Ruff, Robert A Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. Deep semi-supervised anomaly detection. *arXiv preprint arXiv:1906.02694*, 2019.

[100] Philipp Liznerski, Lukas Ruff, Robert A Vandermeulen, Billy Joe Franks, Marius Kloft, and Klaus-Robert Müller. Explainable deep one-class classification. *arXiv preprint arXiv:2007.01760*, 2020.

[101] Naoki Abe, Bianca Zadrozny, and John Langford. Outlier detection by active learning. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 504–509, 2006.

[102] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1920–1929, 2019.

[103] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[104] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations. *Advances in neural information processing systems*, 31, 2018.

[105] Liron Bergman and Yedid Hoshen. Classification-based anomaly detection for general data. *arXiv preprint arXiv:2005.02359*, 2020.

[106] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. A novel anomaly detection scheme based on principal component classifier. Technical report, MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL AND COMPUTER ENGINEERING, 2003.

[107] Charu C Aggarwal. Outlier analysis. In *Data mining*, pages 237–263. Springer, 2015.

[108] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[109] Yingchao Xiao, Huangang Wang, Wenli Xu, and Junwu Zhou. L1 norm based kpca for novelty detection. *Pattern Recognition*, 46(1):389–396, 2013.

[110] Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

[111] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs]*, 2017.

[112] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[113] Paul J Werbos. Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization*, pages 762–770. Springer, 1982.

[114] Jun Deng, Zixing Zhang, Erik Marchi, and Björn Schuller. Sparse autoencoder-based feature transfer learning for speech emotion recognition. In *2013 humaine association conference on affective computing and intelligent interaction*, pages 511–516. IEEE, 2013.

[115] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori. Speech enhancement based on deep denoising autoencoder. In *Interspeech*, volume 2013, pages 436–440, 2013.

[116] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, page 4. ACM, 2014.

[117] Benjamin Berry Thompson, Robert J Marks, Jai J Choi, Mohamed A El-Sharkawi, Ming-Yuh Huang, and Carl Bunje. Implicit learning in autoencoder novelty assessment. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 3, pages 2878–2883. IEEE, 2002.

[118] Tsatsral Amarbayasgalan, Bilguun Jargalsaikhan, and Keun Ho Ryu. Unsupervised novelty detection using deep autoencoders with density based clustering. *Applied Sciences*, 8(9):1468, 2018.

[119] Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14, 2001.

[120] Christopher Bishop and Julia Lasserre. Generative or discriminative? getting the best of both worlds. *Bayesian statistics*, 8(3):3–24, 2007.

[121] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[122] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[123] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[124] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[125] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

[126] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *SNU Data Mining Center, Tech. Rep.*, 2015.

[127] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the 2018 World Wide Web Conference*, pages 187–196, 2018.

[128] Huan-gang Wang, Xin Li, and Tao Zhang. Generative adversarial network based novelty detection usingminimized reconstruction error. *Frontiers of Information Technology & Electronic Engineering*, 19(1):116–125, 2018.

[129] Pramuditha Perera, Ramesh Nallapati, and Bing Xiang. Ocgan: One-class novelty detection using GANs with constrained latent representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2898–2906, 2019.

[130] Masataka Yamaguchi, Yuma Koizumi, and Noboru Harada. Adaflow: Domain-adaptive density estimator with application to anomaly detection and unpaired cross-domain translation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3647–3651. IEEE, 2019.

[131] Maximilian Schmidt and Marko Simic. Normalizing flows for novelty detection in industrial time series data. *arXiv:1906.06904 [cs, stat]*, June 2019.

[132] William Lotter, Gabriel Kreiman, and David Cox. Unsupervised learning of visual structure using predictive generative networks. *arXiv preprint arXiv:1511.06380*, 2015.

[133] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[134] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

[135] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with

conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[136] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[137] Sang-gil Lee, Wei Ping, Boris Ginsburg, Bryan Catanzaro, and Sungroh Yoon. Bigvgan: A universal neural vocoder with large-scale training. *arXiv preprint arXiv:2206.04658*, 2022.

[138] Michael Maschler, Shmuel Zamir, and Eilon Solan. *Game theory*. Cambridge University Press, 2020.

[139] Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, and Jieping Ye. A review on generative adversarial networks: Algorithms, theory, and applications. *IEEE transactions on knowledge and data engineering*, 2021.

[140] Ian Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv:1701.00160 [cs]*, 2017.

[141] Yongjun Hong, Uiwon Hwang, Jaeyoon Yoo, and Sungroh Yoon. How generative adversarial networks and their variants work: An overview. *ACM Computing Surveys (CSUR)*, 52(1):1–43, 2019.

[142] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Georg Langs, and Ursula Schmidt-Erfurth. F-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks. *Medical Image Analysis*, 54:30–44, May 2019.

[143] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[144] Ilyass Haloui, Jayant Sen Gupta, and Vincent Feuillard. Anomaly detection with wasserstein gan. *arXiv preprint arXiv:1812.02463*, 2018.

[145] Mark Kliger and Shachar Fleishman. Novelty detection with GAN. *arXiv:1802.10560 [cs]*, 2018.

[146] Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. Efficient GAN-based anomaly detection. *arXiv:1802.06222 [cs]*, 2019.

[147] Yezheng Liu, Zhe Li, Chong Zhou, Yuanchun Jiang, Jianshan Sun, Meng Wang, and Xiangnan He. Generative adversarial active learning for unsupervised outlier detection. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[148] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv:1605.09782 [cs]*, 2017.

[149] Valentin Leveau and Alexis Joly. Adversarial autoencoders for novelty detection. Research report, Inria - Sophia Antipolis, February 2017.

[150] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[151] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[152] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[153] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

[154] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015.

[155] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *European Conference on Computer Vision*, pages 835–851. Springer, 2016.

[156] Maximilian Sölch, Justin Bayer, Marvin Ludersdorfer, and Patrick van der Smagt. Variational inference for on-line anomaly detection in high-dimensional time series. *arXiv preprint arXiv:1602.07109*, 2016.

[157] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017.

[158] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *arXiv:1706.02262 [cs]*, 2018.

[159] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv:1711.01558 [stat]*, 2019.

[160] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv:1511.05644 [cs]*, 2016.

[161] Yuchen Pu, Weiyao Wang, Ricardo Henao, Liqun Chen, Zhe Gan, Chunyuan Li, and Lawrence Carin. Adversarial symmetric variational autoencoder. In *Advances in Neural Information Processing Systems*, pages 4330–4339, 2017.

[162] Richard A Levine and George Casella. Implementations of the monte carlo em algorithm. *Journal of Computational and Graphical Statistics*, 10(3):422–439, 2001.

[163] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010.

[164] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2391–2400. JMLR. org, 2017.

[165] Clark R Givens and Rae Michael Shortt. A class of wasserstein metrics for probability distributions. *Michigan Mathematical Journal*, 31(2):231–240, 1984.

[166] V Barranco-Lopez, P Luque-Escamilla, and R Roman-Roldan. Texture edge detection by using the jenssen-shannon divergence. In *ISITA'94: International Symposium on Information Theory & Its Applications 1994; Proceedings: International Symposium on Information Theory & Its Applications 1994; Proceedings*, pages 1027–1030. Institution of Engineers, Australia Barton, ACT, 1994.

[167] Jakub Tomczak and Max Welling. VAE with a VampPrior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223, 2018.

[168] Stanislav Pidhorskyi, Ranya Almohsen, and Gianfranco Doretto. Generative probabilistic novelty detection with adversarial autoencoders. In *Advances in Neural Information Processing Systems*, pages 6822–6833, 2018.

[169] Václav Šmídl, Jan Bím, and Tomáš Pevný. Anomaly scores for generative models. *arXiv:1905.11890 [stat]*, 2019.

[170] Bin Dai, Yu Wang, John Aston, Gang Hua, and David Wipf. Hidden talents of the variational autoencoder. *arXiv preprint arXiv:1706.05148*, 2017.

[171] Joao Pereira and Margarida Silveira. Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1275–1282. IEEE, 2018.

[172] Xuhong Wang, Ying Du, Shijie Lin, Ping Cui, Yuntian Shen, and Yupu Yang. advae: A self-adversarial variational autoencoder with gaussian anomaly prior knowledge for anomaly detection. *Knowledge-Based Systems*, 190:105187, 2020.

[173] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[174] Xiaoran Chen and Ender Konukoglu. Unsupervised detection of lesions in brain MRI using constrained adversarial auto-encoders. *arXiv:1806.04972 [cs]*, 2018.

[175] Asimenia Dimokranitou. *Adversarial autoencoders for anomalous event detection in images*. PhD thesis, Purdue University, 2017.

[176] Samet Akcay, Amir Atapour-Abarghouei, and Toby P. Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Asian conference on computer vision*, pages 622–637. Springer, 2018.

[177] Hyojung Ahn, Dawoon Jung, and Han-Lim Choi. Deep generative models-based anomaly detection for spacecraft control systems. *Sensors*, 20:1991, 04 2020.

[178] Samet Akçay, Amir Atapour-Abarghouei, and Toby P Breckon. Skip-ganomaly: Skip connected and adversarially trained encoder-decoder anomaly detection. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[179] Tolga Ergen and Suleyman Serdar Kozat. Unsupervised anomaly detection with LSTM neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 31(8):3127–3141, Aug 2020.

[180] Rong Yao, Chongdang Liu, Linxuan Zhang, and Peng Peng. Unsupervised Anomaly Detection Using Variational Auto-Encoder based Feature Extraction. In *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 1–7, San Francisco, CA, USA, June 2019. IEEE.

[181] Bin Dai and David Wipf. Diagnosing and enhancing vae models. *arXiv:1903.05789 [cs]*, 2019.

[182] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[183] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

[184] Xue-Bo Jin, Wen-Tao Gong, Jian-Lei Kong, Yu-Ting Bai, and Ting-Li Su. Pfvae: a planar flow-based variational auto-encoder prediction model for time series data. *Mathematics*, 10(4):610, 2022.

[185] Madson L. D. Dias, César Lincoln C. Mattos, Ticiana L. C. da Silva, José Antônio F. de Macedo, and Wellington C. P. Silva. Anomaly Detection in Trajectory Data with Normalizing Flows. *arXiv:2004.05958 [cs, stat]*, April 2020.

[186] Tomas Pevny, Vasek Smidl, Martin Trapp, Ondrej Polacek, and Tomas Oberhuber. Sum-product-transform networks: Exploiting symmetries using invertible transformations. *arXiv:2005.01297 [stat]*, 2020.

[187] Marco Rudolph, Tom Wehrbein, Bodo Rosenhahn, and Bastian Wandt. Fully convolutional cross-scale-flows for image-based defect detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1088–1097, 2022.

[188] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv:1605.08803 [cs]*, 2017.

[189] Durk P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224, 2018.

[190] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked Autoregressive Flow for Density Estimation. *arXiv:1705.07057 [cs, stat]*, June 2018.

[191] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. *arXiv:1912.02762 [cs, stat]*, December 2019.

[192] Ivan Kobyzev, Simon J. D. Prince, and Marcus A. Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[193] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Why normalizing flows fail to detect out-of-distribution data. *arXiv:2006.08545 [stat]*, 2020.

[194] Madson LD Dias, César Lincoln C Mattos, Ticiana LC da Silva, José Antônio F de Macedo, and Wellington CP Silva. Anomaly detection in trajectory data with normalizing flows. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[195] Marco Rudolph, Bastian Wandt, and Bodo Rosenhahn. Same same but differnet: Semi-supervised defect detection with normalizing flows. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1907–1916, 2021.

[196] Denis Gudovskiy, Shun Ishizaka, and Kazuki Kozuka. Cflow-ad: Real-time unsupervised anomaly detection with localization via conditional normalizing flows. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 98–107, 2022.

[197] Ian Bird. Computing for the large hadron collider. *Annual Review of Nuclear and Particle Science*, 61:99–118, 2011.

[198] Nicholas M Ball and Robert J Brunner. Data mining and machine learning in astronomy. *International Journal of Modern Physics D*, 19(07):1049–1106, 2010.

[199] Norbert Holtkamp, ITER Project Team, et al. An overview of the iter project. *Fusion Engineering and Design*, 82(5-14):427–434, 2007.

[200] R Panek, J Adámek, M Aftanas, P Bílková, P Böhm, F Brochard, P Cahyna, J Cavalier, R Dejarnac, M Dimitrova, et al. Status of the compass tokamak and characterization of the first h-mode. *Plasma Physics and Controlled Fusion*, 58(1):014015, 2015.

[201] T Markovic, J Seidl, A Melnikov, P Haccek, J Havlicek, A Havranek, M Hron, O Hronova, M Imrisek, F Janky, et al. Alfvén-wave character oscillations in tokamak COMPASS plasma. In *Proc. 42nd EPS Conf. Plasma Physics*, pages 22–26, 2015.

[202] AV Melnikov, T Markovic, LG Eliseev, J Adámek, M Aftanas, P Bilkova, P Boehm, M Gryaznevich, M Imrisek, SE Lysenko, et al. Quasicoherent modes on the COMPASS tokamak. *Plasma Physics and Controlled Fusion*, 57(6):065006, 2015.

[203] T Markovic, A Melnikov, J Seidl, L Eliseev, J Havlicek, A Havránek, M Hron, M Imríšek, and K Kovarík. Alfvén-character oscillations in ohmic plasmas observed on the COMPASS tokamak. In *Proc. 44th EPS Conf. on Plasma Phys*, volume 5, 2017.

[204] RR Mett and SM Mahajan. Kinetic theory of toroidicity-induced alfvén eigenmodes. *Physics of Fluids B: Plasma Physics*, 4(9):2885–2893, 1992.

[205] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[206] Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann L Cun. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2007.

[207] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research*, 15(1):3133–3181, 2014.

[208] B Ravi Kiran, Dilip Mathew Thomas, and Ranjith Parakkal. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *Journal of Imaging*, 4(2):36, 2018.

[209] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton van den Hengel. Deep learning for anomaly detection: A review. *arXiv:2007.02500 [cs]*, 2020.

[210] Vít Škvára, Tomáš Pevný, and Václav Šmídl. Are generative deep models for novelty detection truly better? *arXiv:1807.05027 [cs]*, 2018.

[211] Vít Škvára, Jan Francœu, Matěj Zorek, Tomáš Pevný, and Václav Šmídl. Comparison of anomaly detectors: Context matters. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[212] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[213] Andrew F Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. Systematic construction of anomaly detection benchmarks from real data. In *Proceedings of the ACM SIGKDD workshop on outlier detection and description*, pages 16–21. ACM, 2013.

[214] Norman Mu and Justin Gilmer. MNIST-C: A Robustness Benchmark for Computer Vision. *arXiv:1906.02337 [cs]*, June 2019.

[215] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9592–9600, 2019.

[216] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[217] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747 [cs]*, 2017.

[218] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[219] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[220] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

[221] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[222] Lucas Deecke, Robert Vandermeulen, Lukas Ruff, Stephan Mandt, and Marius Kloft. Image anomaly detection with generative adversarial networks. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 3–17. Springer, 2018.

[223] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114 [stat]*, 2014.

[224] Raghavendra Chalapathy, Edward Toth, and Sanjay Chawla. Group Anomaly Detection using Deep Generative Models. *arXiv:1804.04876 [cs]*, April 2018.

[225] Tomoharu Iwata and Yuki Yamanaka. Supervised Anomaly Detection based on Deep Autoregressive Density Estimators. *arXiv:1904.06034 [cs, stat]*, April 2019.

[226] Hyunsun Choi, Eric Jang, and Alexander A. Alemi. WAIC, but Why? Generative Ensembles for Robust Anomaly Detection. *arXiv:1810.01392 [cs, stat]*, May 2019.

[227] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do Deep Generative Models Know What They Don't Know? *arXiv:1810.09136 [cs, stat]*, February 2019.

[228] Andrew Gordon Wilson. The case for bayesian deep learning. *arXiv:2001.10995 [cs]*, 2020.

[229] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470, 2007.

[230] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. Draem-a discriminatively trained reconstruction embedding for surface anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8330–8339, 2021.

[231] Orna Raz, Philip Koopman, and Mary Shaw. Semantic anomaly detection in online data sources. In *proceedings of the 24th International Conference on Software Engineering*, pages 302–312, 2002.

[232] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Outlier detection in axis-parallel subspaces of high dimensional data. In *Pacific-asia conference on knowledge discovery and data mining*, pages 831–838. Springer, 2009.

[233] Mostafa Rahmani and George K Atia. Randomized robust subspace recovery and outlier detection for high dimensional data matrices. *IEEE Transactions on Signal Processing*, 65(6):1580–1594, 2016.

[234] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-VAE. *arXiv preprint arXiv:1804.03599*, 2018.

[235] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR, 2018.

[236] Babak Esmaeili, Hao Wu, Sarthak Jain, Alican Bozkurt, Narayanaswamy Siddharth, Brooks Paige, Dana H Brooks, Jennifer Dy, and Jan-Willem Meent. Structured disentangled representations. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2525–2534. PMLR, 2019.

[237] Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

[238] Junwen Bai, Weiran Wang, and Carla P Gomes. Contrastively disentangled sequential variational autoencoder. *Advances in Neural Information Processing Systems*, 34:10105–10118, 2021.

[239] Minyoung Kim, Yuting Wang, Pritish Sahu, and Vladimir Pavlovic. Bayes-factor-vae: Hierarchical bayesian deep auto-encoder models for factor disentanglement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2979–2987, 2019.

[240] Lucas Deecke, Lukas Ruff, Robert A Vandermeulen, and Hakan Bilen. Transfer-based semantic anomaly detection. In *International Conference on Machine Learning*, pages 2546–2558. PMLR, 2021.

[241] Jakub M Tomczak and Max Welling. Vae with a vampprior. *arXiv preprint arXiv:1705.07120*, 2017.

[242] Jaewoong Choi, Geonho Hwang, and Myungjoo Kang. Discond-vae: disentangling continuous factors from the discrete. *arXiv preprint arXiv:2009.08039*, 2020.

[243] Axel Sauer and Andreas Geiger. Counterfactual generative networks. *arXiv preprint arXiv:2101.06046*, 2021.

[244] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

[245] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016.

[246] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[247] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[248] Luc P Devroye and Terry J Wagner. The strong uniform consistency of nearest neighbor density estimates. *The Annals of Statistics*, pages 536–540, 1977.

[249] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[250] Quoc Phong Nguyen, Kar Wai Lim, Dinil Mon Divakaran, Kian Hsiang Low, and Mun Choon Chan. Gee: A gradient-based explainable variational autoencoder for network anomaly detection. In *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 91–99. IEEE, 2019.

[251] Tung Kieu, Bin Yang, Chenjuan Guo, Razvan-Gabriel Cirstea, Yan Zhao, Yale Song, and Christian S Jensen. Anomaly detection in time series with robust variational quasi-recurrent autoencoders. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1342–1354. IEEE, 2022.

[252] Jiayu Sun, Xinzhou Wang, Naixue Xiong, and Jie Shao. Learning sparse representation with variational auto-encoder for anomaly detection. *IEEE Access*, 6:33353–33361, 2018.

[253] David Zimmerer, Simon AA Kohl, Jens Petersen, Fabian Isensee, and Klaus H Maier-Hein. Context-encoding variational autoencoder for unsupervised anomaly detection. *arXiv preprint arXiv:1812.05941*, 2018.

[254] Samet Akçay, Amir Atapour-Abarghouei, and Toby P Breckon. Skip-ganomaly: Skip connected and adversarially trained encoder-decoder anomaly detection. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[255] Mahdyar Ravanbakhsh, Moin Nabi, Enver Sangineto, Lucio Marcenaro, Carlo Regazzoni, and Nicu Sebe. Abnormal event detection in videos using generative adversarial nets. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 1577–1581. IEEE, 2017.

[256] Yezheng Liu, Zhe Li, Chong Zhou, Yuanchun Jiang, Jianshan Sun, Meng Wang, and Xiangnan He. Generative adversarial active learning for unsupervised outlier detection. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1517–1528, 2019.

[257] Houssam Zenati, Manon Romain, Chuan-Sheng Foo, Bruno Lecouat, and Vijay Chandrasekhar. Adversarially learned anomaly detection. In *2018 IEEE International conference on data mining (ICDM)*, pages 727–736. IEEE, 2018.

[258] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

[259] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR, 2019.

[260] Ilyes Khemakhem, Diederik Kingma, Ricardo Monti, and Aapo Hyvarinen. Variational autoencoders and nonlinear ica: A unifying framework. In *International Conference on Artificial Intelligence and Statistics*, pages 2207–2217. PMLR, 2020.

[261] Aviv Gabbay, Niv Cohen, and Yedid Hoshen. An image is worth more than a thousand words: Towards disentanglement in the wild. *Advances in Neural Information Processing Systems*, 34:9216–9228, 2021.

[262] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[263] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[264] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.