Visual Inspection of 3D Printed Products

Jiapeng Li j.li3@student.tue.nl
Technische Universiteit Eindoven, PO Box 513 5600 MB Eindhoven, The Netherlands

Abstract

In the 3D printing industry, visual inspection of the defects is necessary to ensure the quality of the products. The goal of the thesis is to use deep learning to automate the visual inspection. However, the industrial data is typically unbalanced where only a few images for each defects type can be obtained, which makes traditional classifier hard to train. Therefore, in this project, two generative methods are proposed for anomaly detection, leveraging only images of good products. One is based on Generative Adversarial Network(GAN), and the other is based on Variational Autoencoder(VAE). We used the generative models to reconstruct images, and reveal the defects by comparing the original images and their reconstruction. Their distance is measured on both pixel level and feature level where the feature level distance is learned by Siamese Network. Visual results show that both GAN and VAE can reconstruct the good images well and not generate the defects. The classification results are improved by anomaly detection combined with Siamese Network, compared with the baseline, which is a modified pre-trained VGG16 classifier.

1. Introduction

In 3D printed products, defects normally take place on the contact area between the foundation of the printer and the surface of the products. It is important to detect those defects as soon as possible to save inspection and maintenance cost. Furthermore, it is also beneficial to classify those defects and capture their unique features. Those features can be applied for further use, such as improving the manufacturing process and the quality of the printers. However, the amount of defective products is so small that it is difficult to automatically distinguish those defects from the good products.

Nowadays two methods are widely adopted for visual inspection in the industrial field, namely statistical method (Huang Jiexian, 2010) and spectral method (Zhang Xuewu, 2011). The former uses the spatial distribution of pixels to describe the defects feature. In the latter method, the defects feature can be described by different filters such as Wavelet filters and Gabor filters. However, the data is often high dimensional and includes a lot of redundant information. And the defects display a huge diversity, which brings the bottleneck to both methods mentioned above.

Recently deep learning has gained great success in computer vision. It uses a bionic neural network to imitate how visual information is processed by the human brain. Convolutional Neural Network (CNN) (Alex Krizhevsky, 2012) is a typical example. It uses iterative convolution and pooling operations to extract features on different levels. What's more, the increasingly faster computational power enables the emergence of variant deep and complicated networks such as VGG (Karen Simonyan, 2014) and Inception model (Christian Szegedy, 2015).

However, deep networks contain a massive number of parameters which a large size of data is trained on. Although nowadays it is easy to obtain a large volume of data, manual labeling still demands high effort. In reality, the data is often not well labeled, in which semi-supervised and unsupervised learning play an important role to capture the intrinsic feature of the data.

1.1. Problem Statement

The goal of this thesis is to investigate algorithms for visual inspections that are capable to distinguish between proper surface and defective ones, using a combination of state-of-the-art supervised and semi-supervised image analysis methods.

Master Thesis, Department of Mathematics and Computer Science

The proposed methods are evaluated on the industrial dataset with the following properties:

Quantity The training data consists of 1784 samples, which is too little to train the parameters of any state-of-art neural networks. Another problem concerning the quantity of the dataset is the class imbalance. From the reality of manufacturing, we observe that defective products happen far less frequently than good products. This situation is also reflected in our dataset distribution. On the one hand, the proportion of good images is much larger than any type of defective images. On the other hand, images of different defect types are not evenly distributed.

Quality When we look into the dataset to observe the defects it is not easy to visually distinguish between good images and some defects. For example images with dirty spot defects usually have a blurred line between good images because the dirty spot is rare and small. The good images also have some flaws which we do not consider as defects. Our image size is 224x224. However, what usually happens is that the defect takes on a tiny part of the image. What's more, multiple types of defects can exist on the same image, which can lead binary classification to multi-label classification.

Research Points In this thesis, based on the problems mentioned above, we will investigate the following research points for our industrial dataset.

- We will investigate the impact of the data imbalance on the results of classification.
- We will compared the classification performance of supervised learning and semi-supervised learning on the customized dataset.
- We want to learn the features of different types of defects and visualize them.

1.2. Contributions

Corresponding to the research points mentioned above, we present in this thesis the following contributions:

- We used a modified pre-trained VGG16 to investigate the impact of data imbalance on classification performance.
- We proposed a pipeline of anomaly detection that takes advantage of only good images, to avoid the problem of data imbalance.
- We compared the results of anomaly detection based on two generative models, namely VAE and GAN.
- We extracted the features of the images and visualized where the defects localize .

1.3. Evaluation Metrics

In this thesis, we use recall and precision on negative samples to evaluate the performance of each model. We consider good images as positive and defective images as negative. In the industrial field, to ensure the products quality, it is important to recall all the defective products. However, using recall as the only metric will lead the model to be biased to defective images, and the model can not learn the accurate representation of those defect. Therefore, we also adopt the precision on defective images as our metric. To balance between precision and recall we use the F1 measure as described in Equation 1.

$$F - measure = \frac{2 * precision * recall}{precision + recall} \tag{1}$$

2. Background

In this section, we introduce some previous works which are related to our thesis. The models that we adopted are divided into two categories, namely discriminative models (VGG16) and generative models (GAN & VAE).

2.1. Terminology

- P_r : Distribution of the training samples
- P_g : Distribution of the generated images learned by generator
- z : Input vector of generator in GAN and points from latent space in VAE
- $D(\cdot)$: Discriminator function
- $G(\cdot)$: Generator function
- $Enc(\cdot)$: Encoder function
- $Dec(\cdot)$: Decoder function
- $f(\cdot)$: The output of the last convolutional layer in the Discriminator
- x/X : Symbol for an image sample/ set of all images
- $\hat{y}/y$: Predicted / target label of model input
- $A(\cdot)$: Anomaly score
- α : Learning rate of gradient descent
- λ : Proportion weight of feature-wise loss
- I : Intensity of an image
- m, n : Length and width of the image

- e : The length of the embedding from the output of $f(\cdot)$

2.2. VGG16

In 2014 Oxford University has proposed VGG network (Karen Simonyan, 2014) in order to improve the performance of convolutional neural network using deeper network . It uses multiple small filters of size 3 to replace a larger filter of size 7. Therefore, the model has deeper layers and wider feature maps. On the one hand, using a small-size filter significantly reduces the calculation of convolutions. On the other hand, richer feature maps enable the model to have a better discernibility.

As the neural network becomes deeper and more complicated, it is difficult to interpret the model and reveal the internal process. (Wei Yu, 2016) has used deconvolutional layers to visualize the intermediate representations in VGG network. As is shown in this paper, the features extracted by VGG are simple in the initial layers, and can be shared among different datasets. Therefore, in the reality when the customized dataset is small, we can transfer(Sinno Jialin Pan, 2009) the feature extracted by a pre-trained model which has been trained on millions of data to our own dataset.

Global Average Pooling Originally in CNN, the fully connected layer contributes significantly to the number of training parameters that the model is likely to overfit. Global Average Pooling (Min Lin, 2013) was proposed to reduce the training parameters, as illustrated in Figure 1. Besides, the feature extracted can be easily interpreted since the feature vector is directly connected to where the classification decision is made.

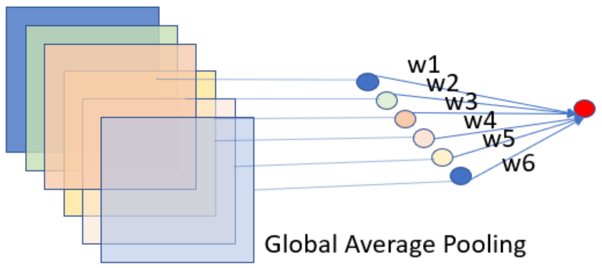


Figure 1. Global Average Pooling

2.3. Generative Model

The supervised learning, as mentioned in VGG16, is to predict labels given data points. However, in the reality, the given data is not usually well labeled. Therefore, we have to learn the underlying hidden structure of the dataset without labels. In unsupervised learning, the generative model attempts to generate new samples ($\hat{x} \sim P_{model}$) which come from the same distribution as the training data ($x \sim P_{data}$). That is, we can say P_{model} is similar to P_{data} based on the observable samples $\hat{x}$ and x. In the following, two models are introduced to determine whether $\hat{x}$ is similar to x or not.

2.3.1. Generative Adversarial Networks (GANs)

Generative adversarial networks (Goodfellow & Bengio, 2014) is a generative model which is made up of a generator G and a discriminator D. G maps the latent space to actual data space while D assigns the input data with the probability between $[0, 1]$. D gives higher value if the input is from real distribution P_r and in the contrast gives lower value if the input is from generator P_g. In other words, G tries to generate images which can fool D while D tries to tell the difference between real images and generated images. Therefore GAN's objective is to find the binary classifier that gives the best possible discrimination between true and generated data and simultaneously encouraging G to fit the true data distribution. The dynamics between the generator and the discriminator are mathematically formulated in:

$$\begin{aligned} \underset{\theta_g}{argmin}\ \underset{\theta_d}{argmax}\ & E_{x\sim P_r}(log(D(x)))+ \\ & E_{z\sim P_g}(log(1-D(G(z)))) \end{aligned} \tag{2}$$

The first term is the assigned probability to the real data and the second term is the assigned probability to the generated data. The *minmax* in the loss function reflects the adversarial training between the generator and the discriminator. Ever since the GAN came out many GAN-based variants have been proposed such as conditional GAN(Mehdi Mirza, 2014), CycleGAN(Jun-Yan Zhu, 2017) etc., among which deep convolutional GAN (DCGAN)(Alec Radford, 2016) is relevant to our project. DCGAN is a simple GAN built up with convolutional layers so that large dimensional data like images can be generated. DCGAN also provides some tricks for training the GAN since vanilla GAN's performance suffers from instability and mode collapse. It uses strided convolutional layer and deconvolution layer instead of pooling layer. The DCGAN also uses Batch Normalization(Sergey Ioffe, 2015) to prevent the problem of vanishing gradients.

Speaking of the application of GAN, closely related to our thesis, GAN has already been applied into anomaly detection(Thomas Schlegl, 2017), which is referred as Ano-GAN. It defines the *Anomaly Score* to measure the distance between good images and defective images, because the larger the distance is, the more defective the image is. The anomaly score consists of two parts: pixel-wise loss and feature-wise loss.

$$L_{pixel}(x,z) = \sum_{i=0}^{m}\sum_{j=0}^{n} |x - G(z)| \tag{3}$$

$$L_{feature}(x,z) = \sum_{i=0}^{e} |f(x) - f(G(z))| \tag{4}$$

$$A(x,z) = \lambda * L_{feature}(x,z) + L_{pixel}(x,z) \tag{5}$$

Equation 3 sums up the difference over all the pixels, where m and n stand for the length and width of the image. Equation 4 sums up the difference over all the embedding, where e stands for the length of flattened embedding. The function of λ is to control the value of feature-wise loss because the pixel-wise loss is the major difference. The pixel-wise loss measures the visual dissmilarity between real images and generated images. The idea of using feature-wise loss is from (Salimans, 2016). It used a technique called feature-matching which takes advantage of a richer intermediate feature representation from the discriminator, forcing the generator to generate realistic images.

2.3.2. Variational Autoencoder (VAE)

In generative model, we can only observe the samples, $x \sim P_{data}$ and $\hat{x} \sim P_{model}$, to measure the distance between P_{model} and P_{data}. GAN directly uses neural network to learn the distance between x and $\hat{x}$ without defining P_{model}. However, VAE(Diederik P Kingma, 2013) uses an explicit density with latent z to describe the likelihood of the training data $p(x)$.

$$p(x) = \int_z p(x|z)p(z)dz \tag{6}$$

Equation 6 explains the motivation of VAE, where $p(x|z)$ describes a model that uses z to generate x. We can set $p(z)$ to be simple such as $N(0,1)$. However, the conditional probability $p(x|z)$ is too complex that it should be represented by a neural network, which is called decoder. The posterior distribution, as is described in Equation 7, is intractable because $p(x)$ is unknown. Therefore, an encoder $q(z|x)$ (Equation 8) is used to approximate the real posterior distribution.

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} \tag{7}$$

Therefore, the VAE consists of two probabilistic networks:

$$z \sim Enc(x) = q(z|x), \tilde{x} \sim Dec(z) = p(x|z) \tag{8}$$

The $Enc(\cdot)$ is the encoder function that encodes a data sample x to the latent representation z. The $Dec(\cdot)$ is the decoder function that decodes the latent representation back to data space. The main insight of VAE is that it can be trained by maximizing the variational lower bound, which is Equation 9.

$$\begin{aligned} L(\theta,\phi,x) = -KL(q_\phi(z|x) \parallel p_\theta(z)) + \\ E_{q_\phi(z|x)}[\log(p_\theta(x|z))] \end{aligned} \tag{9}$$

θ is the parameters of decoder and ϕ is the parameters of encoder.The first term is the KL divergence that measures the distance between latent space and the prior distribution. The second term is reconstruction loss, which maximize the likelihood of original input being reconstructed.

When it comes to the decoder $p(x|z)$, which is a probabilistic generator, we can construct distribution using Bernoulli or Gaussian distribution. In our thesis, since our images are in one-channel gray scale, we choose Bernoulli distribution.

$$p(x|z) = \prod_{i=0,j=0}^{m,n} \rho(i,j)^{x(i,j)} \cdot (1-\rho(i,j))^{(1-x(i,j))} \tag{10}$$

Equation 10 describes the generated probability of Bernoulli distribution, where $\rho(i,j)$ is the probability of pixel at index (i,j). We can use logarithm to explain the reconstruction loss of a single image.

$$\begin{aligned} \log(p(x|z)) = \sum_{i=0,j=0}^{m,n} [x(i,j)] \cdot \log(\rho(i,j)) \\ +(1-x(i,j)) \cdot \log(1-\rho(i,j))] \end{aligned} \tag{11}$$

As Equation 11 shows, the reconstruction loss is in the same form as cross-entropy. Therefore, we need to use sigmoid at the end of the model to make sure the output is in $(0,1)$, and use cross-entropy as loss function for the second term of Equation 9. The reconstruction loss of the whole image can be calculated by summing up the reconstruction loss over all the pixels, where m,n is the index of pixels within the image.

2.4. Siamese Network

One extreme task of solving data imbalance is one-shot learning, where in the dataset, only one image is different from others. In this case, even pre-trained model will be biased to the majority class. However, Siamese Network (Bromley & Shah, 1993) is an efficient model to identify the image by pairing different images as input, and assigning the pair either *similar* or *different*. Symmetrical Siamese Network uses two shared-weights network with two inputs and extracts their features for comparison.

An important property of Siamese Network is its ability to measure the similarity of two input images. Firstly, the Siamese Network projects two images from a pixel level to a feature level by extracting their representations. Secondly, Contrastive loss such as Euclidean distance or Cosine similarity are used to calculate the distance between two embeddings. In this thesis we use the Siamese Network to calculate the distance between two images on feature level.

3. Methodology

In this section, we propose a pipeline of one class anomaly detection, to distinguish defective images from the good ones. The pipeline consists of two steps. The first step is to reconstruct an input image, using generative models that are trained on only good images. The second step is to classify the good and defective images based on the distance between themselves and their reconstructions. Since the generative models are trained on only good images, the model can not recognize defective images well. Therefore, the distance between good images and their reconstructions is smaller than the one between defective images and their reconstructions. We will first describe the motivation of one class anomaly detection with the choice of two generative models, and then two types of metrics are proposed that we can use to measure the distance between the input images and their reconstructions.

3.1. One Class Anomaly Detection

Anomaly detection means recognizing the data points which are very different from others. From Table 1 we can see that the dataset is highly imbalanced, where good images appear more frequently than other types of defective images. Therefore, we can train a model to learn the features and representations that only belong to the good images, and ones that can not be recognized by the model is considered as anomalies. From a probabilistic perspective, we want to learn the distribution $p(x)$ that can describe all the good images, and any other image that is out of $p(x)$ is an anomaly. One class anomaly detection takes advantage of only good images, and train the model that is able to fit their distribution.

To capture the distribution of good images, we adopted generative models. Because the generative models are able to map a simple distribution, e.g.$N(0, 1)$, to the distribution of training data. In this thesis, we adopted two typical generative models, which are GAN and VAE.

3.2. General Pipeline

Figure 2 describes the general pipeline of one class anomaly detection, which is made up of three components. The first one is *Reconstructor*, which is already trained on good images before the anomaly detection. The reconstructor takes in a real image and tries to reconstruct the image with minimal loss. In the following sections, we use GAN and VAE to perform as the reconstructor respectively and then compare their results.

The second component, the triangle in Figure 2, measures the distance between the input image and its reconstruction. In the following sections, we measure the distance in two levels respectively, namely pixel level and feature level. In the pixel level, GAN uses anomaly score (Thomas Schlegl, 2017) and VAE uses reconstruction loss to measure the distance between the input image and its reconstruction. In feature level measurement, we use Siamese Network to learn the distance between the input image and its reconstruction based on their extracted features.

The third component is the threshold based on the output of distance measurement. Since the model is trained on only good images, the good images are better reconstructed, which means the output of distance measurement (Figure 2) is smaller. From the probabilistic perspective, the distance for good images is more likely to be small, and for defective images is more likely to be large. We define the distance as the threshold, where the probabilities of the calculated distance belonging to good images and defective images are equal. The threshold functions as a classifier to give *good* or *defective* label to the input image. The distance is calculated on both good and defective images in the training set and then evaluated on the test set.

3.2.1. VAE based Anomaly Detection

The idea of VAE based anomaly detection is that the decoder of the VAE learns how to reconstruct the image from an approximate posterior distribution, which is learned by the encoder. The latent representation captures the distribution of all the good images in the training set. When feeding a defective image, the latent representation deviates from the prior distribution, therefore, the decoder fails to reconstruct the de-

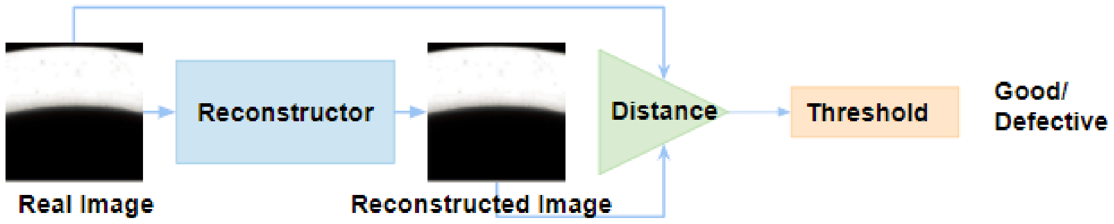


Figure 2. General pipeline of one class anomaly detection. The reconstructor is already trained on good images

fects.

Pipeline In the pipeline of VAE based anomaly detection, as is shown in Figure 3, the VAE functions as the reconstructor. The input image goes through the encoder to the latent space, which is represented by the mean μ and the variance σ. Then the decoder reconstructs the image from the latent representation. The distance measurement in Figure 3 is based on Equation 11, which consists of the cross-entropy between the input image and its reconstruction, and the KL divergence in the latent space as well. Therefore, apart from the inputs of the real image and its reconstruction, another input of KL divergence in the latent space is added to the reconstruction loss.

Image Reconstruction In Equation 11, the posterior distribution $p_\phi(z|x)$ is intractable that we can use Monte Carlo sampling (Haugh, 2004) to approximate the expectation. The Monte Carlo sampling states that the more samples the model uses, the more likely the model will reach the optimum solution. Therefore, during the training process, we train the model with large batch size and multiple epochs, thus we only need to sample from the latent distribution once for each image batch. However, in the testing phase when we only input a single query image, we need to sample multiple times from the latent space. Because in Equation 11, we need to calculate the integral of complex continues function, thus we normally use Monte Carlo sampling to approximate the integral. The detailed process of image reconstruction is described in Algorithm 1.

After we calculate the reconstruction loss for each image in the training dataset, as is shown in Table 1, we build up a histogram of reconstruction loss for good images and defective images respectively. In the histogram, we find the threshold where the frequencies of reconstruction loss are equal for good images and defective images. Then we apply this threshold on the test set for evaluation.

Algorithm 1 VAE Based Anomaly Detection

```
Input: data $x_{1..n}$, Trained VAE $Dec$,$Enc$
for i=1 to n do
  $\mu_{x^i}, \sigma_{x^i} \leftarrow Enc(z^i|x^i)$.
  $L_{KL} = KL(Enc(z^i|x^i)||N(0,1))$
  draw samples from prior $z_{1...L} \sim N(0,1)$
  declare $z_{min}$ latent representation with least loss
  for j=1 to L do
    $\hat{z^j} \leftarrow \mu_{x^i} + z^j * \sigma_{x^i}$ Reparameterization
    $L_{recon} += logDec(x^i|\hat{z^j})$
    if $logDec(x^i|\hat{z^j}) < logDec(x^i|\hat{z_{min}})$ then
      $z_{min} \leftarrow \hat{z^j}$
    end if
  end for
  $L_{x_i} = L_{KL} + \frac{1}{L} * L_{recon}$
  Output: $L_{x^i}$, $Dec(x^i|z_{min})$
end for
```

3.2.2. GAN based Anomaly Detection

Via adversarial training, the generator learns the representations of good images, with the ability to generate various good images. We can compare the real images and the generated image to identify the defects region.

Pipeline: The pipeline of GAN based anomaly detection is shown in Figure 4, where the reconstructor and the distance measurement are radically modified. Firstly, only the generator of the GAN performs as the reconstructor. Secondly, we adopt the anomaly score in Equation 5, which consists of pixel-wise loss and feature-wise loss, to measure the distance between the input image and its reconstruction. The feature-wise loss, as is shown in Equation 4, is based on the output of the last convolutional layer in discriminator. Last but not least, the data flow in the pipeline is not uniform, where we need to backpropagate the anomaly score from the current iteration to update the latent representation z. Because the GAN does not have an inference procedure, so we have to iteratively return to the latent space.

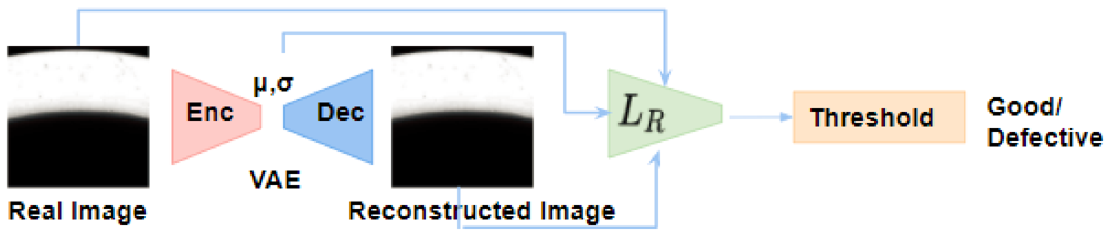


Figure 3. Pipeline of VAE based anomaly detection. The VAE is already trained on good images. *Enc* is the encoder and *Dec* is the decoder of VAE. L_R means the reconstruction loss that is used to measure the distance between input image and its reconstruction.

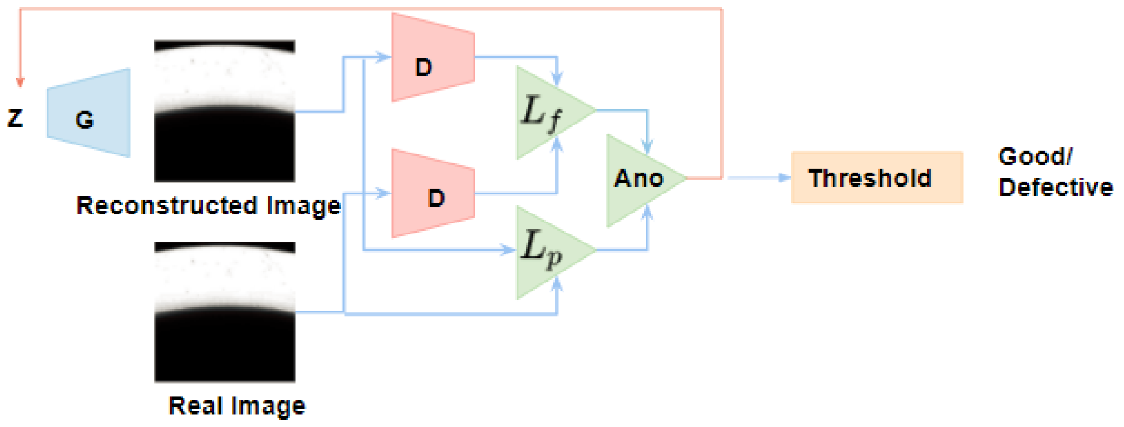


Figure 4. Pipeline of GAN based anomaly detection. The GAN is already trained on good images. D represents the discriminator and G represents the generator. L_f measures feature distance based on the output of discriminator. L_p means the pixel-wise distance between the input image and its reconstruction. ANO is the anomaly score that combines the pixel-wise loss and feature-wise loss. The red arrow implies that the latent representation z is updated based on the anomaly score from previous iteration.

Iterative searching within generated manifold The goal of iteratively returning to the latent space is to find the reconstructed image that is closest to the query image. Since the GAN is trained on only good images, it learns the representations and underlying structures of only good images. The generator of GAN can generate a manifold that contains all the good images, as is shown in Figure 5. We need to find the closest image within the manifold in order to reconstruct the query image. We use anomaly score defined in Equation 5 to measure the distance between the image from the generator and the query image. If the query is a good image, the closest image within the manifold is itself. Otherwise, for a defective image, the closest image should be one that produces minimal anomaly score in the pipeline (Figure 4). A random point in the latent space is unlikely to generate the closest image for a certain query image. Therefore, we can iteratively search the next closet image based on the previous anomaly score by updating the latent representation:

$$z^{k+1} = z^k - \alpha * \frac{d\left(L_{pixel} + \lambda * L_{feature}\right)}{dz^k} \tag{12}$$

When the z converges, that is to say when the anomaly score produced by z does not reduce any more, the closest image is reached and we can obtain a residual image by pixel-wise subtraction between the real image and generated image. The detailed process of reconstructing images based on GAN is described in Algorithm 3.

After we calculate the anomaly score for each image in the training dataset, as shown in Table 1, we find the threshold between good images and defective images in the same way of VAE based anomaly detection. Then we evaluate this threshold on test set.

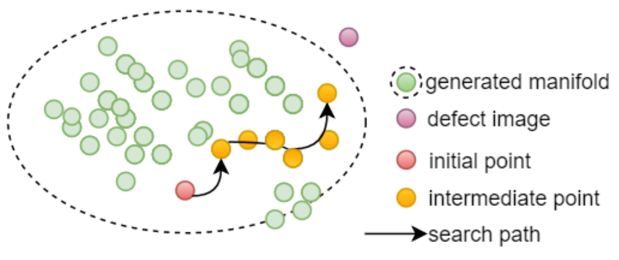


Figure 5. The dotted ellipse demonstrates a generated manifold that contains all the possible good images. The defects is outside this scope, the closest image should be near the edge. Each iteration depends on the loss and learning rate as well as the optimizer's momentum.

Algorithm 2 GAN Based Anomaly Detection

```
Input: data $x_i$, Trained GAN $G$,$f$ from $D$
Random initialization in latent space $z \sim N(0,1)$.
Initialize closest image $img_{closest} = zeros$
counter $k = 0$
repeat
    $L_{feature} = \sum |f(x) - f(G(z_k))|$
    $L_{pixel} = \sum |x - G(z_k)|$
    $gradient = \frac{\sigma(L_{pixel} + \gamma * L_{feature})}{\sigma z_k}$
    update $mean$ and $variance$ of $Adam$ optimizer
    $img_{closest} = G(zk)$
    $z_{k+1} = z_k - Adam(learning\_rate, gradient)$
until $z$ converge
Output:
$img_{closest}$
$img_{residual} = |img_{closest} - x_i|$
$anomaly\_score = L_{pixel}(x_i, img_{closest}) + \gamma * L_{feature}(x_i, img_{closest})$
```

3.3. Distance on Feature Level

So far GAN and VAE based anomaly detection use different approaches to reconstruct images. For each image, the distance measurement in Figure 2 outputs a score, which is the distance between the image and its reconstruction. Then we find the threshold between good images and defective images based on the score. However, both GAN and VAE measure the distance between the query image and its reconstruction on the pixel level. Anomaly score (Equation 5) in GAN focuses on $L1$ distance over all the pixels, and reconstruction loss (Equation 11) in VAE sums up cross-entropy of all the pixels. As is mentioned in the general pipeline of anomaly detection, we not only measure the distance on the pixel level but also on the feature level.

Pipeline: Figure 6 and Figure 7 describe the pipeline of VAE and GAN based anomaly detection with distance measurement on feature level. The features of the query image and its reconstruction are extracted by Siamese Network, which consists of two shared-weights VGG16. We use global average pooling to generate $1-D$ vector from the feature map of the last convolutional layer in modified VGG16. The sigmoid assigns the probability of whether the input images are similar or not. The Siamese Network, together with global average pooling and sigmoid, function as the distance measurement in the general pipeline (Figure 2). The output from sigmoid for good images is higher than the one for defective images, which means that the probability of a good image being similar to its reconstruction is higher than a defective image. Since the output of sigmoid is between $[0,1]$, the threshold here is 0.5.

Distance Learned by Siamese Network Unlike distance on pixel level such as anomaly score (Equation 5) and reconstruction loss (Equation 11), the distance on feature level does not have a explicit expression. The feature distance is learned by the Siamese Network with the following loss function.

$$L(x, x_{rec}) = -\frac{1}{n}\sum_{i=1}^{n}[\hat{y} * ln(S(x_i, x_{rec})) + (1-\hat{y}) * ln(1 - S(x_i, x_{rec}))] \tag{13}$$

x is the query image and x_{rec} is its reconstructed image. $S(\cdot)$ represents the predicted output of Siamese Network, which is the probability between $[0,1]$. The target label $\hat{y}$ is 1 if x_i in X_{good} and 0 otherwise.

4. Experiment

4.1. Data Description & Preprocessing

In this section, the dataset built for this project is described. The images in the dataset are in the same shape, an arc-shaped ring, which represents a part of the product that needs to be inspected. Multiple types of defects can exist in a same slice with different locations, different forms as well as different levels.

Gamma Correction We capture the image by scanning the products in a black box. Due to the over-exposure of scanning, the displayed images will have a luminance bias compared with the real images. Gamma correction is a technique that can non-linearly modify the luminance of the image by enlarging or narrowing the contrast between darkness and brightness. It adds an power index *gamma* to all the pixel values. If $gamma < 0$ then the contrast is narrowed. Otherwise the contrast is enlarged. In this project we set the $gamma = 6$.

From Figure 8 we can see that the original image on

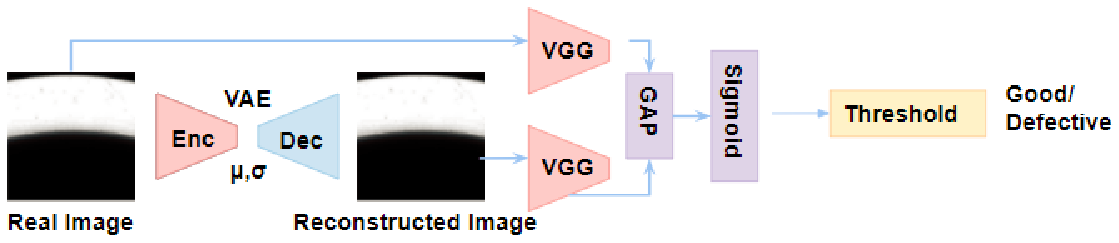


Figure 6. Pipeline of VAE based anomaly detection with distance measurement on feature level. The VAE is already trained on good images. *Enc* is the encoder and *Dec* is the decoder of VAE. VGG is convolutional layers in the modified pre-trained VGG16. GAP is global average pooling.

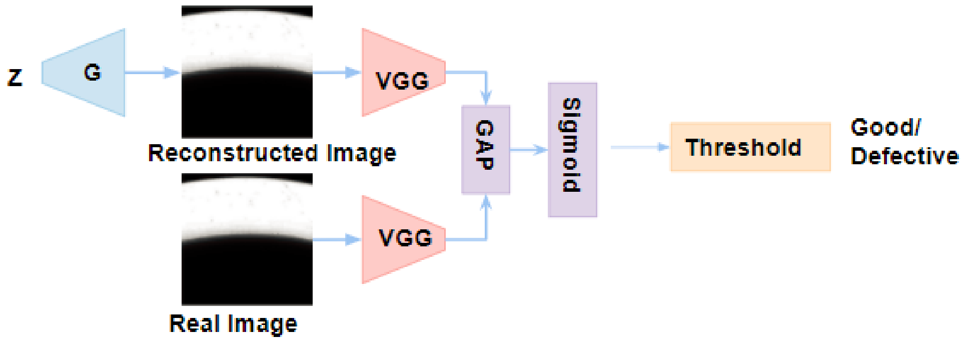
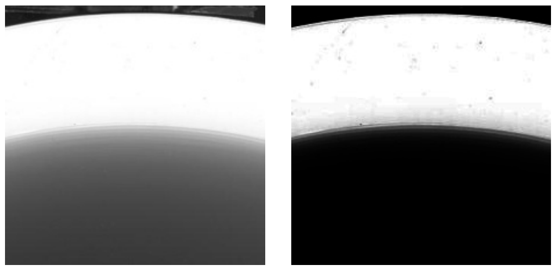


Figure 7. Pipeline of GAN based anomaly detection with distance measurement on feature level. The GAN is already trained on good images. *G* is the generator. VGG is the convolutional layers in modified pre-trained VGG16. GAP is global average pooling. In this figure, the iterative procedure back to the latent space is left out.



(a) Real image (b) Gamma correction

Figure 8. Comparison between original image and image after gamma correction. The example is a good image which has small imperfections

the left is too bright that some small stains are not explicitly revealed. The gradual change of the pixel value on the downside border blurs the edge of the slice, which affects the ability of the generator. After gamma correction, we can visually observe the unusual parts on the slice and a more obvious border.

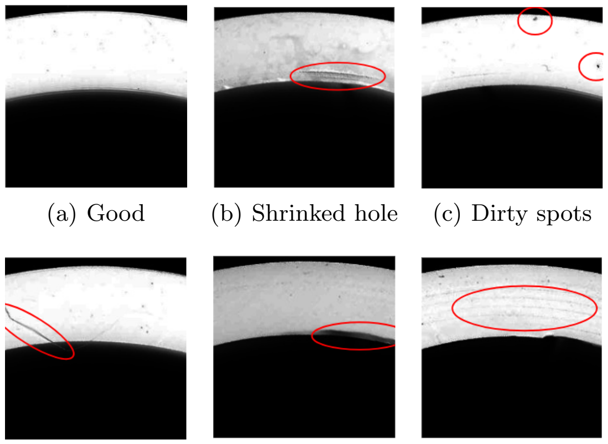
Dataset Categories The dataset is divided into good and defective groups. Within the defective group, the images are subdivided into 5 classes of defects. From the Figure 9 we can see that each defect has its own feature such as shape, position and defect level. However, the labelling procedure is subjective, for example it is hard to distinguish between good images and dirty spot defects since the good images are not perfectly clean. Furthermore, the dataset is highly imbalanced that defective images appear much less than the good images.

4.2. Supervised Learning for Data Imbalance

One of the major problems we target is data imbalance. Therefore we want to measure how the data imbalance influences the results. We set different imbal-

Image class	Sample quantity	Train	Test
Ok	619	495	124
Shrinked hole	138	117	21
Dirty spot	75	54	21
Discoloered line	24	17	7
Edge erosion	239	183	56
Printed line	190	154	36

Table 1. Image class and train-test split


(a) Good (b) Shrinked hole (c) Dirty spots

(d) Discolored line (e) Edge erosion (f) Printed lines

Figure 9. 6 categories of images

ance ratio for each defects class in the range of $[0, 0.5]$. Considering the small amount of the training data we did not use much larger ratio. To build up the training dataset for each defect ratio, we simply over-sample the defectives images or down-sample the good images from the original dataset.

Modified Pre-trained VGG16 To investigated the influence of imbalance ratio on the classification decision, we used a modified pre-trained VGG16 to distinguish between good images and images in a specific defect class. Thus we use a supervised learning as a baseline.

Architecture The original VGG16 has 5 convolutional blocks with the number of filters gradually increasing. The output of each convolution layer has different semantic meaning. As illustrated in related work, the deeper the layer is, the more complex the feature it represents.

The modification is implemented on top of the network where we discard all fully connected layers and the last 3 convolutional layers and the last max pooling layer. The size of the dataset we use is quite small, with 1184 images in total. Such amount of data will not cover millions of trainable parameters. Therefore, we used a pre-trained model which has been trained on the ImageNet dataset, and transfer the learned feature to our target domain. In this experiment, we only trained the last convolutional block and fix the rest layers for fine-tuning.

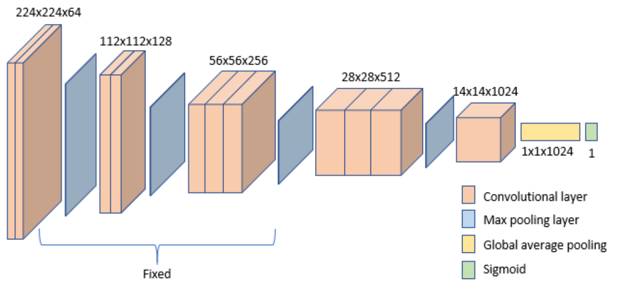


Figure 10. Detailed architecture of modified pre-trained vgg16. Activation layer is not included in the figure. The first 3 blocks are fixed

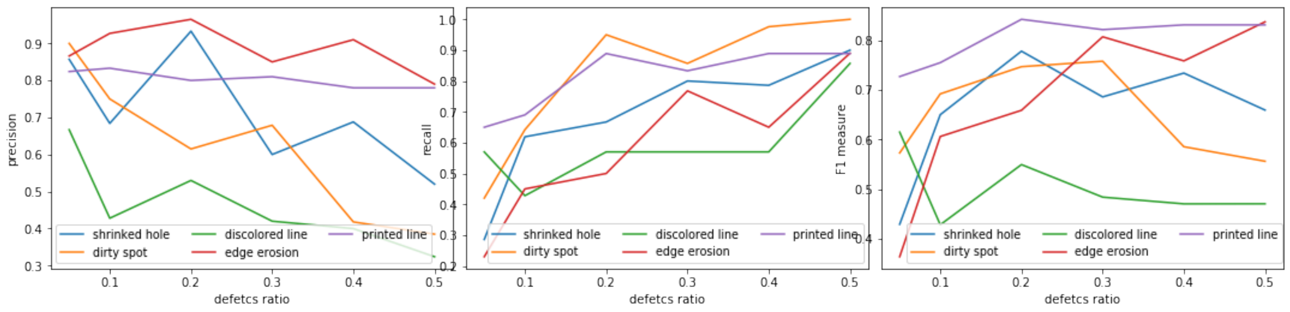
Results of VGG16 From Figure 11 we can observe that with the increase of defects proportion the precision drops while the recall rises with small fluctuations. It indicates that the classification decision gradually deviates from good images to defective images until the model overfits because it learns too much noise from the defective images.

The Figure 11 shows that for different defects class the optimal imbalance ratios are different. One exception is *discoloredline*. In the training set, they only have 24 samples, which is only 3% of the good images. Even if we oversample them, the performance remains the worst.

4.3. GAN Based Anomaly Detection

The overall anomaly detection includes GAN training, finding the closest image and calculating anomaly score. The challenge here is to generate images with large size of 224x224x1. So far the DCGAN can stably generate images with size of 64x64x3. The convolutional layers and deconvolutional layers have strides of 2, so we need a deeper network because our image size is larger. To cope with this challenge, we use fewer filters in the convolutional layers, because the images in our dataset do not have complex features.

Architecture We adopted the basic structure of DCGAN. First, unlike conventional CNN structure, the DCGAN combines the convolutional layer and pooling layer as a strided convolutional layer. Second, the DCGAN uses batch normalization to stabilize the training process by ensuring that the gradient can be propagated to the next layer. Third, the DCGAN uses LeakyReLU layer to prevent dead neurons and sparse



(a) precision (b) recall (c) F1_measure

Figure 11. Precision, recall and F1 for different imbalance ratio using model pre-trained VGG16. The x axis means the proportion of defective images within the whole training set. Different colors represent different classes of defects.

representations in the network.

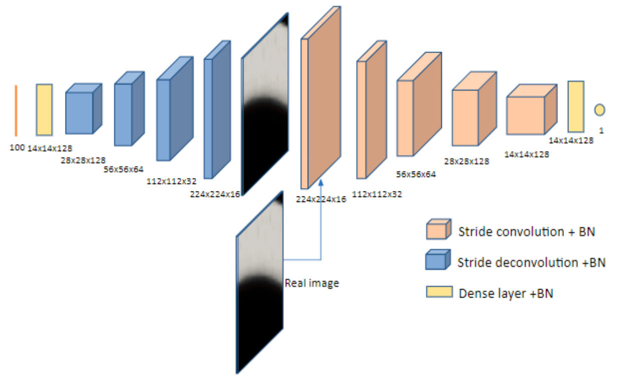


Figure 12. Detailed architecture of DCGAN. Activation layer is not included in the figure. Batch normalization is excluded in the end of generator and the input of discriminator.

Training process The training set has, according to Table 1, only 493 good images. We augmented the dataset by horizontally flipping. The main point in the training process is to make sure that the generator always has something to learn, which means the gradient from the discriminator should not decrease fast. Otherwise, the generator will not have enough gradient to update its parameters and learn how to generate realistic images. Therefore, we use two tricks to make the discriminator's task harder, which are label smoothing (Gabriel Pereyra, 2017) and instance noise (Lars Mescheder, 2018).

Label smoothing is a regularization trick to reduce overfitting by preventing a network from assigning the full probability to each training example. In our experiment we reduce the target label of real images from 1 to 0.8. We also apply instance noise on both of the generated images and real images before feeding them into discriminator. Both tricks contribute to confusing the discriminator. The discriminator does not convergent quickly, and always provides loss to the generator.

Algorithm 3 DCGAN training

```
Input: training data $x$
initialize parameters $w_G$,$w_D$
$epoch = 0$
repeat
    randomly fetch batch size data $x_{1...bs}$
    random augmentation $x_j + N(0, \mu_{epoch})$ for $j \subseteq (1..\frac{bs}{2})$
    sample batch noise $z \sim N(0, 1)$
    $x_{gen} \leftarrow G(z)$
    instance noise $x_{gen}+N(0, \mu_{epoch})$, $x+N(0, \mu_{epoch})$

    real_target $\leftarrow 1 * \alpha$, label_smoothing
    $loss_{real}, loss_{fake} \leftarrow D(x, real\_target), D(G(z), 0)$

    update $w_D$
    fix $w_D$, $loss_G \leftarrow D(G(z), 1)$
    update $w_G$
    $epoch + 1$
until $w_D$,$w_G$ converge
Output: $G(z \rightarrow X)$ , $D(\cdot)$
```

Visual Results of GAN As is illustrated in 3.2.2, after the GAN is trained, we can use it to generate a manifold with infinite number of good images. For a query image, we can find the closest image within this manifold. As is shown in Figure 13, the right column is the pixel-wise subtraction between the original image and its corresponding closest image. For good images there should not be any red region, while for defective images the defects should be exaggerated, marked with red. However, from the right column we can see there are small regions where the pixel values on the original image are larger than on the generated image. It means the generator does not perfectly fit the non-

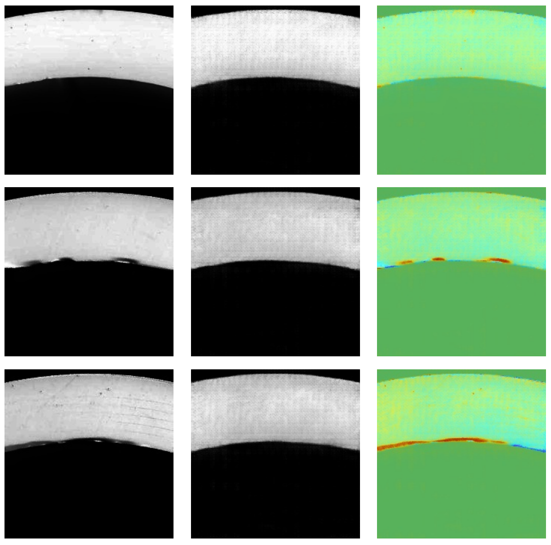

(a) Real image (b) Clean_up (c) Blend

Figure 13. The left column is the original images, in the middle is the closest image found in the generated manifold, and in the right column is subtraction between the left and middle column and mapping to the original image. The top row is the good image while the rest are defective. On the right column the red part implies the pixel value on the generated image is larger than on the original image, which means there is a defect. The blue part implies the pixel value on the generated image is smaller than on the original image, which means the model does not perfectly reconstruct the good areas.

defective parts, which has an impact on calculating the anomaly score.

4.4. VAE Based Anomaly Detection

The VAE based anomaly detection includes training the VAE with only good images, reconstructing the query image and calculating the reconstruction loss. The challenge is that generally for one image the defects take up a small area, which means the distance is not large enough when we use reconstruction loss over all the pixels as the measurement of anomaly detection.

Architecture The model consists of an encoder and a decoder with latent space transformation in the middle. We take the same hyper-parameter as adopted in GAN for the VAE, such as filter size and the number of convolutional layers. Experiments showed that the KL divergence in the latent space can be influenced by the output length of last convolutional layer. Therefore, unlike GAN, we directly connect the latent space to the flattened output of last convolutional layer, in VAE we add an extra dense layer with 256 length to control the value of KL divergence.

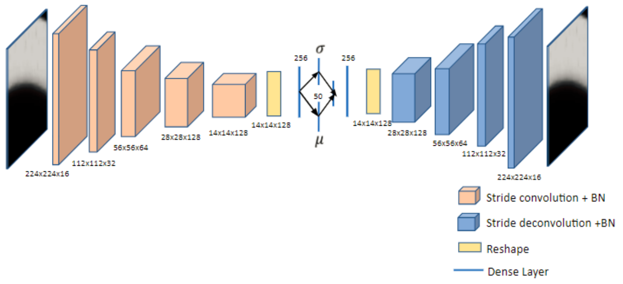


Figure 14. Detailed architecture of VAE. Activation layer is not included in the figure. Batch normalization is excluded in the end of encoder and the input of decoder.

Training process The goal of VAE training is to optimize the lower-bound of log-likelihood of x. The model should balance between reconstruction accuracy and the KL divergence. For the parameter estimation of decoder $p(x|z)$ we maximize the log-likelihood $\log(p_\theta(x))$. Since we can not directly optimize the log-likelihood, we turn to optimize its lower-bound.

In the training process, we need to sample from the latent space, e.g. $N(\mu, \sigma)$. However, the sampling process is non-differentiable so we can not backpropagate the gradient by chain-rule. In the standard VAE training mechanism, *Reparameterization* is used to sample from a standard normal distribution $\epsilon \sim N(0, 1)$ and then build the latent space by $z = \epsilon * \mu + \sigma$. This linear operation is differentiable, and standard stochastic gradient descent (SGD) algorithm can be used for optimization.

Algorithm 4 VAE training

```
Input: training data $x$
initialize parameters $\theta$,$\phi$
$epoch = 0$
repeat
    randomly fetch batch size data $x_{1...bs}$
    random augmentation $x_j + N(0, \mu_{epoch})$ for $j \subseteq (1..\frac{bs}{2})$
    $\hat{z} \leftarrow \mu_x + z * \sigma_x$ Reparameterization
    $L_{KL} = \sum_{i=1}^{bs} -D_{KL}(q(z|x^i)||p(z))$
    $L_{recon} = \sum_{i=1}^{bs} logp(x^i|\hat{z})$
    $L = \frac{1}{bs} * (L_{recon} + L_{KL})$
    update $\theta$ $\phi \leftarrow L$
    $epoch + 1$
until $\theta$,$\phi$ converge
Output: $E_\theta$ , $D_\phi$
```

Visual Results of VAE As is shown in Figure 15, the reconstructed images from VAE are more blurred

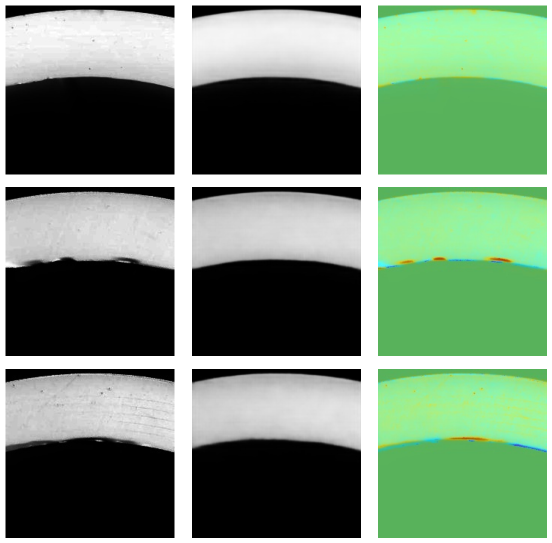


Figure 15. The left column is the original images, in the middle is the reconstruction image with minimum loss and in the right column is subtraction between the left and middle column and mapping to the original image. The top row is the good image while the rest are defective. On the right column the red part implies the pixel value on the generated image is larger than on the original image which mean there is a defect.

compared with the visual result of GAN, and the texture details are not well reconstructed. The defects revealed by the residual images are not as clear as the one shown in Figure 13.

4.5. Determine the Threshold

In this section, we introduce the approach how we determine the threshold of VAE and GAN based anomaly detection, which calculate the distance between the input image and its reconstruction on pixel level. After the GAN and VAE are trained on the good images, they can reconstruct images based on the pipeline proposed in Section 3. The distance calculator assigns either anomaly score or reconstruction loss to the input image. The outputs of distance calculator are divided into two categories based on the label of the input image, namely good and defective. The dataset for determining the threshold is the same for training the supervised learning (VGG16) in order to compare the performance.

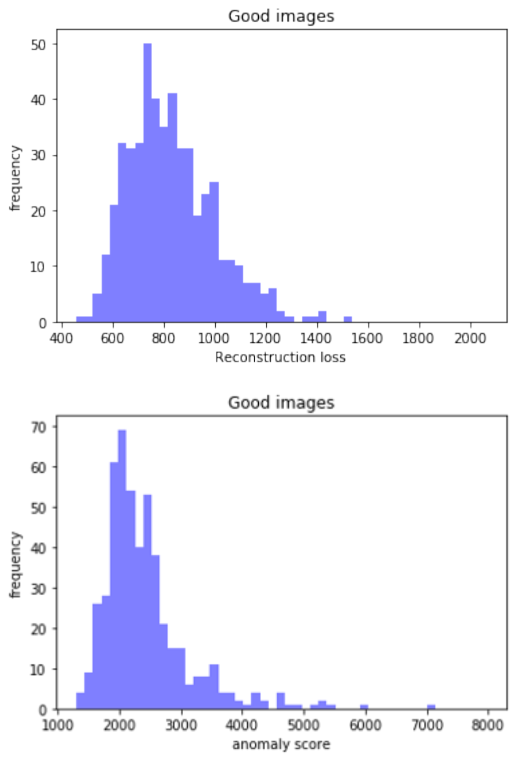
After the reconstruction loss or anomaly score for each image in the training set is obtained, we drew the histograms for good images and defective ones respectively as is shown in Figure 16. Those histograms approximately follow a normal distribution. We calculated the mean and variance for each subplot and fit the distributions to all histograms. Figure 17 shows the merged distribution of reconstruction loss and anomaly score. The intersection point of two distributions is the threshold. As can be seen from Figure 17, the performance of VAE is better than GAN when the distance is calculated on pixel level. Because the distribution of defective images based on reconstruction loss are more separated from the good ones compared with the distribution based on anomaly score.

4.6. Distance on Feature Level

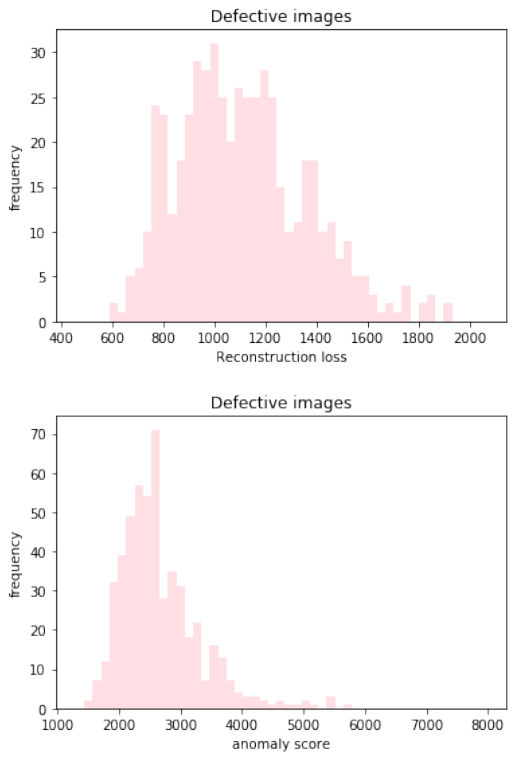
Architecture To project the distance of two images on feature level, we use a Siamese network with the input of original images and their reconstructions that are either from GAN or VAE. The architecture of our proposed Siamese Network simply uses two shared weights modified VGG16 mentioned in Figure 10 as a feature extractor with two inputs channels. Then we concatenate the outputs of the two channels into a single vector and feed this vector to the last sigmoid node to decide whether two input images are similar or not.

Learned Feature Comparison To visualize if the model is capable of capturing the features of different defects we can build salience map for each query image. The salience map is generated by the weighted sum of the output feature map of the last convolutional layer where the weights are the connections between the feature vector and the last neuron with sigmoid output. This implies that the output of the feature extractor can capture the most distinctive features which are only relevant for classification decision.

We visualize the feature maps in both simple VGG16 and our Siamese Network. The Figure 20 shows salience maps for three defective images, all of which are correctly predicted in both VGG16 and Siamese Network. The Figure 19 shows salience maps for other three defective images, but they are only correctly predicted in Siamese network. We can observe that features captured in VGG16 are finer compared with ones in Siamese Network. For the same defects, the feature maps of VGG16 fit more accurately than Siamese Network, while the activation regions in Siamese network are broader. However, for defects *printed line* that diffuse in the image the Siamese network can capture the complete feature of the defects while VGG16 focuses only on a small region.



(a) Histogram of good images



(b) Histogram of defective images

Figure 16. Histograms of good and defective images. The upper row is the histograms of reconstruction loss. The lower row is the histograms of anomaly score. The blue parts are the histograms of good images. The pink parts are the histograms of defective images.

4.7. Quantitative Results

In this section, we compare the quantitative results of GAN and VAE, with combinations of pixel level distance and feature level distance. Regarding the pixel level distance, VAE using reconstruction loss performs better than GAN using anomaly score. Regarding the feature level distance, Siamese Network together with results of GAN performs better than Siamese Network together with results of VAE. From the table 2 we can see that the feature level distance is more effective than pixel level distance. For defect *shrinkedhole*, although the defects recall remains the same but we improved the precision. For *dirtyspot*, *edgeerosion* and *printedline* feature level distance significantly improved the defects recall with only a little trade-off of precision.

However, the quantity of sample also influence the results. For defects *discolored line*, all of the methods perform very poorly. Because the images are extremely rare in the training set, with only 14 samples, and the defects are not so distinctive compared with good images.

5. Discussion

5.1. Impact of Data Imbalance

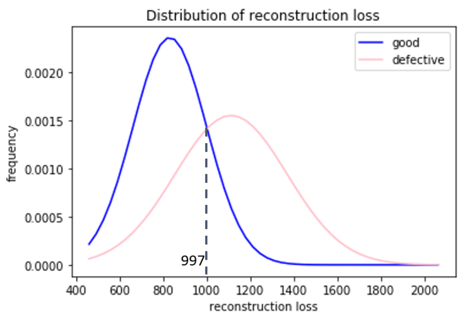
From Figure 11 we can observe the impact of the data imbalance on the classification results. We fix the total amount of the overall dataset and vary the proportion of defective images. We use the F-measure to balance between precision and recall because either metric can be extreme because of the imbalance ratio. For binary classification the ideal distribution is $1 : 1$ so the network will not bias to either of the classes. From data level perspective we can either over-sample the defective image or under-sample the good images to change data distribution.

Normally as the defects ratio increases, the F-measure will rise and then drop again after reaching the optimal point, which is shown in defects classes *shrinked hole* and *dirty spot*. Because extreme over-sampling can lead to overfitting on the defective images.

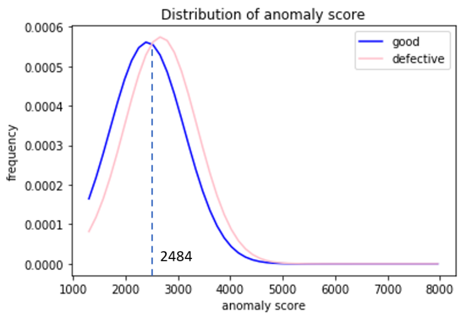
If we look at the defects class *printed line*, the F-measure always stays high. Because the printed line defects cover more areas than other defects. This im-

	GAN			VAE		
	Precision	Recall	F1	Precision	Recall	F1
Shrinked hole(21)	0.14	0.18	0.16	0.4	0.7	0.52
Dirty spot(21)	0.09	0.12	0.1	0.22	0.43	0.3
Discoloered line(7)	0.08	0.43	0.14	0.08	0.29	0.125
Edge erosion(56)	0.34	0.46	0.39	0.65	0.7	0.67
Printed line(36)	0.09	0.02	0.04	0.48	0.56	0.61
	Siamese Network + GAN			Siamese Network + VAE		
	Precision	Recall	F1	Precision	Recall	F1
Shrinked hole(21)	0.83	0.71	0.765	0.79	0.71	0.75
Dirty spot(21)	0.83	0.69	0.754	0.67	0.47	0.55
Discoloered line(7)	0.18	0.14	0.157	1	0.14	0.25
Edge erosion(56)	0.82	0.93	0.872	0.86	0.93	0.89
Printed line(36)	0.81	0.97	0.883	0.82	0.9	0.86

Table 2. The results of GAN, VAE, GAN + Siamese Network, VAE+Siamese Network



(a) Distribution of reconstruction loss



(b) Distribution of anomaly score

Figure 17. The upper figure is the distribution of reconstruction loss obtained from VAE. The figure below is the distribution of anomaly score obtained from GAN. The blue line represents good images which the pink line represents defective images. The intersection points are threshold. The threshold for reconstruction loss is 993, for anomaly score is 2484

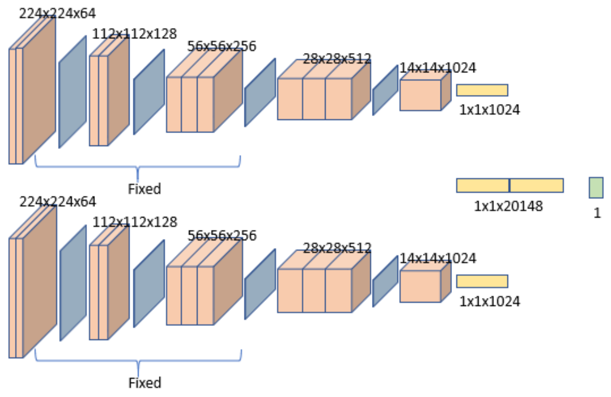


Figure 18. Detailed architecture of metric learning using modified pre-trained vgg16. Activation layer is not included in the figure. Two VGG16 are weights shared.

plies that besides the imbalance ratio, how distinctive the defect is also plays a role in classification that large difference does not necessarily requires much data to train. In future work, we should collect more defective images which contain non-distinctive defects, in order to make up to the small quantity.

5.2. Transfer Learning and VGG16 Modification

The pre-trained VGG16 is trained on the ImageNet which contains millions of data among 1000 classes. When we freeze the model and adapt it to our dataset, we try to capture the features which are unique for our dataset. From previous work of visualization of VGG network (Wei Yu, 2016), we can see that the captured feature become more complicated when the model develops deeper. The feature in the first several

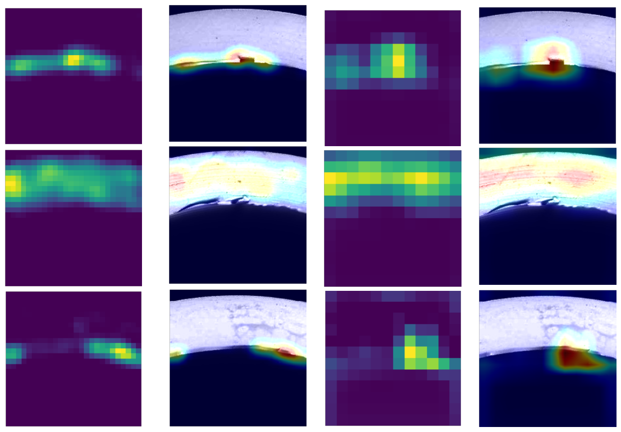


Figure 19. The first column is the salience map generated by VGG16. The third column is the salience map generated by Siamese network. The second and the last column map the salience map back to the original image. Each row represents a certain defective images. The second row is classified as edge erosion. The third row is edge erosion. The first row is printed line.

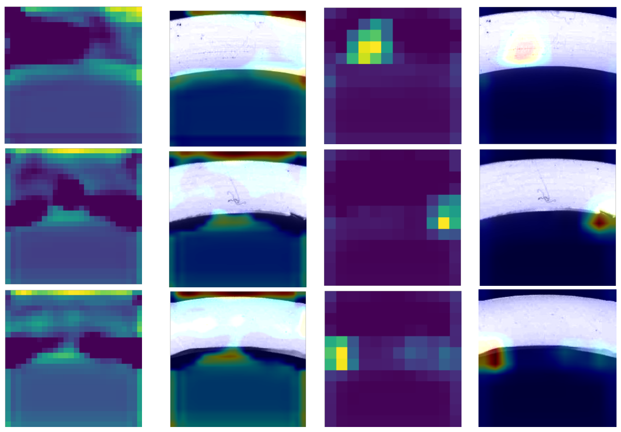


Figure 20. The first column is the salience map generated by VGG16. The third column is the salience map generated by Siamese network. The second and the last column map the salience map back to the original image. Each row represents a certain defective images. The first row and the third row is edge erosion. The middle row is categorized as printed lines.

convolutional layers are usually lines, edges and simple shapes which can be widely adapted to different image domain. Therefore we only freeze initial layers in the model, and train the top layers with our dataset.

Global Average Pooling Global Average Pooling has the following advantages. On the one hand, it replaces the fully connected layer and reduces the training parameters. On the other hand, it reserves the spatial information, and we can use it to determine the area in the image where the classification decision is made. However, the Global Average Pooling is a rough operation since a lot of information is lost. Without fully connected layer the network can not correlate the features in different areas.

However, the simplicity of our dataset can ease this problem. Firstly the image in our dataset does not contain complex features like faces or animals. It consists of an arc-shaped ring and a simple texture. Therefore we do not need to capture complex features. Secondly, VGG16 is very deep, and the last several convolutional layers have feature-length of 512 which contains a lot of combination possibilities. We do not even need the whole VGG16 during the experiment. From the Figure 21 we can compare the feature map from the last convolution block and the second last convolutional block. Features from the last layer are sparser, and they do not focus on the defects region. However, features from the shallower layer are more

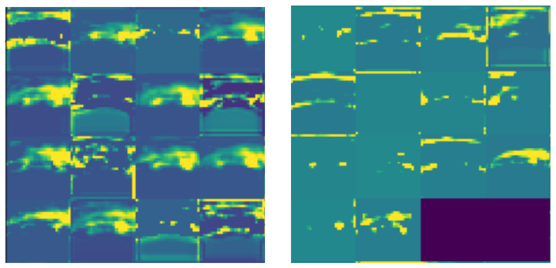

(a) Examples of feature map of layer 11
(b) Examples of feature map of layer 15

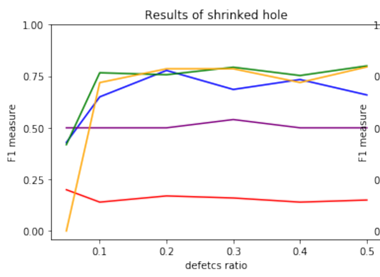
Figure 21. Comparison between feature maps of two convolutional layers

centralized on the defects areas and a lot of repetitive features maps imply that the capacity of the VGG16 is more than enough for our dataset.

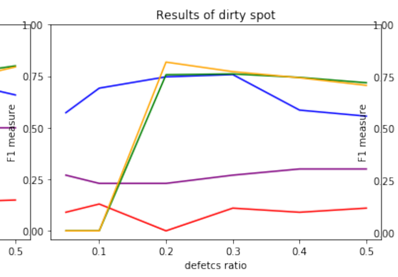
5.3. Supervised vs Semi-supervised Learning

In this section we compare the results of supervised learning and semi-supervised learning. The supervised learning is the first step in experiment, where we used the VGG16 to evaluate the data imbalance. The semi-supervised learning is the one class anomaly detection using GAN and VAE, combining with distance measurement on pixel and feature level. Here we present the F1-measure for each defect class respectively.

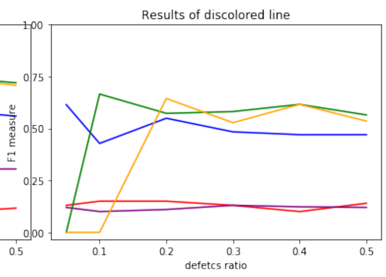
From Figure 22 we can see that the results of tradi-



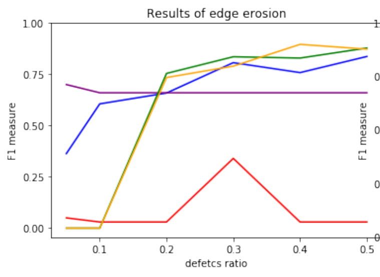
(a) F1 of *shrinked hole*



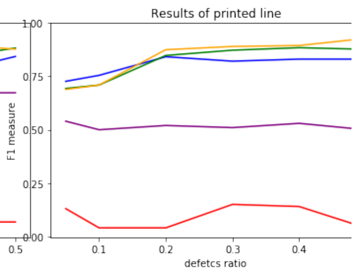
(b) F1 of *dirty spot*



(c) F1 of *discolored line*



(d) F1 of *edge erosion*



(e) F1 of *printed line*

VGG16
GAN
VAE
Siamese Network + GAN
Siamese Network + VAE

(f) legend

Figure 22. Comparison of F1 between different methods for each defect class. Blue line stands for VGG16. Pink line stands for GAN using anomaly score. Yellow line stands for VAE using reconstruction loss. Green line stands for Siamese Network based on GAN. Orange line stand for Siamese Network based on VAE.

tional classification vary a lot with the defect ratio. Furthermore, even the training process of the reconstructor does not have a problem of data imbalance, the defects ratio has a strong influence over the threshold determination. In most cases, the VAE/GAN plus Siamese Network perform the best, while the performance of GAN using anomaly score stays at bottom. In the extreme imbalance case, the supervised learning performs better as semi-supervised learning. From Figure 22 we can also tell the best performer at different defect ratios, which can be used later to decide which method should be adopted, when we come across dataset with different defect proportions.

5.4. Feature Learned in VAE & GAN

Both GAN generated images x_{gan} and the VAE reconstructed images x_{vae} can reconstruct good images, which means their positions and shapes are well learned. However, the x_{vae} loses the details of texture that make the images vivid while the x_{gan} can generate the gradual change of shades. One problem that influences the verisimilitude of the x_{gan} is the checkerboard effect(Odena et al., 2016). In general, the generated images of both models are *cleaner* than the original good images.

Reconstruction Images in GAN & VAE The good images are not perfectly clean where there existing minor flaws which may affect the classification results. Especially some imperfections are not considered as defects because of their small defect level, even if they have the similar feature as the real defects.

The reconstruction image can be considered as clean images, as is shown in the first column of Figure 13, where the small imperfection is not reconstructed by the generator. In Figure 15, the generated images from VAE are even cleaner than ones from GAN, which means the GAN and VAE did not learn the texture details of good images. Lacking of detail is also a possible reason why pixel level distance performs poorly. While the image size is so large that summing up minor difference overall the pixels overwhelms the difference brought by the defects.

5.5. Problem of Definition *Closest*

When we reconstruct the image based on GAN, we observe an interesting problem that the reconstructed image does not fit perfectly the area where the product should be. The situation can be considered within error range if the non-fitting area is not significant. However, they appear very often, and are sometimes significant enough to influence the anomaly detection.

The problem lies in the definition of the *Closest*. When

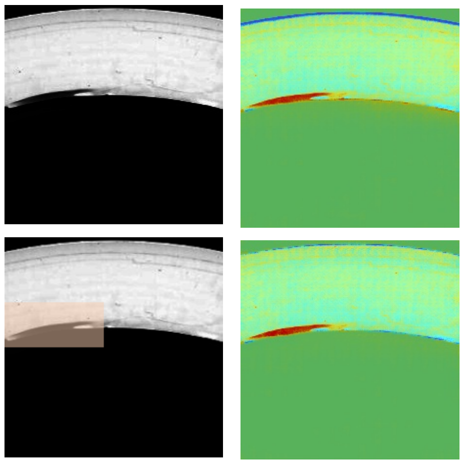

(a) Input image (b) Blend after reconstruction

Figure 23. The left column is the Input image for reconstruction. The top row is the original image with edge erosion defect. The second row is when we mask the defect. Here to visualize we use different color.The right column is blending of residual image and the original image after subtraction

we reconstruct the image, we want to find the *closest* image to the original image. According to Equation 3 and 4 we defined the pixel-wise and feature-wise loss where pixel-wise loss takes dominant place. Because in the training set the position of the product varies as well as the width of the product slice. Therefore the generator will have the ability to adapt to the defects and try to balance the loss brought up by the defects.

To validate our hypothesis we masked the defects (which is pink in the Figure 23) and input the image into the generator for reconstruction. It means we will reconstruct the defects based on the context of the rest of the image. In the experiment we modified the loss function backpropagated to the latent space.

$$\begin{aligned} L_{pixel}(x, z, b(h, h', w, w')) &= \sum_{i=0}^{m}\sum_{j=0}^{n} |x - G(z)| \\ &- \sum_{i=h}^{h'}\sum_{j=w}^{w'} |x - G(z)| \end{aligned} \tag{14}$$

Equation 14 is a modified pixel-wise loss. The first term is the same as Equation 3. The second term is the pixel-wise loss within the defective region, where h, h', w, w' indicate the lower and upper index of the bounding box within the image. We use only the rest of the image by backpropagating the loss outside the bounding box.

As shown in Figure 23 when we reconstruct the original image, there is a blue margin appearing on the top of the edge. This implies that the generator tries to enlarge the width of the slice to balance the loss occurred by defects. When we mask the defects, the margin disappears.

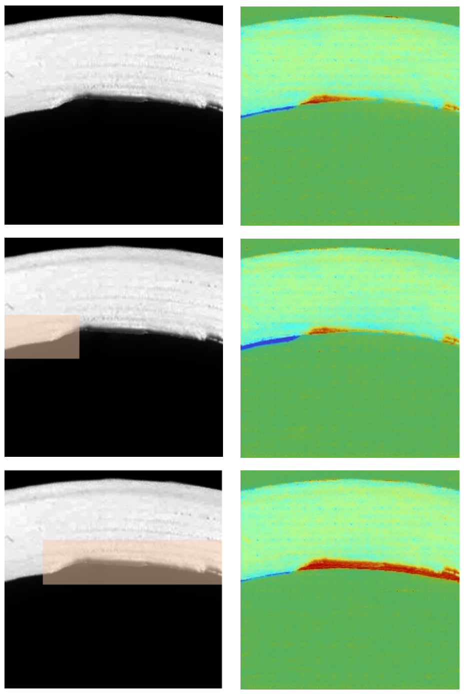

(a) Input image (b) Blend after reconstruction

Figure 24. The left column is the Input image for reconstruction. The top row is the original image with edge erosion defect. The second row is when we mask the a region instead of defect. The third row is when we mask defect.The right column is blending of residual image and the original image after subtraction

Another example to explain the adaptability of the generator is shown in Figure 24. The original image has a large gap in the lower edge while the reconstructed image displays only a small difference region. The generator tries to fit this gap by partially raising the edge in the gap area. When we mask the defects, the generated image follows the arc of the remaining edge and displays the full defects.

Masking the defects and reconstructing based on the remaining part shows a better accuracy to reveal the defects. However, pre-knowledge of where the defect lies is indispensable. This leads to future work to come up with an effective masking strategy for reconstruction.

6. Conclusion

In this thesis, we proposed a pipeline that uses generative models (GAN and VAE) for anomaly detection, in order to avoid the problem of data imbalance where good images take the majority of the dataset. The generative models are used to reconstruct images and reveal the defects based on the distance between the original images and their reconstructions. Furthermore, we measured the distance between the images and their reconstructions on both pixel level and feature level. We also compared the anomaly detection with traditional supervised learning. The results show that, in most cases, anomaly detection using feature level distance performs the best.

From the visual results, both GAN and VAE are able to reconstruct good images and reveal the defects. However, the anomaly score defined in formula 3 and the reconstruction loss defined in formula 11 are unable to distinguish between good images and defective images. Because the defects only take up small area of the whole image and the image size is large(224x224), sum up all the pixel difference decreases the salience of defects.

In the feature level, The Siamese Network using pretrained VGG16 can capture the distinctive features of different defects. Compared with a baseline of single VGG16, Siamese Network improves the recall of defects in all defect classes. Furthermore, using Global Average Pooling can build up a salience map for defects localization.

During the experiment, we observed that defects can be more accurately revealed when we reconstruct images, with defects being masked. However, masking strategy that does not based on the pre-knowledge of defects location is the future work of this project. What's more, in this thesis, we only experiment on the binary classification, ignoring the fact that multiple defects can exist on the same image.

6.1. Future Work

In this thesis, we only experiment with the binary classification, ignoring the fact that multiple defects can exist on the same image. In the future, we would like to investigate how each class of defects differentiates from each other, using multi-labelling classification. Besides, we would like to investigate an efficient masking strategy to reconstruct images more accurately.

For training the GAN, there are several things which are missing in our experiments. There are few variants of GAN that have been proposed recently, claiming better performance than the simple GAN. Particularly Wasserstein GAN (WGAN)(Martin Arjovsky, 2017), which uses Wasserstein distance to solve instability and mode collapse of GAN. Besides, during the experiment, we observed a mild checkerboard-effect brought out by the deconvolution operations (Odena et al., 2016), which could have influenced pixel level distance between original images and their reconstructions. We could later directly up-pooling of feature maps instead of deconvolutional layer, and see if the performance improves.

References

Alec Radford, Luke Metz, Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR 2016*, pp. 2–6, 2016.

Alex Krizhevsky, Ilya Sutskever. Imagenet classification with deep convolutional neural networks. *NIPS*, 22(3):1106–1104, 2012.

Bromley, Jane, Bentz James W Bottou Leon Guyon Isabelle-LeCun-Yann Moore Cliff Sackinger Eduard and Shah, Roopak. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence,*, 7(4):669–688, 1993.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe Jonathon Shlens Zbigniew Wojna. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*, pp. 2–10, 2015.

Diederik P Kingma, Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, pp. 3–5, 2013.

Gabriel Pereyra, George Tucker. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, pp. 1–4, 2017.

Goodfellow, Ian, Pouget-Abadie Jean Mirza Mehdi Xu Bing Warde-Farley David Ozair Sherjil Courville Aaron and Bengio, Yoshua. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 7(3):26722680, 2014.

Haugh, M. Overview of monte carlo simulation, probability review and introduction to matlab,. Technical report, Columbia University, USA, 2004.

Huang Jiexian, Li Di, Ye Feng et al. Detection of surface of solder on flexible circuit[j]. *Optics and Precision Engineering*, pp. 2443–2453, 2010.

Jun-Yan Zhu, Taesung Park, Phillip Isola Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, pp. 2, 2017.

Karen Simonyan, Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, pp. 3–4, 2014.

Lars Mescheder, Andreas Geiger, Sebastian Nowozin. Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*, pp. 2–6, 2018.

Martin Arjovsky, Soumith Chintala, Lon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, pp. 3–4, 2017.

Mehdi Mirza, Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, pp. 2, 2014.

Min Lin, Qiang Chen, Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, pp. 2–4, 2013.

Odena, Augustus, Dumoulin, Vincent, and Olah, Chris. Deconvolution and checkerboard artifacts. *Distill*, 2016. doi: 10.23915/distill.00003. URL `http://distill.pub/2016/deconv-checkerboard`.

Salimans, T., Goodfellow I. Zaremba W. Cheung V. Radford A.-Chen X. Improved techniques for training gans. *In: Advances in Neural Information Processing Systems*, 22(3):22262234, 2016.

Sergey Ioffe, Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, pp. 2–6, 2015.

Sinno Jialin Pan, Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(3):2–3, 2009.

Thomas Schlegl, Philipp Seebck, Sebastian M. Waldstein Ursula Schmidt-Erfurth-Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *arXiv preprint arXiv:1703.05921*, pp. 4–5, 2017.

Wei Yu, Kuiyuan Yang, Yalong Bai. Visualizing and comparing alexnet and vgg using deconvolutional layers. *icml*, 7(3):5–7, 2016.

Zhang Xuewu, Ding Yanqiong, Duan Dunqin et al. Surface defects inspection of copper strips based on vision bionics[j]. *Journal of Image and Graphics*, pp. 594–599, 2011.