Y-GAN: Learning Dual Data Representations for Anomaly Detection in Images

Marija Ivanovska[a], Vitomir Štruc[a,*]

[a]*Faculty of electrical engineering, University of Ljubljana, Trzaska cesta 25, 1000 Ljubljana, Slovenia*

Abstract

We propose a novel reconstruction-based model for anomaly detection in image data, called 'Y-GAN'. The model consists of a Y-shaped auto-encoder and represents images in two separate latent spaces. The first captures meaningful image semantics, which are key for representing (normal) training data, whereas the second encodes low-level residual image characteristics. To ensure the dual representations encode mutually exclusive information, a disentanglement procedure is designed around a latent (proxy) classifier. Additionally, a novel representation-consistency mechanism is proposed to prevent information leakage between the latent spaces. The model is trained in a one-class learning setting using only normal training data. Due to the separation of semantically-relevant and residual information, Y-GAN is able to derive informative data representations that allow for efficacious anomaly detection across a diverse set of anomaly detection tasks. The model is evaluated in comprehensive experiments with several recent anomaly detection models using four popular image datasets, i.e., MNIST, FMNIST, CIFAR10, and PlantVillage. Experimental results show that Y-GAN outperforms all tested models by a considerable margin and yields state-of-the-art results. The source code for the model is made publicly available at https://github.com/MIvanovska/Y-GAN.

Keywords: anomaly detection, one-class learning, disentangled data representations

1. Introduction

Anomaly detection in images represents a challenging computer-vision problem, where the goal is to distinguish *anomalous* data from data considered to be *normal*. Here, the term *normal* usually corresponds to images (or other types of visual data) that conforms to some predefined characteristics, and is, in general, application dependent (Perera et al., 2021). Most recent solutions approach anomaly detection from a *one-class* classification perspective and attempt to learn detection models using normal training data only. Such an approach has led to successful deployment of anomaly detection techniques in a wide variety of application domains where the anomalous data is not readily available or is difficult to collect, including (visual) quality inspection (Racki et al., 2018; Zavrtanik et al., 2021a; Ristea et al., 2022), surveillance and security (Doshi & Yilmaz, 2020; Sabokrou et al., 2017; Ionescu et al., 2019; Bakalos et al., 2022; Thakare et al., 2022; Sultani et al., 2018), information forensics (Khalid & Woo, 2020; Wang et al., 2020),

*Corresponding author.
Email addresses: marija.ivanovska@fe.uni-lj.si (Marija Ivanovska), vitomir.struc@fe.uni-lj.si (Vitomir Štruc)

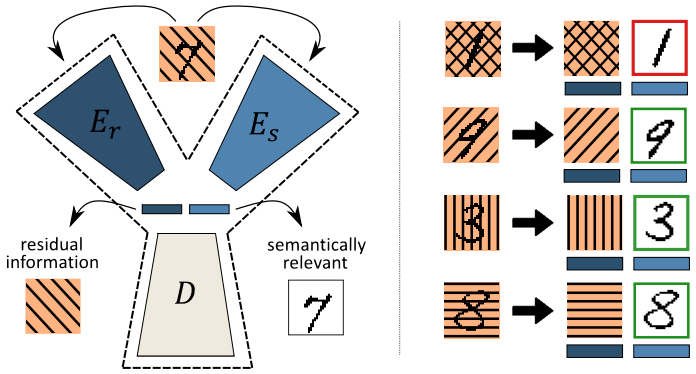


Figure 1: We propose Y-GAN, a novel anomaly detection model for images built around a Y-shaped auto-encoder network. The model disentangles semantically-relevant image information from irrelevant, residual characteristics and facilitates efficacious anomaly detection based on selective image encoding. As illustrated on the right, the removal of residual characteristics allows for easier detection of the digit "1", considered anomalous in this illustrative example.

biometrics, (Fatemifar et al., 2019; Oza & Patel, 2019; Yadav et al., 2020; Perera & Patel, 2019; Ivanovska & Štruc, 2023) or medical imaging (Schlegl et al., 2017, 2019) among others.

Contemporary research on one-class image-based anomaly detection is dominated by reconstruction-oriented models and typically relies on powerful auto-encoders (Gong et al., 2019; Nguyen & Meunier, 2019) or generative adversarial networks (GANs) (Schlegl et al., 2019; Zhou et al., 2020). These models commonly learn some latent representation that can be used to reconstruct normal data samples with high fidelity. Because no anomalous data is seen during training, the basic assumption here is that such (anomalous) samples will lead to poor reconstructions. As a result, differences in reconstruction quality are commonly exploited to differentiate between normal and anomalous data. Reconstructive approaches have been shown to perform well across a broad range of anomaly detection tasks and to provide competitive results across several popular benchmarks (Akçay et al., 2019b,a). However, as emphasized in Fei et al. (2021), the learning objectives typically utilized for learning reconstructive models predominantly focus on low-level pixel comparisons instead of image semantics intrinsic to the training data. This results in latent representations that encode low-level data characteristics that are likely to be shared between normal and anomalous data samples (Dosovitskiy & Brox, 2016) instead of more discriminating higher-level semantics. Additionally, when data with rich visual characteristics and complex appearances is used for training, the likelihood of high-fidelity reconstructions of anomalous data increases as well, rendering reconstruction-based models less effective in such cases. This problem is further exacerbated by the high generalization capabilities of modern generative models, where high-quality reconstructions of anomalous samples can already be expected under more relaxed assumptions (Fei et al., 2021; Gong et al., 2019). A key challenge with these techniques is, therefore, to learn latent representations that encode important image semantics and are uninformative with respect to low-level visual cues commonly shared by the normal and anomalous data. Such semantics are of paramount importance for a successful description of normal data, help to better normal image samples from anomalous ones, and address the over-generalization problem seen

with reconstruction-oriented anomaly detection models.

Based on this insight, we propose in this paper a novel anomaly detection model, called 'Y-GAN,' that aims to address the above-mentioned challenges and explicitly separates useful image semantics from uninformative (noisy, residual) data characteristics.. As illustrated in Fig. 1, Y-GAN is designed around a Y-shaped auto-encoder model that encodes input images in two distinct latent representations. The first representation captures semantically meaningful image cues useful for representing key properties of normal data, while the second encodes irrelevant, residual data characteristics. This dual encoding is enabled by an effective disentanglement procedure that can be learned automatically in a *one-class learning* setting, i.e., without the use of anomalous data. To control the information content in the two latent representations, Y-GAN utilizes a latent classifier and trains it to discriminate between sub-classes/groups of normal data. In other words, it exploits differences within the normal data to learn meaningful data semantics that can later be used for anomaly detection. Additionally, a novel *representation consistency* mecahnism is introduced for the training procedure of Y-GAN that ensures that the encoded information in the dual latent representations is mutually exclusive. Using this approach, Y-GAN is able to learn highly descriptive data representations that facilitate effective anomaly detection across a variety of problem settings. This includes *object-level* anomaly-detection problems, where a group of object classes is considered normal, while an unseen group of objects is considered anomalous, but also *pixel-level* anomaly detection tasks, where unusual/abnormal image regions are considered anomalous. The model is evaluated in extensive experiments on four anomaly detection benchmarks and compared with several state-of-the-art anomaly detection models presented recently in the literature. The results of the evaluations show that Y-GAN offers significant performance improvements over all considered competing models.

In summary, our key contributions in this paper are:

- We propose Y-GAN, a novel anomaly detection method, that disentangles semantically-relevant image characteristics from residual information for efficacious data representation and addresses some of the key challenges associated with reconstructive anomaly-detection models, including over-generalization and the inability to focus on key data semantics that are relevant for the definition of normal data. As shown in the experimental section, the model leads to state-of-the-art performance on several popular datasets, across a variety of anomaly detection tasks.

- We introduce a novel disentanglement strategy that enforces representation consistency and allows Y-GAN to exclude uninformative image information from the anomaly detection task in a *one-class* learning setting. We note that the same strategy is also applicable to other problem domains in need of effective disentanglement.

- We show the benefit of the proposed dual data representation over several state-of-the-art anomaly detection mechanisms by reporting superior results on multiple benchmarks and across different anomaly detection tasks.

2. Background and Related Work

A considerable amount of research has been conducted in the field of anomaly detection over the years. While early approaches considered statistical models (Yamanishi et al., 2004; Xu et al., 2012), one-class classifiers (Tax & Duin, 2004; Lanckriet et al., 2003) or sparse representations (Lu et al., 2013; Zhao et al., 2011) for this task, more recent solutions leverage advances in deep learning to learn powerful (one-class) anomaly detectors, e.g., Schlegl et al. (2019); Abati et al. (2019); Ruff et al. (2018); Markovitz et al. (2020); Bergmann et al. (2020); Pang et al. (2020); Zaheer et al. (2020); Bergman & Hoshen (2020). In this section, we present the most important background information with respect to such one-class models to provide the necessary context for our work. For a more comprehensive coverage of the area of anomaly detection and a broader discussion of existing solutions, the reader is referred to some of the excellent surveys in this field, e.g., Perera et al. (2021); Chandola et al. (2009); Pang et al. (2021).

2.1. Reconstruction-based Anomaly Detection

Reconstruction-based models represent one of the most widely studied groups of anomaly detectors in the literature. Such models try to discriminate between normal and anomalous data by evaluating reconstruction errors produced by generative networks trained exclusively on normal data. Schlegl et al. (2017), for example, proposed to project probe samples into a GAN latent space learned in this manner in their AnoGAN model and generate reconstructions from the computed latent representation for scoring. While this approach relied on two separate steps (i.e., latent-space learning and reconstruction), later improvements, such as f-AnoGAN (Schlegl et al., 2019) or EGBAD (Zenati et al., 2018), demonstrated the benefits of learning the latent representation jointly with the reconstructive mapping. Akçay et al. (2019b) further enhanced the capability of reconstruction-based models with an adversarial auto-encoder, called 'GANomaly.' Different from previous work, the model derived an anomaly score by comparing latent representations of original and reconstructed images to facilitate anomaly detection. Later, the same authors introduced Skip-GANomaly (Akçay et al., 2019a) (GANomaly extension based on U-Net (Ronneberger et al., 2015)) in an attempt to capture descriptive multi-scale information. In addition to the standard reconstruction based criteria, Massoli et al. (2022) also proposed to explicitly optimize intermediate representations of each layer in the anomaly detection model.

More recent work on reconstruction-based models capitalized on the importance of designing informative/discriminating latent spaces that can widen the gap between reconstruction errors observed with normal and anomalous data. Perera et al. (2019), for example, designed a constrained latent space for their OCGAN model, such that only samples belonging to the class observed during training were reconstructed well, while anomalous samples were not. Zhou et al. (2020) split images into two distinct parts (i.e., texture and structure) in their P-Net model. The two parts were then encoded separately with the goal of making the generated representations more informative for anomaly detection. The model was later extended in Zhou et al. (2021), with an additional module, that memorizes the correspondence between the structure and its texture. Integrated memory modules were also utilized by Park et al.

(2020) in their anomaly-detection approach, to lessen the representation capacity of their model for anomalous data and further improve detection results. Conceptually similar solutions were presented in Gong et al. (2019); Yang et al. (2021).

The Y-GAN model proposed in this paper, follows the general idea of reconstruction-based methods, but unlike competing solutions encodes the input data in two distinct latent representations that allow for the separation of relevant information from information irrelevant for the anomaly detection task. As we show in the following sections, this separation of information is: (i) achieved without any assumptions regarding the source of relevant information (e.g., texture, color, structure, etc.), (ii) is learned automatically in an end-to-end manner from normal data only, and (iii) leads to significant performance improvements over existing reconstruction-based models on various anomaly detection tasks.

2.2. Anomaly Detection with Proxy Tasks

To address some of the limitations of reconstruction-based anomaly detectors, another major group of existing models uses proxy tasks when learning to discriminate between normal and anomalous data. The main idea behind solutions from this group is that models trained on normal data will fare badly in the considered proxy task when subjected to anomalous samples. Fei et al. (2021), for instance, explored image restoration in this context and showed that differences in restoration performance can be used for anomaly detection. Noroozi & Favaro (2016) investigated jigsaw solving as a proxy task, the solutions from Zavrtanik et al. (2021a); Haselmann et al. (2018); Collin & Vleeschouwer (2021); Zavrtanik et al. (2021b) utilized image inpainting as the proxy for anomaly detection, and the work from Bergman & Hoshen (2020); Hendrycks et al. (2019); Gidaris et al. (2018); Golan & El-Yaniv (2018) investigated classification objectives defined over self-annotated recognition problems to facilitate anomaly detection. The proposed Y-GAN is related to these models in that it also relies on a proxy classifier, which, however, aims at distinguishing between different sub-groups of the normal data. By defining the proxy task as a classification problem over normal data, Y-GAN is able to: (i) ensure a *compact* representation of the normal data in the model's latent space, and (ii) automatically learn semantically relevant information for the anomaly detection task. Both of these characteristics are beneficial for anomaly detection performance, as we demonstrate in the experimental section.

2.3. Pre-trained Models for Anomaly Detection

In an effort to capture the most discriminating characteristics of the input data, many recent anomaly detection techniques try to leverage the representational power of features extracted by pre-trained (large-scale) classification models. Defard et al. (2021) and Rippel et al. (2020), for example, have shown that such features can be successfully used for explicitly modeling the distribution of normal samples. Anomalies are in this case detected as out-of-distribution samples. Similar solutions based on shallower models have also been investigated in the literature, e.g., Cohen & Hoshen (2020); Roth et al. (2022). Reiss et al. (2021) proposed to fine-tune existing pretrained models using normal training data and showed that such an approach leads to impressive anomaly detection performance. Mai

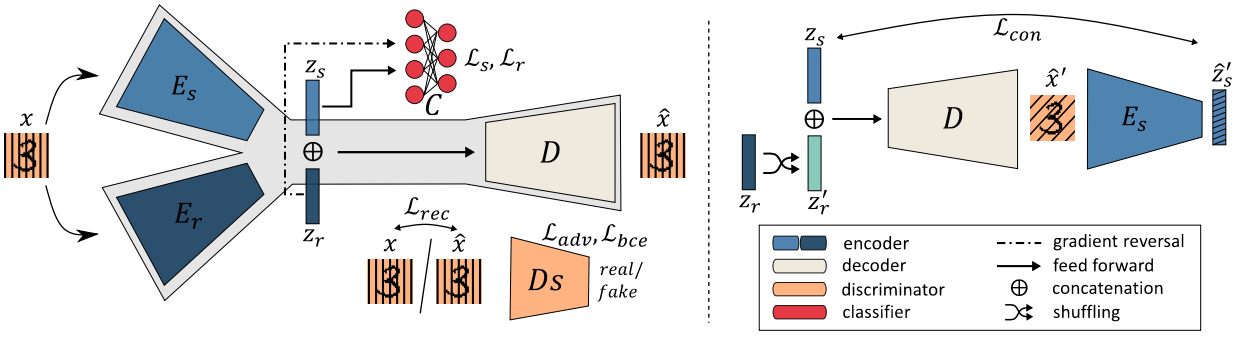


Figure 2: Overview of the proposed Y-GAN model and training loss terms. The model consist of a Y-shaped auto-encoder (illustrated on the right) with encoders, E_s and E_r, for dual data representation, a decoder D for image reconstruction, a latent classifier C for data disentanglement, and an adversarial discriminator Ds, which ensures that data reconstructions $\hat{x}$ cannot be distinguished from the original input samples x. Using an effective disentanglement procedure, Y-GAN aims to learn a semantically meaningful data representation in z_s, that encodes only characteristics relevant for representing normal data, while capturing irrelevant, residual data characteristics in z_r.

et al. (2021) explored the use of knowledge distillation to limit the set of pretrained features considered when building anomaly detection models. Similar ideas have also been pursued in Bergmann et al. (2020); Wang et al. (2021a); Salehi et al. (2021). Wang et al. (2021b) extended these studies by fine-tuning the distilled student network and further improved the detection rates on several anomaly detection benchmarks. In contrast to the outlined solutions, Y-GAN does not rely on pre-trained models for data representation, but instead learns discriminating encodings from scratch by separating uninformative data characteristics from meaningful data semantics relevant with respect to the normal training data. By doing so, it is able to selectively encode part of the characteristics that are relevant for discrimination without the need for large-scale datasets and resource hungry (pre-trained) classification models.

3. Methodology

The main contribution of this work is a novel (powerful) anomaly detection model, called Y-GAN. In this section we present the proposed model in detail and describe its main characteristics.

3.1. Proposed Model

Y-GAN, illustrated in Fig. 2, represents a generative adversarial network built around a Y-shaped auto-encoder. The key idea behind Y-GAN is to split the latent space of the auto-encoder into two distinct parts by disentangling informative *image semantics* (e.g., shapes, appearances, textures), relevant with respect to some *normal* training data from uninformative, *residual image information* (e.g., noise, background, illumination changes). The goal of this disentaglement process is to address some of the shortcoming of existing reconstruction-based models to anomaly detection in images and (in a sense) enhance/improve the information content relevant for the detection of normal

data by splitting away noisy residual information that may be present in both, normal as well as anomalous samples. This separation of image content is achieved through an effective disentanglement procedure (facilitated by a latent classifier) and allows our model to learn highly descriptive data representations for anomaly detection even in the challenging one-class learning regime. Details on the individual components of Y-GAN are given below.

The Auto-Encoder Network. To be able to separate relevant image content from residual information, we design a Y-shaped auto-encoder network and use it as the generator for Y-GAN. As illustrated in Fig. 2, this Y-shaped network consist of two separate encoders and a single decoder. The two encoders are identical from an architectural point of view, but have distinct parameters that can be learned independently one from the other. The first encoder E_s maps the input image $x \in \mathbb{R}^{w \times h}$ into a *semantically-relevant* latent representation z_s and the second E_r maps x into the dual, *residual* representation z_r, i.e.:

$$z_s = E_s(x) \in \mathbb{R}^d \text{ and } z_r = E_r(x) \in \mathbb{R}^d, \tag{1}$$

where d stands for the dimensionality of z_s and z_r. The complete latent representation of x is computed as a concatenation of the two partial representations, i.e., $z = z_s \oplus z_r$, and passed to the decoder D for reconstruction, i.e.,

$$\hat{x} = D(z) \in \mathbb{R}^{w \times h}. \tag{2}$$

To ensure that all of the image content in x is captured by the concatenated representation z, we use a standard L_1 reconstruction loss $\mathcal{L}_{rec}$ when learning the parameters of the auto-encoder (Isola et al., 2017):

$$\mathcal{L}_{rec} = \|x - \hat{x}\|_1 = \|x - D\big(E_s(x) \oplus E_r(x)\big)\|_1. \tag{3}$$

Moreover, an adversarial loss term and additional learning objectives that control the information content in the latent representations z_s and z_r are also utilized during training. We discuss these in the following sections.

The Discriminator. The expressive power of the latent representations, z_s and z_r, critically depends on the fidelity of the image reconstructions $\hat{x}$. To improve fidelity and ensure that the reconstructed samples follow the distribution of the normal training data, we include a discriminator Ds in the training procedure of Y-GAN and use an additional adversarial loss $\mathcal{L}_{adv}$ when learning the model. Following the recommendations from Salimans et al. (2016), we update the weights of the auto-encoder, i.e., the generator of Y-GAN, based on the following feature-matching objective that reduces training instability and avoids GAN over-training, i.e.:

$$\mathcal{L}_{adv} = \|f(x) - f(\hat{x})\|_2^2, \tag{4}$$

where $f(\cdot)$, in our case, denotes the activations of the last convolutional layer of Ds. Conversely, we encourage the discriminator to distinguish between real and fake images by optimizing a standard binary cross-entropy loss:

$$\mathcal{L}_{bce} = -\big[\log(Ds(x)) + \log(1 - Ds(\hat{x}))\big]. \tag{5}$$

The Latent Classifier. The Y-shaped design of Y-GAN's auto-encoder allows for the partitioning of the latent space into two representations, z_s and z_r. We force these representations to encode mutually exclusive information

by using a disentanglement procedure based on a latent classifier C. The goal of this classifier is two-fold: (i) to encourage semantic information relevant for representing normal data to be encoded in z_s and (ii) to force the irrelevant residual information into the latent representation z_r.

To be able to learn C, we assume that the normal training data can be partitioned into N sub-classes[1]. The classifier is then trained to predict correct class labels from the latent representations z_s and to misclassify input samples x given their latent representation z_r. This training procedure controls the information content in the dual latent representations and helps to learn meaningful data characteristics for anomaly detection without examples of anomalous data.

A cross-entropy loss is utilized to maximize the classification performance of C based on the semantically-relevant latent representation z_s, i.e.:

$$\mathcal{L}_s = -\sum_{i=1}^{N} y(i)\ \log(\hat{y}_s(i)), \tag{6}$$

where $y \in \mathbb{R}^N$ is the one-hot encoded ground truth label of x and $\hat{y}_s = C(E_s(x)) \in \mathbb{R}^N$ is the classifier prediction. As can ne seen from the above formulation, the classifier C is trying to learn to classify the semantically-relevant latent representations z_s as accurately as possible. During the back-propagation procedure, the gradients from the classification loss are propagated through the respective (semantically-relevant) encoder E_s, which, as a result, is updated to produce semantically-relevant latent representations z_s that are as informative as possible for C.

While minimizing $\mathcal{L}_s$ forces the latent representation z_s to be informative with respect to the classification of the normal data, our goal is to achieve the opposite for z_r. To this end, we transform z_r with a gradient reversal layer (Ganin & Lempitsky, 2015). This transformation layer R acts as an identity function in the forward pass through the model, i.e., $R(z_r) = z_r$, but reverses the gradients from the subsequent layer during back-propagation, i.e.,

$$\frac{dR}{dz_r} = -\lambda_R I, \tag{7}$$

where λ_R is a hyper-parameter and $I \in \mathbb{R}^{d \times d}$ is an identity matrix. In other words, during training the encoder E_r is encouraged to produce residual representations that are as uninformative as possible for the classifier C. Thus, the gradient reversal for the residual encoder E_r ensures that the residual representations z_r encode no useful semantic information for classification. The minimization of relevant information content in z_r is thus achieved through the following objective:

$$\mathcal{L}_r = -\sum_{i=1}^{N} y(i)\ \log(\hat{y}_r(i)), \tag{8}$$

where $\hat{y}_r = C(R(E_r(x))) \in \mathbb{R}^N$.

Enforcing Representation Consistency. The loss functions in Eqs. (6) and (8) provide for a first level of disentanglement, but do not completely prevent information leakage between the latent representations z_s and z_r. For

[1]Note that this is a reasonable assumption for many applications and holds for a wide variety of datasets and experimental protocols from the literature (Perera et al., 2021; Ruff et al., 2021; Tang et al., 2019), including the ones used in this paper.

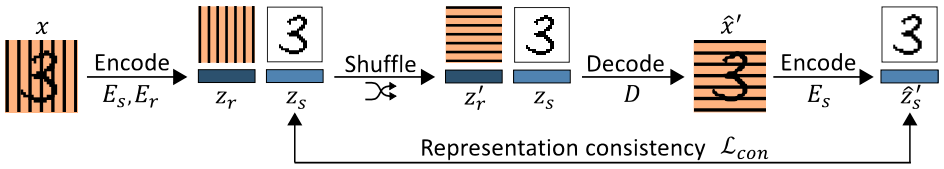


Figure 3: Illustration of the procedure used to enforce *representation consistency*. The input image is represented in two latent spaces that need to encode mutually exclusive information. Y-GAN ensures that the representation in the latent representation z_s is independent of that in z_r by encouraging the model to produce the same z_s even when changes in z_r are introduced. Shown is an illustrative toy example involving digit shapes (assumed to be relevant) and background textures (assumed to be irrelevant).

this purpose, we introduce a *consistency loss* ($\mathcal{L}_{con}$) that penalizes the encoder E_s in case it extracts inconsistent (semantically-relevant) information in z_s, when changes in the residual representation z_r are introduced. The overall idea of this procedure is illustrated on the right part of Fig. 2 and using a simple visual example in Fig. 3.

To calculate $\mathcal{L}_{con}$, we first randomly shuffle the set of residual representation z_r, generated from the samples in a given training batch, so that each z_s vector is concatenated with a z'_r vector, belonging to a randomly chosen sample from the batch. These artificially created concatenations are then passed to the decoder D, which generates hybrid reconstructions $\hat{x}'$, such that $\hat{x}' = D(z_s \oplus z'_r)$. Next, the reconstructed images are fed to the encoder E_s, which is expected to extract latent representations $\hat{z}'_s$ that are equivalent to the vector z_s, initially used for the generation of the hybrid reconstructions $\hat{x}'$. An angular dissimilarity measure is used to penalize differences between z_s and $\hat{z}'_s$, i.e.:

$$\mathcal{L}_{con} = -\frac{z_s \cdot \hat{z}'_s}{\|z_s\| \cdot \|\hat{z}'_s\|}. \tag{9}$$

By minimizing $\mathcal{L}_{con}$, we encourage the encoder E_s to extract image information that is invariant to the residual data characteristics encoded in z_r. Note that, $\mathcal{L}_{con}$ is calculated and optimized for each batch in the training set separately. The size of the training batches, therefore, has to be at least equal or greater than two, otherwise $\mathcal{L}_{con}$ has no effect. It also worth noting that $\mathcal{L}_{con}$ does not control what is encoded in the latent representations z_s and z_r but only ensures that the information in the two representations is mutually exclusive, or in other words, that the representations are properly disentangled.

3.2. Y-GAN Training

Y-GAN is trained in an end-to-end fashion, using a multi-step procedure. For each training batch, the losses from Eqs. (3) to (6) are calculated first. Next, the set of residual representations z_r in the given batch is randomly shuffled and processed, as described above for the calculation of the consistency loss $\mathcal{L}_{con}$ in Eq. (9). Finally, the weights of the generator (i.e., the Y-shaped auto-encoder) are updated, based on the combined objective $\mathcal{L}_{\mathcal{G}}$, i.e.:

$$\mathcal{L}_{\mathcal{G}} = \lambda_1 \mathcal{L}_{rec} + \lambda_2 \mathcal{L}_{adv} + \lambda_3 \mathcal{L}_s + \lambda_4 \mathcal{L}_r + \lambda_5 \mathcal{L}_{con}. \tag{10}$$

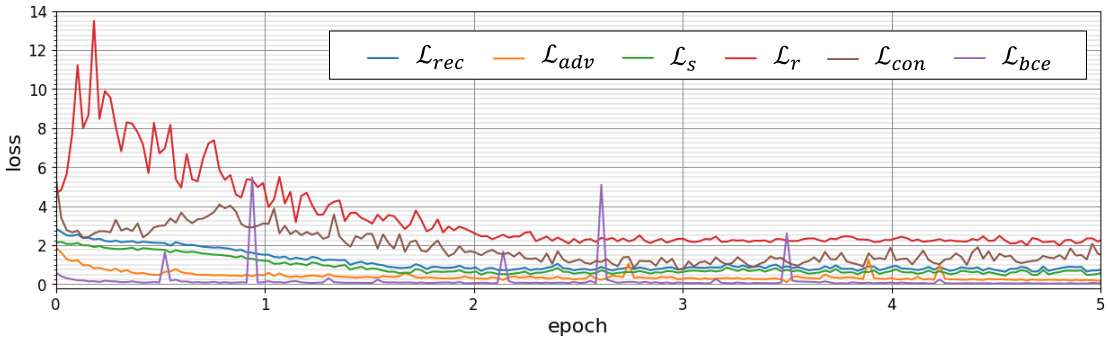
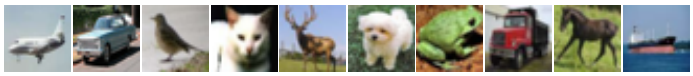


Figure 4: Graph illustrating the change of individual training loss components of Y-GAN during the optimization of the model. The visualization was generated during the training of the model on the MNIST dataset.

(a) MNIST: Examples of the 10 digit categories in the dataset.

(b) FMNIST: Examples of the 10 fashion-item categories in the dataset



(c) CIFAR10: Examples of the 10 object categories in the dataset

Figure 5: Selected samples from the three standard anomaly detection benchmarks used in the experiments: (a) MNIST (LeCun et al., 1998; LeCun et al.), (b) FMNIST (Xiao et al., 2017) and (c) CIFAR10 (Krizhevsky, 2009). Each dataset consists of 10 different object classes.

Similarly, the weights of the adversarial discriminator Ds are updated based on the combined loss $\mathcal{L}_{\mathcal{D}}$, i.e.:

$$\mathcal{L}_{\mathcal{D}} = \lambda_1 \mathcal{L}_{rec} + \lambda_6 \mathcal{L}_{bce} \tag{11}$$

The generator and discriminator are updated alternately for a fixed number of epochs during training. An example of the model's learning dynamics throughout the training phase is presented in Figure 4.

3.3. Anomaly Detection with Y-GAN

Similarly to Golan & El-Yaniv (2018), predictions of the latent classifier C are used to calculate anomaly scores. Given a probe sample x, the latent representation $z_s = E_s(x)$ is first computed and passed to the classifier C. Next, the activations of the output layer of the classifier are normalized $\{p_i(x)\}_{i=1}^{N}$, so they behave like probabilities, i.e.,

$\sum_{i=1}^{N} p_i = 1$. Finally, the highest of these "probabilities" is used to compute the anomaly score s, i.e.:

$$s = 1 - \max(p_i(x)), \tag{12}$$

for the given test sample x. Due to the normalization procedure, the generated anomaly scores are bounded to $s \in [0, 1]$, with 0 representing *ideal normal* data. Note that for the calculation of the anomaly scores only E_s and C are needed, which significantly shortens inference time.

4. Experimental Datasets and Setup

We evaluate Y-GAN in comprehensive experiments over four standard datasets used in the anomaly-detection literature. In this section, we describe the selected datasets and discuss the experimental protocols used to evaluate the proposed model. Finally, we also discuss implementation details related to Y-GAN, meant to foster reproducibility.

4.1. Datasets

We evaluate Y-GAN on three standard image-based anomaly detection benchmarks, i.e., MNIST (LeCun et al., 1998; LeCun et al.), FMNIST (Xiao et al., 2017), and CIFAR10 (Krizhevsky, 2009), as well as the real-world PlantVillage dataset from Hughes & Salathé (2015). A few example images from the four datasets are presented in Figs. 5 and 6. We provide details on the selected datasets below.

- **MNIST** contains $70,000$ grayscale images of handwritten digits, divided into 10 (approximately balanced) classes, where each class represents one digit (0 through 9). The images ship with a resolution of 28×28 pixels and exhibit variations in terms of digit appearance (LeCun et al., 1998; LeCun et al.).

- **FMNIST** (Fashion MNIST) was developed as a more comprehensive alternative to MNIST. The dataset again consists of $70,000$ grayscale images of size 28×28 pixels split into 10 balanced classes. However, FMNIST exhibits a larger degree of appearance variability than MNIST. Images in FMNIST depict clothing items grouped into different categories, i.e., T-shirts, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots (Xiao et al., 2017).

- **CIFAR10** contains $60,000$ color images of size 32×32 pixels representing animals and vehicles from 10 categories, i.e., airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, trucks. Different from MNIST or FMNIST, images in CIFAR10 do not have a uniform background and exhibit considerable diversity in terms of appearance even within the same category. These characteristics make it particularly challenging for anomaly detection tasks (Krizhevsky, 2009).

- **PlantVillage** is a recent real-world dataset of leafs and contains $54,305$ color images of 256×256 pixels. Each image represents a single leaf, photographed on a homogeneous background. Images are divided into 14 unbalanced categories of different plant species, 9 of which contain both, healthy and ill leafs. 3 categories contain only healthy

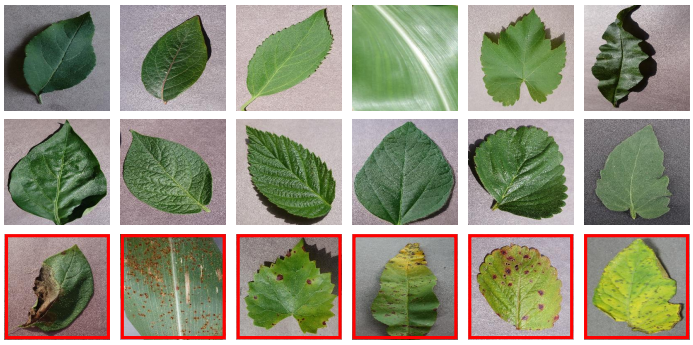


Figure 6: Selected samples from the real-world dataset PlantVillage (Hughes & Salathé, 2015). The dataset consists of healthy and ill leafs of 14 different plant species. Anomalies usually represent changes in the leaf shape and color or they appear as irregular pattern on the leaf's surface. Anomalous samples are in this figure marked with red color.

samples, while the remaining 2 have no disease-free leafs. Plant diseases are usually manifested as changes in the shape and the color of the leaf, but can also appear as a subtle pattern, that covers the leaf base, as shown in Fig. 6.

The selected datasets allow for the evaluation of anomaly detection models in different problem settings, i.e.: (i) with anomalies representing homogeneous classes in MNIST, FMNIST and CIFAR10, and (ii) with challenging and diverse real-world (spatially local) anomalies in the PlantVillage dataset. As we show in the results section, Y-GAN achieves state-of-the-art performance for both types of problems.

4.2. Experimental Setup

All models evaluated in the experimental section are trained in a *one-class learning* setting, where no examples of anomalous data are seen during training. Experiments on the MNIST, FMNIST, and CIFAR10 datasets are conducted in accordance with the standard *k-classes-out* experimental setup (Akçay et al., 2019b; Zenati et al., 2018; Ruff et al., 2021), where nine classes are defined as normal, while the 10th class is considered anomalous. We implement the standard $80/20$ split rule commonly used in the literature (Perera et al., 2019), where 80% of the normal data is randomly selected for training, while the rest is combined with anomalous samples, forming a balanced testing set. All experiments are repeated ten times, each time with a new class defined as anomalous. Images from CIFAR10 are used with the original resolution, while MNIST and FMNIST images are re-scaled to 32×32 pixels to fit the Y-GAN's architecture.

For the experiments with the PlantVillage dataset, we again follow the $80/20$ split rule, by randomly selecting 80% of the normal data for training purposes. The rest of the normal data along with anomalous samples is used for evaluation. However, because the number of training samples in PlantVillage is relatively small and not sufficient for deep learning tasks, the training data is augmented. For pixel-level augmentations we use techniques such as image sharpening, embossing, histogram manipulations and random changes of brightness and contrast. Additionally, we also apply horizontal flipping and random affine transformations. By carefully adjusting parameter values of the

augmentation operations we ensure that the original dataset is significantly enlarged without inducing anomaly-like samples. Following prior work (Akçay et al., 2019b; Schlegl et al., 2017; Zaheer et al., 2020), we report the Area Under the Receiver Operating Characteristic (ROC) curve (AUC) as a scalar performance score in the experiments. In PlantVillage, we also report true positive (TPR) and true negative rates (TNR) for each class in the dataset. Both metrics are calculated at the equal error rate (EER) point of the ROC curve generated for all training samples from the dataset.

4.3. Implementation Details

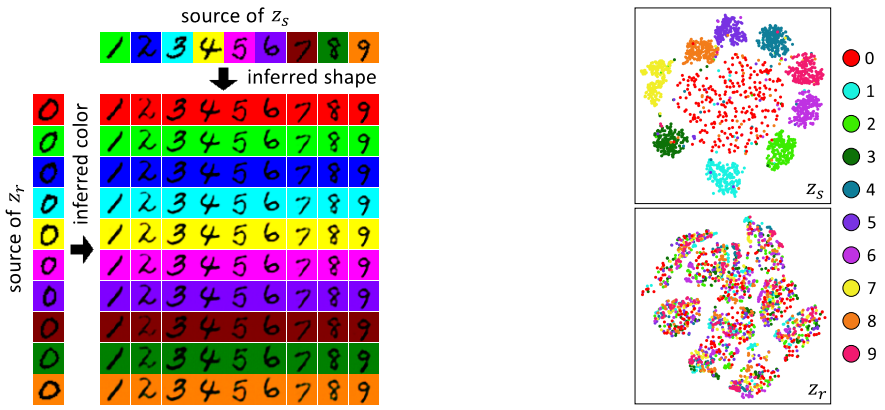
Model Architecture. Y-GAN consists of five architectural building blocks, i.e., two encoders, E_s and E_r, a decoder, D, a discriminator, Ds and a classifier, C. The first four components are designed after DCGAN (Radford et al., 2016), whereas the topology of the classifier is determined experimentally.

The two encoders, E_s and E_r, consist of convolutional layers with stride 2. Each convolutional layer is followed by a Leaky ReLU activation with negative slope 0.2 and a batch normalization layer. The two encoders have an identical architecture and each map the input image x to a $d = 100$ dimensional latent vector. The upscaling in the decoder D is performed with transposed convolutions with stride 2, each followed by ReLU activations and a batch normalization layer. The last convolutional layer uses an *tanh* activation function for bounded support. The disciminator Ds has the same architecture as E_s and E_r, up until the last convolutional layer, which is followed by a standard *sigmoid* activation. The latent classifier C is a multi-layer perceptron (MLP) with one hidden layer and 30 hidden units. The size of the input layer is determined by the dimensionality of the latent representation, z_s, and the size of the output layer by the number of classes N of the (normal) training data. In our case $N = 9$ for the experiments on MNIST, FMNIST, and CIFAR10, since there are 9 normal sub-classes defined by our experimental protocol. For the PlantVillage N is set to $N = 12$, which is the number of non-anomalous plant categories in the dataset[2].

Training Setting. The learning objectives in Eqs. (10) and (11) are minimized using the Adam optimizer (Kingma & Ba, 2015) with a learning rate of $l_r = 0.0002$ and momentums $\beta_1 = 0.5$ and $\beta_2 = 0.999$ (Radford et al., 2016). The weights in $\mathcal{L}_\mathcal{G}$ and $\mathcal{L}_\mathcal{D}$ are determined empirically through an optimization procedure on validation data. For the experiments we use $\lambda_1 = \lambda_5 = 50$, and $\lambda_2 = \lambda_3 = \lambda_4 = \lambda_6 = 1$. While these weights are kept constant, the weight associated with the gradient reversal layer is initialized to a value of $\lambda_R = 0$ and is then gradually increased as the training progresses, as suggested in Ganin & Lempitsky (2015). All models are trained for 100 epochs on MNIST, FMNIST, and CIFAR10 and for 200 epochs on PlantVillage, where less data data is available for the learning procedure.

Implementation. Y-GAN is implemented in Python 3.7. using PyTorch 1.5. and CUDA 10.2. All source code, model definitions, and trained weights are made publicly available to facilitate reproducibility from https://github.com/MIvanovska/Y-GAN.

[2]Recall that 2 out of the total of 14 classes have only anomalous samples and are not considered during training.



(a) Disentanglement analysis

(b) t-SNE plots

Figure 7: Proof-of-concept study. Y-GAN is trained on normal data (digits 1 to 9) with the goal of flagging anomalous data (digit 0). (a) Disentanglement results: the model learns to separate digits from background in the latent spaces z_s and z_r. Shown are examples of hybrid reconstructions, where z_s is taken from the examples on the top and z_r from the samples on the left. (b) t-SNE plots in 2D: normal data forms compact well-separated clusters in the semantically-relevant latent space (marked z_s) and overlaps considerably in the residual latent space (marked z_r).

Using a personal desktop computer with an Intel® Core™ i7-8700K CPU and an NVIDIA® GeForce RTX 2080 Ti GPU, it takes around four hours to train Y-GAN on MNIST, FMNIST, and CIFAR10. For the higher resolution images in PlantVillage, the training stage takes around five hours. Once the model is learned, a single image is processed in around $2.6\,ms$ for the smallest 32×32 images and $13.5\,ms$ for the largest 256×256 images.

5. Proof-of-Concept Studies

To explore the characteristics of the dual latent-space representation and evaluate the effectiveness of the disentanglement process performed by Y-GAN, we first conduct a number of proof-of-concept studies. To this end, we generate a color version of MNIST (*Color-MNIST* hereafter) by inverting the thresholded black and white images of the dataset and replacing the white background with one of the following colors: red, green, blue, cyan, yellow, purple, violet, brown, dark green, or orange. We make sure all colors are represented equally in the generated dataset. Next, we train Y-GAN on the constructed dataset by considering digits 1 to 9 as normal data, and 0 as anomalous. In the test phase, we present the model with unseen samples and compute their latent representations, z_s and z_r. Finally, we generate hybrid reconstructions by concatenating the latent vector z_s taken from one test sample with the latent vector z_r of another randomly selected test sample and pass the concatenated vector through the decoder. The goal of these experiments to illustrate what kind of information is encoded in the semantically-relevant and residual latent representations, z_s and z_r, in a human-interpretable form. Example reconstructions produced with the described process are shown in Fig. 7(a).

As can be seen, Y-GAN learns to disentangle data characteristics relevant for the digit representation task, from

Figure 8: Disentanglement results on samples from two different CIFAR10 categories. The synthesized samples show that the model learns to successfully disentangle semantically relevant from residual image characteristics. Each object's shape is inferred from the source image of the semantically-relevant vector z_s (first row), while color and style are inferred from the source images of z_r (left most image in each example). Best viewed electronically.

characteristics that are irrelevant, i.e., background color in this toy example. Consequently, the replacement of the original latent vector z_r causes a change in the background color, which is now inherited, from the randomly selected sample - shown on the left part of Fig. 7(a). Meanwhile, the shape of the digit in the original image is preserved well. It is worth noting again that the *semantic relevance*, and in turn, the content encoded in z_s is defined by the latent classifier C and the learned classification task. Given that the goal of C in this experiments is to classify digits, z_s correctly encodes the shape of the digits, while the noisy residual information in the form of background colour is encoded in z_r.

Next, we use t-distributed Stochastic Neighbor Embeddings (t-SNE) (Van der Maaten & Hinton, 2008) to visualize the distribution of the generated data in the dual latent spaces in Fig. 7(b). Here, 250 random samples of each of the Color-MNIST digit classes are used for visualization. Note that for the semantically-relevant latent space, samples corresponding to digits 1 to 9 form compact and well separated clusters (marked z_s), while samples for the anomalous 0 are considerably less compact despite the fact that they come from a single (homogeneous) class. Nevertheless, they do not overlap (significantly) with the normal data. In the residual latent space (marked z_r), 10 clusters corresponding to the background colors used in Color-MNIST can be identified in the t-SNE plot. However, each cluster contains samples from all 10 digits, suggesting that this representation has limited discriminating power for anomaly detection.

To test the behavior of Y-GAN on more complex data, we repeat the same experiment using CIFAR10. Selected synthesized samples and their respective source images are shown in Fig. 8. We again observe that Y-GAN learned to successfully disentangle semantically relevant attributes from those that are irrelevant for representing classes in the normal data. The shape of the objects and other visually important characteristics are obviously inferred from the source image of the relevant latent vector z_s, while background style and colors and inferred from the residual latent vector z_r. Different from the Color-MNIST example, where the digits represent homogeneous classes with limited variability, the large intra-class variability of CIFAR10 images leads to lower quality reconstructions, which is expected given Y-GAN's learning objectives. Nevertheless, the results validate that meaningful separation of information content is achieved in the latent space even with challenging input images.

Finally, we investigate the entanglement of extracted information in the latent space of Y-GAN, by generating t-SNE embeddings of computed latent representations from MNIST, FMNIST and CIFAR10. Here, we first generate

t-SNE embeddings of the concatenaned latent vectors z_s and z_r. We then repeat the t-SNE mapping procedure for each latent encoding separately. The visualizations of these t-SNE embedding for the 10-class MNIST, FMNIST and CIFAR10 datasets are shown in Figure 9. As can be seen, the joint concatenated representations (marked $z_s + z_r$) contain poorly separated information, which is well split into noisy, uninformative residual representations in z_r and more structured, better separated representations in z_s. These visualizaation point to the effectivness of the proposed disentaglement scheme and feasbility of the anomaly detection process with Y-GAN.

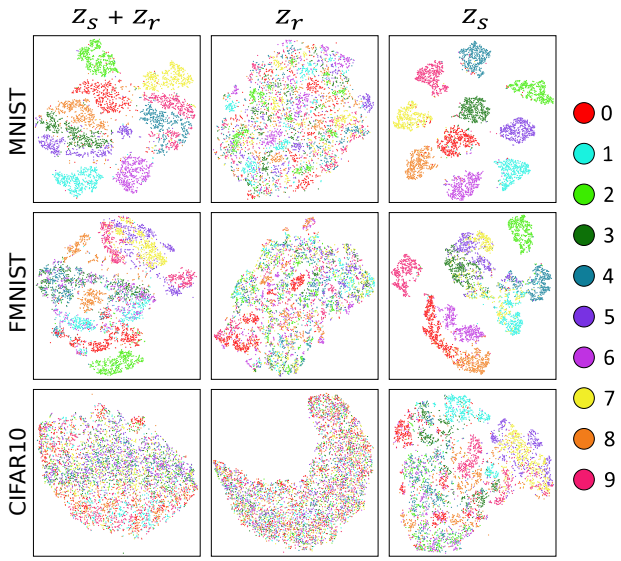


Figure 9: t-SNE embeddings generated for the concatenated representation $z_s + z_r$, as well as the separate embeddings, z_s and z_r, on the MNIST, FMNISTand CIFAR10 datasets. Note how the data in the semantically-relevant representations is more structured and better separated with respect to the sub-classes of the normal data.

6. Results and Discussion

To illustrate the performance of Y-GAN, we report in this section results that: (i) compare Y-GAN to state-of-the-art techniques from the literature, (ii) were generated through a comprehensive ablation study and demonstrate the contribution of various components of Y-GAN, (iii) highlight some of the model's characteristics, and (vi) investigate the behavior of Y-GAN in a qualitative manner.

6.1. Quantitative Evaluation

We evaluate Y-GAN in comparative experiments with several state-of-the-art anomaly-detection models. Specifically, we compare against GANomaly (Akçay et al., 2019b), Skip-GANomaly (Akçay et al., 2019a), OCGAN (Perera et al., 2019), f-AnoGAN (Schlegl et al., 2019), P-Net (Zhou et al., 2020, 2021), and MOCCA (Massoli et al., 2022) which represent powerful reconstruction-based (RB) anomaly detection models and are Y-GAN's *main competitors*.

Model	Type†	0	1	2	3	4	5	6	7	8	9	Mean ± Std
GANomaly (Akçay et al., 2019b)	RB	0.899	0.701	0.954	0.820	0.829	0.891	0.875	0.735	0.926	0.667	0.830 ± 0.094
Skip-GANomaly (Akçay et al., 2019a)	RB	0.845	0.919	0.754	0.734	0.530	0.573	0.761	0.532	0.765	0.637	0.705 ± 0.127
OCGAN (Perera et al., 2019)	RB	**0.958**	0.934	**0.959**	**0.969**	0.929	0.920	0.904	0.775	**0.970**	0.732	0.905 ± 0.079
f-AnoGAN (Schlegl et al., 2019)	RB	0.880	**0.983**	0.954	**0.969**	0.928	0.896	0.892	0.782	0.949	0.702	0.894 ± 0.084
P-Net (Zhou et al., 2020)	RB	0.788	0.608	0.678	0.553	0.528	0.467	0.612	0.526	0.618	0.474	0.585 ± 0.093
MOCCA (Massoli et al., 2022)	RB	0.709	0.668	0.727	0.637	0.647	0.656	0.633	0.617	0.609	0.526	0.643 ± 0.053
ARNet (Fei et al., 2021)	PT	0.879	0.798	0.880	0.752	0.767	0.816	0.940	0.636	0.811	0.685	0.796 ± 0.087
Patch SVDD (Yi & Yoon, 2020)	PT	0.774	0.709	0.849	0.646	0.570	0.656	0.730	0.515	0.573	0.609	0.663 ± 0.098
PaDiM (Defard et al., 2021)	PC	0.551	0.670	0.828	0.647	0.608	0.690	0.820	0.779	0.610	0.561	0.676 ± 0.097
CS-Flow (Rudolph et al., 2022)	PC	0.940	**0.983**	0.945	0.898	**0.976**	**0.958**	**0.961**	**0.967**	0.966	**0.918**	**0.951 ± 0.025**
Y-GAN [Ours]	RB*	**0.993**	**0.993**	**0.984**	**0.989**	**0.984**	**0.986**	**0.985**	**0.980**	**0.988**	**0.987**	**0.987 ± 0.004**

† RB - reconstruction based, PT - proxy task-based, PC - utilizing pre-trained classification models ; RB* - reconstruction-based but with a latent proxy task

Table 1: MNIST results in terms of AUC scores. The best model in each column is marked blue, the runner-up red.

Model	Type†	T-shirt	Trousers	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot	Mean ± Std
GANomaly (Akçay et al., 2019b)	RB	0.607	0.936	0.600	0.767	0.678	**0.973**	0.533	**0.895**	0.961	0.867	0.782 ± 0.158
Skip-GANomaly (Akçay et al., 2019a)	RB	0.615	0.639	0.698	0.523	0.632	0.816	0.612	0.683	0.780	0.722	0.672 ± 0.082
OCGAN (Perera et al., 2019)	RB	0.681	0.912	0.643	0.680	0.615	0.942	0.563	0.679	0.959	0.844	0.752 ± 0.140
f-AnoGAN (Schlegl et al., 2019)	RB	0.642	0.875	**0.758**	0.656	0.671	0.889	**0.690**	0.852	0.945	**0.911**	0.789 ± 0.112
P-Net (Zhou et al., 2020)	RB	0.590	0.586	0.566	0.564	0.694	0.340	0.473	0.542	0.720	0.714	0.579 ± 0.110
MOCCA (Massoli et al., 2022)	RB	0.598	0.773	0.622	0.505	0.465	0.873	0.603	0.513	0.869	0.617	0.644 ± 0.139
ARNet (Fei et al., 2021)	PT	0.647	**0.966**	0.749	0.773	**0.740**	0.877	0.687	0.751	0.973	0.896	0.808 ± 0.112
Patch SVDD (Yi & Yoon, 2020)	PT	0.742	0.572	0.661	0.597	0.536	0.727	0.674	0.577	0.876	0.763	0.673 ± 0.101
PaDiM (Defard et al., 2021)	PC	0.729	0.676	0.475	0.642	0.497	0.821	0.502	0.615	0.915	0.794	0.667 ± 0.142
CS-Flow (Rudolph et al., 2022)	PC	**0.813**	**0.988**	0.698	**0.836**	0.687	0.947	0.666	0.777	**0.980**	0.853	**0.825 ± 0.114**
Y-GAN [Ours]	RB*	**0.912**	0.915	**0.904**	**0.949**	**0.880**	**0.957**	**0.877**	**0.903**	**0.982**	**0.975**	**0.925 ± 0.036**

† RB - reconstruction-based, PT - proxy task-based, PC - utilizing pre-trained classification models ; RB* - reconstruction-based but with a latent proxy task

Table 2: FMNIST results in terms of AUC scores. The best model in each column is marked blue, the runner-up red.

Model	Type†	Airplane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck	Mean ± Std
GANomaly (Akçay et al., 2019b)	RB	0.655	0.705	0.420	0.580	0.359	0.588	0.515	0.571	0.630	0.712	0.574 ± 0.109
Skip-GANomaly (Akçay et al., 2019a)	RB	0.581	0.718	0.494	0.487	0.518	0.480	**0.722**	0.559	0.554	0.701	0.581 ± 0.092
OCGAN (Perera et al., 2019)	RB	0.548	**0.731**	0.386	0.582	0.348	0.597	0.422	0.626	0.488	0.713	0.544 ± 0.125
f-AnoGAN (Schlegl et al., 2019)	RB	0.578	0.692	0.564	0.555	0.459	0.580	0.591	0.643	0.610	0.682	0.595 ± 0.064
P-Net (Zhou et al., 2020)	RB	0.582	0.671	0.455	0.611	0.476	0.596	0.602	0.538	0.523	0.629	0.563 ± 0.065
MOCCA (Massoli et al., 2022)	RB	**0.671**	0.500	0.631	0.502	0.521	0.547	0.518	0.474	0.531	0.451	0.535 ± 0.064
ARNet (Fei et al., 2021)	PT	0.598	0.635	0.466	0.706	0.435	0.697	0.512	0.662	0.630	**0.727**	0.607 ± 0.098
Patch SVDD (Yi & Yoon, 2020)	PT	0.517	0.542	0.505	0.548	0.493	0.567	0.529	0.538	0.511	0.542	0.529 ± 0.021
PaDiM (Defard et al., 2021)	PC	0.542	0.668	0.502	0.546	0.335	0.612	0.433	0.549	0.386	0.625	0.520 ± 0.102
CS-Flow (Rudolph et al., 2022)	PC	0.670	0.698	**0.692**	**0.747**	**0.675**	**0.741**	0.658	**0.712**	**0.690**	0.671	**0.695 ± 0.028**
Y-GAN [Ours]	RB*	**0.729**	**0.767**	**0.749**	**0.768**	**0.759**	**0.764**	**0.778**	**0.780**	**0.722**	**0.811**	**0.763 ± 0.024**

† RB - reconstruction-based, PT - proxy task-based, PC - utilizing pre-trained classification models ; RB* - reconstruction-based but with a latent proxy task

Table 3: CIFAR10 results in terms of AUC scores. The best model in each column is marked blue, the runner-up red.

Additionally, we also include the recent ARNet (Fei et al., 2021) and Patch SVDD (Yi & Yoon, 2020) approaches as representatives of proxy-task (PT) models, and PaDiM (Defard et al., 2021) and CS-Flow (Rudolph et al., 2022)

as an example of solutions utilizing pre-trained classification (PC) models in the evaluation. For a fair comparison, the official GitHub implementations are used for the experiments (where available) together with the advocated hyper-parameters to ensure optimal performance.

MNIST. The MNIST results, reported in Table 1, show that Y-GAN significantly outperforms all evaluated baselines. It improves on the mean AUC score of the runner-up CS-Flow by 5% and on the standard deviation (computed over all runs) by a factor of 6. Compared to the competing models, Y-GAN ensures the most consistent results, regardless of which class is considered anomalous. This can be seen particularly well from the results for digits 7 and 9, where almost all tested models exhibit a drop in AUC scores, while Y-GAN retains performance similar to other settings.

FMNIST. Compared to MNIST, the AUC scores obtained on FMNIST are lower for all tested models due to the larger image diversity in this dataset, as summarized in Table 2. The proposed Y-GAN achieves a mean AUC score of 0.925, compared to 0.825 for CS-Flow, 0.808 for ARNet and 0.789 for f-AnoGAN which are the next three models (in this order) in terms of performance. A performance improvement of more than 11% over the second best performing model, CS-Flow, points to the descriptiveness of the representation learnt by Y-GAN in the semantically-relevant latent space. Y-GAN again achieves the most consistent results across different experimental runs.

CIFAR10. Images in CIFAR10 were captured in unconstrained settings, which makes this dataset extremely challenging, as evidenced by the results in Table 3. Almost all competing models result in mean AUC scores close to (or below) 0.6, which speaks of the difficulty of learning meaningful representations on CIFAR10. The only competitive model, that has significantly higher performence is again CS-Flow, with an AUC score of 0.695. Still the dual representation strategy of Y-GAN, improves on this runner-up by more than 9%, yielding a mean AUC score of 0.763. The proposed model also convincingly outperforms all competing models in all 10 experimental runs.

PlantVillage. Results for the PlantVillage dataset are reported in Table 4. As can be seen, Y-GAN is again the top performer with an AUC of 0.962, outperforming the state-of-the-art runner-up CS-Flow by more than 6%. Y-GAN exceeds the detection accuracy of other models with respect to normal and anomalous samples and generates fewer misses on average as shown by the TPR and TNR scores. However, we also observe that the (global) calibration of the models results in unbalanced TPR and TNR values for certain classes (e.g., Corn, Grape, Potato). Despite these calibration issues, Y-GAN performs best on average, even if the individual TPR and TNR scores are considered. We attribute this performance to the implemented dual data representation strategy, that allows for excluding irrelevant data characteristics when deciding whether a sample is anomalous or not.

Cross-dataset analysis. If we look at the result across all experimental datasets, it is worth noting that Y-GAN is the clear top performer on all four datasets, whereas CS-Flow is consistently the runner-up. These two models appear to be the most robust to various data characteristics and perform well across the different anomaly detection tasks. Other models favor certain data characteristics and detection problems. OCGAN, for example, is very competitive on the MNIST dataset, but less so on the more challenging CIFAR10 and PlantVillage datasets. Similarly, f-AnoGAN produces among the highest detection scores with the k-classes-out datasets, i.e., MNIST, FMNIST and CIFAR10,

Model	Type†	Error	Apple	Blueberry	Cherry	Corn	Grape	Orange	Peach	Pepper	Potato	Raspberry	Soybean	Squash	Strawberry	Tomato	AUC
GANomaly (Akçay et al., 2019b)	RB	TNR	0.228	0.482	**0.953**	0.644	**0.941**	/	0.375	0.635	**0.968**	0.853	**0.911**	/	0.750	0.680	0.781
		TPR	0.630	/	0.436	0.826	0.864	**0.926**	0.784	0.790	0.601	/	/	0.214	0.711	0.656	
Skip-GANomaly (Akçay et al., 2019a)	RB	TNR	0.675	**0.847**	0.842	0.253	**0.929**	/	0.222	0.774	0.742	**0.867**	0.876	/	**0.880**	0.085	0.746
		TPR	0.632	/	**0.832**	**0.989**	0.309	0.843	0.917	0.578	0.569	/	/	**0.969**	0.390	0.656	
OCGAN (Perera et al., 2019)	RB	TNR	0.584	0.266	0.877	0.605	0.882	/	0.111	0.338	0.484	0.640	0.763	/	0.315	0.389	0.608
		TPR	0.562	/	0.216	0.758	0.793	0.426	0.598	**0.943**	0.736	/	/	0.201	0.886	0.547	
f-AnoGAN (Schlegl et al., 2019)	RB	TNR	0.375	0.645	0.579	0.224	0.847	/	0.486	0.625	0.581	0.840	0.767	/	0.587	0.260	0.623
		TPR	0.592	/	0.465	0.772	0.282	0.574	0.655	0.541	0.474	/	/	0.845	0.192	0.631	
P-Net (Zhou et al., 2020)	RB	TNR	0.495	0.532	0.520	0.528	0.529	/	0.444	0.530	0.613	0.467	0.526	/	0.533	0.489	0.524
		TPR	0.508	/	0.520	0.517	0.519	0.511	0.532	0.524	0.505	/	/	0.525	0.526	0.517	
MOCCA (Massoli et al., 2022)	PC	TNR	0.371	0.571	0.596	0.103	0.671	/	0.986	0.645	0.839	0.840	0.586	/	0.250	0.922	0.635
		TPR	0.394	/	0.603	0.583	0.450	0.382	0.641	0.567	0.533	/	/	0.456	0.925	0.591	
ARNet (Fei et al., 2021)	PT	TNR	0.608	0.698	0.889	0.116	0.824	/	0.000	0.767	0.774	0.813	0.846	/	0.511	0.633	0.736
		TPR	0.586	/	0.669	0.979	**0.928**	0.237	0.909	0.782	0.760	/	/	0.835	0.905	0.672	
Patch SVDD (Yi & Yoon, 2020)	PT	TNR	0.672	0.299	0.708	0.682	**0.929**	/	0.375	0.578	0.774	0.707	0.730	/	0.326	0.404	0.670
		TPR	0.512	/	0.339	0.772	0.746	0.512	0.659	0.857	0.687	/	/	0.335	**0.933**	0.593	
PaDiM (Defard et al., 2021)	PC	TNR	0.605	0.751	0.409	0.176	0.529	/	0.111	0.507	0.645	**0.867**	0.796	/	0.674	0.580	0.671
		TPR	0.736	/	0.526	0.966	0.878	0.120	**0.928**	**0.887**	0.754	/	/	0.882	0.835	0.556	
CS-Flow (Rudolph et al., 2022)	PC	TNR	**0.745**	0.714	0.906	**0.785**	0.824	/	**0.778**	**0.834**	**0.871**	0.800	0.887	/	0.674	**0.831**	**0.905**
		TPR	**0.819**	/	0.737	**0.986**	0.908	0.903	0.881	0.794	**0.882**	/	/	0.928	**0.914**	**0.764**	
Y-GAN [Ours]	RB*	TNR	**0.833**	**0.993**	**0.965**	**1.000**	0.847	/	**0.847**	**0.770**	0.677	**0.893**	**0.945**	/	**0.772**	**0.950**	**0.962**
		TPR	**0.802**	/	**0.909**	0.800	**0.943**	**0.964**	**0.934**	0.795	**0.932**	/	/	**0.946**	0.722	**0.926**	

† RB - reconstruction-based, PT - proxy task-based, PC - utilizing pre-trained classification models ; RB* - reconstruction based but with a latent proxy task

Table 4: PlantVillage results in terms of per-class TPR and TNR scores and mean AUC values over all classes. The best model in each column and for each performance score is marked blue, the runner-up is marked red. The TPR and TNR scores were computed at a (global) decision threshold defined by the equal error rate (EER) on the training data of all classes. Note that the two scores are not necessarily well calibrated for category in the dataset – see results for Corn, Grape, or Potato leaves.

but is much less sucessfuly with the pixel-level anomalies present in the PlantVilage dataset. This suggests that the anomaly detection mechnisms in many of the competing models are quite specific and suited best for certain problems, while the dual data representation scheme of Y-GAN is applicable equally well to different types of anomaly detectiont tasks and across a variety of datasets.

6.2. Time complexity analysis.

In the next series of experiments, we explore the computational complexity of Y-GAN and the 10 competing models by measuring the inference time needed for the processing of an image of size 32×32 pixels. Here, we run the anomaly detection task over 100 images and then report the average time as a results. To ensure a fair comparison, all experiments are conducted on the same hardware (a desktop PC with an i7-8700K CPU and a GeForce RTX 2080 Ti GPU) and the official code repositories of the models. To illustrate the trade-off between run-time performance and accuracy, we also report the average AUC performance of the models across three experimental datasets that contain images of size 32×32 pixels, i.e. MNIST, FMNIST, and CIFAR10. As can be seen from Figure 10, MOCCA has the shortest (average) inference time of only 0.15 ms per image, but the average AUC score achieved by this model is among the lowest. The runner-up in terms of speed, GANomaly, has an inference time of 0.56 ms and a very

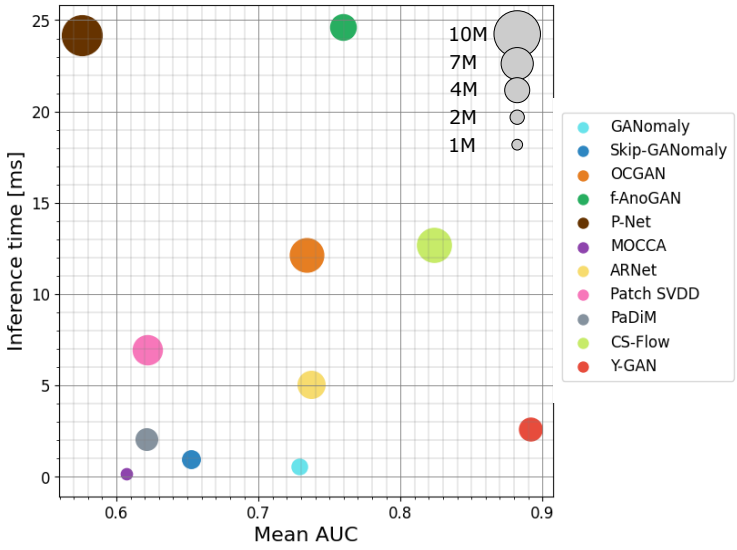


Figure 10: Computational complexity of Y-GAN and the competing models, along with their respective AUC scores and number of parameters. The reported inference time of the individual models represents the (average) processing time needed for one image of size 32×32. The AUC score is calculated as the average performance, achieved across three experimental datasets, i.e. MNIST, FMNIST, CIFAR10. Y-GAN, as the best performing model in terms of average anomaly detection accuracy, is observed to also have a very competitive inference time of only 2.6 ms per image on average.

competitive performance score, similar to those of ARNet and OCGAN, which take 5.1 ms and 12.1 ms, respectively. Y-GAN is among the top 5 fastest models, with a processing time of 2.6 ms per image (with non-optimized code). At the same time, we observe that the average AUC score of Y-GAN is at least 18% higher than any of the faster models. For comparison, CS-Flow, the second-best performing model, has an inference time of 12.7 ms, which is more than $4.5\times$ longer than Y-GAN.

6.3. Ablation Study

To demonstrate the importance of different components of Y-GAN, we perform a two-part ablation study, where we first remove various parts of the learning objective $\mathcal{L}_{\mathcal{G}}$, and then ablate parts of the model architecture.

Impact of Loss Terms. For the first part of the ablation study, three Y-GAN variants are trained with different versions of the generator loss $\mathcal{L}_{\mathcal{G}}$: (i) $\mathcal{L}_{\mathcal{G}}$ without the consistency loss $\mathcal{L}_{con}$ (A1), (ii) $\mathcal{L}_{\mathcal{G}}$ without the (gradient reversal) residual information loss $\mathcal{L}_r$ (A2), and (iii) $\mathcal{L}_{\mathcal{G}}$ without both $\mathcal{L}_{con}$ and $\mathcal{L}_r$ (A3). The results in Table 5 show that the removal of both loss terms causes a considerable drop in the AUC scores on FMNIST, CIFAR10, and PlantVillage. Compared to MNIST, where a smaller drop is observed, these three datasets contain a greater amount of residual, semantically-irrelevant information (e.g., various clothing prints, background style, etc.). In such cases, both disentanglement terms, $\mathcal{L}_{con}$ and $\mathcal{L}_r$, play a significant role in the extraction of semantically-relevant

Ablation study	MNIST	FMNIST	CIFAR10	PlantVillage
Complete Y-GAN	0.987	0.925	0.763	0.962
A1: $\mathcal{L}_{\mathcal{G}}$ w/o $\mathcal{L}_{con}$	0.987	0.890	0.745	0.918
A2: $\mathcal{L}_{\mathcal{G}}$ w/o $\mathcal{L}_{r}$	0.980	0.832	0.682	0.865
A3: $\mathcal{L}_{\mathcal{G}}$ w/o $\mathcal{L}_{r}$ and $\mathcal{L}_{con}$	0.962	0.823	0.660	0.861

Table 5: Y-GAN ablation study on the impact of the learning objectives used. Results are reported in the form of AUC scores.

information. Although the two losses are complementary, $\mathcal{L}_r$ results in a slightly greater performance drop than $\mathcal{L}_{con}$ when removed from the training objective. These results suggest that all loss terms are important and contribute to the performance of Y-GAN.

Impact of Architecture. Y-GAN is designed as an autoencoder with two/dual encoders, one decoder, a latent classifier and a discriminator. Six versions of the model are implemented to highlight the importance of the design choices made around this topology: (i) a model without the dual encoders (a single encoder E is used), where the generated latent representation $z = E(x) \in \mathbb{R}^{2d}$ is split into two equally-sized vectors, $z_s \in \mathbb{R}^d$ and $z_r \in \mathbb{R}^d$, on top of which $\mathcal{L}_{\mathcal{G}}$ (10) is applied (B1), (ii) the model from B1, but with a wider encoder, where the number of filters in each layer have been doubled, so the encoder has approximately the same number of parameters as E_r and E_s together (B2), (iii) the model from B1, but with a deeper encoder, where the number of layers at each stage has been doubled, so the encoder again has approximately the same number of parameters as E_r and E_s together (B3), (iv) the model from B1 but with a single (entangled) latent representation - no z_r and associated losses ($\mathcal{L}_r$, $\mathcal{L}_{con}$) are used (B4), (v) the model from B4 but without the classifier C, i.e., an auto-encoder with a reconstruction-based anomaly score (B5), and (vi) the proposed Y-GAN model without the adversarial discriminator Ds, i.e., a dual encoder generator trained without $\mathcal{L}_{adv}$ (B6). The results in Table 5 suggest that the removal of the dual encoder increasingly impacts results, as the complexity of the data (from an anomaly detection point of view) increases, that is, the biggest performance drop is observed on the most challenging data, i.e., PlantVillage. The results in B1, thus, suggest that the Y-shaped architecture with the two encoders contributes to a more effective disentanglement of semantically-relevant and residual information, compared to the use of a single encoder. It can also be seen (from B4) that using a single entangled latent space representation is detrimental for the performance of the anomaly detection task, especially for the more challenging CIFAR10 and PlantVillage datasets. The disentanglement of irrelevant information and its removal from the decision-making process is, hence, critical for the success of Y-GAN. From the results in B2 and B3, we also observe that a simple increase in model complexity (in terms of parameters) does not impact performance in a consistent and meaningful way. While results are improved slightly on PlantVillage through the heavier models in B2 and B3 compared to the single-encoder setting in B1, we see performance degradations on the remaining three datasets. Y-GAN, on the other hand, is always the top performaned due to its effective disentaglement strategy. The exclusion of the classifier also causes a large decrease in the overall anomaly detection accuracy across

Ablation study	MNIST	FMNIST	CIFAR10	PlantVillage
Complete Y-GAN	0.987	0.925	0.763	0.962
B1: Y-GAN w/o dual encoders	0.979	0.881	0.734	0.915
B2: B1 with doubled filters	0.968	0.872	0.698	0.932
B3: B1 with doubled layers	0.973	0.876	0.707	0.935
B4: B1 w/o z_r	0.956	0.810	0.659	0.807
B5: B1 w/o z_r and C	0.640	0.689	0.531	0.610
B6: Y-GAN w/o Ds	0.982	0.896	0.719	0.921

Table 6: Y-GAN ablation study on the impact of the architectural components. Results are reported in the form of AUC scores.

all datasets, suggesting that steered representation learning is key for Y-GAN - see B5 results. Finally, training the originally proposed Y-GAN auto-encoder without the adversarial discriminator, seems to have the least significant impact on the overall detection accuracy, in comparison to other Y-GAN components (B6). Nevertheless, it does contribute to the quality of the generated image reconstructions, which can further affect the performance of the disentanglement process in more complex images, e.g., in CIFAR10 and PlantVillage.

6.4. Model Characteristics

Next, we illustrate some of the characteristics of Y-GAN through a series of additional experiments that target: (i) the application of the proposed model in practical scenarios with (unlabeled normal) training data, where the number of sub-classes N and the corresponding (sub)class assignments may not be available and need to be defined automatically, and (ii) the definition of the anomaly score through the latent classifier C, as opposed to other alternatives used in the literature.

Unlabeled Normal Data. Y-GAN assumes that the (normal) training data comes from multiple sub-classes/groups and that labels for these sub-classes are readily available. This assumption allows for the inclusion of the latent classifier C in the training process, which was shown to be critical for the overall performance of Y-GAN. Here, we show that it is possible to relax this assumption and train Y-GAN with unlabeled training data in a completely unsupervised manner. To this end, we run a (unsupervised) clustering procedure over the unlabeled training data and use the generated cluster assignments as *pseudo (sub-class) labels* that can be utilized when learning Y-GAN. Specifically, we first compute feature representations from the training images used in the given experimental run with the EfficientNet-B4 (Tan & Le, 2019) model, pre-trained on ImageNet. Given input images x, the model then generates 1792-dimensional representations for the experiments. For efficiency reasons, we reduce the dimensionality of these representations to 100 using Principal Component Analysis (PCA) (Turk & Pentland, 1991) and cluster the data with k-means. We then determine the optimal number of clusters (i.e., the number of subclasses N) based on the *average silhouette method* (Kaufman & Rousseeuw, 2009). Finally, we utilize the generated cluster assignments

Model	MNIST	FMNIST	CIFAR10	PlantVillage
Y-GAN (ground truth labels)	0.987	0.925	0.763	0.962
Y-GAN* (pseudo labels)	0.964	0.892	0.733	0.846

Table 7: Mean AUC scores generated with Y-GAN models trained with: (i) manually assigned ground truth labels on the sub-class structure of the normal training data, and (ii) pseudo labels generated through a k-means clustering procedure in an unsupervised manner. Note that even in a completely unsupervised setting, where (noisy) pseudo labels are inferred directly from the normal training data, Y-GAN generates state-of-the-art results that are highly competitive in comparison to the baseline models on all four experimental datasets.

as pseudo labels for Y-GAN training. We note that this procedure is completely unsupervised and does not rely on any kind of prior annotations of human intervention. The term *pseudo labels* is, hence, used to refer to automatically generated cluster assignments for the training procedure.

- **K–Classes–Out Results.** On MNIST, FMNIST, and CIFAR10, the clustering procedure identifies either 9 or 10 clusters, i.e., $N = 9$ or $N = 10$, in any given experimental run ($N = 9$ classes are in fact represented). An analysis of the generated clusters shows that 95% of MNIST samples in each cluster share the same ground truth. This percentage is bit lower in FMNIST and CIFAR10, where it equals 91% and 87%, respectively. This suggests that the clustering reasonably well approximates the actual data classes, but also that part of the data is not assigned correct class labels. A comparison between Y-GAN trained with the ground truth labels and the pseudo labels generated with the clustering procedure in an unsupervised fashion (denoted as Y-GAN*) is presented in Table 7. As can be seen, the pseudo labels result in slight performance degradations compared to the original Y-GAN. However, the Y-GAN* model achieves competitive results on all three datasets and still yields highly encouraging results compared to the baselines evaluated in Tables 1, 2 and 3. The presented results suggest that learning data representations through a latent (proxy) classifier that considers differences between different sub-classes of normal training data is beneficial for performance, even if the sub-classes are not necessarily homogeneous and contain label noise.

- **PlantVillage Results.** For this dataset, 12 distinct clusters are identified by k-means, i.e., N=12, which is corresponds to the number of original categories that include non-anomalous/normal samples. However, the partitioning of the PlantVillage data is in this case less accurate in comparison to the k-classes-out datasets, due to similarities between objects from different classes. After the clustering process, only 80% of PlantVillage images representing the same plant species share the same ground truth label. Although such noisy pseudo labels decrease the anomaly detection performance by approximately 12%, Y-GAN* learned without any human supervision still leads to highly competitive performance compared to the competing models from Table 4. Overall, these results support the observation that a well-performing Y-GAN model can be trained even without access to (manually generated) ground

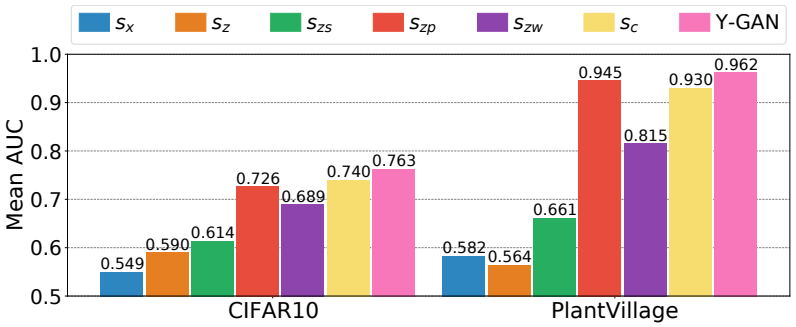


Figure 11: Score analysis on the CIFAR10 and PlantVillage datasets. Various anomaly scores, defined in the image space (s_x), over the latent representations (s_z, s_{zs}), or based on the sub-class structure of the normal data (s_{zp}, s_{zw}, s_c, and s), are explored to better understand the expressive power of different representations generated within Y-GAN. Note that the proposed anomaly score achieves the strongest performance on both datasets.

truth class labels for the normal training data.

Anomaly Score Analysis. The proposed Y-GAN employs a unique anomaly scoring method for the detection of anomalous data, derived from the output of the latent classifier C. Our approach therefore differs from previous methodologies, which often rely on metrics like the disparity between input images and their reconstructions, or utilize L_p norms across latent representations, as seen in works like Perera et al. (2019) and Akçay et al. (2019a), among others. In this section, we compare the proposed anomaly scoring method used in Y-GAN to several other commonly used alternatives. These experiments are meant to provide additional insight into the model and the characteristics of various generated data representations. The analysis is performed using the datasets CIFAR10 and PlantVillage, chosen due to their greater complexity in comparison to the other two datasets utilized in other experiments. To this end, we implement 6 different alternative anomaly scores, defined as follows:

- Image score, s_x:

$$s_x(x, \hat{x}) = ||x - \hat{x}||_2^2, \tag{13}$$

  where the anomaly score is computed-based on the reconstruction quality. Here, x is the input image and $\hat{x}$ is the reconstruction generated by Y-GAN.

- Latent score, s_z:

$$s_z(z, \hat{z}) = ||z - \hat{z}||_2^2, \tag{14}$$

  where the anomaly score is computed in the latent space using the combined latent representations $z = E_s(x) \oplus E_r(x)$ and $\hat{z} = E_s(\hat{x}) \oplus E_r(\hat{x})$. $\oplus$ is a concatenation operator.

- Semantic latent score, s_{zs}:

$$s_{zs}(z_s, \hat{z_s}) = ||z_s - \hat{z_s}||_2^2, \tag{15}$$

where the score is computer-based on the semantic latent space representation only, i.e., $z_s = E_s(x)$ and $\hat{z}_s = E_s(\hat{x})$.

- Prototype-based semantic latent score with ground truth labels, s_{zp}:

$$s_{zp}(z_s, z_s^{(C_i)}) = \min_i ||z_s - z_s^{(C_i)}||_2^2, \tag{16}$$

where the (semantically-meaningful) latent probe vector z_s is compared to the prototypes $z_s^{(C_i)} = 1/|C_i| \sum_{z_s \in C_i} z_s$, computed for the N sub-classes of the normal training data $\{C_i\}_{i=1}^N$ and the minimum distance is used as the anomaly score. $|\cdot|$ is the cardinality of the class.

- Prototype-based semantic latent score with pseudo class labels, s_{zw}:

$$s_{zw}(z_s, z_s^{(C_i^*)}) = \min_i ||z_s - z_s^{(C_i^*)}||_2^2, \tag{17}$$

where the latent probe vector z_s is compared to the class prototypes $z_s^{(C_i^*)} = 1/|C_i^*| \sum_{z_s \in C_i^*} z_s$, computed for the N sub-classes of the normal training data $\{C_i^*\}_{i=1}^N$ defined through k-means clustering. The minimum distance over all class protoypes is used as the anomaly score.

- Classifier uncertainty, s_c:

$$s_c = -\sum_i p_i \log p_i, \tag{18}$$

where $p = [p_1, p_2, \ldots, p_N] \in \mathbb{R}^N$ is the probability distribution for the N sub-classes of the normal data computed by subjecting the output of the latent classifier C to a softmax function given an input probe sample x.

We note that all latent representations used in the above definitions of latent scores are normalized to unit norm prior to score calculation. The results, presented in Fig. 11, show that a simple reconstruction-based score (s_x) results in modest performance in both datasets. The latent space score s_z is slightly more informative on CIFAR10, but generates weaker results on image from PlantVillage. If the residual latent space is removed from the decision making process, we observe additional improvements on both datasets. Thus, the anomaly score defined in the semantically meaningful latent space s_{zs} already ensures better results on CIFAR10 than all of the competing state-of-the-art models evaluated in Table 3 and yields comparable detection results as a large portion of the tested models on PlantVillage. If anomaly scores are defined by also considering the sub–classes present in the (normal) training data (i.e., s_{zp}, s_{zw}, s_c, and the proposed Y-GAN score s), we see another significant jump in AUC results on both datasets, which suggest that the structure (or distribution) of the normal data is an important source of information that can be exploited to improve anomaly detection performance.

Sensitivity Analysis. In all experiments presented so far, Y-GAN was trained with default values for the balancing parameters in Eqs. (10) and (11), i.e., $\lambda_1 = \lambda_5 = 50$, and $\lambda_2 = \lambda_3 = \lambda_4 = \lambda_6 = 1$. In this section, we now conduct a sensitivity analysis to explore how changes from these default values (during training) affect the performance of the

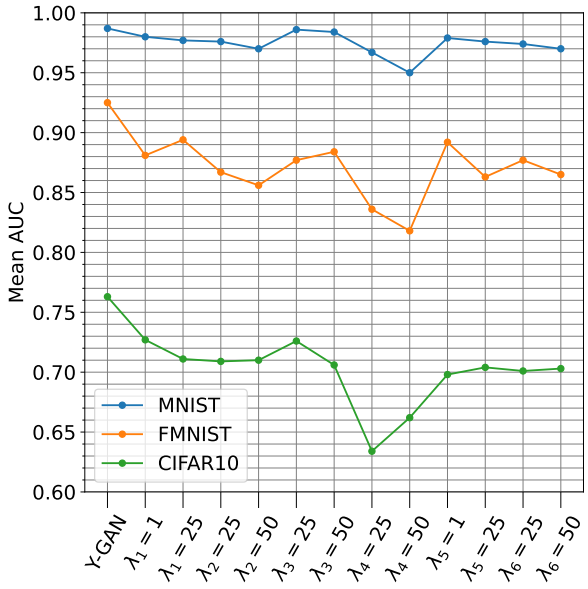


Figure 12: Impact of changes in λ values (from Eqs. (10) and (11)) on the overall anomaly detection performance on MNIST, FMNIST and CIFAR10. The results are reported in terms of mean AUC scores.

model. To this end, we change a single parameter at the time, train the proposed model and then run experiments on the MNIST, FMNIS and CIFAR10 datasets. From Figure 12 it can be seen that any change from the default setting typically leads to weaker performance. Especially detremental appears to be the increase in the importance of the gradient reversal loss in relation to the reconstruction and classification losses. Because the model focuses mostly on inducing missclassifciations through z_r, instead of ensuring proper information encoding through the reconstruction and classificaiton losses, the trained model does not ensure competitive performance on any of the three datasets. Similarly, disrupting the balance of the loss terms through other modifications, in general, also does not help with performance. Overall, these results suggest that suitable balance needs to be ensured between the different learning objectives to ensure optimal results with Y-GAN.

6.5. Visual/Qualitative Evaluation

Grad-CAM Visualizations. To gain better insight into the behavior of the proposed Y-GAN model, we use Grad-CAM visualizations (Selvaraju et al., 2017), to find out which image regions are most informative with respect to the anomaly detection task. Grad-CAM heatmaps are calculated using the gradients of the latent (proxy) classifier C, utilized for the computation of the anomaly classification scores. Examples of correctly classified normal and anomalous samples (marked red) with their corresponding Grad-CAM heatmaps laid on top are shown in Figure 13. As can be seen, in datasets with homogeneous backgrounds and small intra-class variability, such as MNIST and FMNIST, the model tends to focus on the global appearance of the objects, i.e. their overall shape. Conversely, detection on datasets with more complex visual data (such as CIFAR10) is primarily based on local and distinctive

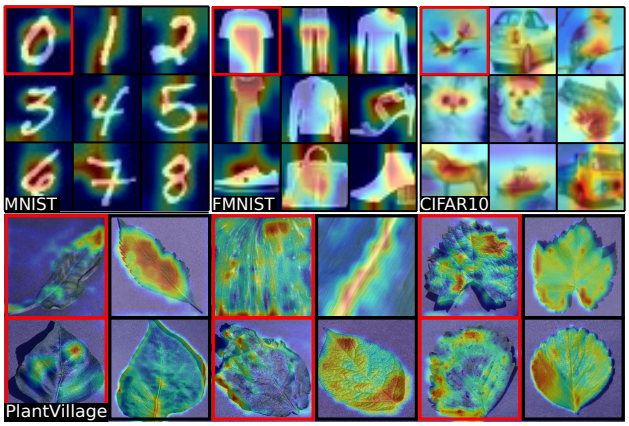


Figure 13: Selected images of correctly detected normal and anomalous samples (marked with red boxes) with their corresponding Grad-CAM visualizations (Selvaraju et al., 2017) laid on top. As can be seen from the Grad-CAM heatmaps, the model focuses on the global appearance of objects coming from low intra-class variability, such as MNIST and FMNIST, while local and distinctive spatial characteristics are more informative in samples from CIFAR10. On PlantVillage, on the other hand, Y-GAN appears to be simultaneously focusing on both, global and local object characteristics, due to the large intra-class variability of non-anomalous samples.

(a) Anomaly: Digit 4 (b) Anomaly: Pullover (c) Anomaly: Horse

Figure 14: Examples of edge cases with the k-classes-out datasets (MNIST, FMNIST, CIFAR10). Undetected normal samples in the top two rows exhibit visual similarities with the anomalous class or are poor representatives of normal data. Similarly, the appearance of undetected anomalous samples in the bottom row (red) is close to the appearance of classes in the normal data.

object and texture characteristics. Different from MNIST, FMNIST, and CIFAR10, PlantVillage exhibits relatively large intra-class variability in terms of the size, shape, illumination, and orientation of the leaves in the images. Anomalies in this dataset can appear either as inconsistencies in the overall shape and color or impact the texture of the leaves at an arbitrary spatial location in this dataset. Therefore, both global and local image characteristics appear to play an important role in the detection of anomalous leaves, as seen in Figure 13. Interestingly, Y-GAN is able to adapt to the anomaly detection task and learn descriptive and informative features from the input data regardless of whether these features correspond to global or local (or both) image characteristics.

Visual Evaluation. To better understand why the model fails to classify certain normal and anomalous samples,

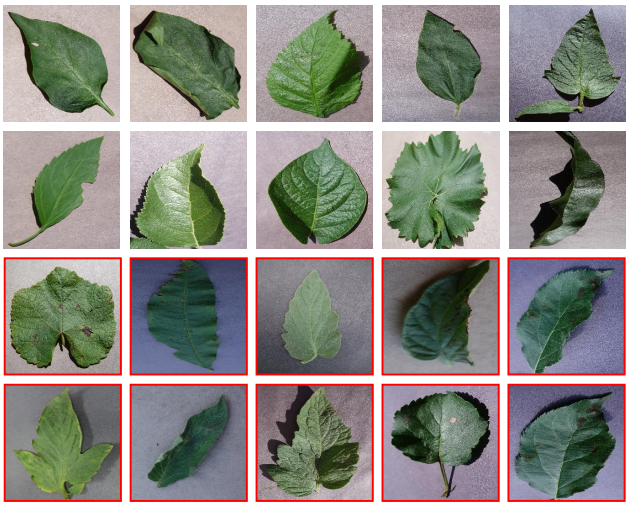


Figure 15: Sample images of undetected normal and anomalous samples (marked red) from the PlantVillage dataset. Misclassifications among non-anomalous data samples occur due to uncommon leaf shapes, holes in healthy leaves, and unusual positions/orientations. Within anomalous samples, problems appear due to subtle, often unnoticeable changes in leaf color or local textures.

we perform an additional visual inspection of edge cases from all four experimental datasets. In Figure 14 we show selected edge samples from the k-classes out datasets MNIST, FMNIST, and CIFAR10. As can be seen, the undetected normal samples represent objects with uncommon appearance, that differs from the rest of the non-anomalous data samples in terms of object structure and texture, i.e., oddly-shaped digits for MNIST, ambiguous fashion classes for FMNIST, and unusual object appearances for CIFAR10. Difficult anomalous samples, conversely, often resemble certain classes from the normal data or exhibit ambiguous appearance. Fig. 15 presents edge cases for the PlanVillage dataset. Here, severely folded healthy leaves and distorted leaf shapes are often detected as anomalous. Similar outcomes are also observed with leaves with holes, although such holes do not necessarily indicate an illness. Shadows darkening various parts of non-anomalous leaves can also trick the model into misclassifying normal samples. Conversely, undetected anomalies typically represent subtle, unnoticeable changes in the leaf color or local textures.

7. Conclusion

The paper introduced a reconstruction-oriented auto-encoder based anomaly detection model, called Y-GAN. Different from competing approaches, Y-GAN learns to disentangle image characteristics that are relevant for representing normal data from irrelevant residual data characteristics and derives anomaly scores from selectively encoded image information. The model was shown to significantly outperform several state-of-the-art anomaly detection models on the MNIST, FMNIST, CIFAR10 and PlantVillage datasets and provide the most consistent performance across different anomaly detection tasks among all tested models. As part of our future work, we plan to extend the model, so it allows for additional functionality, such as anomaly localization/segmentation, which is of interest for various

anomaly detection tasks.

Acknowledgements

Supported in parts by the ARRS Research Program P2-0250 (B) and the ARRS Research Project DeepFake DAD.

References

Abati, D., Porrello, A., Calderara, S., & Cucchiara, R. (2019). Latent Space Autoregression for Novelty Detection. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 481–490).

Akçay, S., Abarghouei, A. A., & Breckon, T. P. (2019a). Skip–GANomaly: Skip Connected and Adversarially Trained Encoder-Decoder Anomaly Detection. In *International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8).

Akçay, S., Atapour-Abarghouei, A., & Breckon, T. P. (2019b). GANomaly: Semi–supervised Anomaly Detection via Adversarial Training. In *Asian Computer Vision Conference (ACCV)* (pp. 622–637).

Bakalos, N., Doulamis, N., Doulamis, A. D., & Makantasis, K. (2022). Multi-property Tensor-Based Learning for Abnormal Event Detection. In *17th International Symposium on Advances in Visual Computing (ISVC)* (pp. 325–335).

Bergman, L., & Hoshen, Y. (2020). Classification-Based Anomaly Detection for General Data. In *International Conference on Learning Representations (ICLR)* (pp. 1–12).

Bergmann, P., Fauser, M., Sattlegger, D., & Steger, C. (2020). Uninformed Students: Student–Teacher Anomaly Detection With Discriminative Latent Embeddings. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 4183–4192).

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Computing Surveys*, *41*, 1–58.

Cohen, N., & Hoshen, Y. (2020). Sub-Image Anomaly Detection with Deep Pyramid Correspondences. *arXiv preprint arXiv:2005.02357*, (pp. 1–17).

Collin, A., & Vleeschouwer, C. D. (2021). Improved Anomaly Detection by Training an Autoencoder With Skip Connections on Images Corrupted With Stain-Shaped Noise. In *International Conference on Pattern Recognition (ICPR)* (pp. 7915–7922).

Defard, T., Setkov, A., Loesch, A., & Audigier, R. (2021). PaDiM: A Patch Distribution Modeling Framework for Anomaly Detection and Localization. In *International Conference on Pattern Recognition (ICPR)* (pp. 475–489).

Doshi, K., & Yilmaz, Y. (2020). Continual Learning for Anomaly Detection in Surveillance Videos. In *Computer Vision and Pattern Recognition Workshops (CVPR–W)* (pp. 1–10).

Dosovitskiy, A., & Brox, T. (2016). Generating Images with Perceptual Similarity Metrics Based on Deep Networks. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 658—666).

Fatemifar, S., Awais, M., Arashloo, S. R., & Kittler, J. (2019). Combining Multiple One–Class Classifiers for Anomaly Based Face Spoofing Attack Detection. In *International Conference on Biometrics (ICB)* (pp. 1–7).

Fei, Y., Huang, C., Jinkun, C., Li, M., Zhang, Y., & Lu, C. (2021). Attribute Restoration Framework for Anomaly Detection. *IEEE Transactions on Multimedia*, (pp. 1–12).

Ganin, Y., & Lempitsky, V. (2015). Unsupervised Domain Adaptation by Backpropagation. In *International Conference on Machine Learning (ICML)* (pp. 1180–1189).

Gidaris, S., Singh, P., & Komodakis, N. (2018). Unsupervised Representation Learning by Predicting Image Rotations. In *International Conference on Learning Representations (ICLR)* (pp. 1–16).

Golan, I., & El-Yaniv, R. (2018). Deep Anomaly Detection Using Geometric Transformations. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 1–17).

Gong, D., Liu, L., Le, V., Saha, B., Mansour, M. R., Venkatesh, S., & Hengel, A. v. d. (2019). Memorizing Normality to Detect Anomaly: Memory–Augmented Deep Autoencoder for Unsupervised Anomaly Detection. In *International Conference on Computer Vision (ICCV)* (pp. 1705–1714).

Haselmann, M., Gruber, D. P., & Tabatabai, P. (2018). Anomaly Detection Using Deep Learning Based Image Completion. In *International Conference on Machine Learning and Applications (ICMLA)* (pp. 1237–1242).

Hendrycks, D., Mazeika, M., Kadavath, S., & Song, D. (2019). Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 1–12).

Hughes, D. P., & Salathé, M. (2015). An Open Access Repository of Images on Plant Health to Enable the Development of Mobile Disease Diagnostics Through Machine Learning and Crowdsourcing. *arXiv preprint arXiv:1511.08060*, (pp. 1–13).

Ionescu, R. T., Khan, F. S., Georgescu, M.-I., & Shao, L. (2019). Object-Centric Auto-Encoders and Dummy Anomalies for Abnormal Event Detection in Video. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 7842–7851).

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image–To–Image Translation With Conditional Adversarial Networks. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 1125–1134).

Ivanovska, M., & Štruc, V. (2023). Face Morphing Attack Detection with Denoising Diffusion Probabilistic Models. In *International Workshop on Biometrics and Forensics (IWBF)* (pp. 1–6).

Kaufman, L., & Rousseeuw, P. J. (2009). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, New Yersey.

Khalid, H., & Woo, S. S. (2020). OC–FakeDect: Classifying Deepfakes Using One–class Variational Autoencoder. In *Computer Vision and Pattern Recognition Workshops (CVPR–W)* (pp. 2794–2803).

Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)* (pp. 1–15).

Krizhevsky, A. (2009). *Learning Multiple Layers of Features from Tiny Images*. Master's thesis University of Toronto.

Lanckriet, G. R., El Ghaoui, L., & Jordan, M. I. (2003). Robust Novelty Detection with Single-class MPM. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 929–936).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient–Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, *86*, 2278–2324.

LeCun, Y., Cortes, C., & Burges, C. J. C. (). The MNIST Database of Handwritten Digits. `http://yann.lecun.com/exdb/mnist/`. Accessed: 2021-03-14.

Lu, C., Shi, J., & Jia, J. (2013). Abnormal Event Detection at 150 FPS in Matlab. In *International Conference on Computer Vision (ICCV)* (pp. 2720–2727).

Van der Maaten, L., & Hinton, G. (2008). Visualizing Data Using *t*–SNE. *Journal of Machine Learning Research*, *9*, 2579–2605.

Mai, K. T., Davies, T., & Griffin, L. D. (2021). Brittle Features May Help Anomaly Detection. *arXiv preprint arXiv:2104.10453*, (pp. 1–9).

Markovitz, A., Sharir, G., Friedman, I., Zelnik-Manor, L., & Avidan, S. (2020). Graph embedded pose clustering for anomaly detection. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 10539–10547).

Massoli, F. V., Falchi, F., Kantarci, A., Akti, S., Ekenel, H. K., & Amato, G. (2022). MOCCA: Multilayer One-Class Classification for Anomaly Detection. *IEEE Transactions on Neural Networks and Learning Systems*, *33*, 2313–2323.

Nguyen, T.-N., & Meunier, J. (2019). Anomaly Detection in Video Sequence With Appearance–Motion Correspondence. In *International Conference on Computer Vision (ICCV)* (pp. 1273–1283).

Noroozi, M., & Favaro, P. (2016). Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. In *European Conference on Computer Vision (ECCV)* (pp. 69–84).

Oza, P., & Patel, V. M. (2019). Active Authentication using an Autoencoder Regularized CNN-based One–Class Classifier. In *Automatic Face & Gesture Recognition (FG)* (pp. 1–8).

Pang, G., Shen, C., Cao, L., & Hengel, A. V. D. (2021). Deep Learning for Anomaly Detection: A Review. *ACM Computing Surveys*, *54*, 1–38.

Pang, G., Yan, C., Shen, C., Hengel, A. v. d., & Bai, X. (2020). Self–Trained Deep Ordinal Regression for End-to-End Video Anomaly Detection. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 12173–12182).

Park, H., Noh, J., & Ham, B. (2020). Learning Memory–Guided Normality for Anomaly Detection. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 14372–14381).

Perera, P., Nallapati, R., & Xiang, B. (2019). OCGAN: One–Class Novelty Detection Using GANs With Constrained Latent Representations. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 2898–2906).

Perera, P., Oza, P., & Patel, V. M. (2021). One–Class Classification: A Survey. *arXiv preprint arXiv:2101.03064*, (pp. 1–19).

Perera, P., & Patel, V. M. (2019). Learning Deep Features for One-Class Classification. *IEEE Transactions on Image Processing*, *28*, 5450–5463.

Racki, D., Tomazevic, D., & Skocaj, D. (2018). A Compact Convolutional Neural Network for Textured Surface Anomaly Detection. In *Winter Conference on Applications of Computer Vision (WACV)* (pp. 1331–1339).

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *International Conference on Learning Representations (ICLR)* (pp. 1–16).

Reiss, T., Cohen, N., Bergman, L., & Hoshen, Y. (2021). PANDA: Adapting Pretrained Features for Anomaly Detection and Segmentation. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 2806–2814).

Rippel, O., Mertens, P., & Merhof, D. (2020). Modeling the Distribution of Normal Data in Pre–trained Deep Features for Anomaly Detection. In *International Conference on Pattern Recognition (ICPR)* (pp. 6726–6733).

Ristea, N.-C., Madan, N., Ionescu, R. T., Nasrollahi, K., Khan, F. S., Moeslund, T. B., & Shah, M. (2022). Self-Supervised Predictive Convolutional Attentive Block for Anomaly Detection. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 13576–13586).

Ronneberger, O., Fischer, P., & Brox, T. (2015). U–Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (pp. 234–241).

Roth, K., Pemula, L., Zepeda, J., Schölkopf, B., Brox, T., & Gehler, P. (2022). Towards Total Recall in Industrial Anomaly Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 14318–14328).

Rudolph, M., Wehrbein, T., Rosenhahn, B., & Wandt, B. (2022). Fully Convolutional Cross-Scale-Flows for Image-Based Defect Detection. In *Winter Conference on Applications of Computer Vision (WACV)* (pp. 1088–1097).

Ruff, L., Kauffmann, J. R., Vandermeulen, R. A., Montavon, G., Samek, W., Kloft, M., Dietterich, T. G., & Müller, K.-R. (2021). A Unifying Review of Deep and Shallow Anomaly Detection. *Proceedings of the IEEE*, (pp. 1–40).

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., & Kloft, M. (2018). Deep One-Class Classification. In *International Conference on Machine Learning (ICML)* (pp. 4393–4402).

Sabokrou, M., Fayyaz, M., Fathy, M., & Klette, R. (2017). Deep-Cascade: Cascading 3d Deep Neural Networks for Fast Anomaly Detection and Localization in Crowded Scenes. *IEEE Transactions on Image Processing*, *26*, 1992–2004.

Salehi, M., Sadjadi, N., Baselizadeh, S., Rohban, M. H., & Rabiee, H. R. (2021). Multiresolution Knowledge Distillation for Anomaly Detection. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 14902–14912).

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., & Chen, X. (2016). Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 1–9).

Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., & Langs, G. (2017). Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. In *Information Processing in Medical Imaging (IPMI)* (pp. 146–157).

Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., & Langs, G. (2019). f–AnoGAN: Fast Unsupervised Anomaly Detection with Generative Adversarial Networks. *Medical Image Analysis*, *54*, 30–44.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization. In *International Conference on Computer Vision (ICCV)* (pp. 618–626).

Sultani, W., Chen, C., & Shah, M. (2018). Real-World Anomaly Detection in Surveillance Videos. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 6479–6488).

Tan, M., & Le, Q. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning (ICML)* (pp. 6105–6114).

Tang, S., He, F., Huang, X., & Yang, J. (2019). Online PCB Defect Detector on a New PCB Defect Dataset. *arXiv preprint arXiv:1902.06197*, (pp. 1–5).

Tax, D. M., & Duin, R. P. (2004). Support Vector Data Description. *Machine Learning*, *54*, 45–66.

Thakare, K. V., Sharma, N., Dogra, D. P., Choi, H., & Kim, I.-J. (2022). A Multi-Stream Deep Neural Network With Late Fuzzy Fusion for Real-World Anomaly Detection. *Expert Systems with Applications*, *201*, 117030.

Turk, M., & Pentland, A. (1991). Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, *3*, 71–86.

Wang, G., Han, S., Ding, E., & Huang, D. (2021a). Student-Teacher Feature Pyramid Matching for Anomaly Detection. In *British Machine Vision Conference (BMVC)* (pp. 1–14).

Wang, R., Juefei-Xu, F., Ma, L., Xie, X., Huang, Y., Wang, J., & Liu, Y. (2020). FakeSpotter: A Simple yet Robust Baseline for Spotting AI–Synthesized Fake Faces. In *International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 3444–3451).

Wang, S., Wu, L., Cui, L., & Shen, Y. (2021b). Glancing at the Patch: Anomaly Localization With Global and Local Feature Comparison. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 254–263).

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*, (pp. 1–6).

Xu, H., Caramanis, C., & Sanghavi, S. (2012). Robust PCA via Outlier Pursuit. *IEEE Transactions on Information Theory*, *58*, 3047–3064.

Yadav, S., Chen, C., & Ross, A. (2020). Relativistic Discriminator: A One–Class Classifier for Generalized Iris Presentation Attack Detection. In *Winter Conference on Applications of Computer Vision (WACV)* (pp. 2635–2644).

Yamanishi, K., Takeuchi, J.-I., Williams, G., & Milne, P. (2004). On-line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms. *Data Mining and Knowledge Discovery*, *8*, 275–300.

Yang, Z., Zhang, T., Bozchalooi, I. S., & Darve, E. (2021). Memory-Augmented Generative Adversarial Networks for Anomaly Detection. *IEEE Transactions on Neural Networks and Learning Systems*, (pp. 1–11).

Yi, J., & Yoon, S. (2020). Patch SVDD: Patch-level SVDD for Anomaly Detection and Segmentation. In *Asian Conference on Computer Vision (ACCV)* (pp. 1–16).

Zaheer, M. Z., Lee, J.-H., Astrid, M., & Lee, S.-I. (2020). Old Is Gold: Redefining the Adversarially Learned One–Class Classifier Training Paradigm. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 14183–14193).

Zavrtanik, V., Kristan, M., & Skočaj, D. (2021a). Reconstruction by Inpainting for Visual Anomaly Detection. *Pattern Recognition*, *112*, 1–9.

Zavrtanik, V., Kristan, M., & Skočaj, D. (2021b). DRÆM – A Discriminatively Trained Reconstruction Embedding for Surface Anomaly Detection. In *International Conference on Computer Vision (ICCV)* (pp. 8330–8339).

Zenati, H., Romain, M., Foo, C., Lecouat, B., & Chandrasekhar, V. (2018). Adversarially Learned Anomaly Detection. In *International Conference on Data Mining (ICDM)* (pp. 727–736).

Zhao, B., Fei-Fei, L., & Xing, E. P. (2011). Online Detection of Unusual Events in Videos via Dynamic Sparse Coding. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 3313–3320).

Zhou, K., Li, J., Xiao, Y., Yang, J., Cheng, J., Liu, W., Luo, W., Liu, J., & Gao, S. (2021). Memorizing Structure-Texture Correspondence for Image Anomaly Detection. *IEEE Transactions on Neural Networks and Learning Systems*, (pp. 1–15).

Zhou, K., Xiao, Y., Yang, J., Cheng, J., Liu, W., Luo, W., Gu, Z., Liu, J., & Gao, S. (2020). Encoding Structure–Texture Relation with P–Net for Anomaly Detection in Retinal Images. In *European Conference on Computer Vision (ECCV)* (pp. 360–377).