The Graduate University for Advanced Studies,

SOKENDAI

DOCTORAL THESIS

Anomaly Detection by Generating Pseudo Anomalous Data

Author:

Hironori MURASE

Supervisor:

Dr. Kenji FUKUMIZU

A thesis submitted in fulfillment of the requirements

for the degree of Doctor of Philosophy

in the

School of Multidisciplinary Sciences

Department of Statistical Science

September, 2022

The Graduate University for Advanced Studies,

SOKENDAI

Abstract

School of Multidisciplinary Sciences

Department of Statistical Science

Doctor of Philosophy

Anomaly Detection by Generating Pseudo Anomalous Data

by Hironori MURASE

Closely related to outlier and novelty detection, anomaly detection refers to the technique of distinguishing between unexpected and normal data. Practical examples include fraud detection, medical diagnosis, surveillance, and optical inspection, and a common feature of these applications is the discovery of undesirable data.

The difficulty with anomaly detection is that, in many cases, anomalous data are rarely observed and are of a wide variety; hence, the learning of anomaly detection models suffers from the difficulty of imbalanced or one-class classification. Additionally, as the preparation of large amounts of anomalous data is difficult, training without anomalous data is a preferable approach. Therefore, the focus of this thesis is upon anomaly detection using only normal training data. A wide variety of methods have already been proposed in this field based on traditional machine learning and statistical techniques, such as one-class classification, likelihood, nearest neighbors, and clustering.

Recently, deep learning methods such as representation learning have been successfully applied to anomaly detection without using anomalous data for training. Taking advantage of the effective representation of deep learning, the features obtained by a pre-trained model, such as VGG and ResNet, can also be applied to unsupervised anomaly detection. Since deep generative models are able to learn probability distributions of normal data, they have been combined for anomaly detection in various ways.

Generative models approximate the true data distribution of observed samples with probabilistic models. However, generative modeling of high-dimensional data distributions such as images is difficult, and hence generative models using deep learning methods have been studied. Most previous anomaly detection studies using deep generative models have taken advantage of the model characteristic of generating only normal samples. By contrast, only a few have focused on generating outlier samples and adding them to training.

In addition to the generative model network, some deep generative models also use subnetworks such as an encoder and a discriminator for training. We aim to utilize these subnetworks to improve the efficiency of anomaly detection.

In this thesis, we propose a method that improves anomaly detection performance by generating pseudo-anomalous data from only normal training data using Generative Adversarial Networks (GANs). Unlike the standard usage of GANs, the generator used in the proposed method provides pseudo-anomalous data and fake-normal data by introducing anomalous states in the latent variable; this model is known as Anomalous Latent GAN (ALGAN). Note that the discriminator of a standard GAN is not necessarily suitable for distinguishing between normal and anomalous data. It is trained to discriminate between real and fake data such that in successful learning, the two classes are almost similar. By contrast, when training is successful, the discriminator of ALGAN distinguishes between the group of real-normal data and the group of fake-normal and pseudo-anomalous data.

We introduce two types of pseudo-anomalous data for training. The first type of pseudo-anomalous data is called fake-anomalous data. ALGAN utilizes the anomalous latent variables with a larger variance to generate fake-anomalous data. The other type of pseudo-anomalous data is called buffered data, which are defined as generated samples during the early stage of the training process. These are expected to differ from the normal training (real-normal) data.

The proposed method follows an adversarial training procedure. It provides a discrimination boundary not only for the real-normal and fake-normal data but also for the real-normal and pseudo-anomalous data, the latter of which has a broader support of the distribution. As the training progresses, the

generator produces samples that resemble real-normal data, and the discriminator cannot distinguish between real-normal and fake-normal data. The pseudo-anomalous data are clearly different from the real-normal data; therefore, the discrimination boundary of the discriminator is used to classify them.

The proposed method for generating pseudo-anomalous data can be applied to both images and feature vectors. We applied it to three anomaly detection benchmarks and demonstrated its high accuracy. On MVTec-AD, ALGAN-image achieved more than 10% higher average accuracy than conventional image-based methods, and ALGAN-feature exhibited comparable ability to the feature-based methods. On the COIL-100 dataset, ALGAN performed almost perfectly.

Real-time prediction is significant to apply anomaly detection in the real world, where the data generation speed has increased. Reducing computational costs will contribute to the expansion of the application. The proposed ALGAN exhibited remarkably fast predictions. Compared with conventional methods trained on image data and features, ALGAN could predict up to tens of times faster while maintaining high performance.

Acknowledgements

My research project would not have been possible without the support of many people.

My supervisor, Dr. Kenji Fukumizu, has provided tremendous support and advice for my research project. For his contribution, I would like to express my sincere gratitude.

I would like to thank my thesis committee, Dr. Ryo Yoshida, Dr. Hironori Fujisawa, and Dr. Makoto Yamada, for their many thought-provoking suggestions.

I would also like to thank the many people at my workplace who graciously agreed to allow me to continue my studies for my degree.

I would not be who I am today without Dr. Mutsumi Yoshino, who first introduced me to the excitement of statistical machine learning.

I was able to pursue a happy research life with the support of my wife Yui, my daughters Shion and Kanon, my father Noritaka, and my mother Takayo. I would like to express my gratitude to all of them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Machine learning has been researched and applied in academic and industrial fields in recent years, and many applications have demonstrated remarkable performance. These results are attributed to the handling of a large amount of data and high-dimensional data based on advances in information technology. The Internet of Things (IoT), which connects familiar devices to the Internet, is growing in popularity, and the data types and collection frequency are increasing as a result.

Real-world machine learning applications suffer from data imbalance problems (Guo et al., 2008; Johnson and Khoshgoftaar, 2019). For example, while manufacturing industries can now utilize machine learning methods and IoT data to improve manufacturing, the collected data becomes more biased as process capability increases. This could mean that as more facilities are improved, it can become more difficult to collect data on equipment and product failures.

In the most severe case, the data to be distinguished can only rarely be obtained or are unavailable entirely. Clearly, manufacturing industries are not viable if they have equipment that breaks down frequently and processes

that produce large amounts of defective products, yet it remains difficult to predict what troubles will occur on a newly launched production line over time. In order to collect such data in advance, it is necessary to investigate possible problems and collect data by artificially generating similar situations.

1.2 Motivation

Anomaly detection refers to the technique of distinguishing between unexpected and normal data and is closely related to outlier detection and novelty detection (Chandola, Banerjee, and Kumar, 2009). Practical examples include fraud detection to identify unauthorized access (Rodda and Erothi, 2016), medical diagnosis to discover lesion sites from medical images (Schlegl et al., 2017), surveillance to find suspicious behavior in real-time videos (Sabokrou et al., 2015), and optical inspection for detecting defects in industrial products (Bergmann et al., 2019a). A common feature of these applications is the discovery of undesirable data.

The difficulty with anomaly detection is that, in many cases, anomalous data are rarely observed and are of a wide variety; hence, the learning of anomaly detection models suffers from the difficulty of imbalanced or one-class classification. Various methods have been proposed to address this issue, such as creating a dataset containing new anomalous data not included in conventional datasets (Perales Gómez et al., 2019), verifying classification performance for anomalous data that are rarely observed (Rodda and Erothi, 2016), and developing a new one-class classifier for complex data such as image and sequence data (Chalapathy, Menon, and Chawla, 2018). The augmentation of anomalous data from different sources has also been proposed (e.g., out-of-distribution data, Kawachi, Koizumi, and Harada, 2018; Hendrycks,

Mazeika, and Dietterich, 2018); however, in such cases, undesired biases may be introduced.

This thesis discusses anomaly detection using only normal training data. Because the preparation of a large amount of anomalous data is difficult, training without anomalous data is a preferable approach. In this line of research, a wide variety of methods have already been proposed based on traditional machine learning and statistical techniques, such as one-class classification, likelihood, nearest neighbors, and clustering. See Chandola, Banerjee, and Kumar (2009) for a comprehensive survey of traditional approaches.

Recently, deep learning methods such as representation learning have been successfully applied to anomaly detection without using anomalous data for training. See Chalapathy and Chawla (2019), Ruff et al. (2021), and Pang et al. (2021) for a comprehensive survey of the deep learning approaches. Taking advantage of the effective representation of deep learning, the features obtained by a pre-trained model, such as VGG (Simonyan and Zisserman, 2015) and ResNet (He et al., 2016), are applied to unsupervised anomaly detection (Andrews et al., 2016). Since deep generative models can learn probability distributions of normal data, they have been combined for anomaly detection in various ways (Schlegl et al., 2017; Zenati et al., 2018; Akçay, Atapour-Abarghouei, and Breckon, 2018; Akçay, Atapour-Abarghouei, and Breckon, 2019; Sabokrou et al., 2018; Liu et al., 2020b; Fan et al., 2020). More details on anomaly detection without using anomalous data for training are presented in Chapter 3.

1.3 Research Goal

Generative models approximate the true data distribution of observed samples with probabilistic models. However, generative modeling of high-dimensional

data distributions such as images is difficult, and generative models using deep learning methods have been studied (Kingma and Welling, 2014; Goodfellow et al., 2014; Rezende and Mohamed, 2015; Sohl-Dickstein et al., 2015; LeCun and Huang, 2005; Larochelle and Murray, 2011; Song and Ermon, 2019). Due to the limited number of methods that can calculate the likelihood directly (Rezende and Mohamed, 2015; Larochelle and Murray, 2011), deep generative models are difficult to apply directly to anomaly detection, and applied methods are developed.

Most previous anomaly detection studies using deep generative models have taken advantage of the model characteristic of generating only normal samples (Schlegl et al., 2017; Zenati et al., 2018; Liu et al., 2020b; Fan et al., 2020). By contrast, few have focused on generating outlier samples and adding them to training.

In addition to the generative model network, most deep generative models use sub-networks such as an encoder and a discriminator during training, and combine the generative model network and the sub-networks for anomaly detection. To the best of our knowledge, there is no method for anomaly detection using only sub-networks.

The use of multiple networks increases the number of parameters and consequently increases the computational cost during training and prediction. Although there are several methods for reducing computational cost during prediction, such as pruning (Han et al., 2015; Han, Mao, and Dally, 2016), knowledge distillation (Hinton, Vinyals, Dean, et al., 2015; Urban et al., 2017), and quantization (Jacob et al., 2018; Krishnamoorthi, 2018), their application to anomaly detection using deep generative models is limited (Zhang, Chen, and Sun, 2021). Applying these techniques after training is inefficient because it requires second steps of training to fine-tune the parameters.

Therefore, the research questions of this thesis are as follows:

- Can outliers be generated from a generative model of normal data and used for anomaly detection?
- Can the sub-networks used in training deep generative models be exploited for anomaly detection directly?
- Is there a more efficient way?

In this thesis, we propose a method that improves anomaly detection performance by generating pseudo-anomalous data from only normal training data using Generative Adversarial Networks (GANs, (Goodfellow et al., 2014)). Unlike the standard usage of GANs, the generator used in the proposed method provides pseudo-anomalous data as well as fake-normal data, by introducing anomalous states in the latent variable. We call this model Anomalous Latent GAN (ALGAN). Note that the discriminator of a standard GAN is not necessarily suitable for distinguishing between normal and anomalous data. It is trained to discriminate between real and fake data such that in successful learning, the two classes are almost similar. By contrast, when training is successful, the discriminator of ALGAN distinguishes between the group of real-normal data and the group of fake-normal and pseudo-anomalous data.

Some state-of-the-art anomaly detection methods specialize in product appearance inspection from images. For example, DifferNet (Rudolph, Wandt, and Rosenhahn, 2021) concatenates feature vectors from three different resolution images and uses them to train a model; furthermore, PatchCore (Roth et al., 2022) uses the hierarchical patches of features to achieve fast and effective performance. However, these methods presume a pre-trained model and cannot be trained directly from the image data. By contrast, the proposed ALGAN can be trained using both images and features.

The proposed ALGAN can detect anomalies using only a discriminator and

achieves fast prediction times without using methods that reduce computational costs, which requires two stages of training. Normal training data can be collected efficiently; however, training time increases with the amount of training data. In addition, real-time prediction is significant in the real world (Sabokrou et al., 2015), where the data generation speed has increased (Ahmad et al., 2017). Additionally, reducing computational costs will contribute to the expansion of the application (Menghani, 2021).

The contributions of this study are as follows:

- We propose a novel method for generating pseudo-anomalous data: adding pseudo-anomalous data to GAN training improves the anomaly detection performance of the discriminator.
- The proposed method can be applied to both images and feature vectors. Experimental results show that it achieves a state-of-the-art performance compared with image-based methods and comparable ability to feature-based methods.
- The proposed ALGAN achieved a remarkably fast prediction time, 10.4 to 54.6 times faster than other image-based methods on the benchmark MVTec-AD dataset.

1.4 Outline

The remainder of this thesis is organized as follows: In Chapter 2, we introduce the theoretical background of generative models and GANs. In Chapter 3, we review relevant anomaly detection methods and explain the evaluation metric for anomaly detection performance. First, likelihood-based traditional supervised and unsupervised anomaly detection are explained; then, GAN-based anomaly detection methods and recently developed pre-trained model's feature-based anomaly detection methods. Third, we explain the

evaluation metric for anomaly detection performance. In Chapter 4, we begin by explaining related works using pseudo-anomalous data, which an immature GAN generator generated or image processed (cut and pasted). Next, we explain the proposed method details for detecting anomalies using GANs by generating pseudo-anomalous data. Moreover, we provide an intuitive understanding of pseudo-anomalous data. In Chapter 5, firstly, we provide the implementation details, datasets, and evaluation method. We examine the anomaly detection performances on various datasets and present the computation time, stability, ablation study, and additional experiments results. Secondly, we explain hyperparameter selection which cannot use validation data or can use a small amount of validation data. Thirdly, we evaluate the impact of differences between normal and anomalous latent variables on anomaly detection performance. In Chapter 6, we discuss the advantages of this study, possible future work, and conclusions.

Chapter 2

Theoretical Background

2.1 Generative Model

A generative model is a model of the data generating process that follows a certain probability distribution. The generative model is trained by minimizing a measure of the difference between the two distributions, which are real and generated data. As a foundation, this section uses Kullback–Leibler (KL) divergence, a typical divergence to measure the discrepancy, for discussion.

Let x, $p_{data}(x)$, and $p_\theta(x)$ be the observed variable, the true data distribution, and the probabilistic model, respectively. KL divergence is used to measure similarity between the two probability distributions, and it is defined as follows:

$$D_{KL}(p_{data}(x)\|p_\theta(x)) = \int p_{data}(x) \log \frac{p_{data}(x)}{p_\theta(x)} dx. \tag{2.1}$$

The probabilistic model is trained to minimize the KL divergence. Eq. (2.1) is written by

$$\begin{aligned} D_{KL}(p_{data}(x)\|p_\theta(x)) &= \int p_{data}(x)\log\frac{p_{data}(x)}{p_\theta(x)}dx \\ &= \int p_{data}(x)\log p_{data}(x)dx - \int p_{data}(x)\log p_\theta(x)dx \\ &= \mathbb{E}_{x\sim p_{data}(x)}[\log p_{data}(x)] - \mathbb{E}_{x\sim p_{data}(x)}[\log p_\theta(x)]. \end{aligned} \tag{2.2}$$

The first term of Eq. (2.2) is constant. Therefore, maximizing the second term of Eq. (2.2) is equivalent to minimizing the KL divergence:

$$\begin{aligned} \theta^* &= \arg\min_\theta D_{KL}(p_{data}(x)\|p_\theta(x)) \\ &= \arg\max_\theta \mathbb{E}_{x\sim p_{data}(x)}[\log p_\theta(x)], \end{aligned} \tag{2.3}$$

where θ^* are optimal parameters for $p_\theta(x)$.

The true data distribution $p_{data}(x)$ is unknown; thus, observed variables x are used to approximate the expectation:

$$\mathbb{E}_{x\sim p_{data}(x)}[\log p_\theta(x)] \simeq \frac{1}{N}\sum_{n=1}^{N}\log p_\theta(x_n). \tag{2.4}$$

From Eq. (2.3) and (2.4):

$$\begin{aligned} \theta^* &= \arg\max_\theta \frac{1}{N}\sum_{n=1}^{N}\log p_\theta(x_n) \\ &= \arg\max_\theta \log\prod_{n=1}^{N} p_\theta(x_n) \\ &= \arg\max_\theta \log p(\mathcal{D}|\theta) := \arg\max_\theta \log L(\theta), \end{aligned} \tag{2.5}$$

where $\mathcal{D} = \{x_1, ..., x_N\}$ is a set of training samples from the data distribution, and L is likelihood function. This method is known as the maximum

likelihood estimation. To evaluate Eq. (2.5), we need to be able to express $p_\theta(x)$ with parameter θ explicitly (Mohamed and Lakshminarayanan, 2016; Grover, Dhar, and Ermon, 2018).

2.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs, Goodfellow et al., 2014) are the implicit generative model using neural networks and replace distribution modeling with a discrimination problem: the generator $G(z;\theta)$ maps latent variables z to the data space, and the discriminator $D(x;\phi)$ distinguishes between real data x and the generated samples $x' = G(z)$. The discriminator outputs the probability of the input, and real data and generated samples are labeled with 1 and 0, respectively. The discriminator is trained to maximize the log-likelihoods $\log(D(x))$ and $\log(1 - D(G(z)))$. Conversely, the generator is trained to minimize $\log(1 - D(G(z)))$ to fool the discriminator. Both network objectives are given by the following equations:

$$\begin{aligned} \max_D V(D) &= \Big(\mathbb{E}_{x\sim p_{data}(x)}[\log(D(x))] \\ &\quad + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]\Big), \end{aligned} \tag{2.6}$$

$$\min_G V(G) = \Big(\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]\Big), \tag{2.7}$$

where p_{data} and p_z denote the distributions of real data and latent variables, respectively.

When the discriminator can no longer distinguish between real data and generated samples, the generator approximately realizes a sampler from the real data distribution. The objective function of the GANs is obtained by combining Eq. (2.6) and (2.7) as follows:

$$\begin{aligned} \min_G \max_D V(D,G) &= \Big(\mathbb{E}_{x\sim p_{data}(x)}[\log(D(x))] \\ &\quad + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]\Big). \end{aligned} \tag{2.8}$$

In Eq. (2.8), when the generator p_g is fixed, the optimal discriminator D^* is given by

$$D^*(x) = \frac{p_{data}}{p_{data} + p_g}. \tag{2.9}$$

Substituting Eq. (2.9) into Eq. (2.8) leads to following the equation:

$$\begin{aligned}
C(G) =& \max_D V(G, D) \\
=& \mathbb{E}_{p_{data}} \left[\log \frac{p_{data}}{p_{data} + p_g} \right] + \mathbb{E}_{p_g} \left[\log \left(1 - \frac{p_{data}}{p_{data} + p_g} \right) \right] \\
=& \int p_{data} \log \frac{p_{data}}{p_{data} + p_g} dx + \int p_g \log \frac{p_g}{p_{data} + p_g} dx \\
=& \int p_{data} \log \frac{2 \cdot p_{data}}{p_{data} + p_g} dx - \int p_{data} \log 2 \, dx \\
&+ \int p_g \log \frac{2 \cdot p_g}{p_{data} + p_g} dx - \int p_g \log 2 \, dx \\
=& \int p_{data} \log \frac{2 \cdot p_{data}}{p_{data} + p_g} dx + \int p_g \log \frac{2 \cdot p_g}{p_{data} + p_g} dx - 2 \log 2 \\
=& 2 \, D_{JS}(p_{data} \| p_g) - 2 \log 2,
\end{aligned} \tag{2.10}$$

where D_{JS} is the Jensen–Shannon (JS) divergence:

$$D_{JS}(p_{data} \| p_g) = \frac{1}{2} D_{KL} \left(p_{data} \| \frac{p_{data} + p_g}{2} \right) + \frac{1}{2} D_{KL} \left(p_g \| \frac{p_{data} + p_g}{2} \right). \tag{2.11}$$

From the above, the generator minimizes the JS divergence under the optimal discriminator.

Given the optimal generator, the discriminator can distinguish between two similar classes. Thus, it is nontrivial to determine whether such a discriminator is suitable for identifying real and anomalous data. When GANs are trained on a dataset $\mathcal{D}$ that includes only normal data, the discriminator has poor discrimination performance on anomalous data (Schlegl et al., 2017). This suggests that the discrimination boundary of GANs is not specialized

in the one-class classification of anomaly detection.

Chapter 3

Anomaly Detection

3.1 Traditional Approach

To begin with, we consider the case where probabilistic models are obtained for the observed data.

3.1.1 Supervised Anomaly Detection

Let us consider the labeled data $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n), ..., (x_N, y_N)\}$ where labels $y = 0$ and $y = 1$ are defined as normal and anomalous, respectively. An anomaly detector $p(y|x, \mathcal{D})$ is a binary classifier trained by $\mathcal{D}$. The classifier parameter w is expressed by the posterior distribution as follows:

$$\begin{aligned} p(\mathcal{D}, w) = p(w|\mathcal{D})p(\mathcal{D}) = p(\mathcal{D}|w)p(w), \\ p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})}, \end{aligned} \tag{3.1}$$

where $p(w)$ is the prior distribution of w.

The likelihood and prior distribution in Eq. (3.1) can be written by:

$$\begin{aligned} p(\mathcal{D}|w)p(w) &= p(w)\prod_{n=1}^{N} p(x_n, y_n|w) \\ &= p(w)\prod_{n=1}^{N} p(y_n|x_n, w)p(x_n). \end{aligned} \tag{3.2}$$

Because the model evidence $p(\mathcal{D})$ and $p(x_n)$ are the functions that not include w, they can be expressed as follows:

$$p(w|\mathcal{D}) \propto p(w)\prod_{n=1}^{N} p(y_n|x_n, w). \tag{3.3}$$

From the above, the classifier is obtained by:

$$p(y|x, \mathcal{D}) = \int p(y|x, w)p(w|\mathcal{D})dw. \tag{3.4}$$

If the classification error is minimized in supervised anomaly detection, the decision rule is defined as follows:

$$\textit{if } \log\frac{p(y=1|x,\mathcal{D})}{p(y=0|x,\mathcal{D})} > 0, \textit{ then } y=1, \textit{else } y=0. \tag{3.5}$$

However, in many cases $p(y=1) \ll p(y=0)$, the anomalous can be detected hardly in this threshold. We think conditional distribution $p(x|y)$ when labels are given. Then, a density ratio or likelihood ratio is used for anomaly score.

$$\textit{if } \log\frac{p(x|y=1,\mathcal{D})}{p(x|y=0,\mathcal{D})} > \tau, \textit{ then } y=1, \textit{else } y=0, \tag{3.6}$$

where τ is a threshold for anomaly detection.

3.1.2 Unsupervised Anomaly Detection

Let us consider the data $\mathcal{D} = \{x_1, ..., x_N\}$ that are not provided labels. Then, we assume that the provided data are not included anomalous data. We estimate the probabilistic model with $\mathcal{D}$, and calculate the negative log-likelihood, so we can detect anomalous.

$$if \ -\log p(x|\mathcal{D}) > \tau, \ then \ y = 1, else \ y = 0, \tag{3.7}$$

where τ is a threshold for anomaly detection.

3.1.3 Issues with Traditional Methods

Traditional anomaly detection methods may not be able to handle large amounts of data or high-dimensional data. For example, probabilistic methods such as kernel density estimation (KDE) (Rosenblatt, 1956; Parzen, 1962) and Gaussian mixture model (GMM) are difficult to apply to high-dimensional data because it is difficult to estimate high-dimensional probability distributions. Distance-based methods such as k-nearest neighbors, k-means (MacQueen, 1967), and local outlier factor (LOF) (Breunig et al., 2000) suffer from the curse of dimensionality. In contrast, one-class support vector machines (OC-SVM) (Schölkopf et al., 2001) face the problem of computational costs as data volume increases.

In recent years, deep learning that can efficiently learn from large amounts of data has been applied to anomaly detection (Chalapathy and Chawla, 2019; Ruff et al., 2021; Pang et al., 2021). Generative models based on deep learning have made it possible to handle probability distributions of high-dimensional data. Feature representations obtained by deep learning overcome the curse of dimensionality and are combined with traditional methods. Section 3.2 presents the former deep generative model, while Section 3.3

presents the latter combination of the feature representation from deep learning and traditional methods.

3.2 GAN-Based Anomaly Detection

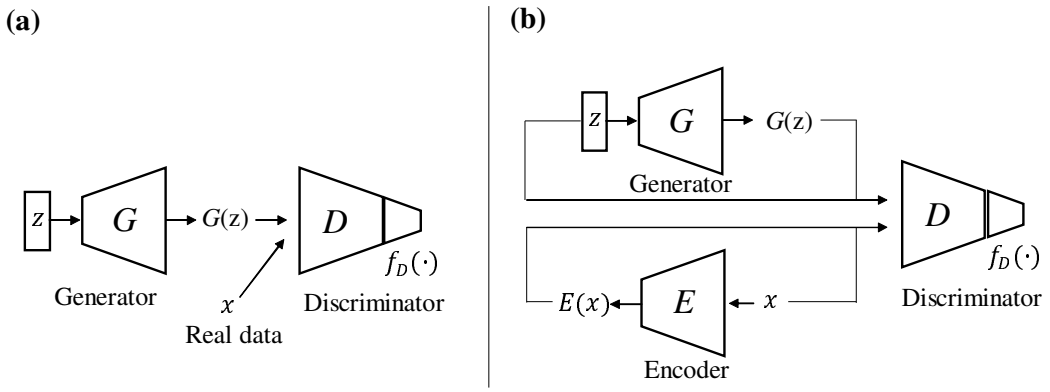

FIGURE 3.1: Overviews of AnoGAN and Efficient GAN-based Anomaly Detection (EGBAD). (a) AnoGAN is trained by only normal data with standard GAN training procedure. The latent variable z corresponding to given test data x is estimated for anomaly detection. (b) EGBAD is trained using only normal data in the manner of BiGAN. The Encoder E estimates the latent variable z corresponding to given test data x.

Anomaly detection methods using GANs can be divided into two categories: those using reconstruction errors and those using one-class classifiers. In early studies (Schlegl et al., 2017; Zenati et al., 2018), the latent variable corresponding to given test data was estimated, and the reconstruction error of the image generated from the latent variable was used as the anomaly score.

AnoGAN (Schlegl et al., 2017) is the earliest method for anomaly detection using GANs. The generator G and the discriminator D are trained by only normal data with standard GAN training procedure. The latent variable z corresponding to given test data x is estimated for anomaly detection. The latent variable is iteratively updated by a gradient descent method with the loss function consisting of image reconstruction error and the discriminator's

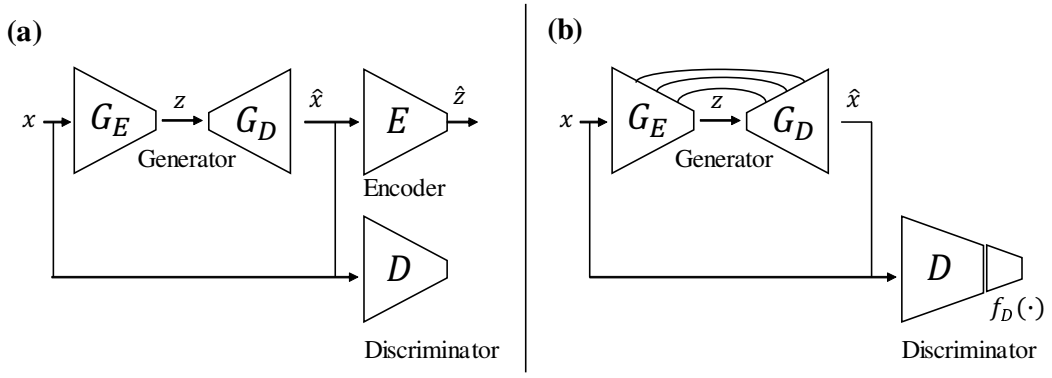

FIGURE 3.2: Overviews of GANomaly and Skip-GANomaly. (a) GANomaly uses two encoders that estimate the latent variables of the input and generated images and detects anomalies using the reconstruction error between the two latent variables. (b) Skip-GANomaly is an improvement that employs U-Net architecture for the GANomaly's generator and detects anomalies using the reconstruction error and the features extracted from the middle layer of the discriminator.

intermediate feature,

$$loss = (1-\lambda)\left\|x-(G(z))\right\|_1 + \lambda\left\|f_D(x)-f_D(G(z))\right\|_1, \tag{3.8}$$

where λ and $f_D(\cdot)$ denote the coefficient of weighted sum and the discriminator's intermediate feature, respectively. The loss function (3.8) is used as the anomaly score.

Efficient GAN-based Anomaly Detection (EGBAD) (Zenati et al., 2018) uses an architecture like BiGAN (Donahue, Krähenbühl, and Darrell, 2017; Dumoulin et al., 2017) to estimate latent variables using an encoder (Fig. 3.1 (b)). In preparation for anomaly detection, the three networks, the generator G, the discriminator D, and the encoder E, are trained in the manner of BiGAN with only normal data. The function A (3.9), similar to AnoGAN's loss function (3.8), is used as the anomaly score.

$$A = (1-\lambda)\left\|x-G(E(x))\right\|_1 + \lambda\left\|f_D(x)-f_D(G(E(x)))\right\|_1. \tag{3.9}$$

GANomaly (Akçay, Atapour-Abarghouei, and Breckon, 2018) uses two encoders that estimate the latent variables of input and generated images. Anomaly detection is performed using the reconstruction error between the two latent variables (Fig. 3.2 (a)).

The generator of GANomaly consists of the encoder G_E and the decoder G_D that encodes image x into latent z, and the latent is reconstructed into image $\hat{x}$. The reconstructed image is again encoded latent $\hat{z}$ by the encoder E. After the training is finished, anomaly detection is performed using the following anomaly score A,

$$A = \|G_E(x) - E(G_D(G_E(x)))\|_2 . \tag{3.10}$$

Skip-GANomaly (Akçay, Atapour-Abarghouei, and Breckon, 2019) is an improvement that employs U-Net (Ronneberger, Fischer, and Brox, 2015) architecture for the encoder–decoder of GANomaly's generator. The generator is trained to reconstruct training data containing only normal data, and the discriminator is trained to distinguish between the training data and the reconstructed data. The anomaly score A is calculated using the reconstruction error and the features extracted from the middle layer of the discriminator,

$$A = \lambda \|x - G_D(G_E(x))\|_1 + (1 - \lambda) \|f_D(x) - f_D(G_D(G_E(x)))\|_2 . \tag{3.11}$$

Adversarially Learned One-Class Classifier (ALOCC) (Sabokrou et al., 2018) exploits reconstructed data and performs anomaly detection with the one-class classification. An encoder–decoder trained only on normal training data can reconstruct normal data but may not be able to reconstruct anomalous data successfully. The generator consisting of the encoder–decoder is trained to reconstruct normal data, and the discriminator learns to discriminate between the training and reconstruction data. The training procedure

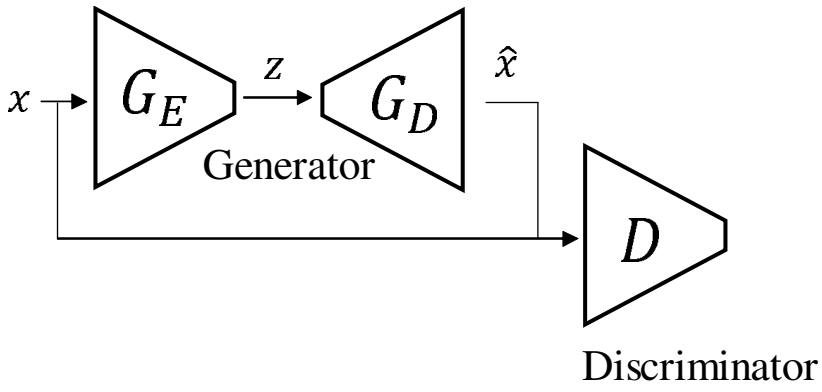

FIGURE 3.3: Overview of Adversarially Learned One-Class Classifier (ALOCC). The ALOCC performs anomaly detection using the discriminator. An encoder–decoder trained only on normal training data may not be able to reconstruct anomalous data successfully. The reconstruction test data is input to the discriminator, and the output is used as an anomaly score.

follows standard GANs procedures. During testing, the test data are input to the generator; the output reconstruction data is input to the discriminator. The discriminator's output is used as an anomaly score for anomaly detection.

$$A = -D(G_D(G_E(x))). \tag{3.12}$$

However, these methods fail to detect anomalies when data reconstruction is successful. By contrast, the proposed method does not depend on the reconstruction error and discriminates directly.

3.3 Anomaly Detection with Pre-Trained Models

Remarkable performance has been demonstrated in a recent work (Bergman, Cohen, and Hoshen, 2020) in which feature representations were exploited from pre-trained models on the ImageNet dataset (Deng et al., 2009).

Mahalanobis-AD (Rippel, Mertens, and Merhof, 2021) performs image-level anomaly detection that an image is normal or anomalous. The hierarchical

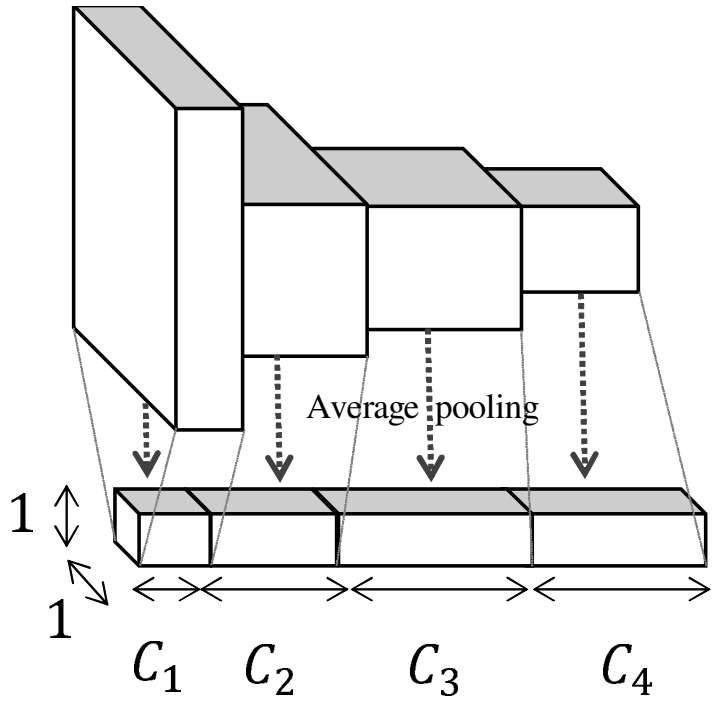

FIGURE 3.4: Feature extraction overview of Mahalanobis-AD. Mahalanobis-AD extracts features from EfficientNet and average-pooling is applied to the extracted features. The mean vector $\mu \in \mathbb{R}^C$ and the covariance matrix $\Sigma \in \mathbb{R}^{C \times C}$ are calculated between the average-pooled features of all training data, where $C \subseteq \{C_1, C_2, C_3, C_4\}$ can be chosen arbitrarily and concatenated if multiple.

features of normal data are extracted from pre-trained EfficientNet (Tan and Le, 2019) and applied average pooling to each channel (Fig. 3.4). The mean vector μ and the covariance matrix Σ on each channel's feature are calculated using normal training data. The anomaly score A is calculated with Mahalanobis distance between multivariate normal distribution $N(\mu, \Sigma)$ and hierarchical features of test data $f_{avg}(x)$ which applied average pooling to each channel.

$$A = \sqrt{(f_{avg}(x) - \mu)^T \Sigma^{-1} (f_{avg}(x) - \mu)}. \tag{3.13}$$

PaDiM (Defard et al., 2021) extends Mahalanobis-AD for pixel-level anomaly detection using hierarchical features separated into patches (Fig. 3.5). Similar to Mahalanobis-AD, the mean vectors $\mu_{i,j}$ and the covariance matrices $\Sigma_{i,j}$ on

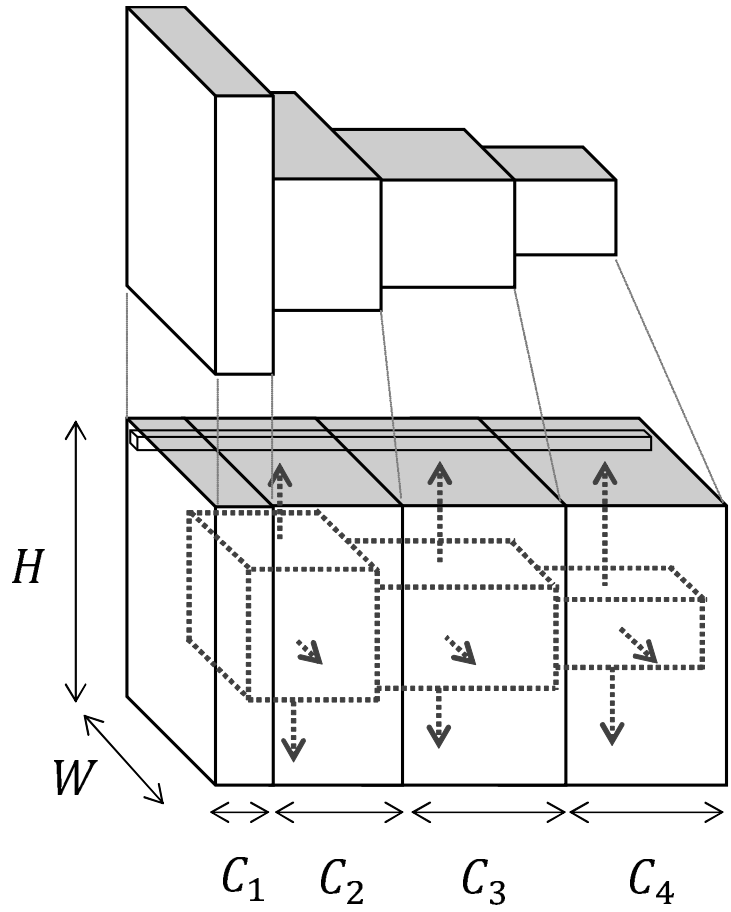

FIGURE 3.5: Feature extraction overview for PaDiM, SPADE, and PatchCore. The features extracted from the pre-trained model are upsampled to the same dimensions and concatenated. The mean vectors $\mu_{i,j} \in \mathbb{R}^C$ and the covariance matrices $\Sigma_{i,j} \in \mathbb{R}^{C \times C}$ are calculated between the features of all training data, where $C \subseteq \{C_1, C_2, C_3, C_4\}$ can be chosen arbitrarily and concatenated if multiple. PatchCore applies 3×3 average-pooling in $[1, 1]$ strides for the feature maps before upsampling to reduce a position sensitivity.

each channel's feature are calculated on each position using normal training data, where $(i, j) \in [1, H] \times [1, W]$ and $H \times W$ denote positions on the feature map and the feature map resolution. The anomaly score on each position $A_{i,j}$ is calculated with Mahalanobis distance between multivariate normal distribution $N(\mu_{i,j}, \Sigma_{i,j})$ and hierarchical features of test data $f(x_{i,j})$.

$$A_{i,j} = \sqrt{(f(x_{i,j}) - \mu_{i,j})^T \Sigma_{i,j}^{-1} (f(x_{i,j}) - \mu_{i,j})}. \tag{3.14}$$

SPADE (Cohen and Hoshen, 2020) also performs pixel-level anomaly detection using hierarchical features separated into patches (Fig. 3.5), but SPADE uses k-nearest neighbors, whereas Mahalanobis-AD and PaDiM use Mahalanobis distance. The memory bank of training data $M_{i,j}$ is created using hierarchical features on each position. K-nearest neighbor distances calculated between the memory bank and test data features are used to the anomaly score according to the following equation,

$$A_{i,j} = \frac{1}{k} \sum_{n=1}^{k} \| M_{i,j}^{n} - f(x_{i,j}) \|_2. \tag{3.15}$$

PatchCore (Roth et al., 2022) subsamples the hierarchical features memory bank of training data and achieves high performance and fast prediction using k-nearest neighbors (Fig. 3.5). The maximum distance among the k samples selected from the memory bank is used as the anomaly score,

$$A_{i,j} = \| M_{i,j}^{*} - f(x_{i,j}) \|_2, \tag{3.16}$$

where $M_{i,j}^{*}$ denotes the maximum distance patch of the hierarchical features in the memory bank.

DifferNet (Rudolph, Wandt, and Rosenhahn, 2021) uses a flow-based model (Dinh, Sohl-Dickstein, and Bengio, 2017), which is typically computation-intensive because each layer is the same as the dimensions of input data. Therefore, pre-trained model features are utilized to reduce data dimensionality. The flow-based model can directly calculate likelihood, which is used for anomaly score.

$$A = -\log p_z(f_{NF}(f(x))), \tag{3.17}$$

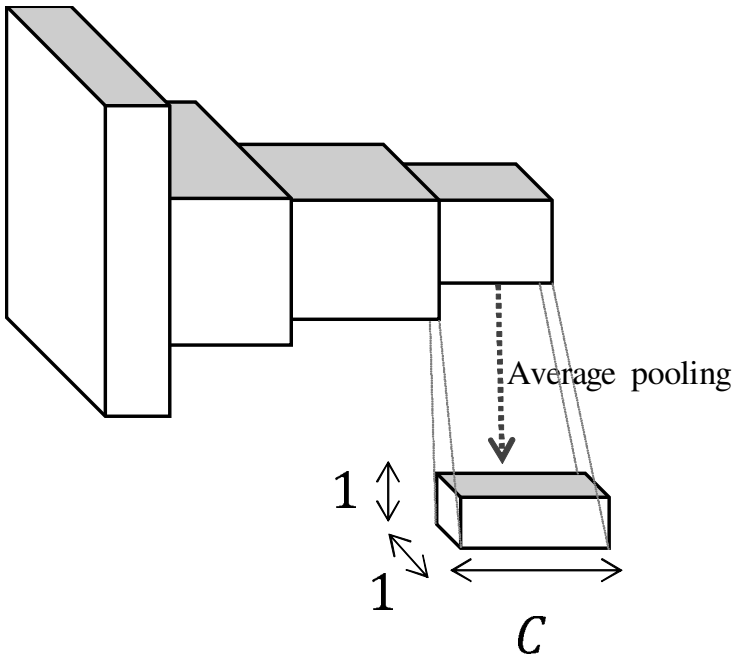

FIGURE 3.6: Feature extraction overview of DifferNet. DifferNet extracts features from the last block of AlexNet on three different resolution images, and average-pooling is applied to the three extracted features. The three extracted features are concatenated and inputted into a flow-based model, where the concatenated feature $\in \mathbb{R}^{3\times C}$.

where f_{NF} and $p_z(z)$ denote the flow-based model and a well-defined distribution.

Most of these methods use traditional anomaly detection techniques that require the use of features, and cannot use images directly for training. By contrast, the proposed method can use both types of training data.

3.4 Evaluation Metric

In anomaly detection, the performance is evaluated commonly in terms of the area under the receiver operating characteristic (AUROC) curve, which is obtained by moving the classification threshold for the ratio of true-positive and false-positive rates.

The true-positive rate (TPR) and false-positive rate (FPR) are defined as follows:

$$TPR = \frac{TP}{TP + FN}, \tag{3.18}$$

$$FPR = \frac{FP}{FP + TN}, \tag{3.19}$$

where TP, FN, TN, and FP mean true positive, false negative, true negative, and false positive, respectively.

If the result is by chance, then AUROC is 0.5; if positive and negative can be completely separated, then AUROC is 1.

Chapter 4

Generating Pseudo Anomalous Data

4.1 Overview

Fig. 4.1 illustrates the training procedure of our proposed ALGAN. The generator is trained in the same manner as in standard GANs. Two additional data types are employed to train the discriminator. One of the data types is generated from the anomalous latent variable, and the other is a buffer of data generated during the training process. The buffer size is twice as large as the training data, and in each epoch, a portion of the old buffer is replaced with newly generated data.

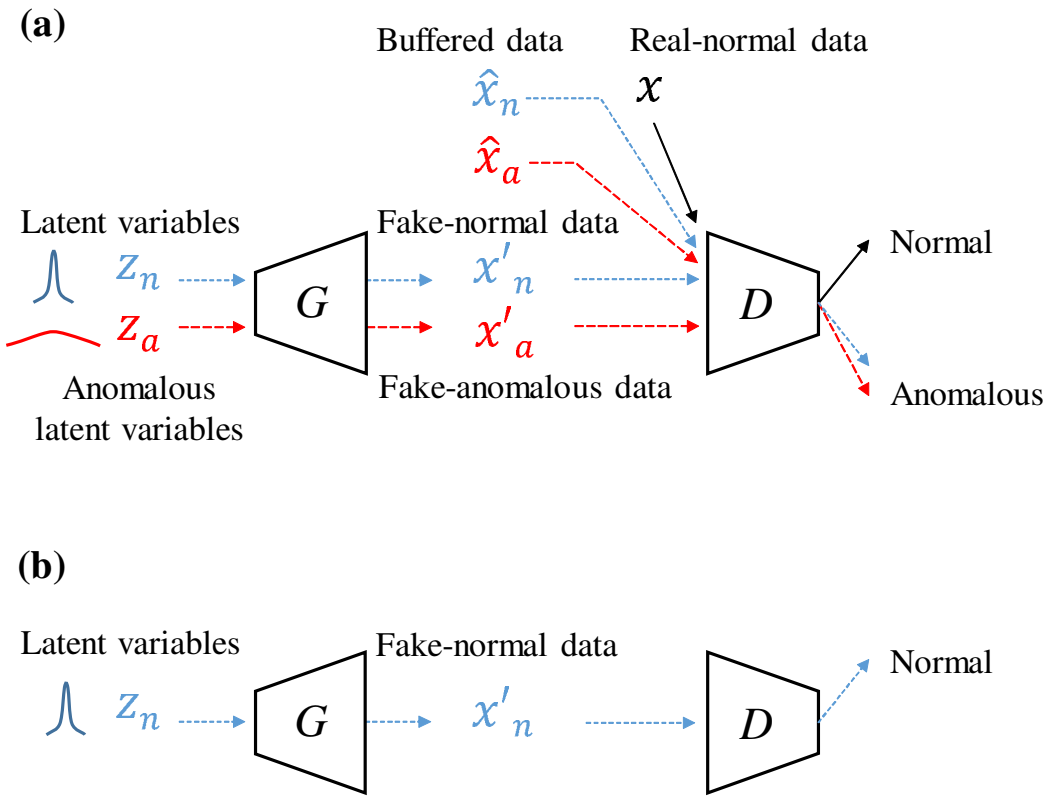

FIGURE 4.1: Overview of ALGAN training. (a) Procedure for training the discriminator D with real-normal data x (black), latent variables $z_n \sim N(0, I)$ (blue), anomalous latent variables $z_a \sim N(0, \sigma^2 I)$ (red), fake-normal data x'_n (blue), fake-anomalous data x'_a (red), buffered fake-normal data $\hat{x}_n$ (blue), and buffered fake-anomalous data $\hat{x}_a$ (red). (b) Procedure for training the generator G.

4.2 Related Work

4.2.1 Pseudo-Anomalous Data

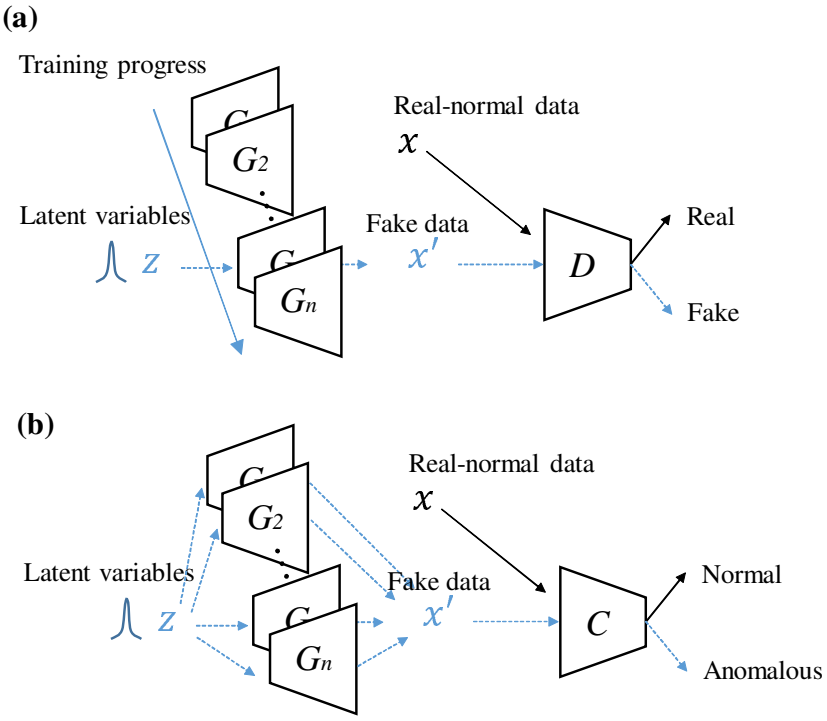

FIGURE 4.2: Overview of G2D training. (a) GANs are trained using only normal training data x, and the generators $\{G_1, G_2, ..., G_n\}$ are saved during training. (b) The saved generators produce immature samples, which are used as pseudo-anomalous data for anomaly detector C training.

Some methods consider the generation of pseudo-anomalous data. Data immaturely generated during the training process of GANs have been used as pseudo-anomalous data for training (Chatillon and Ballester, 2020; Zaheer et al., 2020; Pourreza et al., 2021). G2D (Pourreza et al., 2021) and history-based anomaly detector (Chatillon and Ballester, 2020) save generators during the training process, and the saved generators generate immature samples for the anomaly detector training (Fig. 4.2).

In CutPaste (Li et al., 2021), patches of random sizes and angles are cut out from an image and randomly pasted onto the image. The classifier is trained either from scratch or fine-tuned using normal and pseudo-anomalous data.

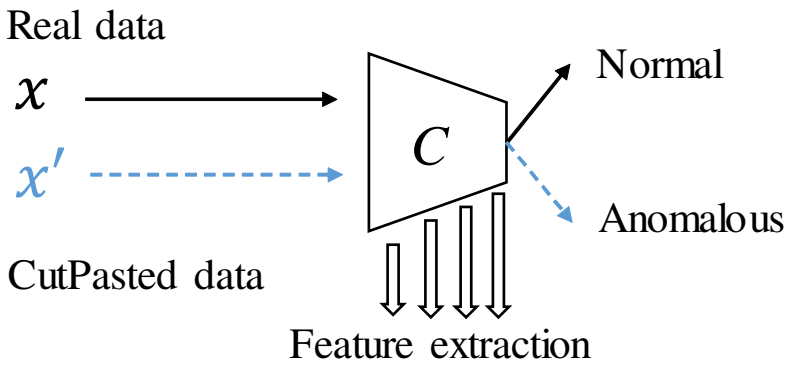

FIGURE 4.3: Overview of CutPaste training. The classifier used for feature extraction is trained by normal training data and cut-pasted pseudo-anomalous data. Pseudo-anomalous data are created by image processing. Patches of random sizes and angles are cut out from an image and randomly pasted onto the image.

The feature representation by the classifier is then used to calculate the anomaly score based on the Gaussian density assumption, as introduced in Section 3.3.

The proposed method generates pseudo-anomalous data by introducing anomalous states into latent variables other than using data generated by an immature generator. Furthermore, it is less biased than techniques that generate pseudo-anomalous data using prior knowledge like image processing.

4.3 Proposed Method for Generating Pseudo Anomalous Data

We introduce two types of pseudo-anomalous data for training. They are in addition to the x and $x'_n = G(z_n)$ used in the standard GANs described in Section 2.2, where x, x'_n, and z_n are defined as real-normal data, fake-normal data, and latent variables from $N(0, I)$, respectively.

The first type of pseudo-anomalous data is called *fake-anomalous data*. AL-GAN utilizes the anomalous latent variables $z_a \sim N(0, \sigma^2 I)$ with a larger variance ($\sigma > 1$) to generate fake-anomalous data $x'_a = G(z_a)$. See Fig. 4.4 for

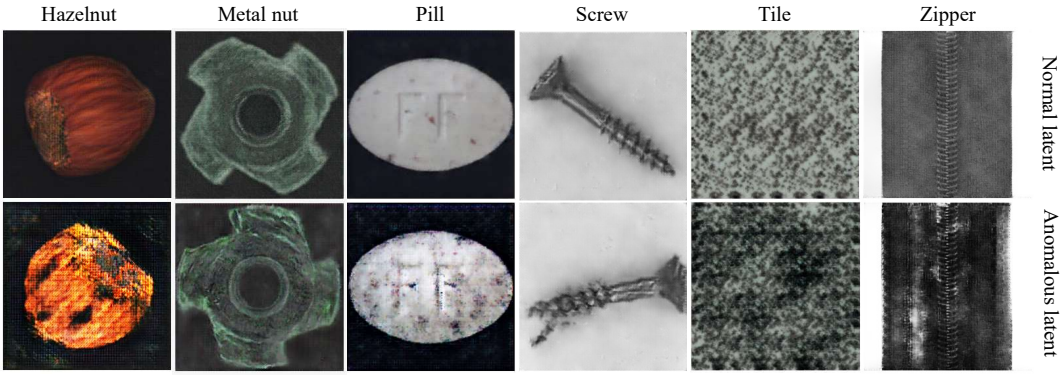

FIGURE 4.4: Images generated by ALGAN-image. Top row: Fake-normal data from normal latent variables $N(0, I)$. Bottom row: Fake-anomalous data from anomalous latent variables $N(0, \sigma^2 I)$, where $\sigma = 4$.

examples of fake-anomalous data. The fake-anomalous images are slightly degraded compared with the fake-normal images.

The other type of pseudo-anomalous data is called *buffered data* $\hat{x} = \{\hat{x}_n, \hat{x}_a\}$, which are defined as generated samples during the early stage of the training process. These are expected to differ from the real-normal data. During training, the fake samples $x' = \{x'_n, x'_a\}$ are stored and used as buffered data.

4.4 Training Methodology

The discriminator is trained to maximize log-likelihoods $\log(D(x))$, $\log(1 - D(x'))$, and $\log(1 - D(\hat{x}))$ to distinguish real-normal data x from other data. Conversely, the generator is trained to generate fake-normal data from z_n, and minimize log-likelihood $\log(1 - D(G(z_n)))$ to fool the discriminator. Both network objectives are given by the following equations:

$$\max_D \Big(\mathbb{E}_x[\log(D(x))] + \xi \mathbb{E}_{z_n}[\log(1 - D(x'))] + (1 - \xi)\mathbb{E}_{z_n}[\log(1 - D(\hat{x}))] \Big), \tag{4.1}$$

$$\min_G \Big(\mathbb{E}_{z_n}[\log(1 - D(G(z_n)))] \Big), \tag{4.2}$$

where ξ is the ratio of generated data to buffered data. Let α be the ratio of fake-normal data to fake-anomalous data; then, the objective of the discriminator is given by the following equation:

$$\begin{aligned}\max_{D} \Big(& \mathbb{E}_x[\log(D(x))] + \alpha\{\xi \mathbb{E}_{z_n}[\log(1 - D(x'_n)] \\ & + (1-\xi)\mathbb{E}_{\hat{x}_n}[\log(1 - D(\hat{x}_n))]\} \\ & + (1-\alpha)\{\xi \mathbb{E}_{z_a}[\log(1 - D(x'_a))] \\ & + (1-\xi)\mathbb{E}_{\hat{x}_a}[\log(1 - D(\hat{x}_a))]\}\Big). \end{aligned} \tag{4.3}$$

Thus, the discriminator of ALGAN learns the discrimination boundary between real-normal data and the other types of data. The objective function of ALGAN is obtained by combining Eq. (4.2) and (4.3) as follows:

$$\begin{aligned}\min_{G}\max_{D} \Big(& \mathbb{E}_x[\log(D(x))] + \alpha\{\xi \mathbb{E}_{z_n}[\log(1 - D(G(z_n)))] \\ & + (1-\xi)\mathbb{E}_{\hat{x}_n}[\log(1 - D(\hat{x}_n))]\} \\ & + (1-\alpha)\{\xi \mathbb{E}_{z_a}[\log(1 - D(x'_a))] \\ & + (1-\xi)\mathbb{E}_{\hat{x}_a}[\log(1 - D(\hat{x}_a))]\}\Big). \end{aligned} \tag{4.4}$$

The proposed method follows an adversarial training procedure (Fig. 4.1). It provides a discrimination boundary not only for the real-normal and fake-normal data but also for the real-normal and pseudo-anomalous data, the latter of which has a broader support of the distribution. As the training progresses, the generator produces samples that resemble real-normal data, and the discriminator cannot distinguish between real-normal and fake-normal data. The pseudo-anomalous data are clearly different from the real-normal data; therefore, the discrimination boundary of the discriminator is used to classify them.

Algorithm 1 Training algorithm of ALGAN.

Notation: Number of batches, m; latent variables for D, z_d; latent variables for G, z_g.

Hyperparameters: Training epochs (e), update frequency of latent variables (n_z), ratio of normal and anomalous latent variables (α), standard deviation of anomalous latent (σ), and number of updates for D (n_{dis}).

for $i = 1, \cdots, e$ **do**
  if i mod $n_z = 0$ **then**
    Sample $z_d \sim \alpha N(0, I)$ and $(1-\alpha)N(0, \sigma^2 I)$
    Sample $z_g \sim N(0, I)$
  end if
  for $j = 1, \cdots, m$ **do**
    Sample $x \sim p_{data}$
    for $k = 1, \cdots, n_{dis}$ **do**
      $x' \leftarrow G_\theta(z_d)$
      if $i = 1$ **then**
        $Loss_D \leftarrow D_\phi(x) + D_\phi(x')$
      else
        Sample buffered data $\hat{x} \sim$ Buffer
        $Loss_D \leftarrow D_\phi(x) + D_\phi(x') + D_\phi(\hat{x})$
      end if
      $\phi \leftarrow Adam(Loss_D, \phi)$
    end for
    Buffer $\leftarrow x'$
    $Loss_G \leftarrow D_\phi(G_\theta(z_g))$
    $\theta \leftarrow Adam(Loss_G, \theta)$
  end for
end for
return D_ϕ, G_θ

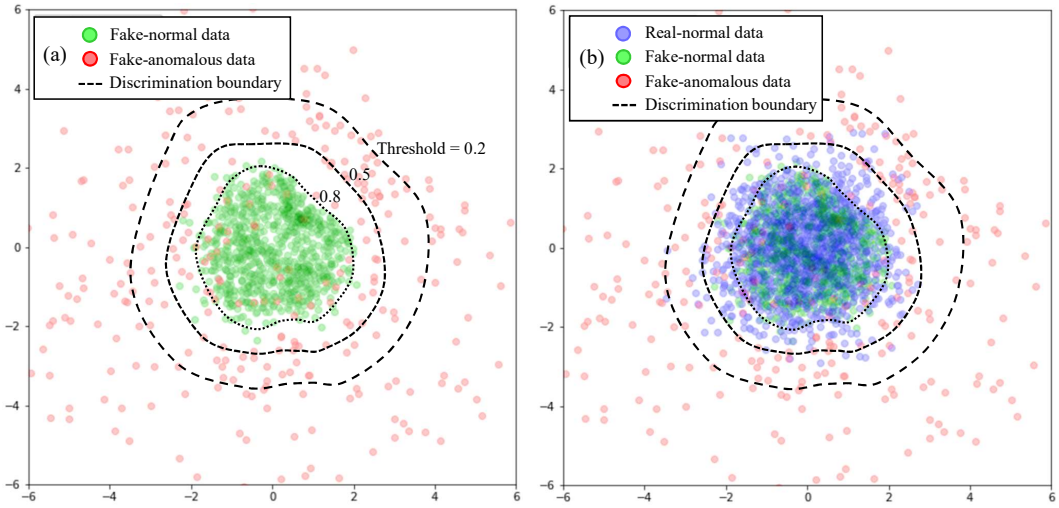

FIGURE 4.5: Intuitive interpretation of fake-anomalous data in the toy problem: fake-anomalous data are added to training GANs that map 100-dimensional latent variables to 2-dimensional normal distribution. (a) Fake-anomalous data x'_a (red) generated by $\sigma = 4$ are distributed to surround the fake-normal data x'_n (green). (b) Real-normal data x (blue) are overlaid on the left figure. From the figure, it can be seen that the generator has been trained successfully and has generated fake-normal data that approximate the real-normal data distribution. Thus, the discriminator cannot distinguish between the real-normal and fake-normal data. By contrast, the fake-anomalous data can be properly distinguished by adjusting the discrimination threshold because they are distributed outside.

The pseudo-code for training is presented in Algorithm 1. A feature of AL-GAN training is the method for updating the parameter ϕ of the discriminator. In line 3 of the pseudo-code, normal and anomalous latent variables are sampled, and in line 9, fake-normal and fake-anomalous data are produced from the generator. In line 11, loss is calculated by identifying both types of data, and in line 16, the parameter ϕ of the discriminator is updated. Fake-normal and fake-anomalous data are buffered in line 18. As there are no buffered data in the first epoch, the conditional branch for $i = 1$ is required in line 10. The loss of buffered data is calculated in addition to fake-normal and fake-anomalous data in line 14 after the branch. The parameter θ of the generator is updated by only fake-normal data from normal latent variables in lines 4, 19, and 20.

Fig. 4.5 provides an intuitive understanding of the training for anomaly detection using fake-anomalous data. The generator is trained to produce fake-normal data that approximate the distribution of real-normal data. Given anomalous latent variables, the generator produces fake-anomalous data with a large variance. The discriminator is trained to distinguish between real-normal data and the other types of data. Even after real-normal and fake-normal data become indistinguishable, fake-anomalous data can be identified because the discrimination boundary is laid between real-normal and fake-anomalous data.

Chapter 5

Experiments

An advantage of ALGAN is that it can be used for both the images and features extracted from a pre-trained model, whereas some state-of-the-art methods for visual inspection depend on features (Cohen and Hoshen, 2020; Defard et al., 2021; Roth et al., 2022; Rudolph, Wandt, and Rosenhahn, 2021). To demonstrate this advantage experimentally, we used two different types of implementations. We call them ALGAN-image and ALGAN-feature, and compare them with the relevant methods.

5.1 Implementation Details

5.1.1 Network Architecture and Hyperparameter

ALGAN-image employs an architecture similar to that of DCGAN (Radford, Metz, and Chintala, 2016). The generator and discriminator use seven transposed convolutional and convolutional layers, respectively.

For ALGAN-feature, WideResNet101 (Zagoruyko and Komodakis, 2016) is applied to the feature extractor to obtain 2048-dimensional vectors from the last block with global average pooling (in Fig. 5.1). Both the generator and discriminator have three fully connected layers.

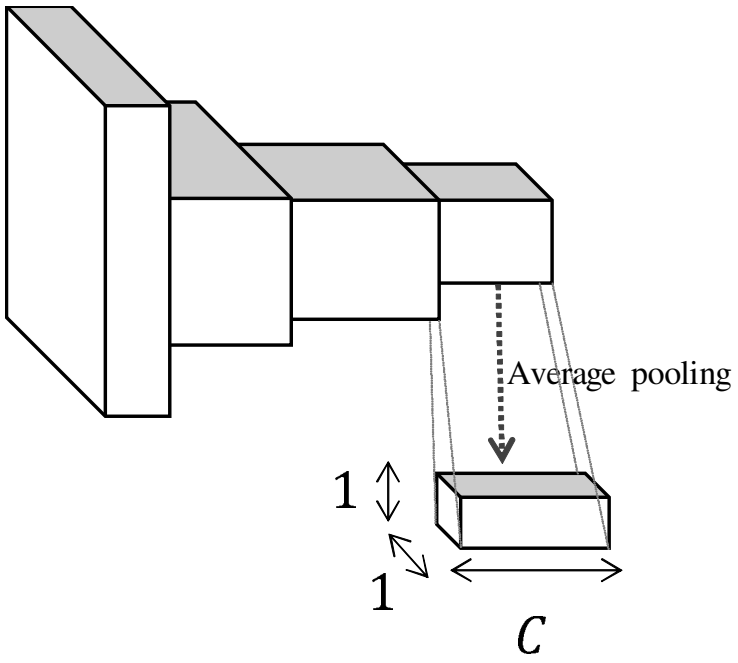

FIGURE 5.1: Feature extraction overview of ALGAN-feature. ALGAN-feature extracts features from the last block of WideResNet-101 and average-pooling is applied to the extracted features. Difference from DifferNet, ALGAN-feature uses a single resolution image feature $\in \mathbb{R}^C$.

In both architectures, the generator uses batch normalization (Ioffe and Szegedy, 2015) and the ReLU activation function (Nair and Hinton, 2010), whereas batch normalization is removed from the output layer. The discriminator employs spectral normalization (Miyato et al., 2018) and the Leaky-ReLU activation function (Maas, Hannun, and Ng, 2013).

The networks were optimized using Adam (Kingma and Ba, 2015) with momentum $\beta_1 = 0$ and $\beta_2 = 0.9$, and the learning rates of the generator and discriminator were set to 2×10^{-4} and 1×10^{-4}, respectively. The latent variable z had 100 dimensions, and the standard deviation of the anomalous latent variable used $\sigma = 4$. The parameters that determine the ratio of the pseudo-anomalous data were set to $\alpha = 0.75$ and $\xi = 0.75$. The parameters of Algorithm 1 were set as $n_z = 2$ and $n_{dis} = 2$. The buffer holds twice the batch size, half of which is randomly replaced by newly generated data.

The comparison methods were implemented using a DCGAN-like architecture similar to ALGAN-image. In GANomaly (Akçay, Atapour-Abarghouei, and Breckon, 2018) and Skip-GANomaly (Akçay, Atapour-Abarghouei, and Breckon, 2019), the dimensions of the latent variables were set to 100 and 512, respectively. The image resolution used for each method was 256×256. All the models were trained using a batch size of 16.

5.1.2 Software and Hardware

All the models were implemented using Python 3.8.8 and PyTorch 1.8.1 on Ubuntu 20.04 LTS. An AMD EPYC 7542 32-core processor with 512 GB memory and an NVIDIA A100-SXM4 40 GB GPU were used for the computations.

5.2 Datasets

For performance evaluation, we used three different datasets.

5.2.1 MVTec-AD Dataset

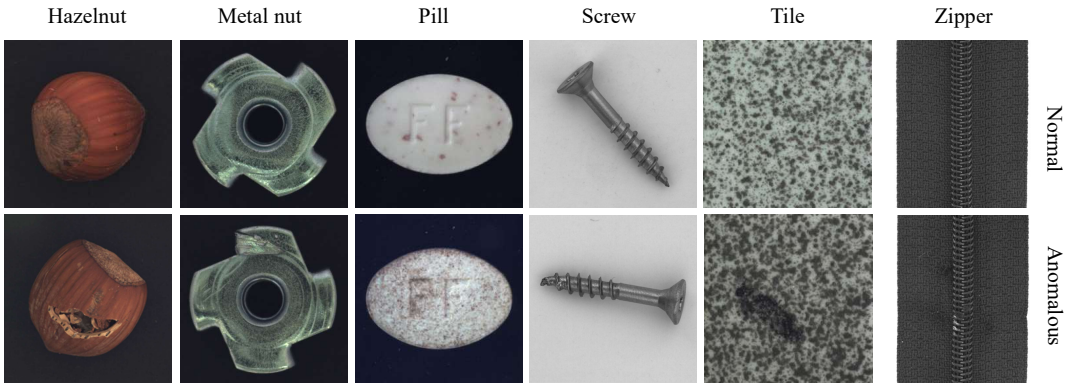

FIGURE 5.2: Overview of the expmerimental dataset: MVTec-AD dataset. Top row: Normal data. Bottom row: Anomalous data.

This dataset, designed for visual inspection, consists of 5,354 images, comprising five texture and 10 object categories (Bergmann et al., 2019b). The training set contains 3,629 defect-free (normal) images, and the test set contains 467 defect-free (normal) images and 1,258 defective (anomalous) images (Fig. 5.2, Table 5.1).

5.2.2 Magnetic Tile Defects Dataset (MTD)

This dataset consists of grayscale images with different aspect ratios, including 952 defect-free (normal) images and 392 images containing five defect types (anomalous) (Huang, Qiu, and Yuan, 2020) (Fig. 5.3).

5.2.3 COIL-100 Dataset

This dataset contains 100 different object categories and 7,200 images (Nene, Nayar, Murase, et al., 1996). Each object category has 72 images rotated every 5° (Fig. 5.3).

TABLE 5.1: Details of MVTec-AD Dataset. First column: category name. Second column: number of normal training data. Third column: number of normal test data. Fourth column: number of anomalous test data.

	Train (normal)	Test (normal)	Test (anomalous)
Carpet	280	28	89
Grid	264	21	57
Leather	245	32	92
Tile	230	33	84
Wood	247	19	60
Bottle	209	20	63
Cable	224	58	92
Capsule	219	23	109
Hazelnut	391	40	70
MetalNut	220	22	93
Pill	267	26	141
Screw	320	41	119
Toothbrush	60	12	30
Transistor	213	60	40
Zipper	240	32	119
Total	3629	467	1258

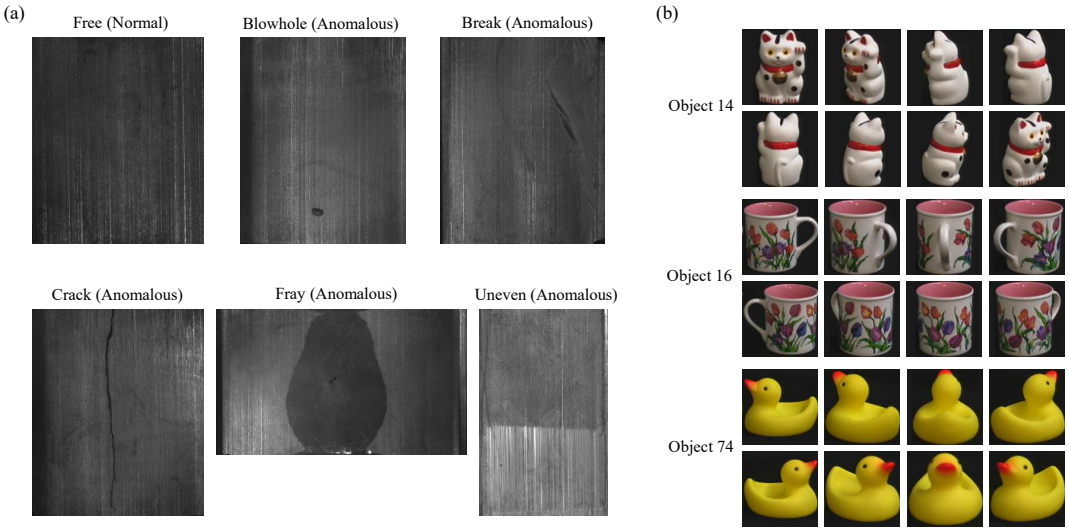

FIGURE 5.3: Overview of the experimental datasets. (a) Magnetic Tile Defects dataset (MTD); (b) Columbia University Image Library dataset (COIL-100)

5.2.4 Splitting into Training and Validation Sets

Because the training sets of MVTec-AD and MTD contain no defective (anomalous) images, 50% of their respective test sets was used as the validation set, which was used as the stopping rule.

In MTD, the test set is not provided separately from the training data; therefore, 50%, 25%, and 25% of the defect-free (normal) images were used for training, validation, and testing, respectively.

For COIL-100, 10 categories were selected for normal data and the remaining 90 categories for anomalous data. We used 60%, 20%, and 20% of the normal data for training, validation, and testing, respectively, and 50% of the anomalous data were used for validation and 50% for testing.

5.2.5 Pre-Processing and Data Augmentation

In MVTec-AD using ALGAN-image, images were resized to 256×256. To highlight defects, two images were concatenated with the original image and used as an input image: one was created with max pooling and the other with average pooling in the channel axis dimension. For the texture categories, vertical and horizontal flips were applied (however, only horizontal flip was applied for wood). In the object categories, vertical flip, horizontal flip, and random rotation were applied to Bottle and Hazelnut. Horizontal flip was applied to Toothbrush, Transistor, and Zipper. Toothbrush was converted into grayscale. Random rotation was applied to Metal nut and Screw. Cable, Capsule, and Pill were only resized.

In MVTec-AD using ALGAN-feature, the texture categories were resized to 224×224. Furthermore, the object categories were resized to 256×256, and then center-cropped to 224×224. Following Rudolph, Wandt, and Rosenhahn (2021)'s procedure, we applied 24 and 64 different angular rotations

during training and prediction, respectively, but using a single-resolution image.

The images in MTD and COIL-100 were resized to 256×256 for ALGAN-image and 224×224 for ALGAN-feature. For ALGAN-image, a horizontal flip was applied, and for ALGAN-feature, Rudolph, Wandt, and Rosenhahn (2021)'s procedure was applied for both training and prediction.

5.3 Evaluation Method and Metric

In ALGAN, the real and fake labels are assigned probability 1 and 0, respectively. The real label corresponds to normal data; thus, the anomaly detection rule is given by the following:

$$\text{ALGAN}(x) = \begin{cases} \text{Normal}, & \text{if } D(x) > \text{threshold}, \\ \text{Anomalous}, & \text{otherwise.} \end{cases} \tag{5.1}$$

Because the significance of false positives or false negatives depends on their application, the threshold is chosen by the user.

Accounting for the randomness of the dataset split, we performed 10 experiments with different seed values for each dataset. Performance was evaluated in terms of the area under the receiver operating characteristic (AUROC) curve.

ALGAN-image was trained on 512 epochs and ALGAN-feature on 192 epochs. The performances were validated every eight epochs on the validation set, and the model that showed the best AUROC was saved. The best model was evaluated on the test set after training.

The image-based methods GANomaly, Skip-GANomaly, and ALOCC, were trained on 512 epochs and validated every eight epochs as in ALGAN-image.

The feature-based method, DifferNet, was trained on 192 epochs and validated every eight epochs, as in ALGAN-feature. The model that showed the best AUROC in the validation set was saved, and the test set was evaluated after training.

5.4 Results

5.4.1 Anomaly Detection on MVTec-AD

Training with Image Data

The results on the test data with the model with the best AUROC for validation (Section 5.3) are listed in Table 5.2. ALGAN-image significantly outperformed other state-of-the-art image-based methods, such as GANomaly, Skip-GANomaly, and ALOCC. ALGAN-image showed an average accuracy of more than 10% compared with the others and attained the best accuracy for 13 out of the 15 categories.

Our method uses the discriminator to distinguish between normal and anomalous directly from the images. In the Hazelnut and Screw categories, the rotation angle of the object is different in each image. Thus, small changes in image details caused by slight defects may be buried by large changes in the image caused by rotation. Consequently, the anomalous data detection performance could not be better than that of the comparison methods in these categories.

Because the comparison methods use reconstruction error, they fail to detect anomalous data if the reconstruction is successful. By contrast, our proposed method, ALGAN-image, uses the discriminator to classify the data and does not suffer from detection errors due to reconstruction.

TABLE 5.2: Results obtained on MVTec-AD. ALGAN-image is compared with methods trained on image data. Top row: mean AUROC. Bottom row: standard deviation. We report the results of 10 experiments using each method. The best performance for each category is indicated in boldface.

	GANomaly	Skip-GANomaly	ALOCC	ALGAN -image
Carpet	0.803 0.070	0.829 0.056	0.736 0.054	**0.846** 0.041
Grid	0.924 0.088	0.816 0.079	0.847 0.087	**0.938** 0.057
Leather	0.796 0.079	0.787 0.095	0.768 0.042	**0.920** 0.041
Tile	0.852 0.034	0.941 0.041	0.648 0.071	**0.914** 0.034
Wood	0.939 0.029	0.969 0.020	0.849 0.062	**0.972** 0.022
Bottle	0.704 0.063	0.674 0.096	0.801 0.075	**0.948** 0.024
Cable	0.686 0.057	0.636 0.064	0.686 0.053	**0.877** 0.036
Capsule	0.717 0.075	0.691 0.068	0.707 0.073	**0.805** 0.083
Hazelnut	0.765 0.058	**0.955** 0.036	0.697 0.111	0.836 0.047
Metal nut	0.698 0.064	0.553 0.105	0.771 0.066	**0.811** 0.060
Pill	0.772 0.050	0.787 0.083	0.659 0.043	**0.819** 0.053
Screw	0.567 0.357	**1.000** 0.001	0.938 0.076	0.811 0.090
Toothbrush	0.774 0.098	0.808 0.084	0.778 0.102	**0.933** 0.048
Transistor	0.783 0.067	0.769 0.094	0.745 0.085	**0.865** 0.074
Zipper	0.693 0.052	0.675 0.070	0.656 0.079	**0.879** 0.025
mean	0.765	0.793	0.752	**0.878**

Training with Pre-Trained Features

The results for the feature-based methods are listed in Table 5.3. The results for DifferNet are the means of 10 different test datasets (see Section 5.3). The results for GANomaly were obtained from Rudolph, Wandt, and Rosenhahn (2021) and the other methods from their respective papers (Cohen and Hoshen, 2020; Rippel, Mertens, and Merhof, 2021; Defard et al., 2021; Li et al., 2021; Roth et al., 2022). ALGAN-feature achieved results comparable to those of DifferNet, whereas PatchCore-25, PaDiM, and CutPaste achieved better performance. Note, however, that the PatchCore, PaDiM, and CutPaste methods are strongly specialized for pre-trained features.

ALGAN-feature and DifferNet use only a single feature from the last layer of the pre-trained model; better performing methods use features from multiple layers. Using features from multiple layers increases the computational costs owing their high dimensionality. For example, PatchCore (Roth et al., 2022) divides high-dimensional features into patches and subsamples them to select useful patches, which results in much higher computational costs. The prediction time is also longer than that of ALGAN-feature (see Section 5.4.3). Furthermore, DifferNet (Rudolph, Wandt, and Rosenhahn, 2021) concatenates features from three different image resolutions, whereas our proposed method, ALGAN-feature, achieves comparable performance with only one resolution.

TABLE 5.3: Results obtained on MVTec-AD. ALGAN-feature is compared with methods learned from features of pre-trained models. For DifferNet and ALGAN-feature, we report the mean AUROC results of 10 experiments. The best performance for each category is indicated in boldface.

	SPADE	Mah.AD	PaDiM	CutPaste	PatchCore-25	GANomaly	DifferNet	ALGAN-feature
Texture	-	0.978	0.990	0.962	**0.991**	0.775	0.915	0.925
Object	-	0.950	0.972	0.955	**0.992**	0.755	0.934	0.905
ALL	0.855	0.958	0.979	0.961	**0.991**	0.761	0.927	0.911

5.4.2 Anomaly Detection on Other Datasets

The results obtained on MTD and COIL-100 are listed in Tables 5.4 and 5.5, respectively. These are the means of 10 trials with the same hyperparameters as those used for MVTec-AD. ALGAN-image achieved state-of-the-art performance on MTD after training with image data and achieved comparable results to that of PatchCore. On MTD, ALGAN-feature performed worse than ALGAN-image. This may be because the features useful for anomaly detection could not be extracted from the last block of WideResNet-101. Because the features from the deep block in ResNet are biased towards ImageNet (Roth et al., 2022), the features from the shallow block should be used to identify the more abstract features required for MTD. GANomaly-features, which extracted features from the last block of the same WideResNet-101, did not perform well the same as ALGAN-features. On the other hand, PatchCore, which performed well, used shallow 2- and 3-block features close to the input of WideResNet-50. Because DifferNet used three types of last layer features (448, 224, and 112 pixels) of AlexNet, which has fewer convolutional layers, the 448-pixel features are considered to perform better because they can contain abstract features. On the COIL-100 benchmark, all the methods performed almost perfectly.

TABLE 5.4: Results obtained on Magnetic Tile Defects. For training with the image data, we report the mean AUROC results of 10 experiments. Numbers in boldface indicate the best performance.

	Training with image data			Training with pre-trained features			
	GANomaly	Skip-GANomaly	ALGAN-image	PatchCore-10	GANomaly-feat.	DifferNet	ALGAN-feature
AUROC	0.683	0.504	**0.956**	**0.979**	0.766	0.977	0.824

TABLE 5.5: Results obtained on COIL-100. Top row: mean AUROC. Bottom row: standard deviation. We report the results of 10 experiments using each method. The best performance is indicated in boldface.

	DifferNet	PatchCore-1	ALGAN -image	ALGAN -feature
AUROC	0.999	**1.000**	0.999	**1.000**
	0.003	0.000	0.001	0.000

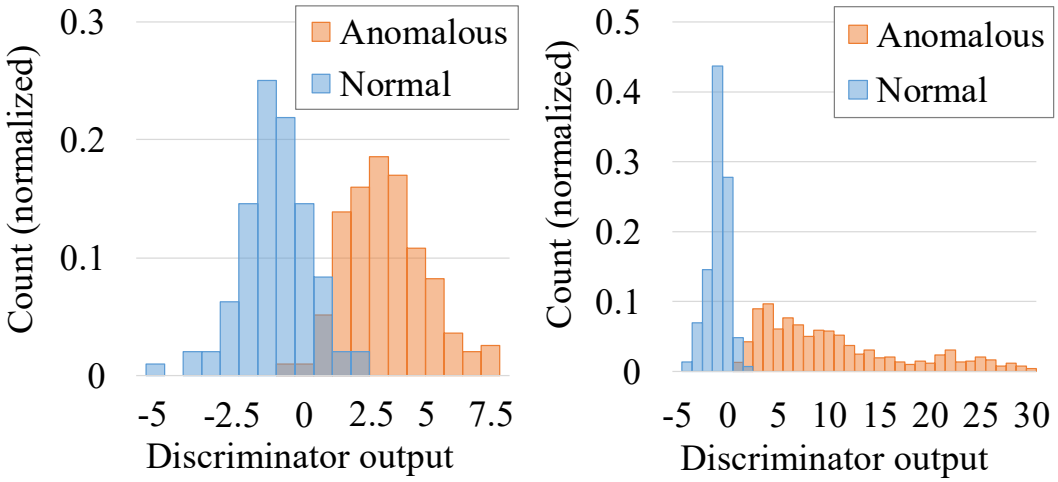

FIGURE 5.4: Histograms of raw output values of the discriminator before input to the sigmoid function in ALGAN-image. Left: Magnetic Tile Defects. Right: COIL-100. The sign is reversed so that the horizontal axis represents the anomaly score.

Fig. 5.4 shows histograms of the raw output values of the discriminator before they are input into the sigmoid function of ALGAN-image. The distributions of normal and anomalous data are significantly separated. In COIL-100, normal data have a peaky distribution with fewer variations, whereas the distribution of anomalous data exhibits a long tail, reflecting the large variations of anomalous data.

5.4.3 Prediction and Training Times

Table 5.6 compares the prediction times of the models for MVTec-AD. We can see that ALGAN-image achieved a significantly faster prediction time (10.6 ms), which is 10.4 to 54.6 times faster than those of the other image-based methods. ALGAN-feature is 1.3 to 2.2 times faster than the other methods trained on the feature. Fig. 5.5 depicts the prediction time and AUROC performance of the selected methods. ALGAN-image is the fastest and has a high AUROC, but it is not the highest. ALGAN-feature is faster than the other feature-based methods while maintaining a competitive AUROC. ALGAN and PatchCore exhibit a trade-off between performance and speed.

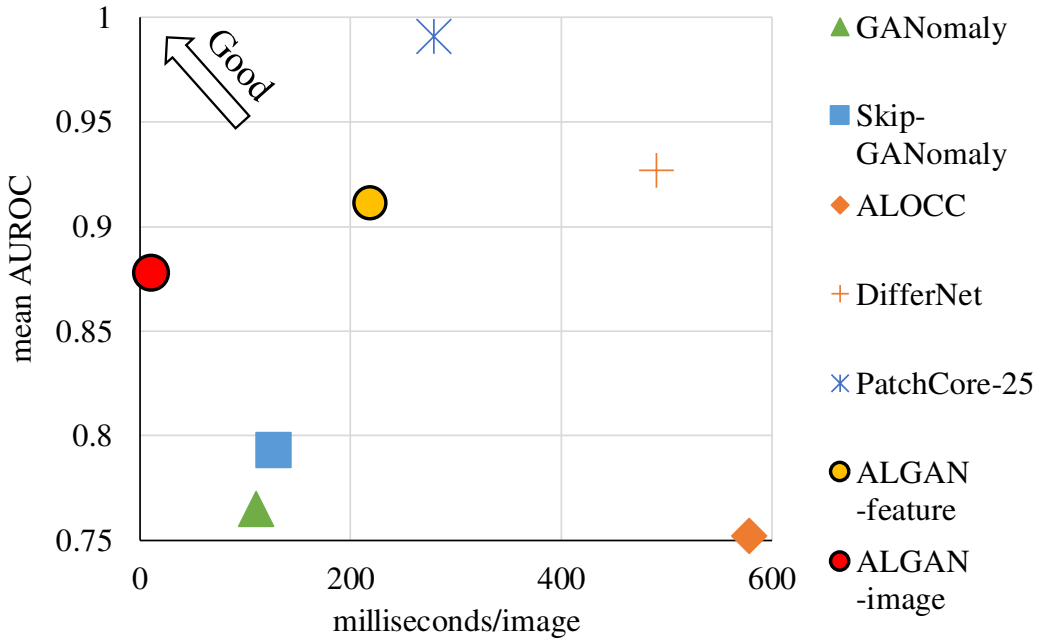

FIGURE 5.5: Mean AUROC vs. prediction time for each method on MVTec-AD.

Considering the fast prediction of ALGAN-image, it can be applied to expensive tasks such as real-time prediction with a large number of bounding boxes obtained from object detection (Liu et al., 2020a).

Table 5.7 compares the training times obtained on MVTec-AD. ALGAN-image is the fastest among the compared methods, with the default number of epochs described in Section 5.3. Because there is no official implementation of PatchCore, we did not use the Faiss library (Johnson, Douze, and Jégou, 2021)[1], which is used in the original study, but applied our own implementation.

[1]Our implementation uses the scikit-learn (Pedregosa et al., 2011) library for core-set selection (Sener and Savarese, 2018) and random projection (Sinha et al., 2020). We confirmed that our implementation produces an AUROC similar to the results in (Roth et al., 2022).

TABLE 5.6: Mean prediction times per single test image for all categories on MVTec-AD. Boldface indicates the best result.

	GANomaly	Skip-GANomaly	ALOCC	DifferNet	PatchCore -25	ALGAN -feature	ALGAN -image
milliseconds/image	109.8	126.3	577.8	489.5	278.6	218.9	**10.6**

TABLE 5.7: Mean training times for all categories on MVTec-AD. Boldface indicates the best result.

	GANomaly	Skip-GANomaly	ALOCC	DifferNet	PatchCore -25	ALGAN -feature	ALGAN -image
Training Epochs	512	512	512	192	-	192	512
Training Time (min)	23	47	401	55	98	41	**22**

5.4.4 Training Stability

Zaheer et al. (2020) reported that GAN-based anomaly detection exhibited unstable validation results during the training process. Upon validation, our proposed method exhibited stable results in terms of AUROC (Fig. 5.6).

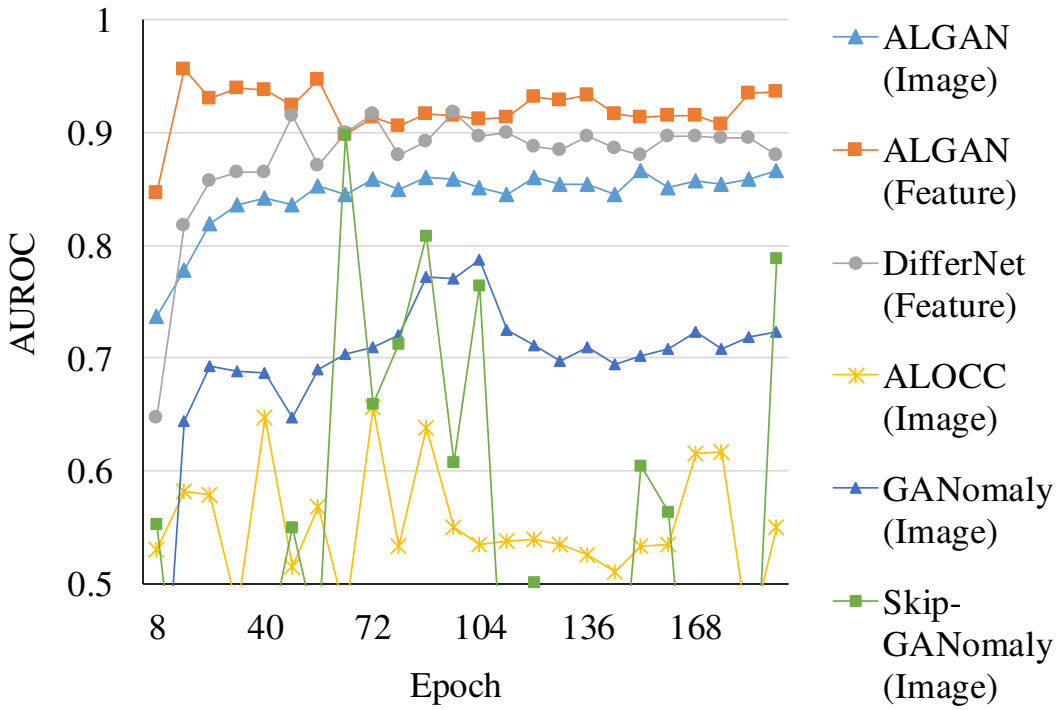

FIGURE 5.6: Validation results plotted every 8 epochs during the training of the transistor category. ALGAN-image and ALGAN-feature exhibit stable AUROCs compared with other GAN-based methods.

5.5 Ablation Study

Ablation studies were performed to verify the effect of the two types of pseudo-anomalous data on performance. In ALGAN, buffered data and the data generated by anomalous latent variables were used as pseudo-anomalous data. The checkmark in Table 5.8 indicates the type(s) used. Although either of the two can improve performance, using both works best and reduces variance. These results validate that the proposed pseudo-anomalous data are useful for improving anomaly detection performance.

TABLE 5.8: Ablation study results obtained on MVTec-AD with ALGAN-image. Each experiment was conducted 10 times. The left side of the mean AUROC column lists the mean and the right side lists the standard deviation. The best performance is indicated in boldface.

Buffered Data	Anomalous Latent	Mean AUROC	
		0.643	0.097
✓		0.761	0.102
	✓	0.779	0.104
✓	✓	**0.878**	0.049

5.6 Hyperparameter Study

We also studied the impact of hyperparameters on anomaly detection performance. ALGAN employs the following hyperparameters: σ is the standard deviation of the anomalous latent variable, n_z is the epoch frequency to update the latent variables, n_{dis} is the number of updates per batch on the discriminator, and α and ξ are the fake-anomalous and buffered data balanced parameter, respectively. Each experiment was conducted 10 times using the Bottle category of the MVTec-AD dataset.

5.6.1 Standard Deviation σ for Anomalous Latent Variable

For a small σ, performance is low and peak performance is reached at $\sigma = 4$ or 5. If the σ value is too large, the support for normal and anomalous data may be separated, which can degrade performance (Table 5.9).

TABLE 5.9: Impact of σ changes on performance. Top row: mean AUROC. Bottom row: standard deviation. Boldface indicates the best result.

σ	2	3	4	5	6	8
AUROC	0.859	0.932	**0.948**	0.947	0.937	0.917
	0.058	0.042	0.024	0.025	0.033	0.036

5.6.2 Balance Parameter α and ξ for Fake-normal and Buffered Data

When α is small, the effect of fake-anomalous is large, and when ξ is small, the effect of buffered data is large. The performance is high around the values of $\alpha = 0.75$, $\xi = 0.75$ reported in this study. Recalling Eq. (4.3), ξ is multiplied by α. When $\alpha = 0.25$, $\xi = 0.25$, the effect of the fake-anomalous buffer is greater, and the performance is improved. By contrast, when $\alpha = 0.85$, $\xi = 0.85$, the effect of fake-anomalous and buffered data is smaller, and the

performance is lower (Table 5.10). The results suggest that the effect of the fake-anomalous buffer is large, and the parameters α and ξ should be adjusted so that the effect is not too small.

TABLE 5.10: Impact of α and ξ change on performance. First row: mean AUROC. Second row: standard deviation. Boldface indicates the best result.

		ξ			
		0.25	0.5	0.75	0.85
α	0.25	0.808 0.085	0.720 0.077	0.749 0.155	0.656 0.162
	0.5	0.755 0.148	0.814 0.081	0.892 0.068	0.794 0.157
	0.75	0.826 0.138	0.860 0.048	**0.948** 0.024	0.808 0.122
	0.85	0.816 0.094	0.846 0.110	0.910 0.060	0.765 0.153

5.6.3 Update Frequency for Latent Variable and Discriminator

In this study, n_z peaked at 2, and n_{dis} was good above 2, but performance decreased slightly when discriminator updates became excessive (Table 5.11, 5.12). Therefore, n_z has a considerable impact on the performance and should be carefully considered and chosen in practical applications.

TABLE 5.11: Impact of n_z changes on performance. Top row: mean AUROC. Bottom row: standard deviation. Boldface indicates the best result.

n_z	1	2	3	4
AUROC	0.813	**0.948**	0.903	0.874
	0.121	0.024	0.035	0.104

TABLE 5.12: Impact of n_{dis} changes on performance. Top row: mean AUROC. Bottom row: standard deviation. Boldface indicates the best result.

n_{dis}	1	2	3	4
AUROC	0.887	**0.948**	0.943	0.933
	0.082	0.024	0.031	0.049

5.7 Robustness to Other Anomalous Data

FIGURE 5.7: CutPaste-style processed images. An image patch sampled from the full image, then color-jittered and pasted onto the original image.

TABLE 5.13: Results obtained on other anomalous data. Second row: number of normal test data (real). Third row: number of anomalous test data (generated). Fourth row: AUROC. Normal test data and generated anomalous test data were perfectly classified.

	$\sigma = 5$	$\sigma = 6$	$\sigma = 7$	$\sigma = 8$	CutPaste-style
Normal	10	10	10	10	10
Anomalous (generated)	32	32	32	32	32
AUROC	1.000	1.000	1.000	1.000	1.000

We also examined the robustness of anomalous data not included in the test data. From Fig. 4.5, we hypothesized that the generator learns the distribution of real-normal data, and the discriminator lays the discrimination boundary between high-density and low-density regions of normal data. The following two types of data were used for the evaluation: fake-anomalous data generated by $\sigma = 5$ to 8, and CutPaste-style processed images (Li et al., 2021) on normal data included in the test data. An image patch was sampled from the full image with an area ratio of 20 to 30% and an aspect ratio between 3 : 1 and 1 : 3, then color-jittered and pasted onto the image.

As in the previous experiments, the discriminator was trained using fake-anomalous data generated by $\sigma = 4$ in the Bottle category. Test anomalous

data were generated as the same number of real test data. All the anomalous data, generated by $\sigma = 5$ to 8 and CutPaste images, were classified as anomalous (in Table 5.13). Thus, these results suggest that the discriminator learns similar to one-class classification between normal data and the other data types.

5.8 How to Determine Hyperparameters?

The hyperparameters described in Section 5.1.1 are recommended basically when using the proposed method for other practical data. This study conducted experiments on three datasets with the same hyperparameters and obtained high-performance results (in Section 5.4.1 and 5.4.2).

Assume that training is performed in a situation where anomalous data are not available. In that case, it is advisable to monitor the discrimination results of fake-normal data, fake-anomalous data, and buffered data during training. Once the discriminator is acquiring anomaly detection performance, fake-normal data will not be well classified, and fake-anomalous and buffered data will be classified as fake (anomalous). When the generator is well trained, the anomaly detection performance of the discriminator is also high. The performance of the generator can be quantitatively evaluated by monitoring the inception score (Salimans et al., 2016) and Fréchet inception distance (Heusel et al., 2017) of fake-normal data.

If even a small amount of anomalous data are available, such data should be used for validation. It is difficult to determine an appropriate threshold discussed in Section 5.3 with the small data. A conservative initial threshold is required, and the ratio of false-positive and false-negative data during operation should be checked to determine a practical value.

5.9 Selection of Latent Variables

The previous experiments in Sections 5.4.1 and 5.4.2 had a wide overlap between two distributions of the normal and anomalous latent variables. A wider overlap will generate more similar fake-normal and fake-anomalous data. On the other hand, with a narrow overlap, the fake-anomalous data generated from the anomalous latent variables are more different from the fake-normal data, which may affect anomaly detection performance.

We evaluate the effect of differences in normal and anomalous latent variables used for training ALGAN on anomaly detection performance. Three experiments are conducted on the MVTec-AD dataset by changing the latent variables.

The latent variables used in the three experiments are depicted in Fig. 5.8. Fig. 5.8 (a) depicts $N(0,1)$ used for normal latent variables and $N(0,\sigma^2)$ where $\sigma = 4$ used for anomalous latent variables. These latent variables were used in Sections 5.4.1 and 5.4.2. Fig. 5.8 (b) depicts $N(0,1)$ used for normal latent variables and $N(0,\sigma^2)$ where $\sigma = 4$ truncating the values between -2 and $+2$ used for anomalous latent variables. Fig. 5.8 (c) depicts $U(-1,1)$ used for normal latent variables and $U(-2,2)$ truncating the values between -1 and $+1$ used for anomalous latent variables. Compared to the latent variables in Fig. 5.8 (a), which were used in the previous experiment, the latent variables in Fig. 5.8 (b) overlap less. In Fig. 5.8 (c), the normal latent variables have a uniform distribution, and there is no overlap with the anomalous latent variables.

The other experimental conditions follow the ALGAN-feature in Sections 5.1.1, 5.2.4, 5.2.5, and 5.3, but 64 different angular rotations during prediction are not performed.

The experimental results are shown in Table 5.14. Although there were some

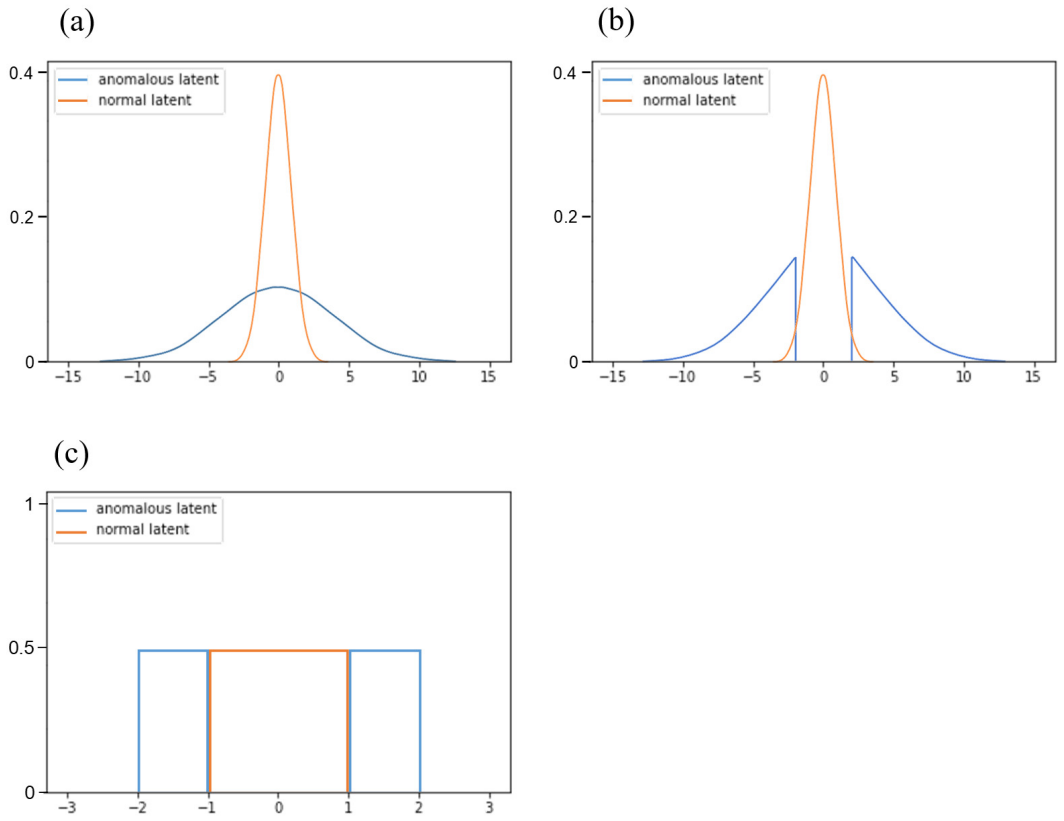

FIGURE 5.8: The latent variables used in the three experiments. (a) depicts $N(0,1)$ used for normal latent variables and $N(0,\sigma^2)$ where $\sigma = 4$ used for anomalous latent variables. (b) depicts $N(0,1)$ used for normal latent variables and $N(0,\sigma^2)$ where $\sigma = 4$ truncating the values between -2 and $+2$ used for anomalous latent variables. (c) depicts $U(-1,1)$ used for normal latent variables and $U(-2,2)$ truncating the values between -1 and $+1$ used for anomalous latent variables.

performance changes in some categories (e.g., Grid, Bottle), there were no significant differences in mean performance across the three experiments. From Fig. 4.5, we hypothesized that ALGAN's discriminator would lay a discrimination boundary between the high-density and low-density regions of the normal data. The overlap between the normal and anomalous latent variables had less effect on the discrimination boundary.

TABLE 5.14: Results obtained on three different combinations of normal and anomalous latent variables. Second column: results on the latent variables in Fig. 5.14 (a). Third column: results on the latent variables in Fig. 5.14 (b). Fourth column: results on the latent variables in Fig. 5.14 (c). th indicates the truncated values for the anomalous latent variables. Boldface indicates the best result.

Normal latent	$N(0,1)$	$N(0,1)$	$U(-1,1)$
Anomalous latent	$N(0,\sigma^2)$ $\sigma = 4$	$N(0,\sigma^2)$ $\sigma = 4, th = \pm 2$	$U(-2,2)$ $th = \pm 1$
Carpet	**0.813**	0.783	0.811
Grid	0.549	**0.643**	0.561
Leather	0.980	**0.982**	0.981
Tile	**0.989**	0.986	0.987
Wood	**0.827**	0.813	0.817
Bottle	0.913	**0.928**	0.806
Cable	0.880	0.857	**0.882**
Capsule	0.571	**0.583**	0.580
Hazelnut	0.929	0.954	**0.971**
Metalnut	0.729	**0.760**	0.739
Pill	0.714	**0.748**	0.728
Screw	0.731	0.755	**0.789**
Toothbrush	**0.578**	0.522	0.500
Transistor	**0.863**	0.833	0.855
Zipper	0.974	**0.981**	0.977
mean	0.803±0.151	**0.809**±0.146	0.799 ±0.157

Chapter 6

Conclusion

The key finding of this study is that adding pseudo-anomalous data to training improves the anomaly detection performance of the discriminator, as shown in Table 5.8. Fake-anomalous data, one of the pseudo-anomalous data types, are generated from anomalous latent variables with high entropy. This method has fewer concerns about biases than adding out-of-distribution data to the training (Kawachi, Koizumi, and Harada, 2018; Hendrycks, Mazeika, and Dietterich, 2018) or using prior knowledge to generate pseudo-anomalous data (Li et al., 2021).

Furthermore, our proposed method uses only the discriminator for anomaly detection, whereas other image-based methods use multiple networks such as encoders, decoders, or both. Thus, the prediction time of our method is faster.

PatchCore (Roth et al., 2022) is a state-of-the-art feature-based method that uses pre-trained models for images and cannot utilize other types of data. The same is true for other high-performance methods. By contrast, our proposed method can be directly applied to both images and features. Standard GANs have the potential to approximate any data distribution. ALGAN, an extension of standard GANs, has the same potential.

The evaluation in this study focused on image data. However, it would be interesting to see the performance of ALGAN for anomaly or novelty detection in other data types, such as signals (Brophy et al., 2021) and text (de Rosa and Papa, 2021). This is an important topic for future work.

In this thesis, we proposed a novel GAN-based anomaly detection method called ALGAN. The ALGAN generator provides pseudo-anomalous data as well as fake-normal data, by introducing anomalous states in the latent variable. The ALGAN discriminator distinguishes between the group of real-normal data and the group of fake-normal and pseudo-anomalous data.

The proposed method for generating pseudo-anomalous data can be applied to both images and feature vectors. We applied it to three anomaly detection benchmarks and demonstrated its high accuracy.

On MVTec-AD, ALGAN-image achieved more than 10% higher average accuracy than conventional image-based methods, and ALGAN-feature exhibited comparable ability to the feature-based methods. On the COIL-100 dataset, ALGAN performed almost perfectly.

ALGAN exhibited remarkably fast predictions. Compared with methods trained on image data and features, ALGAN-image could predict 10.4 to 54.6 times faster while maintaining high performance, and ALGAN-feature could predict 1.3 to 2.2 times faster.

Bibliography

Ahmad, Subutai et al. (Nov. 2017). "Unsupervised real-time anomaly detection for streaming data". In: *Neurocomputing* 262, pp. 134–147. DOI: 10.1016/j.neucom.2017.04.070.

Akçay, Samet, Amir Atapour-Abarghouei, and Toby P Breckon (Dec. 2018). "GANomaly: Semi-supervised anomaly detection via adversarial training". In: *Asian Conference on Computer Vision*. Springer, pp. 622–637. DOI: 10.1007/978-3-030-20893-6_39.

– (July 2019). "Skip-GANomaly: Skip connected and adversarially trained encoder-decoder anomaly detection". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–8. DOI: 10.1109/IJCNN.2019.8851808.

Andrews, Jerone et al. (July 2016). "Transfer representation-learning for anomaly detection". In: *International Conference on Machine Learning, Anomaly Detection Workshop*. URL: https://sites.google.com/site/icmlworkshoponanomalydetection/.

Bergman, Liron, Niv Cohen, and Yedid Hoshen (Feb. 2020). "Deep nearest neighbor anomaly detection". In: *arXiv preprint arXiv:2002.10445*. URL: https://arxiv.org/abs/2002.10445.

Bergmann, Paul et al. (Feb. 2019a). "Improving unsupervised defect segmentation by applying structural similarity to autoencoders". In: *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. INSTICC, pp. 372–380. DOI: 10.5220/0007364503720380.

Bergmann, Paul et al. (June 2019b). "MVTec AD–a comprehensive real-world dataset for unsupervised anomaly detection". In: *Conference on Computer*

Vision and Pattern Recognition. IEEE, pp. 9592–9600. DOI: 10.1109/CVPR.2019.00982.

Breunig, Markus M. et al. (June 2000). "LOF: Identifying density-based local outliers". In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. Vol. 29. Association for Computing Machinery, pp. 93–104. ISBN: 1581132174. DOI: 10.1145/342009.335388.

Brophy, Eoin et al. (July 2021). "Generative adversarial networks in time series: A survey and taxonomy". In: *arXiv preprint arXiv:2107.11098*. URL: https://arxiv.org/abs/2107.11098.

Chalapathy, Raghavendra and Sanjay Chawla (Jan. 2019). "Deep learning for anomaly detection: A survey". In: *arXiv preprint arXiv:1901.03407*. URL: https://arxiv.org/abs/1901.03407.

Chalapathy, Raghavendra, Aditya Krishna Menon, and Sanjay Chawla (Feb. 2018). "Anomaly detection using one-class neural networks". In: *arXiv preprint arXiv:1802.06360*. URL: https://arxiv.org/abs/1802.06360.

Chandola, Varun, Arindam Banerjee, and Vipin Kumar (July 2009). "Anomaly detection: A survey". In: *ACM Computing Surveys* 41.3. ISSN: 0360-0300. DOI: 10.1145/1541880.1541882.

Chatillon, Pierrick and Coloma Ballester (Aug. 2020). "History-based anomaly detector: An adversarial approach to anomaly detection". In: *Proceedings of SAI Intelligent Systems Conference*. Springer, pp. 761–776. DOI: 10.1007/978-3-030-55180-3_58.

Cohen, Niv and Yedid Hoshen (May 2020). "Sub-image anomaly detection with deep pyramid correspondences". In: *arXiv preprint arXiv:2005.02357*. URL: https://arxiv.org/abs/2005.02357.

de Rosa, Gustavo H. and João P. Papa (Nov. 2021). "A survey on text generation using generative adversarial networks". In: *Pattern Recognition* 119, p. 108098. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2021.108098.

Defard, Thomas et al. (Mar. 2021). "PaDiM: A patch distribution modeling framework for anomaly detection and localization". In: *International Conference on Pattern Recognition*. Springer, pp. 475–489. DOI: 10.1007/978-3-030-68799-1_35.

Deng, Jia et al. (June 2009). "Imagenet: A large-scale hierarchical image database". In: *Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (Apr. 2017). "Density estimation using Real NVP". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=HkpbnH9lx.

Donahue, Jeff, Philipp Krähenbühl, and Trevor Darrell (Apr. 2017). "Adversarial feature learning". In: *International Conference on Learning Representation*. URL: https://openreview.net/forum?id=BJtNZAFgg.

Dumoulin, Vincent et al. (Apr. 2017). "Adversarially learned inference". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=B1ElR4cgg.

Fan, Yaxiang et al. (June 2020). "Video anomaly detection and localization via Gaussian mixture fully convolutional variational autoencoder". In: *Computer Vision and Image Understanding* 195, p. 102920. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2020.102920.

Goodfellow, Ian et al. (Dec. 2014). "Generative adversarial nets". In: *Advances in Neural Information Processing Systems*, pp. 2672–2680. URL: https://papers.nips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html.

Grover, Aditya, Manik Dhar, and Stefano Ermon (Apr. 2018). "Flow-GAN: Combining maximum likelihood and adversarial learning in generative models". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. URL: https://dl.acm.org/doi/abs/10.5555/3504035.3504410.

Guo, Xinjian et al. (Oct. 2008). "On the class imbalance problem". In: *International Conference on Natural Computation*. Vol. 4. IEEE, pp. 192–201. DOI: 10.1109/ICNC.2008.871.

Han, Song, Huizi Mao, and William J Dally (May 2016). "Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding". In: *International Conference on Learning Representations*. URL: https://arxiv.org/abs/1510.00149.

Han, Song et al. (Dec. 2015). "Learning both weights and connections for efficient neural network". In: *Advances in Neural Information Processing Systems*. Vol. 28, pp. 1135–1143. URL: https://dl.acm.org/doi/10.5555/2969239.2969366.

He, Kaiming et al. (June 2016). "Deep residual learning for image recognition". In: *Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

Hendrycks, Dan, Mantas Mazeika, and Thomas Dietterich (May 2018). "Deep anomaly detection with outlier exposure". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=HyxCxhRcY7.

Heusel, Martin et al. (Dec. 2017). "GANs trained by a two time-scale update rule converge to a local Nash equilibrium". In: *Advances in Neural Information Processing Systems*. Vol. 30, pp. 6629–6640. URL: https://dl.acm.org/doi/10.5555/3295222.3295408.

Hinton, Geoffrey, Oriol Vinyals, Jeff Dean, et al. (Mar. 2015). "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531*. URL: https://arxiv.org/abs/1503.02531.

Huang, Yibin, Congying Qiu, and Kui Yuan (Jan. 2020). "Surface defect saliency of magnetic tile". In: *The Visual Computer* 36.1, pp. 85–96. DOI: 10.1007/s00371-018-1588-5.

Ioffe, Sergey and Christian Szegedy (July 2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International Conference on Machine Learning*. PMLR, pp. 448–456. URL: https://dl.acm.org/doi/10.5555/3045118.3045167.

Jacob, Benoit et al. (June 2018). "Quantization and training of neural networks for efficient integer-arithmetic-only inference". In: *Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 2704–2713. URL: https://openaccess.thecvf.com/content_cvpr_2018/html/Jacob_Quantization_and_Training_CVPR_2018_paper.html.

Johnson, Jeff, Matthijs Douze, and Hervé Jégou (July 2021). "Billion-scale similarity search with gpus". In: *IEEE Transactions on Big Data* 7.3, pp. 535–547. DOI: 10.1109/TBDATA.2019.2921572.

Johnson, Justin M and Taghi M Khoshgoftaar (Mar. 2019). "Survey on deep learning with class imbalance". In: *Journal of Big Data* 6.1, p. 27. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0192-5.

Kawachi, Yuta, Yuma Koizumi, and Noboru Harada (Apr. 2018). "Complementary set variational autoencoder for supervised anomaly detection". In: *International Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 2366–2370. DOI: 10.1109/ICASSP.2018.8462181.

Kingma, Diederick P and Jimmy Ba (May 2015). "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=8gmWwjFyLj.

Kingma, Diederik P and Max Welling (Apr. 2014). "Auto-encoding variational bayes". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=33X9fd2-9FyZd.

Krishnamoorthi, Raghuraman (June 2018). "Quantizing deep convolutional networks for efficient inference: A whitepaper". In: *arXiv preprint arXiv:1806.08342*. URL: https://arxiv.org/abs/1806.08342.

Larochelle, Hugo and Iain Murray (Jan. 2011). "The neural autoregressive distribution estimator". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR, pp. 29–37. URL: https://proceedings.mlr.press/v15/larochelle11a.html.

LeCun, Yann and Fu Jie Huang (Jan. 2005). "Loss functions for discriminative training of energy-based models". In: *International Workshop on Artificial Intelligence and Statistics*. PMLR, pp. 206–213. URL: https://proceedings.mlr.press/r5/lecun05a.html.

Li, Chun-Liang et al. (June 2021). "CutPaste: Self-supervised learning for anomaly detection and localization". In: *Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 9664–9674. DOI: 10.1109/CVPR46437.2021.00954.

Liu, Li et al. (Feb. 2020a). "Deep learning for generic object detection: A survey". In: *International Journal of Computer Vision* 128.2, pp. 261–318. DOI: 10.1007/s11263-019-01247-4.

Liu, Wenqian et al. (June 2020b). "Towards visually explaining variational autoencoders". In: *Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 8642–8651. DOI: 10.1109/CVPR42600.2020.00867.

Maas, Andrew L., Awni Y. Hannun, and Andrew Y. Ng (June 2013). "Rectifier nonlinearities improve neural network acoustic models". In: *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. URL: https://sites.google.com/site/deeplearningicml2013/.

MacQueen, James (Jan. 1967). "Some methods for classification and analysis of multivariate observations". In: *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 51, pp. 281–297.

Menghani, Gaurav (June 2021). "Efficient deep learning: A survey on making deep learning models smaller, faster, and better". In: *arXiv preprint arXiv:2106.08962*. URL: https://arxiv.org/abs/2106.08962.

Miyato, Takeru et al. (Apr. 2018). "Spectral normalization for generative adversarial networks". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=B1QRgziT-.

Mohamed, Shakir and Balaji Lakshminarayanan (Oct. 2016). "Learning in implicit generative models". In: *arXiv preprint arXiv:1610.03483*. URL: https://arxiv.org/abs/1610.03483.

Nair, Vinod and Geoffrey E. Hinton (June 2010). "Rectified linear units improve restricted Boltzmann machines". In: *International Conference on Machine Learning*. PMLR, pp. 807–814. URL: https://dl.acm.org/doi/10.5555/3104322.3104425.

Nene, Sameer A, Shree K Nayar, Hiroshi Murase, et al. (1996). "Columbia Object Image Library (COIL-100)". In: URL: https://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php.

Pang, Guansong et al. (Mar. 2021). "Deep learning for anomaly detection: A review". In: *ACM Computing Surveys* 54.2. ISSN: 0360-0300. DOI: 10.1145/3439950.

Parzen, Emanuel (Sept. 1962). "On estimation of a probability density function and mode". In: *The annals of mathematical statistics* 33.3, pp. 1065–1076. DOI: 10.1214/aoms/1177704472.

Pedregosa, F. et al. (Oct. 2011). "Scikit-learn: Machine learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830. URL: https://www.jmlr.org/papers/v12/pedregosa11a.html.

Perales Gómez, Ángel Luis et al. (Dec. 2019). "On the generation of anomaly detection datasets in industrial control systems". In: *IEEE Access* 7, pp. 177460–177473. DOI: 10.1109/ACCESS.2019.2958284.

Pourreza, Masoud et al. (Jan. 2021). "G2D: Generate to detect anomaly". In: *Winter Conference on Applications of Computer Vision*. IEEE, pp. 2003–2012. DOI: 10.1109/WACV48630.2021.00205.

Radford, Alec, Luke Metz, and Soumith Chintala (May 2016). "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *International Conference on Learning Representations*. URL: https://arxiv.org/abs/1511.06434.

Rezende, Danilo and Shakir Mohamed (July 2015). "Variational inference with normalizing flows". In: *International Conference on Machine Learning*. PMLR, pp. 1530–1538. URL: https://proceedings.mlr.press/v37/rezende15.html.

Rippel, Oliver, Patrick Mertens, and Dorit Merhof (Jan. 2021). "Modeling the distribution of normal data in pre-trained deep features for anomaly detection". In: *International Conference on Pattern Recognition*. IEEE, pp. 6726–6733. DOI: 10.1109/ICPR48806.2021.9412109.

Rodda, Sireesha and Uma Shankar Rao Erothi (Mar. 2016). "Class imbalance problem in the network intrusion detection systems". In: *International Conference on Electrical, Electronics, and Optimization Techniques*. IEEE, pp. 2685–2688. DOI: 10.1109/ICEEOT.2016.7755181.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (Oct. 2015). "U-Net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

Rosenblatt, Murray (Sept. 1956). "Remarks on some nonparametric estimates of a density function". In: *The Annals of Mathematical Statistics* 27.3, pp. 832–837. DOI: 10.1214/aoms/1177728190.

Roth, Karsten et al. (June 2022). "Towards total recall in industrial anomaly detection". In: *Conference on Computer Vision and Pattern Recognition*. URL: https://arxiv.org/abs/2106.08265.

Rudolph, Marco, Bastian Wandt, and Bodo Rosenhahn (Jan. 2021). "Same same but DifferNet: semi-supervised defect detection with normalizing

flows". In: *Winter Conference on Applications of Computer Vision*. IEEE, pp. 1907–1916. DOI: 10.1109/WACV48630.2021.00195.

Ruff, Lukas et al. (May 2021). "A unifying review of deep and shallow anomaly detection". In: *Proceedings of the IEEE* 109.5, pp. 756–795. DOI: 10.1109/JPROC.2021.3052449.

Sabokrou, Mohammad et al. (June 2015). "Real-time anomaly detection and localization in crowded scenes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. IEEE, pp. 56–62. DOI: 10.1109/CVPRW.2015.7301284.

Sabokrou, Mohammad et al. (June 2018). "Adversarially learned one-class classifier for novelty detection". In: *Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 3379–3388. DOI: 10.1109/CVPR.2018.00356.

Salimans, Tim et al. (Dec. 2016). "Improved techniques for training GANs". In: *Advances in Neural Information Processing Systems*. Vol. 29, pp. 2234–2242. URL: https://papers.nips.cc/paper/2016/hash/8a3363abe792db2d8761d6403605aeb7-Abstract.html.

Schlegl, Thomas et al. (May 2017). "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery". In: *International Conference on Information Processing in Medical Imaging*. Springer, pp. 146–157. DOI: 10.1007/978-3-319-59050-9_12.

Schölkopf, Bernhard et al. (July 2001). "Estimating support of a high-dimensional distribution". In: *Neural Computation* 13, pp. 1443–1471. ISSN: 0899-7667. DOI: 10.1162/089976601750264965.

Sener, Ozan and Silvio Savarese (Apr. 2018). "Active learning for convolutional neural networks: A core-set approach". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=H1aIuk-RW.

Simonyan, Karen and Andrew Zisserman (May 2015). "Very deep convolutional networks for large-scale image recognition". In: *International Conference on Learning Representations*. URL: https://arxiv.org/abs/1409.1556.

Sinha, Samarth et al. (Apr. 2020). "Small-GAN: Speeding up GAN training using core-sets". In: *International Conference on Machine Learning*. PMLR, pp. 9005–9015. URL: https://proceedings.mlr.press/v119/sinha20b.html.

Sohl-Dickstein, Jascha et al. (July 2015). "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International Conference on Machine Learning*. PMLR, pp. 2256–2265. URL: https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Song, Yang and Stefano Ermon (Dec. 2019). "Generative modeling by estimating gradients of the data distribution". In: *Advances in Neural Information Processing Systems* 32. URL: https://dl.acm.org/doi/10.5555/3454287.3455354.

Tan, Mingxing and Quoc Le (June 2019). "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International Conference on Machine Learning*. PMLR, pp. 6105–6114. URL: http://proceedings.mlr.press/v97/tan19a.html.

Urban, Gregor et al. (Apr. 2017). "Do deep convolutional nets really need to be deep and convolutional?" In: *International Conference on Learning Representation*. URL: https://openreview.net/forum?id=r10FA8Kxg.

Zagoruyko, Sergey and Nikos Komodakis (Sept. 2016). "Wide residual networks". In: *British Machine Vision Conference*. British Machine Vision Association, pp. 87.1–87.12. DOI: 10.5244/C.30.87.

Zaheer, Muhammad Zaigham et al. (June 2020). "Old is gold: Redefining the adversarially learned one-class classifier training paradigm". In: *Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 14183–14193. DOI: 10.1109/CVPR42600.2020.01419.

Zenati, Houssam et al. (Feb. 2018). "Efficient GAN-based anomaly detection". In: *arXiv preprint arXiv:1802.06222*. URL: https://arxiv.org/abs/1802.06222.

Zhang, Zhiwei, Shifeng Chen, and Lei Sun (Jan. 2021). "P-KDGAN: Progressive knowledge distillation with GANs for one-class novelty detection". In: *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence (IJCAI-20)*, pp. 3237–3243. DOI: https://dl.acm.org/doi/abs/10.5555/3491440.3491888.