EAI Endorsed Transactions
on AI and Robotics

Research Article EAI.EU

Two-Stage Metal Surface Defect Detection and Classification System Based on VAEGAN and Class-Embedding

Sheng-Tzong Cheng [1], Chun-Wei Yeh [2, *]

[1]1st Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan, stcheng@mail.ncku.edu.tw
[2]2nd Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan, nf6101078@gs.ncku.edu.tw

Abstract

Surface defect detection is crucial in maintaining product quality across various industries. Traditional manual inspection methods are often time-consuming and subjective, which can result in inaccuracies and higher production costs. With the use of deep learning techniques, significant advancements have been made in automating the process of surface defect detection in recent years. Moreover, deep learning includes a variety of techniques, and image recognition-based deep learning is especially relevant to our field of study, which is the main focus of this research paper.

In the industrial surface defect detection field, researchers have always aimed to create a deep learning-based intelligent defect detection system that achieves near-zero defect rates while maintaining a lightweight, efficient, and cost-effective solution. However, these objectives often conflict with each other, and it is unrealistic to develop a model that can achieve all of them simultaneously. Some trade-offs must be made. If accuracy is the top priority, a large amount of defective data labeled for supervised learning is usually required. If lightweight and low cost is prioritized, a simple small model such as Auto-Encoder is usually used, along with a large number of flawless images for unsupervised learning to minimize the cost of labeling.

As mentioned before, it is very difficult to design a single model that can achieve all of them simultaneously. However, present-day studies frequently center on accomplishing those tasks using a single model and rarely address the multi-model architecture. This paper presents a Surface Defect Detection and Classification System that builds on the current state-of-the-art model in the field of surface defect detection, along with the zero-shot learning (ZSL) classifier based on VAEGAN and the Variational Auto-Encoder developed by our laboratory.

We have developed a Surface Defect Detection and Classification System that effectively integrates the aforementioned three models. It has been validated on a dataset of metal surface defects, yielding excellent results. This system not only achieves defect detection rates that comply with industrial standards and low false positive rates but also maintains characteristics such as lightweight, low latency, and low annotation cost. In addition to achieving the above goals, this system can also instantly identify and issue anomaly notifications when there are unseen anomalies, which is generally impossible to do with supervised learning anomaly detection models.

Keywords: Industry 4.0, Intelligent manufacturing, Quality control, Industrial surface defect detection, Anomaly detection, Zero-shot learning

Received on 10 August 2023, accepted on 27 November 2023, published on 29 November 2023

doi: 10.4108/airo.3695

*Corresponding author. Email: nf6101078@gs.ncku.edu.tw

1. Introduction

Anomaly detection is an area of deep learning with many practical applications. These applications range from signal analysis for detecting credit card fraud to predicting machine maintenance. Image analysis is also part of the anomaly detection field, such as examining brain tumor histopathology and detecting product surface defects. Instead of being a single technique, it is more of a problem-solving approach.

Today, many manufacturers still rely on manual methods for surface defect detection in production, which is time-consuming and labor-intensive. Moreover, the accuracy heavily depends on the experience of technical personnel to correctly identify and differentiate various defects hidden on the surface. Numerous studies have demonstrated that utilizing deep learning techniques to address the issue of product surface defect detection has significantly outperformed human experts in terms of time, cost, and accuracy.

This paper approaches the problem of product surface defect detection in production from the perspective of anomaly detection, utilizing deep learning techniques to achieve automated defect detection and intelligent analysis.

At present, most of the research using deep learning technology and applied to industrial surface defect detection can be roughly divided into two directions: supervised learning and unsupervised learning.

In order to achieve high standards in industrial defect detection, a high level of accuracy is necessary, which is the strength of supervised learning. However, supervised learning typically requires a large amount of annotated abnormal data, which may not be available in practical production scenarios. On the other hand, the advantage of unsupervised learning is that it can perform anomaly detection without the need for labeled data (1). One common technique is to use Auto-Encoders and Variational Auto-Encoder (2), which learn to reconstruct the distribution of normal data by inputting a large amount of unlabeled normal data. Therefore, any data the model cannot reconstruct can be identified as anomalies. The benefit of using this approach is that it does not require labeled data, and the model can still successfully detect unseen anomalies that have not occurred in the past on the production line, which is not possible with supervised models. However, there are two main disadvantages of utilizing unsupervised learning for detecting surface defects. The first drawback is the need for more accuracy, as the correct rate often falls below the standard required for industrial production. The second drawback is that while the model can identify unseen anomalies, it can only perform simple binary classification. The model typically can only determine whether the input image is similar enough to the training data. Any data that deviates significantly from the training data is treated as unseen anomalies.

In the field of supervised learning for detecting surface defects, the Segmentation and Decision Network (Seg-Dec Net) (3) has demonstrated excellent performance in terms of accuracy, labeling cost, and recognition time. This model uses a supervised learning approach and is broken down into two stages: segmentation and decision. Its primary objective is to identify defects in images and mark their locations. Several studies have shown that supervised learning combined with defect location annotation can effectively improve the performance of defect detection. However, Seg-Dec Net only treats surface defect detection as a binary classification problem, focusing solely on identifying the presence of defects in images without the ability to distinguish between different types of surface materials or different types of defects.

The proposed Surface Defect Detection and Classification System not only achieves high accuracy in surface defect detection and classification tasks, but also has the ability to recognize unseen anomalies. To enable the recognition of unseen anomalies, we have incorporated techniques from the field of zero-shot learning, combining class embedding to allow the model to learn semantic information from the seen classes it has observed. By integrating visual and semantic feature information, the model can distinguish between seen and unseen classes. This system allows for high accuracy using a small amount of labeled data, which meets industry requirements. Additionally, it can also efficiently recognize unseen anomalies and achieve soft real-time detection capabilities.

To summarize, the main contributions of this paper are as follows:

- Propose a novel Surface Defect Detection and Classification System that combines a zero-shot learning classifier with multiple binary CNN defect detectors, enabling the model to simultaneously detect and classify defects, as well as recognize unknown anomalies
- Introduce DAGM_MIX dataset, which combines the benign and defect data together in each class
- Introduce class embedding (CE) of DAGM_MIX dataset for training the zero-shot learning (ZSL) classifier
- Proposed the concept of Reverse Autoencoder (R-AE), emphasizing the reconstruction quality of class embedding (CE)
- Utilize the Customized Variational Auto Encoder (VAE) model developed by our laboratory to enable the model to generate its own class embedding (referred to as Learned-CE), which further enhances the accuracy of the zero-shot learning (ZSL) classifier
- The surface defect detection and classification system is designed to simulate real-world scenarios in factory production lines, with features such as anomaly detection and classification, defect detection, high accuracy, low latency, and cost-effectiveness

2. Related work

Research on anomaly detection has a long history, with early work going back as far as Edgeworth (1887) and is concerned with finding unusual or anomalous samples in a corpus of data. An extensive overview of traditional anomaly detection methods as well as open challenges, can be found in Chandola et al. (4).

Defect detection on product surfaces is a critical task in manufacturing and quality control. Traditional defect detection methods rely on manual inspection, which is time-consuming and error-prone. With the recent advancements in deep learning, there has been growing interest in using deep learning techniques for defect detection on product surfaces.

There are many different approaches to solving this problem. One popular approach is to use convolutional neural network (CNN) for defect detection. CNN is well-suited for image analysis tasks and can learn to identify complex patterns in images. Researchers have proposed various CNN-based models for defect detection, such as Region-based CNN (R-CNN), Faster R-CNN, and Mask R-CNN. These models can detect defects of different shapes and sizes and can achieve high accuracy in defect detection.

The Segmentation-Based Defect Detector(SBDD) model, which will present later, is based on the current state-of-the-art paper using CNN (3) (5) (6). They considered the surface defect detection task as a binary classification issue and obtained significant results using two-stage training and label optimization.

Recently, several object detection techniques (7) (8) have demonstrated promising results in automating surface defect detection. One popular choice is You Only Look Once (YOLO) due to its speed and accuracy.

YOLO is a deep learning model that uses convolutional neural network (CNN) to detect and classify objects in real-time. It is an improvement over previous versions of YOLO in terms of accuracy, speed, and robustness to occlusions and small object sizes. YOLO is based on the anchor box approach, which involves defining a set of boxes of different sizes and aspect ratios that are used to predict object locations and sizes. This approach allows for efficient training and detection of objects, making it well-suited for surface defect detection. However, there are some drawbacks to using YOLO for surface defect detection:

- This method requires a large amount of image dataset for training, and collecting and annotating these data may require a significant amount of time and manpower
- The method may have missed detections and false alarms, especially when the defect appearance and location differ from known samples, which is one of the main problems this paper aims to solve
- Using YOLO for surface defect detection requires model debugging and optimization to improve its accuracy and performance

Another approach to surface defect detection is the use of generative models, such as Variational Autoencoder (VAE) and Generative Adversarial Networks (GAN).

Recently, Generative Adversarial Network (9) (GAN) have been employed to synthesize unseen class features, which are then used in a fully supervised setting to train ZSL classifiers (10). GAN is composed of two neural networks that collaborate to produce realistic images. A conditional Wasserstein GAN (11) (WGAN) is used along with a seen category classifier to learn the generator for unseen class feature synthesis (10). This is achieved by using a WGAN loss and a classification loss.

Researchers have proposed using GAN for anomaly detection on product surfaces. GAN is mainly for defect image reconstruction and recognition (12), where the model learns to distinguish between normal and abnormal samples. This approach has shown promising results in defect detection tasks, as it can learn to identify subtle defects that are challenging to detect with conventional methods. VAE-GAN (13) (14) is a hybrid model that combines the advantages of VAE and GAN. The VAE-GAN model is trained on a dataset of images with and without surface defects. The encoder of VAE-GAN learns the latent representation of the image, and the decoder generates the reconstructed image. The generator of GAN generates the fake image, and the discriminator distinguishes between real and fake images. The joint training of VAE and GAN ensures that the generated images are not only similar to the input images but also visually realistic.

To the best of our knowledge, no one has yet applied Zero-shot learning (ZSL) techniques to the field of industrial metal surface defect detection. The proposed Surface-Based Anomaly Detector(SBAD) model in this paper, which will present later, is based on the TF-VAEGAN (13) architecture, which combines VAE and GAN models. It introduces innovative concepts such as incorporating Class Embedding (CE) at all stages of training and using the decoder to reconstruct CE. The model has been demonstrated to achieve significant accuracy improvement on multiple datasets.

Additionally, researchers have explored using transfer learning techniques to improve the performance of defect detection models (15). Transfer learning involves using a pre-trained model on a large dataset and fine-tuning it on a smaller dataset for a specific task. By leveraging the pre-trained model's knowledge, transfer learning can improve the performance of defect detection models, especially in scenarios with limited training data.

In conclusion, deep learning techniques have shown great potential in defect detection on product surfaces. By leveraging the power of CNN, VAE, GAN, and transfer learning, defect detection models can achieve high accuracy and robustness, improving the efficiency and accuracy of quality control in manufacturing. However, there are still challenges to be addressed, such as the requirement for large amounts of labeled data and the robustness of the models to different lighting and viewing conditions. Further research is required to address these challenges and improve the performance of defect detection models in real-world scenarios.

2.1. Transfer Learning

Transfer Learning serves as an approach to address the challenges of data scarcity, especially in obtaining labeled data, by exploring the relatedness between datasets of similar tasks.

In Transfer Learning, the data required for the target task is referred to as target data, while data that is similar to the task but not directly related is termed source data. The objective of Transfer Learning is to leverage the assistance of source data to alleviate the difficulties and costs associated with collecting target data.

Zero-Shot Learning (ZSL) is a subfield of Transfer Learning wherein the target data lacks any labels while the source data is labeled. During training, only the labeled source data is provided to the model, with the expectation that the model can discern the categories among the target data during testing.

It is evident that relying solely on visual information is insufficient to achieve the aforementioned goals. Therefore, ZSL requires the prior definition of attributes, also known as Class-Embedding or CE, based on the characteristics of each class. The collection of attributes defined for each class represents the Class-Embedding, and different combinations of Class-Embedding can represent distinct categories.

Class-Embedding

In Zero-shot learning, Class-Embedding is a technique for representing category information. It is used to convert categories into vector representations for use in zero-shot learning tasks.

In traditional supervised learning, we typically have a large number of labeled training samples, with each sample associated with a specific category. However, in zero-shot learning, the challenge is to learn how to classify new categories that have not appeared in the training set.

Class-Embedding addresses this problem by generating a vector for each category, encoding the semantic information of the category into a continuous feature representation. This allows us to use these class embedding vectors during the training phase and associate them with existing training samples. Then, during the testing phase, we can utilize these embedding vectors to classify new categories.

Class-Embedding (CE) plays a crucial role in Zero-Shot Learning (ZSL) as it represents the key attribute features of a category, encapsulating the core information of that class. Currently, there are several methods for generating Class-Embedding. Apart from relying on domain knowledge and understanding of the dataset itself through manual definition, natural language processing techniques are also employed. These techniques leverage resources available on the internet, such as Wikipedia, and utilize word-to-vector approaches to transform textual descriptions of a category into numerical representations.

To summarize, Class-Embedding is a technique that incorporates category information into vector representations, which is essential in zero-shot learning. This approach allows us to classify new categories by utilizing semantic relationships and attribute information between categories, thereby expanding the capabilities of traditional supervised learning.

3. System Design and Methodology

3.1. Two-Stage Metal Surface Defect Detection and Classification System

In the following chapters, we will start by providing an overview of the system architecture and design, as well as the purpose and benefits of using the two-stage model. We will also cover the advantages and goals of the system design. Then, we will give separate introductions to the SBAD and SBDD models and their architectures. Finally, we will explain the proposed DAGM_MIX dataset and the design of Class-Embedding specifically for this dataset.

The proposed system consists of two main models, namely Surface-Based Anomaly Detector (SBAD) and Segmentation-Based Defect Detector (SBDD); SBDD treats defect detection as a binary classification problem, with the task of detecting whether the input image contains defects and displaying the location of the defects. On the other hand, SBAD can be viewed as a pre-classifier for the input images before SBDD, with its primary task being to recognize unseen anomalies and its secondary task being to classify seen categories.

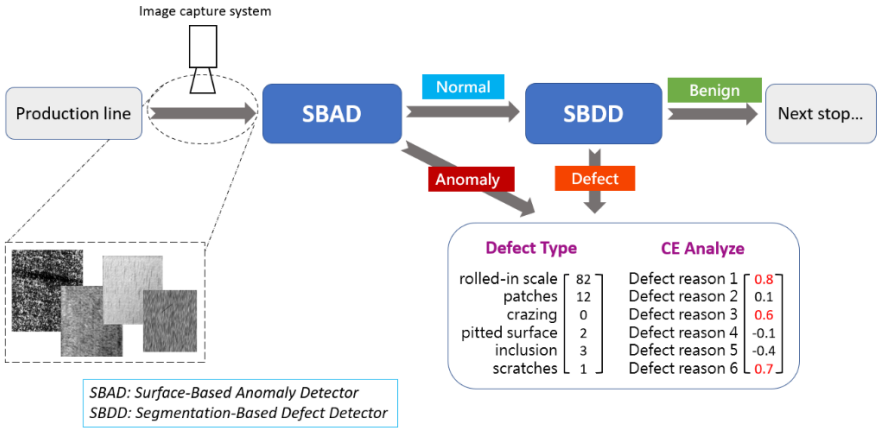

Figure 1. Metal Surface Defect Detection and Classification System flow chart – 1

The system is a diagram that models real-world situations occurring on production lines. It is composed of two main stages: SBAD handles anomaly detection and classification, while SBDD handles defect detection, as displayed in **Figure 1**.

3.1.1. System Design

The entire system can be divided into two stages: the unknown anomaly detection and classification stage, handled by SBAD, and the defect recognition stage, handled by SBDD.

Firstly, the Image Capture System is responsible for capturing images of products on the production line. The captured images are then input to SBAD. The primary task of SBAD is determining whether the current input image belongs to an unknown category. In most cases, the input to SBAD consists of normal seen class images, so SBAD generally does not produce any output or trigger anomaly alerts. However, when SBAD identifies that the current input image does not belong to any known category, referred to as an "unseen anomaly," the model outputs the result, triggers an anomaly alert in the system, and prevents the transmission of that image data to SBDD. The secondary task of SBAD is to classify known categories. When the production line is not isolated, multiple products with different surface materials and defect types may appear simultaneously on a single production line. Therefore, we rely on SBAD to classify these different products into categories, ensuring that the inputs to SBDD belong to the same category. At this stage, the tasks of the first stage are completed.

The second stage is the defect recognition stage, led by SBDD. SBDD can be considered a defect detector that identifies defects in an image, performing a binary classification task. If a defect is detected, it triggers a defect alert and indicates the specific location of the defect. However, whether encountering an unknown category in the first stage or detecting defects in the second stage, the system not only issues an alert but also outputs the corresponding Class-Embedding (CE). This allows inspection personnel to make an initial assessment of the cause behind the alert. Furthermore, for CEs generated from unknown categories, the model's output is based on the learned features from known categories.

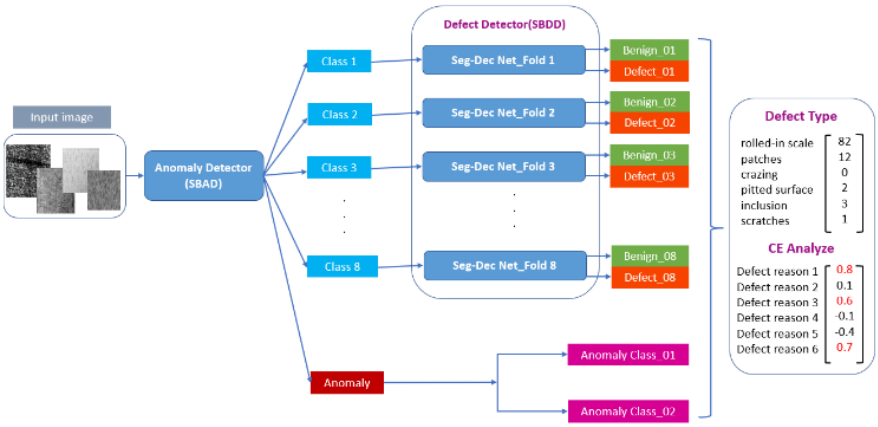

Figure 2. Metal Surface Defect Detection and Classification System flow chart – 2

To be more specific, **Figure 2** illustrates the detailed process of the system operation. Assuming ten classes are in the dataset, each class would have two labels, namely "benign" and "defect." If there are eight known classes and two unknown classes, we need to train eight SBDD models for all known classes. The purpose of each SBDD model is to detect whether there is a defect in the images of its respective class. The SBAD's role is to ensure that each image input to the SBDD belongs to the corresponding class and cannot contain any images of other known or unknown classes. Additionally, when an image of an unknown class is detected, the SBAD model needs to issue an abnormal alert and prevent it from being input into the SBDD.

3.1.2. Surface-Based Anomaly Detector

The Surface-Based Anomaly Detector (SBAD) plays a crucial role in the system. SBAD has two primary tasks: functioning as a classifier for the input of SBDD and performing anomaly detection for the entire system, responsible for detecting unseen anomalies.

The first priority of the SBAD is to identify the difference between Normal and Anomaly. Most of the time, the output of the detector will be normal; it would only output an anomaly when the detector thinks this data does not belong to any known classes. The second priority is classifying the known classes to ensure that every input data of the SBDD belongs to its own category.

SBAD is a classifier that utilizes surface textures as the basis for classification. In order for SBDD to perform binary classification tasks, SBAD needs to learn to treat benign and defective samples within the same class as the same cluster. To achieve this, we require a more flexible model than an Autoencoder (AE) so that the model can still successfully reconstruct the original image even if defects appear on

images with the same surface texture. Therefore, we adopt the Variational Autoencoder (VAE) model to reduce the differences between benign and defective samples in images with the same surface texture. However, due to the characteristics of industrial surface defect datasets, different classes often exhibit high similarity. If a VAE model is used alone, it will result in reduced differences between each class, i.e., images with different surface textures. This would increase the difficulty of classification for SBAD. Hence, we combine the VAE model with a Generative Adversarial Network (GAN) to form the VAEGAN architecture. Through the discriminative power of GAN, the quality of the images reconstructed by VAE for each category is improved, enhancing the difference between each category.

The training process for SBAD is split into two stages: the CE generation stage for training the VAE and the classification stage for training the VAEGAN. In the following sections, we will discuss the model architectures and training methods used in each of these stages.

CE generation stage for training the VAE model:

At the CE generation stage, we use the VAE model alongside the CE we proposed (Table 1). The VAE model used in this paper is based on previous research in our laboratory (16), which has demonstrated that using VAE-generated CE can significantly improve accuracy in some mainstream zero-shot learning datasets, such as Animals with Attributes 2 (AWA2) (17) and Caltech-UCSD birds (CUB) (18) datasets, compared to using expert-defined CE. This paper applies this method to the field of industrial surface defect detection and achieves promising results.

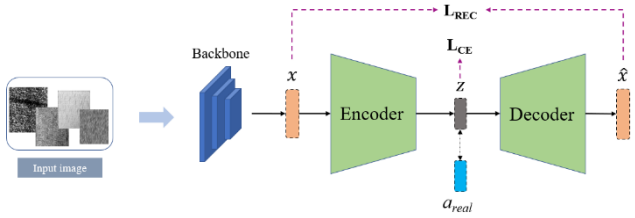

Figure 3. Proposed VAE model architecture

The training process of the VAE model is briefly described below, while the detailed model architecture and training details can be found in the original paper.

Firstly, the feature vector, denoted as $\boldsymbol{x}$, which is extracted by the backbone network, is used as the input of the Encoder. The goal of the Encoder is dimensionality reduction. It simplifies the feature vector into a 29-dimensional vector that matches the same dimensions of the designed Class Embedding, denoted as $\boldsymbol{a_{real}}$, which is presented in Table 1. We expect the Encoder to learn the most representative 29-dimensional vector, which is also known as the latent code, denoted as $\boldsymbol{Z}$, that can best represent the feature vector $\boldsymbol{x}$. Next, the latent code is used as the input of the Decoder. The goal of the Decoder is to reconstruct the feature vector $\boldsymbol{x}$ from the latent code so that the reconstructed vector, denoted as $\hat{\boldsymbol{x}}$, is as similar as possible to the original input $\boldsymbol{x}$.

The VAE model has two loss functions, namely the reconstruction loss function (L_{REC}) and the class embedding constrained loss function (L_{CE}).

$$BCE(y_i, \hat{y}_i) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \quad (1)$$

$$L_{REC} : 0.5 * BCE\left(x_{Input_{(i)}}, x_{Output_{(i)}}\right) \quad (2)$$

$$L_{CE} : -0.5 * [1 + log\sigma^2 + \sigma^2 + (\mu - CE_{REAL})^2] \quad (3)$$

Reconstruction loss (2) is computed using Binary Cross-Entropy (1). In the following, $\boldsymbol{n}$ represents the number of samples, and $\boldsymbol{y_i}$ denotes the true label of the i-th sample, where the label values represent the brightness of each pixel using 0 and 1. On the other hand, $\hat{\boldsymbol{y}}_{\boldsymbol{i}}$ represents the predicted probability of the i-th sample.

Reconstruction loss has been used to measure the difference in probability distribution between the reconstructed $\hat{\boldsymbol{x}}$ output by the Decoder and the original input $\boldsymbol{x}$. The optimization goal is to minimize the difference between them, meaning that we want the probability distribution of $\hat{\boldsymbol{x}}$ to be as similar as possible to the probability distribution of input $\boldsymbol{x}$.

Class embedding constrained loss (3) is measured using KL-divergence loss to evaluate the distance between the probability distributions learned by the Encoder and the real distributions. Unlike a typical KL-divergence loss, the CE-constrained loss adds a constraint on the CE, which means that the latent code produced by the Encoder not only needs to represent the input features well but also needs to be as similar as possible to our designated Real CE. Using these two loss functions, the Encoder can generate a latent code that can accurately reconstruct the original input through the Decoder and be distributed near the Real CE we designed. This indicates that the Encoder can find the 29-dimensional vector that best represents the original input $\boldsymbol{x}$, and each dimension of this vector has a meaningful value. By comparing the differences between the values of the latent code and the Real CE, we can observe the model's interpretation of different features and compare it with our human-designed Real CE.

After training the VAE, we take the average of the latent code produced by the Encoder to obtain the Learned CE that best represents each seen class and replace the original Real CE with it for the subsequent training of the VAEGAN.

Classification stage for training the VAEGAN model:

VAEGAN is a model that merges Variational Auto Encoder (VAE) and Generative Adversarial Networks

(GAN) to generate high-quality synthetic images. It accomplishes this by training a VAE and a GAN simultaneously, capitalizing on the advantages of both models. During training, the VAE's Decoder acts as the GAN's Generator, attempting to reconstruct the original input image based on the latent code produced by the Encoder. Meanwhile, the Discriminator in the GAN identifies the difference between real and synthesized images generated by the Generator. The Generator creates images that closely resemble the original to mislead the Discriminator.

The VAEGAN model used in this paper is fine-tuned based on the TF-VAEGAN (13) model architecture. The paper introduces the concept of using another Decoder to reconstruct CE and suggests incorporating CE throughout all training phases to enhance the model's ability to distinguish between seen and unseen classes. The detailed model architecture and training details can be found in the original paper. The strong classification ability of the ZSL Anomaly Detector can be attributed to the design of the VAEGAN model. During training, VAEGAN not only has to classify seen classes but also has to be able to recognize unseen anomalies.

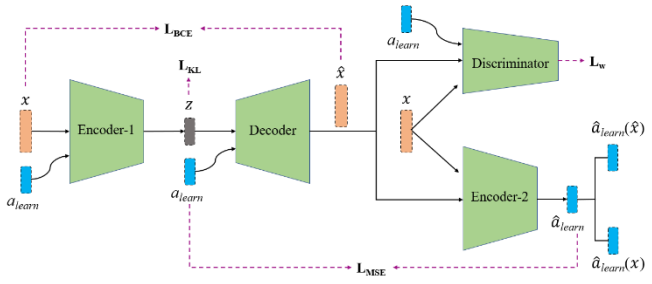

Figure 4. Proposed VAEGAN model architecture

The overall architecture of the proposed VAEGAN model is depicted in **Figure 4**. Unlike TF-VAEGAN, our proposed architecture includes two Encoders (Encoder-1, Encoder-2), a Decoder, and a Discriminator, with an emphasis on the reconstruction quality of Class-Embedding (CE). Next, we will provide a detailed description of the complete training procedure for VAEGAN.

After obtaining the Learned CE for each seen class in the previous stage, we assign the corresponding Learned CE based on the class of the seen class. Firstly, we input the seen class image into the backbone network to extract visual features $\boldsymbol{x}$. These features, along with the corresponding Learned CE (denoted as $\boldsymbol{a_{learn}}$ in **Figure 4**), are used as inputs for Encoder-1. Encoder-1 then outputs a latent code $\boldsymbol{Z}$. Next, $\boldsymbol{Z}$ and the corresponding $\boldsymbol{a_{learn}}$ are used as inputs for the Decoder, which synthesizes the features $\hat{\boldsymbol{x}}$ based on the input. Finally, as mentioned earlier, the Discriminator receives either $\boldsymbol{x}$ or $\hat{\boldsymbol{x}}$ as input. The training objective of the Discriminator is to learn to distinguish between real and synthesized features without knowing whether the current input is $\boldsymbol{x}$ or $\hat{\boldsymbol{x}}$.

Next, Encoder-2 is used to map the visual features $\boldsymbol{x}$ or $\hat{\boldsymbol{x}}$ to a latent attribute $\hat{\boldsymbol{a}}_{\boldsymbol{learn}}$. During training, the dissimilarity between $\boldsymbol{a_{learn}}$ and $\hat{\boldsymbol{a}}_{\boldsymbol{learn}}$ is minimized. During testing, when images of unseen anomalies appear, the dissimilarity between the latent attribute $\hat{\boldsymbol{a}}_{\boldsymbol{learn}}$ and $\boldsymbol{a_{learn}}$ will be enlarged. As a result, given an unseen test image, the high dissimilarity between the latent attribute $\hat{\boldsymbol{a}}_{\boldsymbol{learn}}$ and $\boldsymbol{a_{learn}}$ indicates an unseen anomaly.

On the other hand, we can consider the combination of the Decoder and Encoder-2 as a **Reversed-Autoencoder** (R-AE). The conventional autoencoder (AE) maps high-dimensional features to a lower-dimensional latent code, aiming to learn essential features from the original input data while disregarding less critical details. AE focuses on whether it can produce a sufficiently good latent code that represents the original high-dimensional features. In contrast, R-AE requires mapping a low-dimensional latent code to a high-dimensional feature representation, which is more challenging than AE. Therefore, R-AE heavily relies on the quality of the latent code, i.e., the $\boldsymbol{a_{learn}}$ and $\boldsymbol{Z}$ in **Figure 4**. If the quality of the latent code is good enough, the reconstruction by Encoder-2 becomes easier, and the key concern of R-AE is whether the reconstructed $\hat{\boldsymbol{a}}_{\boldsymbol{learn}}$ is sufficiently similar to the original $\boldsymbol{a_{learn}}$. During training, Encoder-2 receives the same input as the Discriminator without knowing whether the current input is $\boldsymbol{x}$ or $\hat{\boldsymbol{x}}$. The role of Encoder-2 is to reduce the current visual features to $\hat{\boldsymbol{a}}_{\boldsymbol{learn}}$ with the same dimensionality as $\boldsymbol{a_{learn}}$. Next, we will explain the loss functions used in each model.

Both Encoder-1 and Decoder together constitute the VAE, which is trained using Binary Cross-Entropy, denoted as L_{BCE}, and the KL divergence loss, denoted as L_{KL}. Likewise, both Generator(Decoder) and Discriminator form the GAN trained using the WGAN loss (13) (11), denoted as L_W. Finally, as mentioned before, both Decoder and Encoder-2 consist of R-AE, which is trained with Mean Squared Error (MSE) loss, denoted as L_{MSE}. It is worth noting that our proposed VAEGAN and the original reference paper (13) differ in both the architecture and loss function used for CE reconstruction. The model employed in TF-VAEGAN is called Semantic Embedding Decoder, which utilizes L1 reconstruction loss. This loss function subtracts the predicted value from the actual value, i.e., $\hat{\boldsymbol{a}}_{\boldsymbol{learn}} - \boldsymbol{a_{learn}}$, and then computes the L1 norm of the resulting values, followed by taking the average. In contrast, the loss function we utilize is MSE loss (4).

Next, we will provide an explanation of the proposed MSE loss (4). However, it is important to note that other utilized loss functions, including Binary Cross-Entropy, KL divergence, and WGAN loss, remain the same as in the original reference paper (13).

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (\hat{a}_{\text{learn}_i} - a_{\text{learn}_i})^2 \qquad (4)$$

Generally, in the field of industrial surface defect detection, anomaly detection models are expected to minimize the reconstruction error on normal images during training and induce a large reconstruction error on anomalies during the testing stage. However, in this domain, the similarity between classes is often high. SBAD not only

needs to differentiate between these similar seen classes but also needs to recognize unseen anomalies. This requirement makes it insufficient to rely solely on L1 reconstruction loss for effective classification. Therefore, we propose using MSE loss (4) to measure the quality of the reconstructed $\hat{a}_{learn}$. Previous studies (19) have demonstrated that MSE loss performs better than L1 reconstruction loss. Furthermore, the experimental results presented in subsequent chapters demonstrate that using MSE loss improves the overall classification accuracy compared to TF-VAEGAN, which uses L1 reconstruction loss.

3.1.3. Segmentation-Based Defect Detector

The Metal Surface Defect Detection and Classification System simulates possible scenarios in real-world production lines, where SBDD plays the role of real-time defect detector.

Segmentation-Based Defect Detector, abbreviated as SBDD, is a deep convolutional network model based on segmentation. The overall model can be roughly divided into a segmentation network and a decision network. The segmentation network focuses on detecting small surface defects in high-resolution images, aiming to locate the specific positions of defects and perform pixel-level segmentation. The decision network is responsible for determining whether there are defects in the image. Based on this architecture, SBDD can be regarded as a binary classifier whose task is to inform the system whether the current input image contains defects. A defect notification is issued if defects are detected, and the specific defect locations in the image are output.

The characteristics of SBDD are lightweight and cost-effective while maintaining high accuracy in real-time detection. Since the task of SBDD is relatively simple and has already achieved good results in the original papers (3) (5) (6), we have made minimal modifications to its source code, only integrating the input with SBAD. For more details on the model architecture of SBDD, please refer to the original papers (3) (5) (6).

3.2. DAGM_MIX and Class-Embedding

The performance of the Metal Surface Defect Detection and Classification System heavily relies on the accurate and effective classification of SBAD. SBAD needs to be able to classify not only the seen classes but also identify unseen classes that have not been encountered before. The two key factors that determine the effectiveness of SBAD's classification ability are the quality of the dataset and the class embedding. In this chapter, we will explain the DAGM_MIX dataset we proposed and its corresponding class embedding.

DAGM dataset

Before we introduce the DAGM_MIX dataset, we first need to introduce what the DAGM dataset is. The DAGM dataset (20) is a widely recognized benchmark dataset in the field of industrial surface defect detection. The dataset comprises ten classes generated by computers, but similar to real-world problems, each with a different surface texture and containing grayscale image data for both benign(non-defective) and defective samples, such as crazing, scratches, or spots. At first, only six classes were made publicly available and known as the development dataset, while four more were introduced later, known as the competition dataset. As a result, some related methods report results only on the first six classes, while others present results on all ten. Each development (competition) dataset consists of 1000 (2000) 'benign' and 150 (300) 'defective' images saved in grayscale 8-bit PNG format.

DAGM_MIX dataset

As mentioned earlier, due to the system design we proposed, which requires SBAD to treat both benign and defect images within the same class, we have introduced the DAGM_MIX dataset based on the DAGM dataset.

The DAGM_MIX dataset is categorized into ten classes based on surface texture (background). Each class in the dataset contains both benign and defective images, as shown in **Figure 5**.

The DAGM_MIX dataset provides three different types of data annotations: surface-based labels, image-level labels, and pixel-level labels. The surface-based labels classify the images based on surface texture and indicate which class the image belongs to. The image-level labels can be considered as weak labels, informing the model whether the image is "benign" or "defect". On the other hand, pixel-level labels can be seen as strong labels, providing the specific spatial distribution of the background and defect areas in the image. In the pixel-level labels, the values 0 and 255 denote the background and defective area, respectively. The relationship between surface-based, image-level, and pixel-level labels is illustrated in **Figure 5**.

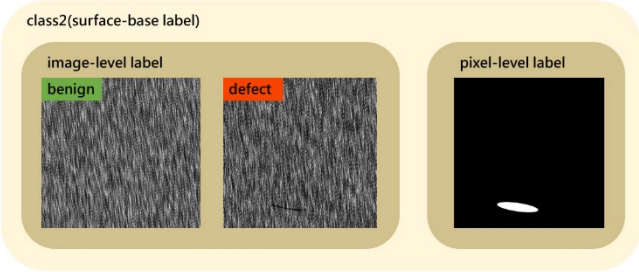

Figure 5. The different level label used in DAGM_MIX dataset

On the other hand, due to the large difference in the amount of benign and defective data, we have also addressed the issue of data imbalance. In addition to downsampling benign images, we have also performed data augmentation on defect images of various types, including rotation, horizontal flip, vertical flip, brightness adjustment, etc.

Class-Embedding

We propose a Class Embedding designed explicitly for the DAGM_MIX dataset based on its characteristics and the relevant expertise of the laboratory in the production and manufacturing field. The Class Embedding consists of 29 dimensions, with each dimension representing a specific attribute, as shown in Table 1.

Table 1. Class-Embedding of DAGM_MIX dataset

Attribute number	Define attributes
1	Color_D: white
2	Color_D : black
3	Color_BG: white_b
4	Color_BG: black_b
5	Surface texture: grainy
6	Surface texture: smooth
7	Surface texture: regular pattern
8	Surface texture: irregular pattern
9	Surface texture: strip
10	Defect scope area: large
11	Defect scope area: medium
12	Defect scope area: small
13	Defect shape: irregular slender strip
14	Defect shape: regular straight slender strip
15	Defect shape: irregular long strip
16	Defect shape: irregular small bump
17	Defect shape: irregular sparse dots
18	Defect shape: small dot
19	Defect shape: irregular ellipse stain
20	Defect distribution: concentration
21	Defect distribution: continuity
22	Defect reason: reason 1
23	Defect reason: reason 2
24	Defect reason: reason 3
25	Defect reason: reason 4
26	Defect reason: reason 5
27	Defect reason: reason 6
28	Defect reason: reason 7
29	Defect reason: reason 8

4. System Design and Methodology

This chapter will describe our experimental results in detail. First, we will explain the datasets we use and our experiment's environment in section 4.1. Next, we will demonstrate several experiments in section 4.2 to evaluate the performance of SBAD and SBDD models separately.

4.1. Datasets and Environment

DAGM_MIX Datasets

In the dataset processing part, we first divided the original ten categories of the DAGM dataset into eight seen classes and two unseen classes. The so-called unseen classes refer to the fact that the model will not use any data from that category during the entire training stage, including image data and the CE of that category. Next, we separated the benign and defect in each of the seen classes and downsized the benign images to 300 per class. Then, we performed data augmentation for the class with only 150 defect images, which is the competition dataset mentioned earlier. After data augmentation, each class increased from the original 150 to 300. At this point, the ratio of benign to defect data in all eight seen classes is 1:1, with a total of 600 data per class and a total of 4800 data. Then, we divided these data into training and validation sets at a ratio of 8:2. As for the unseen class part, we directly downsized the benign images in each category to be consistent with the number of defect images in that category and used them all as test data. Finally, we map the CE to their corresponding classes. Each CE is a 29-dimensional vector, where the semantics of each dimension are shown in Table 1. The experiment environment is shown in Table 2.

Environment

We use the local server built in our lab with Nvidia GeForce RTX3090 GPU to train our models. Following is the detailed experiment environment. In the next section, we will present experimental results to demonstrate our work.

Table 2. Experiment environment

	Setting
OS	Ubuntu 20.04
Platform	Pytorch 1.7.1 / CUDA 11.0
Program language	Python 3.7
CPU	AMD Ryzen 9 5950X 16-Core Processor
GPU	NVIDIA GeForce RTX 3090
memory	32G

4.2. Experimental Results

In this section, we validate the proposed Metal Surface Defect Detection and Classification System for industrial quality control and anomaly detection.

This system consists of two main models, the SBAD model, responsible for the surface material classification and identifying unseen anomalies, and the SBDD model, responsible for detecting whether defects appear in the image and drawing the specific location of the defects. Next, we will verify the performance of these two models separately.

Results on the SBAD Model

In the experimentation of the SBAD model, firstly, In order to verify the effectiveness of the learned CE, we compared the accuracy of using the Real CE proposed in this paper (Table 1) and the Learned CE generated by the VAE model while adopting the Inductive setting standard on TF-VAEGAN model. Furthermore, in order to verify the performance of our proposed VAEGAN, we trained TF-VAEGAN (13) using the DAGM_MIX dataset and Learned CE configuration and compared it with our proposed VAEGAN.

Table 3. Accuracy using TF-VAEGAN

Class-Embedding	GZSL Normal (seen classes)	GZSL Anomaly (unseen classes)	GZSL Harmonic mean(H)	ZSL Anomaly (unseen classes)
Real CE	**96.56**	65.76	78.24	100
Learned CE	96.15	**93.94**	**95.03**	100

Table 4. Accuracy of SBAD using proposed VAEGAN

Class-Embedding	GZSL Normal (seen classes)	GZSL Anomaly (unseen classes)	GZSL Harmonic mean(H)	ZSL Anomaly (unseen classes)
Real CE	96.46	70.61	81.53	100
Learned CE	**98.12**	**95.96**	**97.03**	100

As shown in Table 3 and Table 4, in the evaluation of SBAD, it can be observed that our model achieved perfect classification in all four different experimental setups in ZSL. When comparing VAEGAN and TF-VAEGAN in the case of GZSL, we noticed that using Real CE resulted in a slight decrease of 0.1% in the classification accuracy of seen classes. However, the accuracy of unseen classes improved by almost 5%. We believe this improvement is due to the higher quality of CE reconstruction in Encoder-2.

In addition, regardless of whether VAEGAN or the original TF-VAEGAN was used, we found that using Learned CE can significantly improve the accuracy of recognizing unseen classes than using Real CE. This indicates that the Learned CE obtained from our VAE model can provide more precise semantic feature values compared to manually defined Real CE.

It can be observed that using our proposed **VAEGAN and training with Learned CE can achieve the best performance.** The accuracy of harmonic reach 97%.

Table 5. Accuracy comparison between TF-VAEGAN and proposed VAEGAN

Models	GZSL Normal (seen classes)	GZSL Anomaly (unseen classes)	GZSL Harmonic mean(H)	ZSL Anomaly (unseen classes)
TF-VAEGAN	96.15	93.94	95.03	100
VAEGAN (ours)	**98.12**	**95.96**	**97.03**	100

From Table 5, as mentioned before, we can observe that using our proposed VAEGAN model architecture as the SBAD classifier, the classification accuracy of both seen and unseen classes has been slightly improved compared to training directly with TF-VAEGAN. We believe this improvement can be attributed to the use of the R-AE architecture combined with MSE Loss, which places more power on the quality of CE reconstruction.

Results on the SBDD Model

In the experiments conducted on the SBDD model, we compared four different datasets: DAGM, KSDD, KSDD2, and STEEL.

Table 6. Evaluation of SBDD on four datasets

	DAGM	KSDD	KSDD2	STEEL
AUC	0.99	0.99	0.97	0.99
AP	0.99	0.99	0.94	0.99
f-measure	0.99	0.99	0.91	0.96
FP	0	0	2	35
FN	0	0	17	9

As shown in Table 6. It can be observed that, except for the KSDD2 and STEEL datasets, the SBDD model demonstrated no False Positive (FP) or False Negative (FN) values, indicating that the model was able to distinguish whether defects were present in the input images perfectly. The data results of this experiment were also consistent with the data provided in the original paper (5).

5. Conclusions

Regarding the last chapter of this thesis, we will discuss the results of the experiment of our model and make conclusions in Section 5.1.

5.1. Conclusions

With the development of artificial intelligence, big data, the Internet of Things, and intelligent manufacturing, the digitalization and intelligence transformation of manufacturing factories have become an inevitable trend. Among them, the use of deep learning to address the problem of industrial surface defect detection has undoubtedly demonstrated remarkable effectiveness. It is currently a topic of much research and development.

This paper references many studies that adopt different deep learning methods and finds some common goals in these research efforts, namely:

1. High accuracy at an industrial level of standard, with the current goal of approaching zero defects.
2. Difficulty in obtaining defect data and the cost of labeling, which requires training methods that do not rely on large amounts of labeled data.
3. Lightweight models that can be deployed on edge devices with architectures that are not overly complex.
4. In a mass production environment, low latency of the detection system is usually required.
5. Facing the diversity of production and manufacturing processes, the model often needs to be able to recognize unseen anomalies.

To address the issues mentioned above, we found that most research focuses on developing a single general deep learning model to solve most of the problems in the field or developing a single deep learning model to solve a specific sub-problem. In contrast, this paper takes a different approach to the industrial surface defect detection problem, starting from the perspective of multiple models. We incorporate and combine three different deep learning models and adjust them according to the dataset's characteristics, proposing a Metal Surface Defect Detection and Classification System.

Our proposed VAE model has demonstrated that using learned CE to train a ZSL classifier performs better than using manually defined CE and better reflects the features of the classes. Next, through experiments, we found that using our proposed DAGM_MIX dataset and Learned CE could achieve the best performance in the SBAD classification task and significantly increase the ability of SBAD to identify unseen anomalies.

Finally, we also compared the performance of TF-VAEGAN and our proposed VAEGAN under the optimal configuration mentioned earlier. The experiment in Table 4 showed that using our proposed VAEGAN model, the accuracy of both seen and unseen classes has increased slightly.

In addition, we briefly introduce various deep learning methods currently used to solve industrial surface defect detection problems, including supervised and unsupervised learning techniques, and analyze their advantages and disadvantages in the Introduction and Related Work sections.

References

1. Cohn R, Holm E. Unsupervised machine learning via transfer learning and k-means clustering to classify materials image data. Integrating Materials and Manufacturing Innovation. 2021;10(2):231-44.
2. Vahdat A, Kautz J. NVAE: A deep hierarchical variational autoencoder. Advances in neural information processing systems. 2020;33:19667-79.
3. Božič J, Tabernik D, Skočaj D, editors. End-to-end training of a two-stage neural network for defect detection. 2020 25th International Conference on Pattern Recognition (ICPR); 2021: IEEE.
4. Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey. ACM computing surveys (CSUR). 2009;41(3):1-58.
5. Božič J, Tabernik D, Skočaj D. Mixed supervision for surface-defect detection: From weakly to fully supervised learning. Computers in Industry. 2021;129:103459.
6. Tabernik D, Šela S, Skvarč J, Skočaj D. Segmentation-based deep-learning approach for surface-defect detection. Journal of Intelligent Manufacturing. 2020;31(3):759-76.
7. Li J, Su Z, Geng J, Yin Y. Real-time detection of steel strip surface defects based on improved yolo detection network. IFAC-PapersOnLine. 2018;51(21):76-81.
8. Guo Z, Wang C, Yang G, Huang Z, Li G. Msft-yolo: Improved yolov5 based on transformer for detecting defects of steel surface. Sensors. 2022;22(9):3467.
9. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. Advances in neural information processing systems. 2014;27.
10. Xian Y, Lorenz T, Schiele B, Akata Z, editors. Feature generating networks for zero-shot learning. Proceedings of the IEEE conference on computer vision and pattern recognition; 2018.
11. Arjovsky M, Chintala S, Bottou L, editors. Wasserstein generative adversarial networks. International conference on machine learning; 2017: PMLR.
12. Gao Y, Gao L, Li X. A generative adversarial network based deep learning method for low-quality defect image reconstruction and recognition. IEEE Transactions on Industrial Informatics. 2020;17(5):3231-40.
13. Narayan S, Gupta A, Khan FS, Snoek CG, Shao L, editors. Latent embedding feedback and discriminative features for zero-shot classification. European Conference on Computer Vision; 2020: Springer.
14. Zhao Z, Li B, Dong R, Zhao P, editors. A surface defect detection method based on positive samples. PRICAI 2018: Trends in Artificial Intelligence: 15th Pacific Rim International Conference on Artificial Intelligence, Nanjing, China, August 28–31, 2018, Proceedings, Part II 15; 2018: Springer.
15. Wu H, Lv Q. Hot-rolled steel strip surface inspection based on transfer learning model. Journal of Sensors. 2021;2021:1-8.
16. Pan C-C. Evolution of Class Embedding for Unseen Class in Zero-Shot Learning 2021.
17. Xian Y, Lampert CH, Schiele B, Akata Z. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. IEEE transactions on pattern analysis and machine intelligence. 2018;41(9):2251-65.
18. Catherine Wah, Steve Branson, Peter Welinder, Pietro Per-ona, and Serge Belongie. The caltech-ucsd birds-200-2011dataset. 2011.
19. Liu J, Song K, Feng M, Yan Y, Tu Z, Zhu L. Semi-supervised anomaly detection with dual prototypes autoencoder for industrial surface inspection. Optics and Lasers in Engineering. 2021;136:106324.
20. Weimer D, Scholz-Reiter B, Shpitalni M. Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. CIRP annals. 2016;65(1):417-20.