Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics

Detecting production anomalies using Diffusion Networks

Scientific Students' Association Report

Author:

Nándor Szécsényi

Advisor:

dr. László Lengyel
Gergely Karz

2023

Contents

Kivonat **i**

Abstract **ii**

1 Introduction **1**

2 A brief look on Anomaly Detection Techniques **3**

3 The Measurement Setup **5**

3.1 The Anomaly Detection Process 5

3.2 Metrics 6

3.2.1 Mean Squared Error 7

3.2.2 Area under curve 7

3.3 The baseline method 10

4 The Dataset **11**

4.1 The Solder Images Dataset 11

5 Networks **15**

5.1 Autoencoder 15

5.2 Denoising Diffusion Probabilistic Model 20

6 Analysing the Results **24**

6.1 Results on the Blob Soldering Samples 24

6.2 Results on the Paste Low Samples 27

6.3 Results on the Burnt Samples 31

7 Conclusions **39**

Bibliography **41**

Kivonat

A mesterséges intelligencia korában, amikor naponta olvashatjuk a híreket annak fejlődéséről, természetesnek vehető, hogy az ipari adathalmazok hibadetektálására is számos módszer áll rendelkezésre, melyek mesterséges intelligenciát, azon belül is mélytanulást alkalmaznak. Erre a feladatra jelenthet egy új megközelítést a zajtalanító diffúziós hálózatok (DDPM) alkalmazása, ezen hálózatok ugyanis képesek a tanult mintákat visszaállítani kis minőségveszteség mellett, illetve eltérő, hibás adatminta esetén, azt egy tanulttal megegyező kinézetű mintává alakítják át, így egy teljesen új képet generálva a bemenetiből. A modellek ezen tulajdonságát kihasználhatjuk, ha csak jó minőségű mintákkal tanítjuk őket és ezután a bemenetükre hibás adatokat adunk, amelyeket a hálózat átalakít jó minőségű képekké a tanultak alapján, eltüntetve az eredeti deformitásokat. Ennek köszönhetően a bemeneti és kimeneti képek különbségét vizsgálva szétválasztható, hogy mely minták voltak megfelelő minőségűek és melyek tartalmaztak hibákat, amely módszert már például agyi felvételek elemzésénél sikeresen alkalmaztak [1]. A megközelítésből kiindulva a dolgozatban a diffúziós neurális hálózatok felügyelet nélküli hibadetektációs képessége kerül bemutatására valós ipari adatokon keresztül, nagy hangsúlyt fektetve a nehezen osztályozható minták feldolgozására. A kapott eredmények ezen felül összehasonlításra kerülnek egy autoenkóder modell különbség eloszlásaival is, amely hálózat esetében a hibadetektálási folyamat lényegében ugyanaz, azonban a a visszaállítás minősége gyengébb, a kapott képek elmosódottak, így zajt jelentve a kimenetre nézve, elnyomva a hibák eltüntetéséből eredő különbségeket. Emiatt a diffúziós modell jobb kimeneti képminőségéből adódóan alkalmasabb lehet a kisméretű és kis kontrasztú hibák észlelésére, míg az autoenkóder esetében, a nagyobb kimeneti jel-zaj viszony miatt, ezek nehezen detektálhatóak.

Abstract

We truly live in the era of Artificial Intelligence, reading about its achievements on a day-to-day basis, so it is natural that there are several solutions available for detecting anomalies in industrial datasets that are based on AI, more precisely deep learning models. A new approach to this task can be the application of Denoising Diffusion Probabilistic Models (DDPMs), as these networks are able to reconstruct the training samples with minimal loss in image quality, while also transforming different, faulty samples to ones similar to the training data, which only consists of samples without any anomalies, thus generating an entirely new image from the input. By exploiting this transformation and generative feature of the networks, we can train them with only good samples and after training, faulty samples can be provided to the network for partial inference (when the model has to denoise a sample) and the model, in theory, generates a good sample from the faulty one based on its training, thus eliminating the original's anomalies. Thanks to this behaviour, by evaluating the difference between the input and output images, we can separate which samples were good and which had some deformities in the dataset, which ability has been already proven in the case of brain scans [1]. Originating from this approach, in this thesis, the anomaly detection ability of the Diffusion Networks is detailed using actual industrial production data with a big emphasis on detecting the more difficult samples. The obtained results are also compared to the reconstruction loss distributions of an Autoencoder model, where the overall process is the same, but the reconstruction quality is lower, the images are blurred, resulting in additional noise regarding the output. Based on this fact, thanks to its better reconstruction quality, the general expectation is that the diffusion model is better suited for detecting smaller and less prominent anomalies, which are hard to identify in the case of the Autoencoder due to the its higher output signal-to-noise ratio.

Chapter 1

Introduction

As the constant research and development of artificial intelligence is currently a huge driving force in the world, it is natural that AI and Deep Learning based methods in particular are being applied more and more at different areas. This is the case with the task of Anomaly Detection in a production environment, where it is crucial to provide a reliable method that can separate the incoming samples with high precision and stability. One of the existing approaches is the use of the Autoencoder architecture, which can learn an encoding of the original pictures and reconstruct them using it. The problem with it is that the reconstruction quality is bad because the output images are blurred, which makes the detection of the differences coming from correcting the smaller anomalies much more harder. However, with the emergence of Diffusion Networks, a new, generative approach could be given, as it has much better output image quality, possibly solving this problem and improving the performance of the detection process. Based on this expectation, in this thesis, I present a Diffusion based Anomaly Detection approach that can be applied in a production setting. This presented method is evaluated on an industrial dataset and it is compared to the performance of the Autoencoder architecture. To make the comparison more thorough and informative, both models are trained on various numbers of training samples to help visualize a trend in their performances.

The main structure of the thesis itself can be separated into two parts: the first serves as an introduction to the Anomaly Detection problem, details the measurement setup, contains information about the dataset and the network architectures used throughout the thesis, while the second part showcases the results, their analysis and the obtained conclusions.

After this introduction, the first chapter introduces the basis of the Anomaly Detection problem and lists some of the already used approaches. The second chapter presents the workflow of the Anomaly Detection process used in this thesis, with details on every step, input and output. Moreover, the measurement setup is also introduced and the different metrics used in the evaluation of the obtained results are also discussed. In the third chapter, the Autoencoder and the DDPM network architectures are introduced and analysed with some brief explanations about their core aspects. To help understand these networks, some of their main parameters are also analysed more thoroughly, as they will play a crucial role in optimizing the performance of the detection process. After these, the next chapter details the dataset used for evaluation, with a high emphasis on the appearance and features of each class, as they will be important later in the process. In the last chapter, the results for the different methods on each error class of the dataset are listed and analysed, with an emphasis on the harder to detect anomalies: for these,

outside of the basic analysis, a more in depth evaluation is carried out involving the confusion matrices of the two approaches. Using these results, each approach is compared across different parameters to rank them and establish whether the DDPM based method can outperform the already existing solutions.

Chapter 2

A brief look on Anomaly Detection Techniques

Detecting faulty samples in a given dataset has been a key aspect of the industrialized world for centuries, as by increasing the detection ratio of anomalies, the production performance and stability can also be improved and kept at a much higher level. For low dimensional data, there are several statistical approaches that can be used to select the outlier samples from a given dataset: they range from simple observations to more complex algorithms that cluster the Data into groups (see [2] and [3] for a detailed summary, where the anomaly detection methods are grouped based on their core features and compared to each other in great detail). However, in the case of high dimensional data, like images, many of the traditional approaches cannot be applied due to the larger size of the individual data samples. Here come the Machine Learning and Deep Learning techniques into the highlight, as they have proved to be efficient in several image processing tasks due to their ability to efficiently analyze the spatial information of even large pictures. Moreover, they can also be quite easily integrated into the already developed methods: for example, the neural network training can be paired with Active Learning, which was already a reliable solution on its own in several use-cases [4]. As the neural networks became even more efficient and deeper, having several different type of layers that further optimized their performance, new architectures were developed for detecting outlier samples. One of these was the Autoencoder neural architecture (see it detailed in Section 5.1), which learns a representation of the original image and tries to reconstruct it from this embedding. This architecture can be used effectively in Anomaly Detection, as it is able to learn the main features of the samples and both its embedding and output images can be used for separating the good samples from the bad ones [5]. With the introduction of generative adversarial networks (GANs) [6], the possibilities for outlier detection increased, as now, the networks could generate high quality samples and could also decide which data contained anomalies according to their own knowledge: for example they achieved great results in the case of capturing imaging markers relevant for disease progression and treatment monitoring [7]. Recently, this ability increased with the advent of Denoising Diffusion Probabilistic Models (DDPMs) [8] as the generated image quality is even higher than their previous counterparts: a good example for the high reconstruction quality is the popular DALL-E text-conditional image generator [9], with which the user can generate high quality images based on instructions given in text input without limits. The core feature of the DDPM model is that it gradually applies noise to the input image and then tries to reverse it, learning the distribution of the dataset during the process (this is detailed in Section 5.2). Originally these networks were not used for Anomaly Detection

tasks, but nowadays, they are being used in more and more fields, for example in medical use-cases [1] and surface evaluations [10] with great success. However, there has not been a comprehensive analysis of this process in the case of industrial production datasets, which is the main reason behind this thesis. Moreover, usually the detected anomalies have a trend or a typical appearance, so it is also exciting to see how does the DDPM network deal with very small and hard-to-detect production anomalies which can differ for each and every sample.

Chapter 3

The Measurement Setup

After presenting the existing detection methods, in this section, the Anomaly Detection process is detailed which is the basis of evaluating the performance of the DDPM network. Moreover, the used metrics and measurement setup are also introduced which are used to test the methods and compare their performances to each other.

3.1 The Anomaly Detection Process

In this section, the Anomaly Detection process itself is discussed, as it is important to understand each of its steps in detail, in order to determine its parameters and be able to evaluate the obtained results more effectively. To visualize each step, Figure 3.1 is provided, where the flowchart of the whole process is pictured: each step is symbolized by a rectangle with a solid line, while every input is shown with a dotted border.

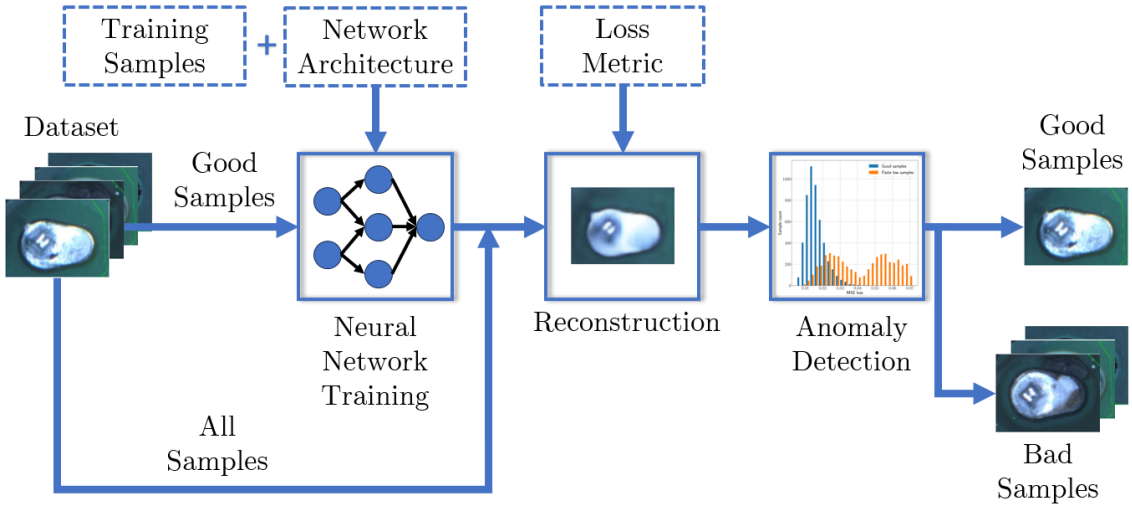


Figure 3.1: Flowchart of the Anomaly Detection Process

Before the first step, the dataset needs to be divided into two groups, where one only contains the good quality samples and the other has the bad quality ones. This division is needed, as the next step is the network training, which should only involve the good samples. It might sound difficult or even counter intuitive to categorize the dataset before the Anomaly Detection begins, but usually in the case of Industrial datasets, the production line outputs good samples in over 99% of the cases. So in case of larger sample sizes, it

does not really affect the training of the neural network, if some bad samples are kept in the training dataset, as their contribution to the overall loss will be minimal, making the network not learn their features.

After the network training is finished, the next step is the selection process, where it is determined for every sample in the dataset whether it is considered a good or bad image. This is done by calculating the reconstruction loss of each sample, which is simply the difference between the input and the output image, summarized into one number (see Subsection 3.2.1 for more details). Based on this loss value, a distribution of the samples can be derived, where the general expectation is that data that are similar to each other in appearance will have similar loss values in the distribution as well. Based on this, since the good production samples should be very alike and also the network was trained on them –meaning their reconstruction loss value should be smaller in general– should be situated at the lower end of this distribution with a small deviation. The samples with errors, on the other hand, can vary in appearance based on the type of anomaly they have (for example the extent of excess soldering paste on the board) and they overall differ from the good samples, so they should be located at the higher end of the distribution –due to their larger reconstruction loss values–, also having a larger deviation due to their uniqueness. If these expectations turn out to be true, then the good and bad samples can be easily separated by a threshold between the two distributions. Unfortunately, this only happens in ideal circumstances, so the two distributions will overlap each other in the majority of cases, due to the problems with the quality of the reconstructions: for example, in the case of the Autoencoder, the blurred nature of the output images causes the two distributions to shift and overlap heavily, as every sample has a constant loss value added because of this noisiness. Due to the overlap, the threshold value cannot be set so easily, since there is no empty space between the good and bad samples.

To provide a better evaluation technique besides analysing the distributions visually, as the last step of the Anomaly Detection process, another metric, the AUC value (see Subsection 3.2.2 for a detailed description) is calculated, which will summarize the extent of this overlap into one single value. Based on this value, the ability to separate the two distributions is rated and the extent of the aforementioned overlap can be evaluated without having to analyze the distributions visually. Moreover, using this metric, the performances of different methods regarding the quality classes can be simply compared to each other, resulting in a definitive ranking.

Finally, to add another dimension to this comparison, the performance of each model should be tested with several different setups regarding the parameters of the detection process, so that we can get a more complete picture of the difference in performance between the two approaches. One way to accomplish this is to run each Anomaly Detection method with a set of different training dataset sizes ranging from small to large and have them evaluated on the same validation dataset. Based on the acquired results, it will be possible to establish a better understanding of each model's main characteristics when it comes to the reconstruction quality of the outputs, as they will learn on an increasingly more diverse training dataset, as the sample size is gradually increased.

3.2 Metrics

After discussing the overall measurement workflow, in this section, the metrics used for evaluating the results are detailed, as they provide the basis for evaluating each model's anomaly detection performance. In this thesis, two metrics are used to evaluate the

Anomaly Detection performance of the different networks: the *MSE* loss value and *AUC* metric. The first one is a stable, well-used metric in the images processing field, as it is very straightforward, easy-to-obtain, although, due to its simple nature, it has some flaws as well. The AUC metric is usually used in classification tasks, however, it can also be used in the case of Anomaly Detection, as it can summarize the performance of a method with just one number instead of relying on visual distributions.

3.2.1 Mean Squared Error

Although the Mean Squared Error (MSE) is a very simple metric, it still needs to be discussed, as it serves as the basis of the Anomaly Detection workflow. Since it is such a well known metric in the field of image processing, in this section, only a brief description is provided.

To begin with, we need to take a look at the formula of this metric, as most of its characteristics can be derived directly from it. Based on solely its name, the exact formula can be written very easily, as it is just the average squared difference between two groups of data, in our case two images with the same size:

$$f(\mathbf{x}, \mathbf{y}) = \frac{1}{N \cdot M} \sum_{i=1}^{N} \sum_{j=1}^{M} (\mathbf{x}_{ij} - \mathbf{y}_{ij})^2 \tag{3.1}$$

where $\mathbf{y}$ is the output and $\mathbf{x}$ is the input image with both having $N \times M$ size. The advantage of this calculation is that it is easy to evaluate, since it only requires basic operations and producing the difference can be done in parallel across the pixels, making it even quicker to execute. It is also beneficial that the more lower an MSE value gets, it means a more smaller difference in pixels, due to squaring the differences of the normalized pixel values which are less then 1. However, this can also cause some difficulties, as it makes comparing two samples just based on their MSE values harder.

To sum up, the MSE value is a very straightforward and easy-to-use metric in the case of calculating the difference between two images, however its drawbacks or hindrances (due to its simple nature) also need to be kept in mind and it should be used with some other techniques to complement it and to provide a more complete analysis.

3.2.2 Area under curve

The second metric used in this thesis, is the Area Under Curve (AUC) value, which is also a widely used metric in evaluating classification performances. However, it can also be intuitively used in the case of Anomaly Detection, so in this section, it will be explained, why it is suitable for this task as well.

The basis of this metric is the Receiver Operating Characteristic (ROC) space which can visually show how good one method is with respect to another one based on its True Positive Rate (TPR) and False Positive Rate (FPR) (see [11] for a detailed description). This plane is separated by a diagonal line representing the performance of the random choice, which serves as the baseline when evaluating the performance of a classification method: the higher a given method is above this diagonal, the better it is at classifying the samples of the given dataset. In the case of continuous probabilistic variables, ROC curves can also be displayed, where each point is treated as a different setup or state for

the analysed model. To plot these curves, the TPR and FPR values need to be defined the following way:

$$\mathrm{TPR}(T) = \int_T^\infty f_1(x)\,dx \tag{3.2}$$

$$\mathrm{FPR}(T) = \int_T^\infty f_0(x)\,dx \tag{3.3}$$

where T is a changeable threshold variable determining whether the continuous probabilistic variable (X) is considered positive ($X > T$) or negative ($X < T$) while $f_1(x)$ and $f_0(x)$ are the density functions of the positive and negative X's, respectively. By plotting this curve, the model's performance can be evaluated in a dynamic way: the goal is to reach the $(0, 1)$ point, which serves as the perfect classification, as soon as possible, while the FPR still stays low during the process of changing the threshold variable. To describe this progression with just one number, the Area Under Curve (AUC) value can be calculated, which is simply the overall area below the plotted ROC curve and the closer it gets to 1, the better the process is at classifying the samples from the dataset.

To utilize this metric, during the Anomaly Detection process, first we need to label the samples and transform the task into a classification problem, which can be done easily by first normalizing the loss values between 0 and 1: giving the good samples the label 0 and annotating the faulty ones with 1. This approach is effective since the good samples are expected to have a lower loss value, so they should be closer to 0, while the faulty images cannot be reconstructed perfectly, giving them a higher MSE value and placing them closer to 1. Plotting the ROC curve this way will represent for each T threshold variable how correctly the two distributions can be separated. If the good and faulty samples form two distinct distributions in the range between 0 and 1, then the ROC curve will achieve high TPR values at most threshold values, while keeping the FPR low, resulting in an overall higher AUC value. This way a higher AUC value means that the given method produces a more structured sample distribution based on reconstruction and it is easier to detect and label the samples that have anomalies. To provide an example of how this evaluation works in practice, Figure 3.2 shows the ability of an Autoencoder detecting the paste low solderings among the correct samples.

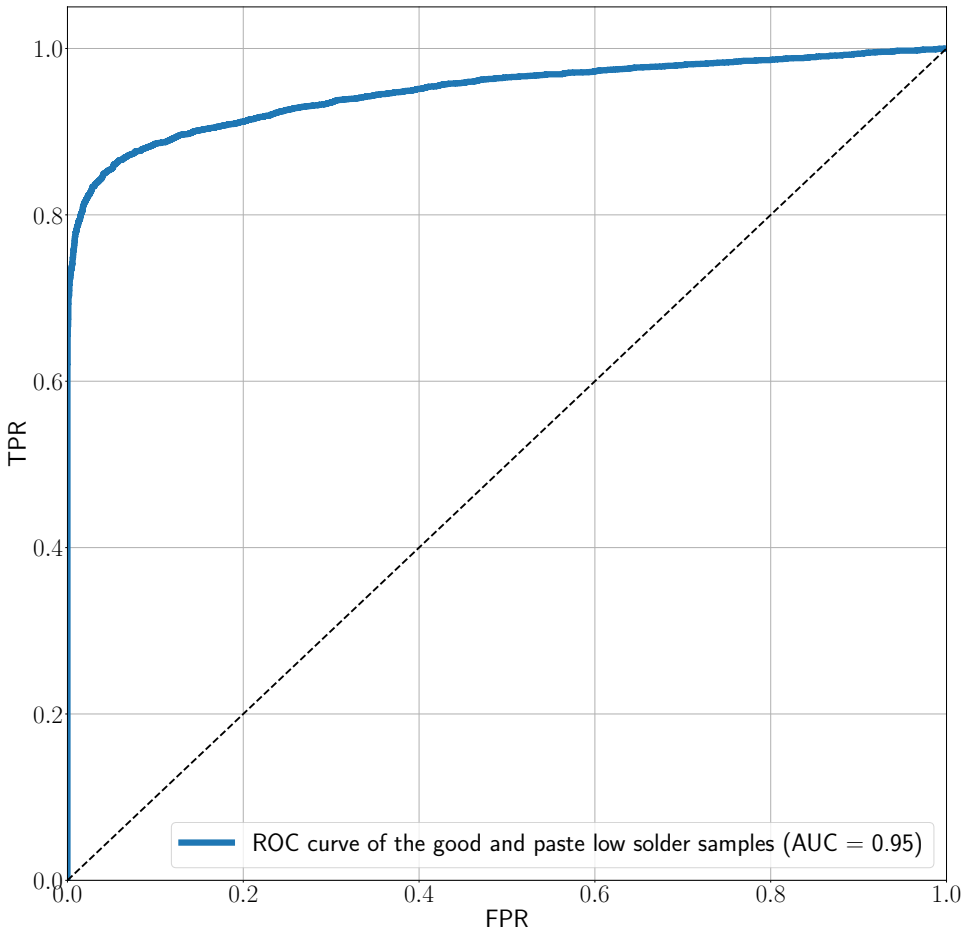


Figure 3.2: Example ROC curve of an Autoencoder on the good and paste low samples from the solder images dataset

At the beginning, the ROC curve is very close to 0 FPR, as there are mainly good samples near the smaller loss values. However, as the threshold value gets larger, the TPR values spike up really fast while the FPR metric only creeps up very slowly. This behaviour results in an ROC curve that always sticks to the top-left corner, near the ideal $(0, 1)$ point, meaning that based on the reconstruction loss of the Autoencoder, the good and paste low samples can be detected with large success. The AUC value also signals a great performance, as with 0.95, it is close to the perfect value.

All in all, based on the above explanation, the AUC metric can be efficiently used in the evaluation of the Anomaly Detection problem. It can also summarize the performance of each model in one single value, instead of having to compare the class distributions visually.

3.3 The baseline method

To provide a baseline approach which can then be used to rank the other methods, in this section, a very simple method is introduced to detect the Anomalies in the Solder Images Dataset. The approach is the simplest autoencoder, as it always produces the average of all training samples as its output. To demonstrate how this output image will look, Figure 3.3 is provided where the average of the good solder samples is shown.

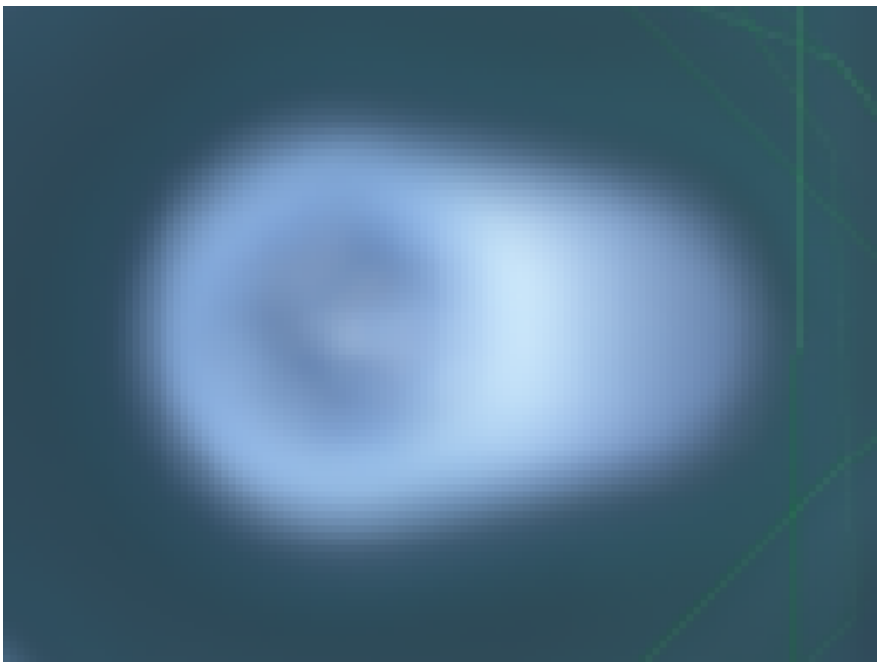


Figure 3.3: Output of the Average Autoencoder on the good solder samples

Based on the output image, it is easy to tell, that the reconstruction quality of the approach is very poor, however, due to the well conditioned samples (same overall position, colors and shapes for all samples), the output still resembles a soldering. After producing the mean image, it is used to calculate its difference from every other sample, which values then will be used to determine which sample is correct and which has some production anomalies.

This approach serves as a great baseline, because it is very simple, needs no training time and still uses some strategy to detect the faulty samples from the rest. The results of the Average Autoencoder will be used as comparison to the other introduced methods, as it will help understand the difference in performance between them better.

Chapter 4

The Dataset

In this chapter, one of the key aspects of the Anomaly Detection process is discussed: the dataset that will be labeled. The features of different datasets are very crucial when it comes to understanding the approach they need in order to obtain the highest performance possible.

The most basic feature of any dataset is the data samples' size, pixel color (color channel size) and class distribution, as these directly affect the parameters of the detection pipeline (for example, the size and capacity of the neural network). Another aspect is the variance of the samples: how much one data point differs from another one, if the pair is from the same class. This usually depends on the method which the data is obtained or if there were some preparations regarding the samples before the initial annotation. Furthermore, the most important feature to consider is the appearance and characteristics of each dataset class: this will rank each category based on their difficulty when it comes to detecting their deformities and defects.

Over this chapter, the dataset used in this thesis will be introduced with emphases on the aspects detailed above. For each class, there will be also some example samples shown to help the Reader better understand their core features and overall appearance.

4.1 The Solder Images Dataset

In this section, the Solder Images Dataset, which is used throughout the thesis, is introduced and analysed in order to provide useful information for the following chapters. This dataset is based on a real industrial application and it is important to delve into the details of the samples, so that when it comes to Anomaly Detection, the process can be efficiently parameterized and will achieve overall better performance.

To begin with, the dataset consists of images of electrical solderings that are on a circuit board. The circuit board has several soldering locations and the camera system takes a picture of the whole board. To better analyze each location, first, it is needed to crop out the important picture regions and then transform them in a way so that all samples (crops) have the same general characteristics: the soldering location is always in the middle, it is rotated, so that it lays horizontally and each image has the same size (192×96 RGB pixels).

After the necessary preparations, it is important to look at each different class individually and analyze their describing features and characteristics. The soldering itself can be broken down into 4 parts that play a role in defining the class aspects: the hole, the pin, the

soldering paste and the board. The outline of the hole should not be seen in the case of a good soldering as the paste covers it entirely. The pin is usually located in the middle of the hole and it should be surrounded with soldering paste to hold it in place properly. The board is the background for each sample and the soldering paste can overlap it when it flows from the pin location.

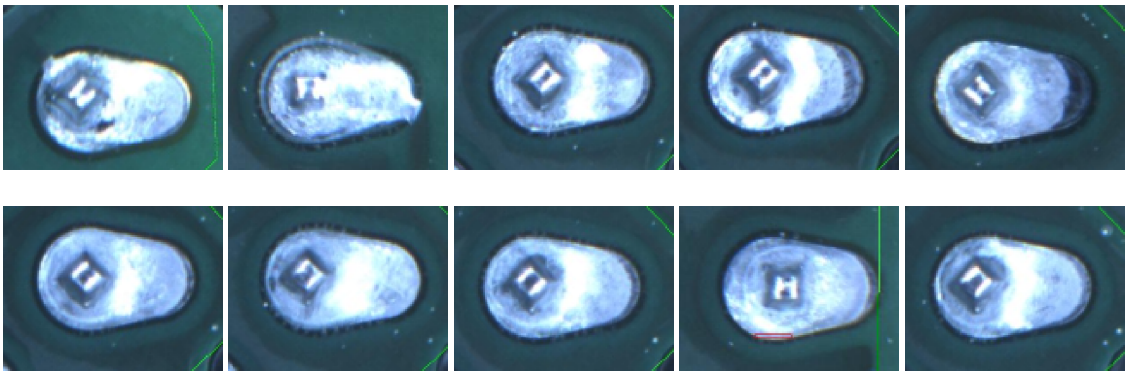


Figure 4.1: Examples for the good soldering class

The good samples are the first to be examined since all images have the same basic layout and based on them the other classes can be differentiated much more easily. Starting with the pin, it is covered in soldering paste to the point that its peak its still visible, while the entire hole is filled with paste, so that it provides good mechanical stability. The paste itself flows to the designated pad on to the board thus accommodating the necessary amount of soldering paste. Figure 4.1 shows some examples for good cases of soldering where the aforementioned characteristics can be observed.

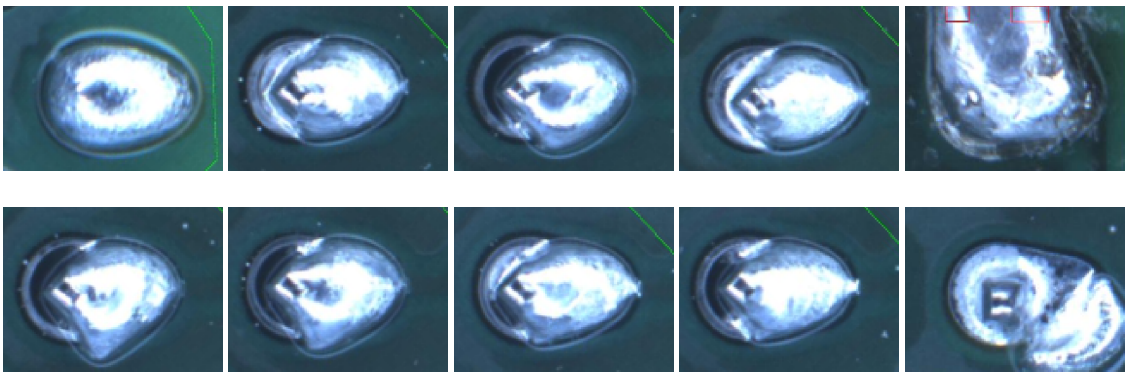


Figure 4.2: Examples for the "blob" soldering class

The next class is called "blob" because the excess soldering paste builds up like a sphere around the pin. These samples can be discovered by looking at the pin location, as usually with too much paste, the pin itself is not visible anymore. Moreover, the paste can flow around the hole and create a much more widespread shape around it, which is also easy to detect in comparison to the good samples. Examples for the blob class can be seen on Figure 4.2, where the main features of the class are presented.

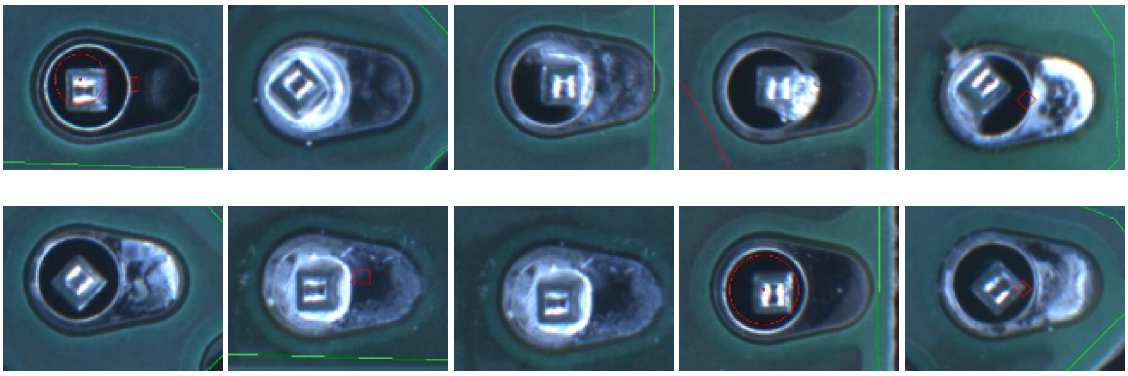


Figure 4.3: Examples for the low paste soldering class

The third class in the dataset is the exact opposite of the "blob" class: there are samples where there was not enough solder paste rationed to the board during production. Due to this, these samples usually lack the necessary bonding between the hole and the pin and with the board itself. However, it is easy to detect them visually, as the hole is partly or in some cases entirely visible, so around the pin there is a darker color present. The level of soldering paste deficit can vary vastly, but usually based on the color difference mentioned before, it is easy to distinguish these samples from the correct ones. Different examples where there is not enough soldering paste can be seen on Figure 4.3.

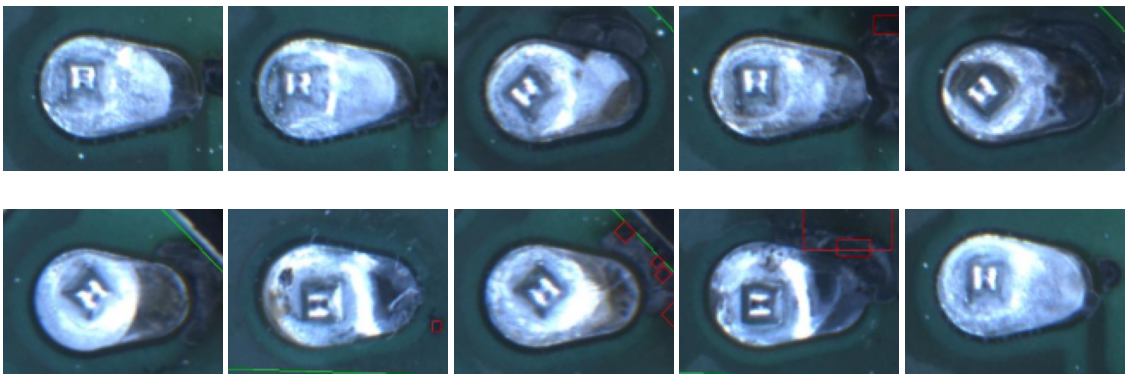


Figure 4.4: Examples for the burnt soldering class

Finally, the last class contains the burnt samples, where during the soldering process, excess heat was applied to parts of the board, thus unintentionally damaging it. In these cases, although the soldering can have enough paste and a good shape over the hole and around the pin, the board itself has burn marks around or directly on the pin location. Comparing this class to the other ones, it can be safely said, that detecting the burnt samples is the hardest task, as they usually do not have a major difference to the good samples (like for example the lack of soldering paste). The difficulty of detection also increases in reverse with the extent of the burnt area on the board: if the damage overlaps the soldering then it can be detected more easily, however, if it only covers a small part of the board, it can become several magnitudes harder to detect the anomaly correctly. Another source of difficulty is the visual appearance of the burnt area, since in some cases,

it can be very subtle due to its small size or lighter level of burn damage, making it seem similar to other lightning phenomenon, not to mention, that it can also appear almost in every location in the examined area. Furthermore, this class is highly underrepresented among the samples, so it is that much harder to collect a dataset which covers every feature described here. Based on these features, the burnt samples are the hardest to identify correctly in this dataset, so this class will be the most emphasized during the Anomaly Detection process with a more in-depth analysis provided during evaluation of the obtained results. To help understand the above mentioned difficulties, some examples of this class are shown on Figure 4.4, where the burnt areas can be seen with a darker contrast around the solder location.

To sum up the Solder Images Dataset, it is a very diverse dataset regarding its samples: it has some easy to identify defections (for example the lack of soldering paste) while some deformations or anomalies are very hard to detect (mostly the burn marks). During the detection process, the emphasis will be put on the harder to detect deformities as those are the samples which the Autoencoder network has trouble identifying, due to them being smaller size or having less contrast. This is because the output images of the Autoencoder are blurred, thus overshadowing the smaller differences that come from correcting the burn damages. As the DDPM network produces images with much higher quality, it is expected, that it solves this problem by eliminating the noise coming from the reconstruction.

Chapter 5

Networks

After the introduction of the dataset, the next important component of the Anomaly Detection process is the Neural Network architecture that will produce the output images, which then can be compared to the original samples. In this way, this step is also crucial, as it sets the basis for the success of the later performed comparisons that will be based on the reconstruction quality of the trained network. For this purpose, in this chapter, the two architectures used in thesis are presented and analysed: one is a true and tested approach, the Autoencoder, while the other one is a relatively new structure, the Denoising Diffusion Probabilistic Model (DDPM). For each one, the center point of interest will be the reconstruction quality, as that will determine the ability to detect the more difficult anomalies: in the case of the Autoencoder, the output images are blurred, making it more difficult the detect the differences coming from correcting the anomalies, while in the case of the DDPM model, the output image quality is much higher which should solve this issue.

5.1 Autoencoder

The first Neural Network architecture is one of the most common choices in Deep Learning when it comes to outlier detection: the Autoencoder model. It consists of 2 components which work together in a pipeline producing an embedding of the original sample into a latent space where the data size is smaller, while also guaranteeing that no important feature is lost during the conversion process. A visual representation of the model can be seen on Figure 5.1, where the exact layer structure can also be observed. In this thesis, I have used the modified U-Net Model [12], which was originally used for performing image segmentation tasks.

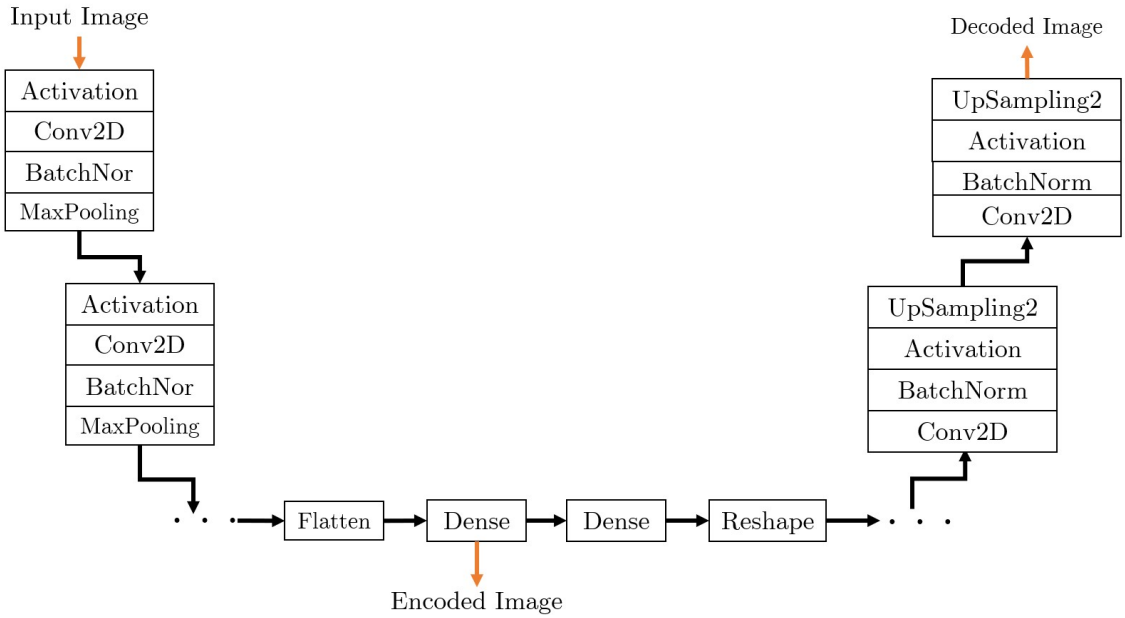


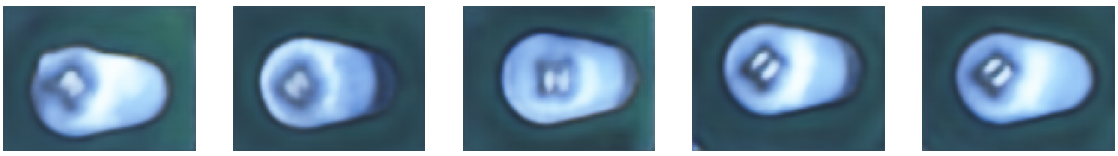
Figure 5.1: Architecture of the Autoencoder model

The input is first given to an encoder module that transforms the original image size to a much smaller matrix with higher channel number with the help of Convolutional Layers. Then the resulting tensor is flattened into a vector that will represent the original image in the latent space. This vector's length, which is also the code length, can be adjusted with the help of a Dense Layer, where the neuron number sets the size of the output. In this way, much of the unnecessary information is thrown away from the original sample until we are only left with the key components that help identify the data.

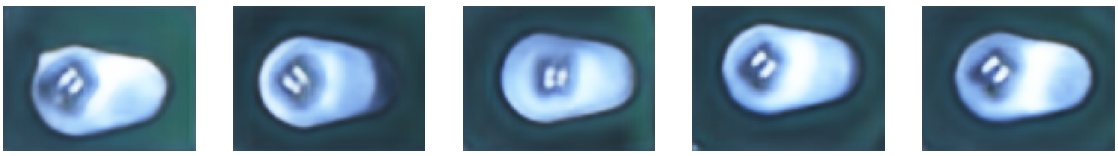
The next part of the model is the decoder, where the produced code is first converted back to a tensor identical to the output of the encoder's last Convolutional Layer. Then it is up-sampled through several Convolutional Blocks which consists of the same layers as in the encoder part, but with reversed order and the goal of enlarging the tensor's size (while also decreasing the channel number) until we get an exact match with the original image. Using this output, we can measure the quality of the encoding and decoding process by comparison with the original sample, for example with a mean-squared loss across the pixels of the two images. Thanks to this method, during training, the Autoencoder will learn a representation that only encodes the necessary features from which the original image can be decoded with as small error in quality as possible.

After the brief introduction, we should focus on some of the parameters that can affect the quality of the output images: these are the size of the code vector and the overall depth of the network. The code size basically affects the structure of the whole embedding, as it creates a bottleneck that only allows information deemed crucial to the reconstruction process. To visualize the impact the code length has on the output images, Figure 5.2 shows the output of models with different code size compared to the original samples.

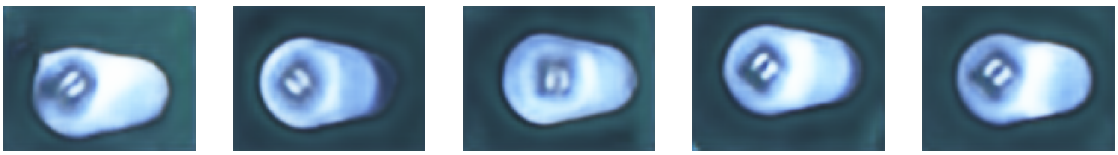
(a) Original Samples



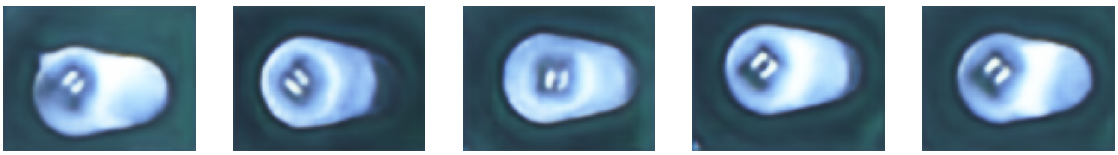
(b) $N = 4, MSE = 0.004682$



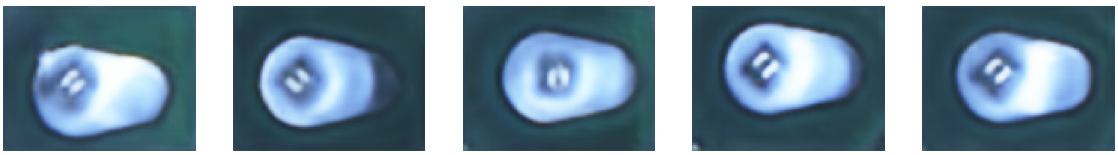
(c) $N = 32, MSE = 0.003639$



(d) $N = 64, MSE = 0.003493$



(e) $N = 128, MSE = 0.003295$



(f) $N = 256, MSE = 0.003557$

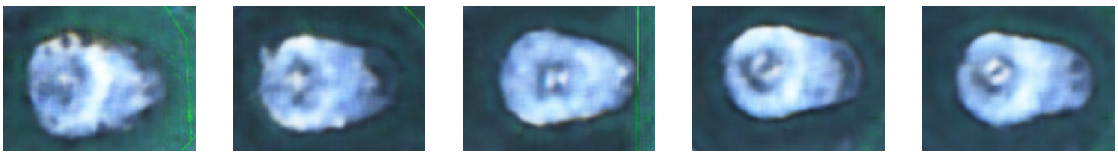
Figure 5.2: Output images of different Autoencoders with N code size and the average MSE loss shown

From the presented samples and MSE values, it is clear that by increasing the code size of the Autoencoder, the reconstruction quality gets better, however, not by a large margin. The main change can be seen from code size 4 (see Figure 5.2b) to 32 (see Figure 5.2c) as both the MSE values and the quality of the output images show a great improvement. After this however, the difference in reconstruction quality is hardly noticeable and even the MSE values change much more slowly between code size 64 and 128 (Figures 5.2d and 5.2e respectively) than in the first case. At size 256 (Figure 5.2f), the performance even becomes worse, as the MSE value starts to become larger, ending up around the same threshold as the previous, smaller code sizes. From this experiment, it is clear that –except at extremely small code lengths–, the code size of the Autoencoder does not affect

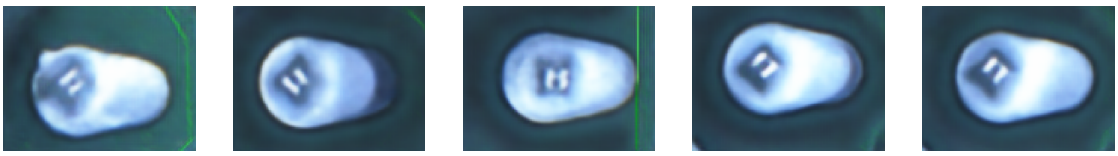
heavily the reconstruction quality of the output images. It is important to keep in mind, however, that the code size should not be larger than the overall size of the vector before flattening, as in that case, the vector size will be the main bottleneck of the network, not the code length.

The second component which can affect the reconstruction quality is the network's layer and parameter count, as they define its overall capacity to learn the features of the samples in the dataset. In the case of the U-Net architecture, the network capacity can be modified in two ways: by adding more blocks to the encoder and decoder module (increasing depth) or by expanding each Convolutional Block to have more layers inside (increasing block size). To demonstrate the effects of them, Figure 5.3 is provided, where the outputs of several Autoencoder networks with different depths are shown.

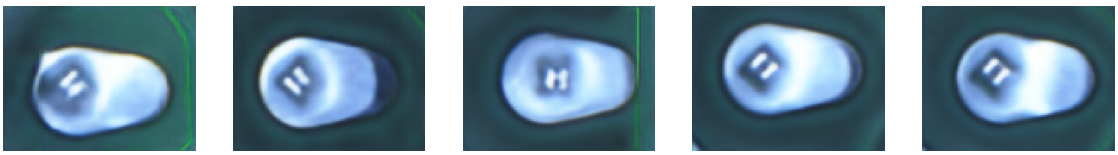
(a) Original Samples



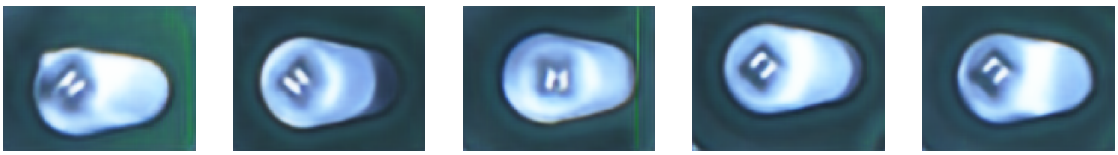
(b) $D = 1, MSE = 0.012891$



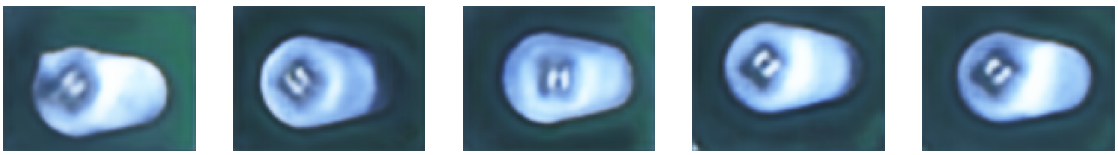
(c) $D = 2, MSE = 0.003298$



(d) $D = 3, MSE = 0.002547$



(e) $D = 4, MSE = 0.002430$



(f) $D = 5, MSE = 0.003519$

Figure 5.3: Output images of different Autoencoders with D depth, 128 code size and the average MSE loss shown

Compared to the code size pictures above, the Autoencoder with depth size 1 produced much worse quality output images both in visual appearance and MSE values (see Figure 5.3b). Increasing the size of the network improved the performance by a large margin, as the MSE values became 1 magnitude smaller (see Figure 5.3c), which is noticeable on the reconstruction quality as well. After this, adding more blocks to the Autoencoder still improved the performance, however the progress slowed down: between depth 3 and 4 (Figures 5.3d and 5.3e respectively) the MSE values only got a little bit smaller. Looking at the results for depth 5 (Figure 5.3f), the MSE values got larger, producing similar or even worse outputs than that of depth 2. This behaviour is very similar to what we have seen at the code size figures, as by continuously increasing the size of the network, it is

becoming much harder to train the model effectively with the same setup resulting in the same performance, as if we had a network which has fewer blocks. This occurrence can be explained by looking at the model structure: when we are adding more and more blocks, the parameter count of the network also increases exponentially, due to always increasing the number of filters (last channel) of the tensors, making the network learn slower. However decreasing the filters too much can also lead to another problem: the size of the last vector before flattening will be smaller than the actual code size, making it the bottleneck of the network, similarly to the case with the code size analysis above. Based on these findings, the conclusion is, when choosing the size of the network, the ideal value should be not too small and also not too large in order to achieve the best possible result with the available resources.

To sum up, the main parameter that directly impact the reconstruction quality of the Autoencoder is the overall network depth, since not enough model parameters can lead to crucial information loss and a network with too few layers and parameters does not have the capacity to learn the detailed features of the dataset. It is important to find the optimal value for each parameter in order to achieve a reconstruction quality as flawless as possible, but is important to keep in mind that in every case it should be a different value, as each dataset has its own number of important characteristics that are needed to be learned during training. For example, the code and network size for the Solder Images dataset needs to be larger than for a dataset with smaller images or having less detailed features that define their classes.

5.2 Denoising Diffusion Probabilistic Model

The main network architecture of this thesis is the Denoising Diffusion Probabilistic Model (DDPM) [8] which is a quite recently emerged model in the field of Deep Learning. It is being used for solving more and more tasks, but in order to understand its structure and how can it be applied efficiently in the case of Anomaly Detection, it is important to detail its basic features.

As the “denoising“ name suggests, the main essence of the DDPM network is the ability to produce images from pure noise that resemble the training data, which the model learned. To achieve this, the model is performing two processes during training: the first one transforms the original image to noise, while the second one reverts this and tries to produce an output as similar to the input sample as possible. The key for success with the “noising“ process is the gradual nature of it, as the input picture is not transformed right away to pure noise, but rather has noise added to it in several steps, creating a transition. This is also done during the reverse process, so starting from the pure noise, in each step, more and more of the original image is reverted until, in the end, the output looks just like the input sample. Adding noise to the images this way makes the learning process possible, as the added noise can be described with a probabilistic distribution function and also the results can be evaluated after every step. Moreover, this is also true for the reverse process, so the distribution of the images generated from noise can be compared to those of the forward process at every step, making the learning process much faster and efficient. To visualize these two processes, Figure 5.4 illustrates the main steps of the noising and denoising workflows.

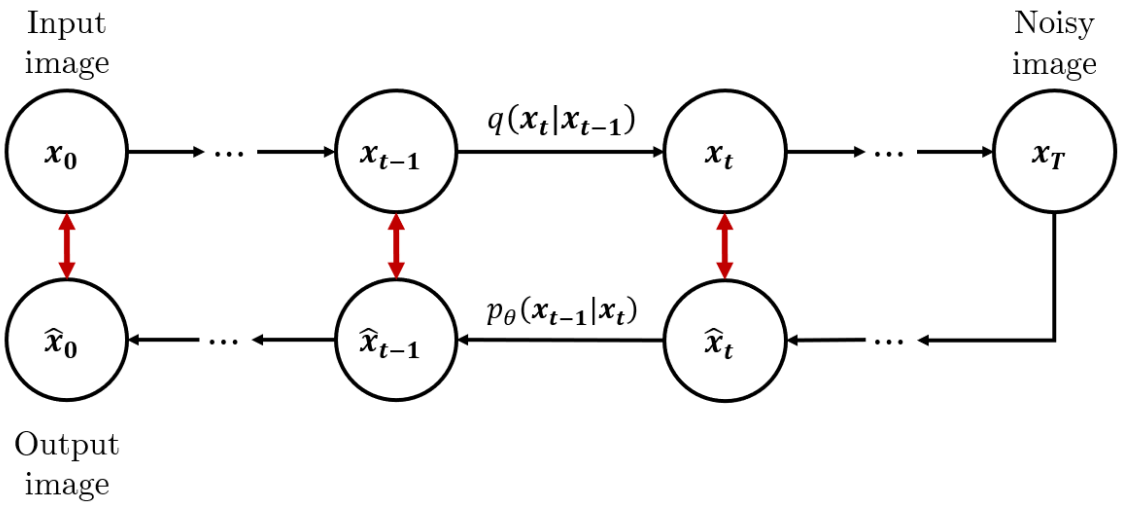


Figure 5.4: Noising and denoising processes of the DDPM network

After introducing the general purpose of the two processes of the DDPM architecture, they should be detailed, in order to understand the relevant parameters of the model for the task of Anomaly Detection. Starting with the forward process, it modifies the input image ($\mathbf{x}_0$) through several steps adding Gaussian noise to it gradually. Since the nature of the Gaussian noise can be described with a probability distribution, a function can be assigned to this process with the following expressions:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{5.1}$$

where t is the indicator of the current step and $\mathcal{N}$ is a normal distribution with a given mean and covariance matrix. Based on this formula, one of the influential parameters of the network is the diffusion step size (T), since it affects how much noise is added to the original image. Exploiting the fact that the distribution of each step is independent from the others, the entire forward process can be formulated with a single expression:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{5.2}$$

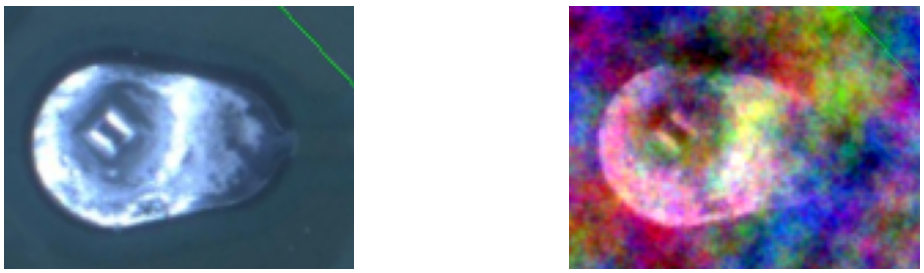
As for the reverse process, the same principles can be used: we can define a probability distribution that describes the nature of each generated image throughout the process. The exact formulas of this function can be seen here:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \mathbf{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{5.3}$$

$$p_\theta(\mathbf{x}_{0:T}|\mathbf{x}_0) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{5.4}$$

where μ_θ (mean values) and $\mathbf{\Sigma}_\theta$ (covariance matrix) are the distribution parameters for each t step that the network needs to learn in order to be able to generate images whose distribution and appearance is similar to the training samples.

Defining the processes this way makes it easier to learn the parameters of these distributions, as we can compare the results for each reverse step ($\hat{\mathbf{x}}_t$ is the estimated image) with the original distributions using a loss function based on the KL divergence formula. After the training is finished, the network is capable of producing a brand new image from seemingly pure noise, since it learned the parameters of each distribution in the reverse process. To further enhance the anomaly detection performance, it was found effective to use Simplex noise [13] instead of the Gaussian, as the corruption is more structured and the denoising process will be able to "repair" those structured anomalies more effectively. Figure 5.5 provides an example soldering sample with added simplex noise to showcase how it affects the original image.



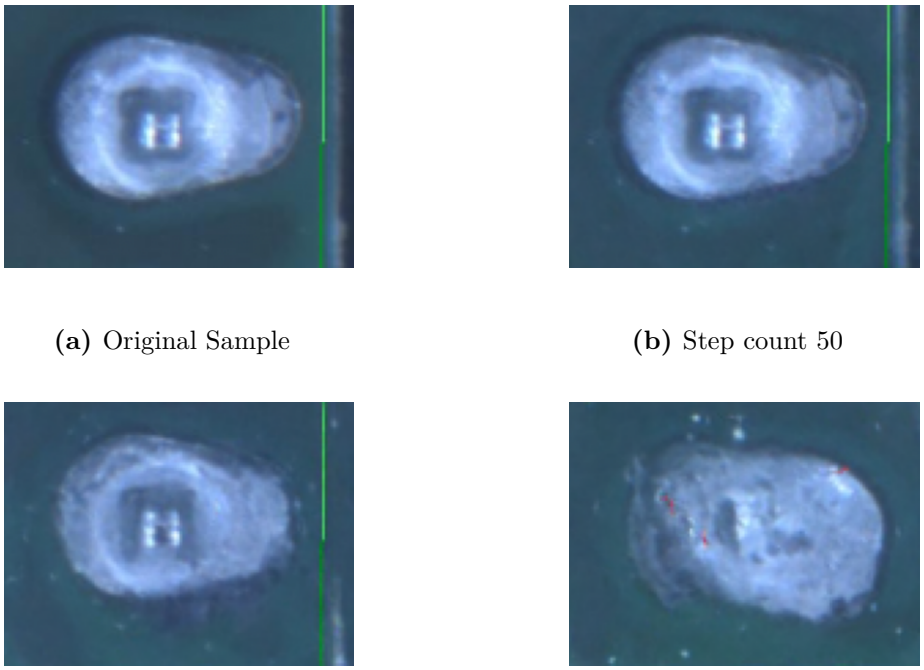
(a) Original Sample **(b)** Sample with simplex noise added

Figure 5.5: Example Soldering Sample with Simplex noise

Looking at the example image, we can still see the main silhouette of the original soldering, which is ideal for Anomaly Detection, as we want to erase only the anomalies from the picture and keep the overall appearance of the sample unchanged. It is important to mention here, that using an unfiltered simplex noise will contain some colour values that the original sample simply lacked, making the training process a bit skewed as the network can detect these more easily. To combat this, the range of the values that we sample from when constructing the simplex noise needs to be adjusted to the channel histograms of the input samples, thus limiting the available colors. In this thesis, due to simplicity, I have only used a generic simplex noise, but filtering its values can be certainly explored in further research.

As with the Autoencoder network, we also need to mention the most important parameter of this approach when it comes to reconstruction quality: the diffusion step count. This is the variable that determines how many times noise is applied to the input in the forward process and how many times it is reversed during generation. In general, it is better to train with a larger diffusion step count, as this way the network can learn the more subtle features of the samples, as well as be able to generate samples more similar to the originals. However, during inference, we need to pay attention to the diffusion step count, as it highly affects the nature of the output picture: if we choose a too small value, the output may just be the same image as the input, while in case of a value too large, the network can loose the appearance of the original sample during the reverse process and generate a brand new image, resulting in a high difference. To illustrate the impact of the

diffusion step count on the quality of the output images, Figure 5.6 shows some examples for different diffusion step counts.



(a) Original Sample

(b) Step count 50

(c) Step count 200

(d) Step count 800

Figure 5.6: Example Good Soldering Sample and its reconstructions with varying diffusion step counts

From these examples, the aforementioned phenomena can be clearly seen: at step count 50 (see Figure 5.6b), the output image is almost identical to the original one, while at step count 800 (Figure 5.6d), the reconstructed image is very distorted with the green marking line on the right entirely missing. In between these two extreme cases, at 200 steps (Figure 5.6c), the main features of the original sample are still present with a little bit of "flaking" around the border of the soldering. Based on this, the ideal diffusion step count is between 50 and 200, but we need to keep in mind, that for smaller anomalies (like the burn marks) the smaller difference can be enough due to their area and position around the soldering. However, for the other, more glaring error classes, where often the soldering itself is more damaged or distorted, a higher step count is needed, as the network has to mold the original input into a correct soldering which requires more steps overall. Based on these findings, it is advised to choose a suitable value for the diffusion step count with which both the output image quality and similarity with the original sample is high on average for all classes to achieve an overall good performance.

Chapter 6

Analysing the Results

After presenting the different Anomaly Detection methods and detailing the measurement setup, in this section, the results obtained for each sample size is listed and analysed. Each experiment is evaluated on the same validation dataset and the network parameters also stay the same to provide an unskewed basis for comparison. First, for each different error class, a joint diagram shows the AUC values of each method regarding the different sample sizes, so that it is easy to compare the different approaches and how they evolve when increasing the size of the training dataset. This way a more in depth analysis can be performed on each class to be able to really determine the nature of each approach. Moreover, as the most difficult class of the Solder Images Dataset is the burnt soldering class, the results on it are discussed in greater detail at the end of this chapter, as the main goal of this thesis is to provide a suitable approach for the harder to detect anomalies.

6.1 Results on the Blob Soldering Samples

In this section, the results of the different Anomaly Detection approaches on the Blob samples are detailed using a training dataset consisting of 5, 50 and 500 good soldering samples. The comparison diagram showcasing the AUC values of each method for the different sample sizes can be seen on Figure 6.1.

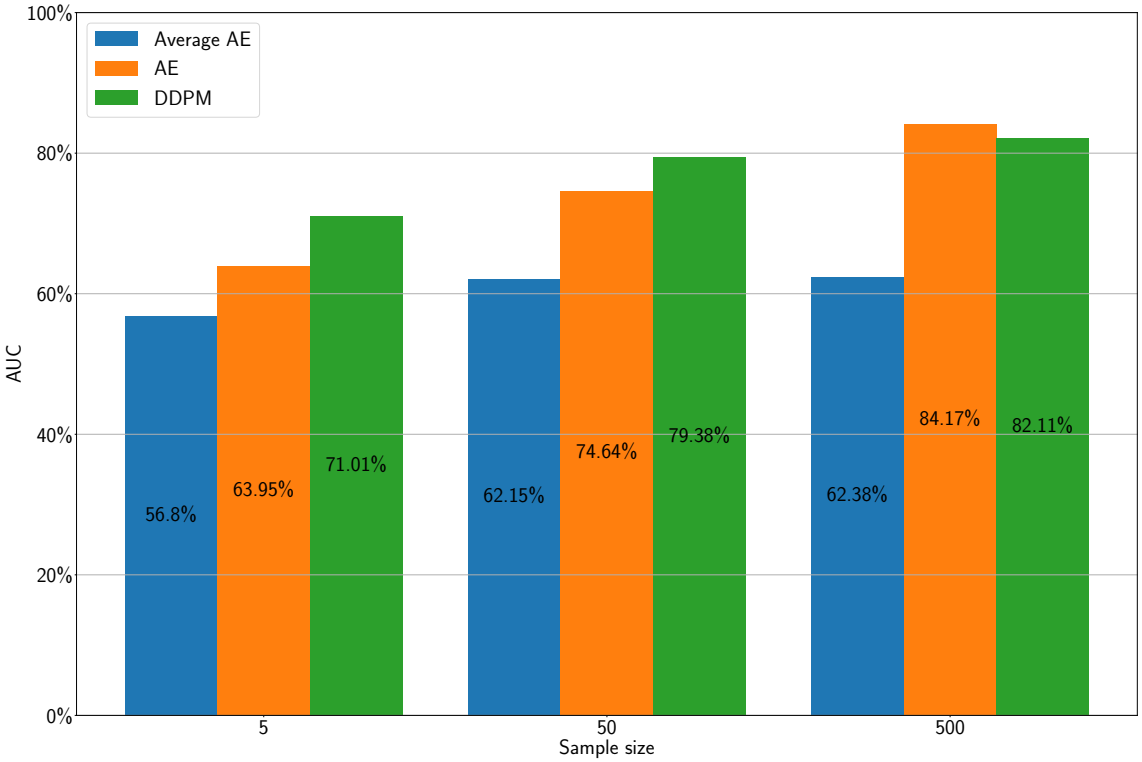
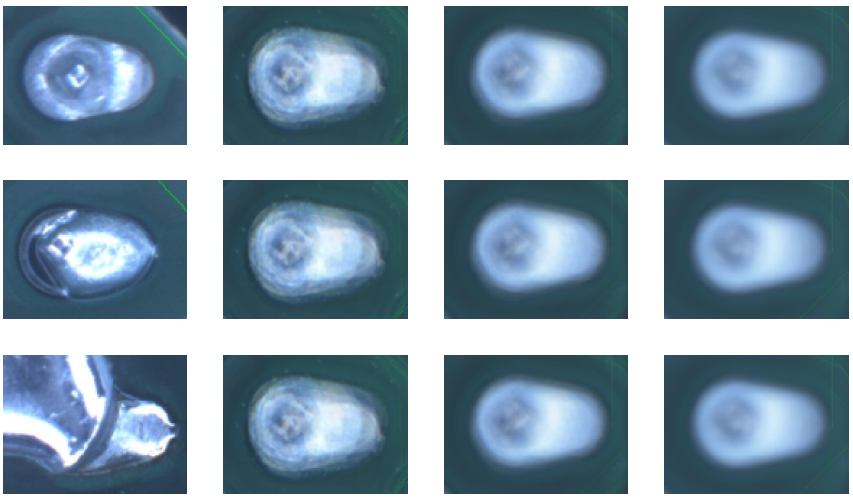
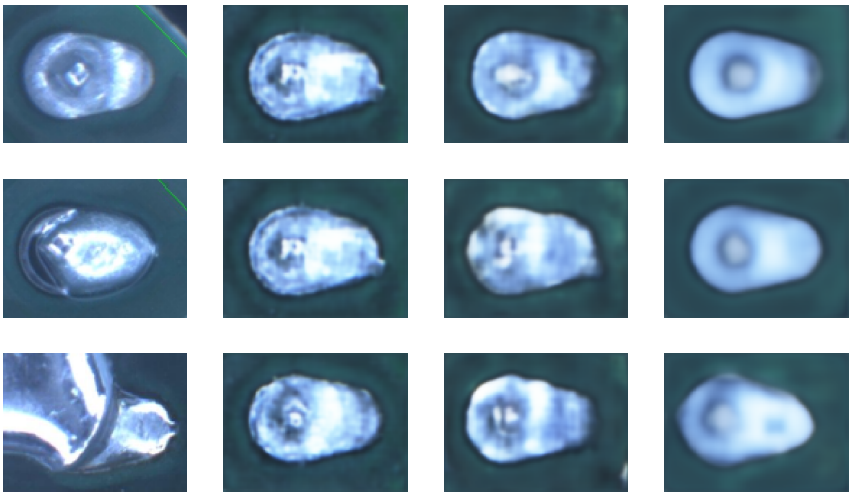


Figure 6.1: AUC values of each approach for the different sample sizes on the Blob Soldering class

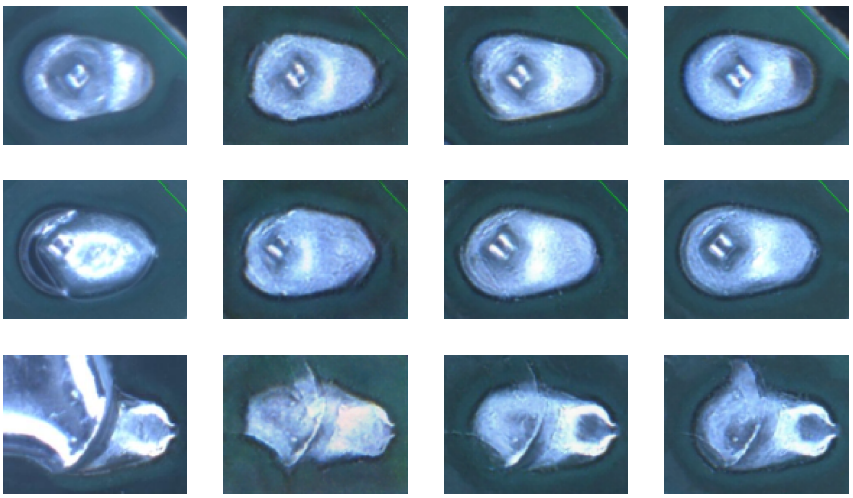
In the case of the Blob classes, the Average Autoencoder class performs the worst which is expected due to the appearance of this error class: in most cases there is not a large amount of excess soldering paste present, meaning it does not cover much of the board itself, so these samples are very similar to the good samples in the case of the averaged image. The Autoencoder performs worse at smaller samples, however, at larger sample sizes and especially at 500 samples, it outshines the other methods. This can be due to the fact, that as it trains on more and more samples, the output becomes very close to that of the Average Autoencoder, mashing the features of the training inputs together thus basically executing a more complex averaging function. The DDPM approach achieves overall great results at 5 and 50 samples, where it is the best approach out of the three methods. To provide a more in depth analysis and understand the reasons behind these AUC values, Figure 6.2 is provided, where one Good and two Blob Soldering samples (one subtle and one extreme) are shown with their reconstructed images, ordered by training sample sizes, for each method. The MSE values for these examples are also provided in Table 6.1 for easier comparison.



(a) Average Autoencoder



(b) Autoencoder



(c) DDPM

Figure 6.2: Example Good and Blob Soldering samples and their reconstructions made by the different methods ordered in increasing training sample size

Methods		Training Dataset Size		
		5	50	500
Average AE	Good Sample	0.01291	0.01374	0.01123
	Subtle Sample	0.01591	0.01766	0.01546
	Extreme Sample	0.06798	0.06735	0.06434
Autoencoder	Good Sample	0.01788	0.01473	0.00919
	Subtle Sample	0.02234	0.02368	0.01914
	Extreme Sample	0.07750	0.07941	0.06576
DDPM	Good Sample	0.01103	0.01273	0.01068
	Subtle Sample	0.02007	0.01728	0.01678
	Extreme Sample	0.04366	0.05721	0.05065

Table 6.1: MSE Values of each method trained with different sample sizes and evaluated on the example good and blob images

Looking at these reconstructed images, it is easy to see that the output of the Average Autoencoder (Figure 6.2a) gets more blurry as more and more training samples are added, but this does not affect the detection process, as the MSE values only change a little between increasing the sample size. It is important to note however, that the subtle sample is in closer range of the good sample at 5 sample size, making it harder to detect until 50 samples, where there is a bit larger gap between them, which attributes to the slight increase in AUC values.

In the case of the Autoencoder (see Figure 6.2b), first at 5 and 50 samples, the output images are relatively clear, which is because it reconstructs the training images it was trained on, as it can learn their finer details due to the small data size, resulting in overall higher MSE values for the error samples. However, by increasing the sample size, the output becomes similar to that of the Average Autoencoder, as it is only able to reconstruct the overall appearance of the training samples with the same code size and model depth, essentially blurring their features together at inference. As for the difference in MSE values, as the training sample size becomes larger, the gap between the 3 samples also widens, which is why the Autoencoder can reach higher and higher AUC values as the sample size increases.

For the DDPM, the output images (Figure 6.2c) at the smaller sample sizes are very close to a good sample except the extreme blob image, where part of the paste overflow is kept, thus resulting in a lower MSE value than in the case of the other methods. As for the extreme sample, although the output image still looks somewhat like a blob sample, the MSE value compared to the good sample is increasingly larger, which means that the method can still detect these extreme cases well. Increasing the diffusion step size might appear as a solution to correcting these deformities, but it has to be kept in mind, that by running a deeper diffusion at inference, even the good samples will change much more (see Figure 5.6 for reference), which will attribute to higher MSE values for all samples across the dataset, keeping the performance about the same or even making it worse.

6.2 Results on the Paste Low Samples

After analysing the Blob images, we can take a look at the Paste Low Samples: Figure 6.3 shows the AUC values for each method and each different training sample size. For this class, the AUC values are overall higher due to the nature of the Paste low images: the lack

of soldering paste compared to the good pictures causes a higher average pixel difference, meaning the two distributions can be separated more precisely with less effort overall.

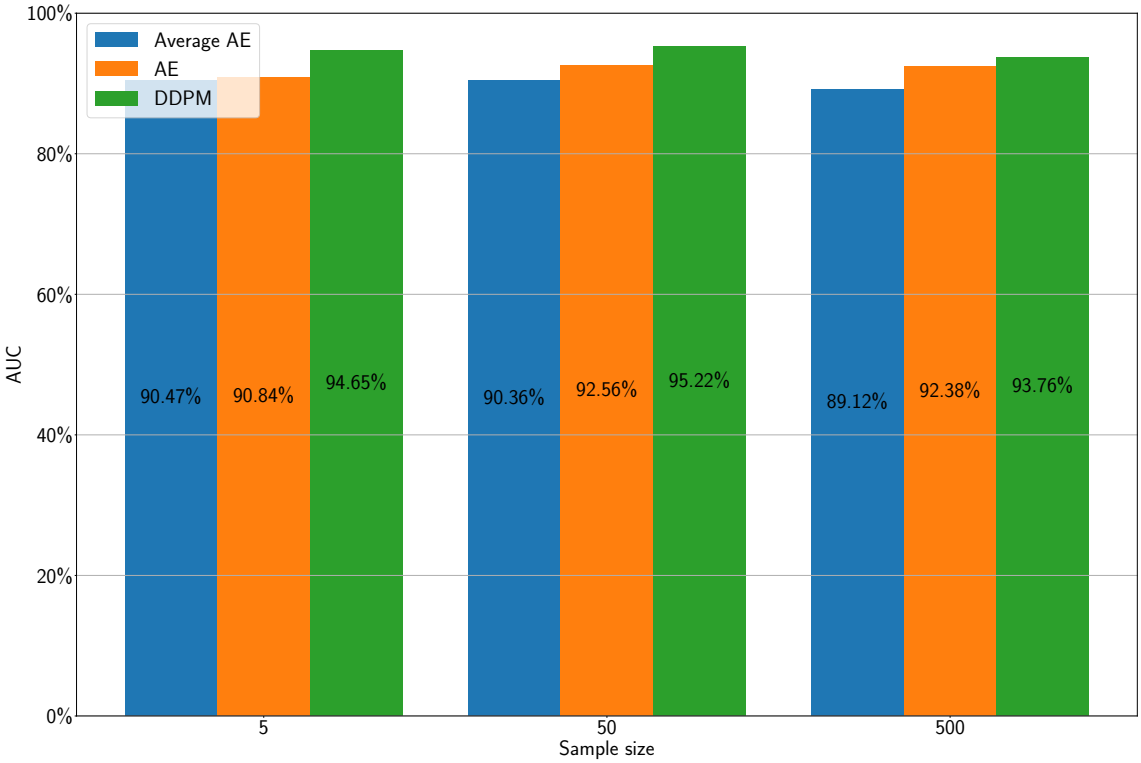
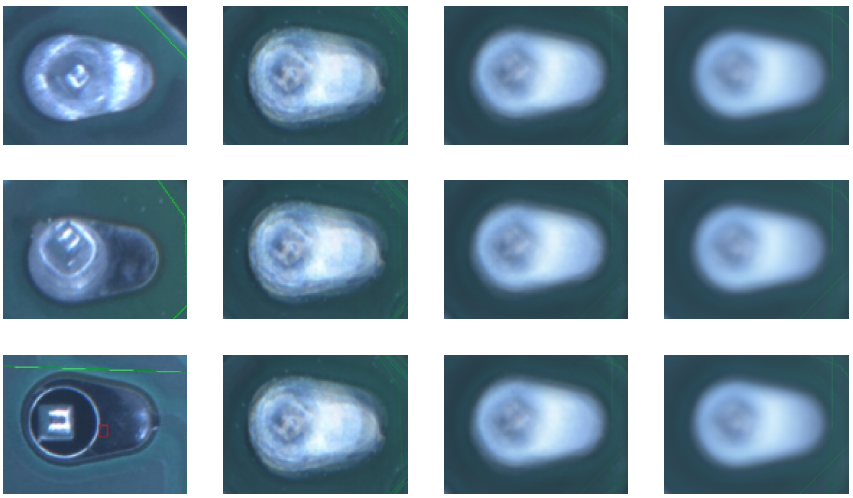
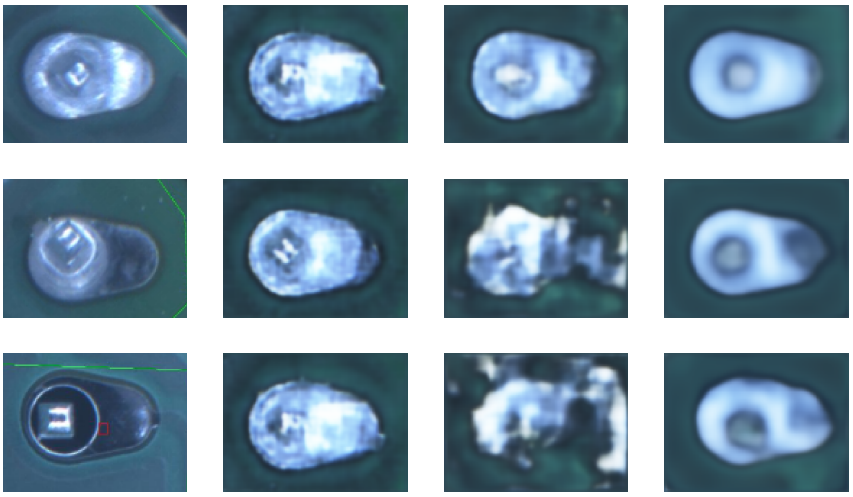


Figure 6.3: AUC values of each approach for the different sample sizes on the Paste Low Soldering class

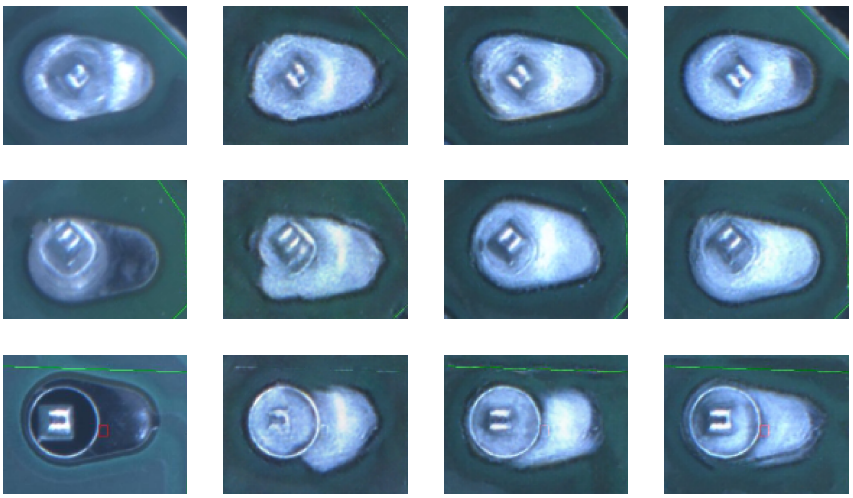
As expected, all of the methods performs overall much better, with the DDPM network being the best across all sample sizes, outshining the Autoencoder and the Average Autoencoder based methods. To visualize these values, Figure 6.4 provides one Good and two Paste Low example samples with their respective reconstructed images produced by the methods. The first Paste Low example is a more subtle sample, with some soldering paste present around the pin, while the second one is an extreme case, where there is almost no paste present on the image. To provide a better comparison Table 6.2 is also provided, where the MSE values of each reconstructed example image for every method can be seen.



(a) Average Autoencoder



(b) Autoencoder



(c) DDPM

Figure 6.4: Example Good and Paste Low Soldering samples and their reconstructions made by the different methods ordered in increasing training sample size

Methods		Training Dataset Size		
		5	50	500
Average AE	Good Sample	0.01291	0.01374	0.01123
	Subtle Sample	0.03407	0.03717	0.03204
	Extreme Sample	0.07228	0.0715	0.06475
Autoencoder	Good Sample	0.04642	0.01432	0.00810
	Subtle Sample	0.02262	0.03120	0.01408
	Extreme Sample	0.09050	0.07199	0.03062
DDPM	Good Sample	0.01103	0.01273	0.01068
	Subtle Sample	0.02922	0.04484	0.03616
	Extreme Sample	0.06084	0.06133	0.05084

Table 6.2: MSE Values of each method trained with different sample sizes and evaluated on the example good and paste low images

Looking at the reconstructed images, the above discussed results can be explained for every training sample size. Starting with the Average Autoencoder's images (Figure 6.4a), they are almost the same across the larger sample sizes, meaning that its performance should not change drastically which is reflected in the results. As mentioned before, since most of the Paste Low samples are much darker in color in contrast to the good samples, the average image of the training sample will achieve a high difference in most cases, which is backed up by the MSE values, as there is a constant gap between the good and paste low example images.

In the case of the Autoencoder (see Figure 6.4b), the sample quality at 5 training samples is very high, as it can confidently reconstruct one of the training samples, which is great for the extreme paste low sample, however, due to the output image having a lighter color even than the good sample, the Autoencoder fails to detect the subtle error sample. At 50 samples, the quality of the reconstruction for the paste low samples drops quite drastically, but it manages to get a large MSE distance for them, making the performance slightly better. This sudden drop in quality can be caused by a few new training samples at 50 sample size, which make the Autoencoder unstable in the case of the paste low samples due to their unique appearance. For 500 samples, the quality of the reconstruction is much better, although the gap between the MSE values shrinks, making the AUC value a bit less than before. Also at this sample size, the Autoencoder provides a closer reconstruction of the subtle paste low sample, which is, similar to the blob class, can be attributed to training samples whose soldering has a slightly darker color making it possible for the network to learn it.

Finally, in the case of the DDPM (Figure 6.4c), the reconstructed samples look like genuine good samples generated from the originals with small modifications across all sample sizes. This is also signalled with a constant gap between the good and subtle samples and also between the extreme and the subtle paste low samples, making the DDPM the best performing approach out of the three methods. It is also interesting to see, that the DDPM preserves the ring around the pin in the case of the extreme sample, although it does not influence the performance negatively due to its small weight compared to the color change of the entire soldering. This can be explained by the variety found in the training dataset which increases as the sample size gets larger, broadening the array of features the DDPM can reconstruct.

6.3 Results on the Burnt Samples

In the previous two sections, the results on the Blob and Paste low classes were showcased and analysed, leaving only the Burnt class to be discussed. Since this class means the most difficulty when it comes to detecting its anomalies, besides making similar observations already seen with the other classes, a more detailed and focused evaluation is needed in order to truly establish the performance of the DDPM approach. To begin with, the overall results can be analysed with the help of Figure 6.5, where the AUC values for each method and training sample size are shown.

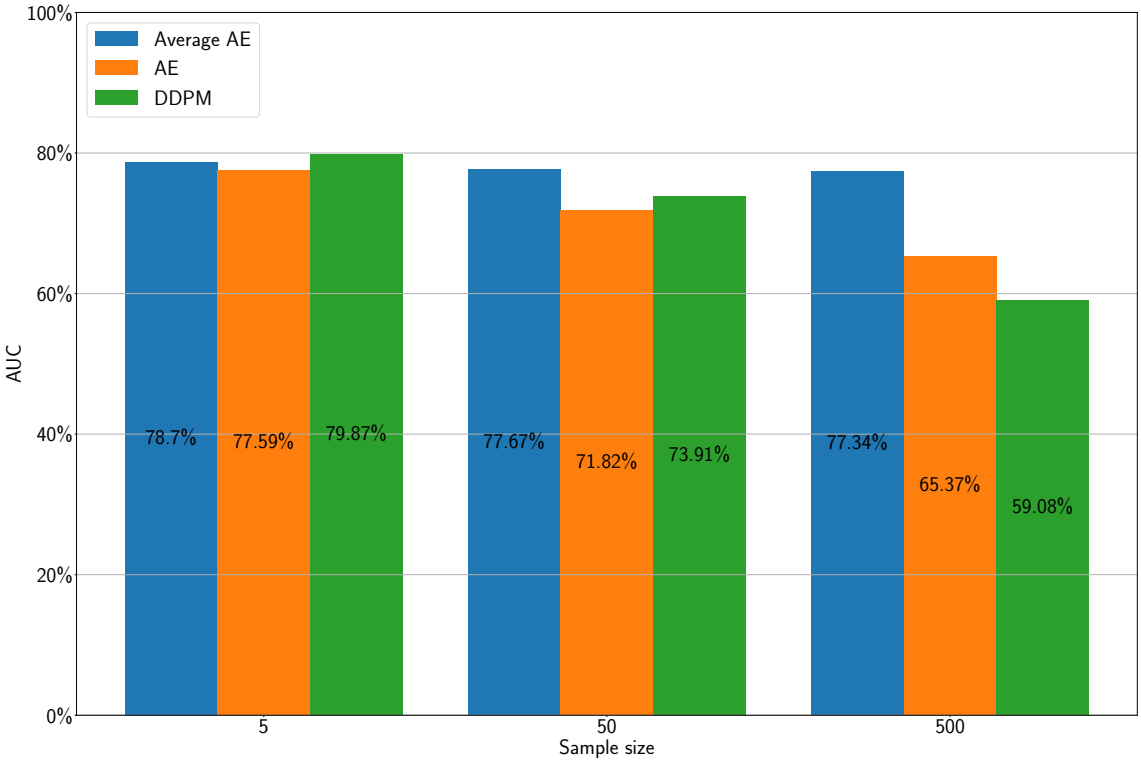
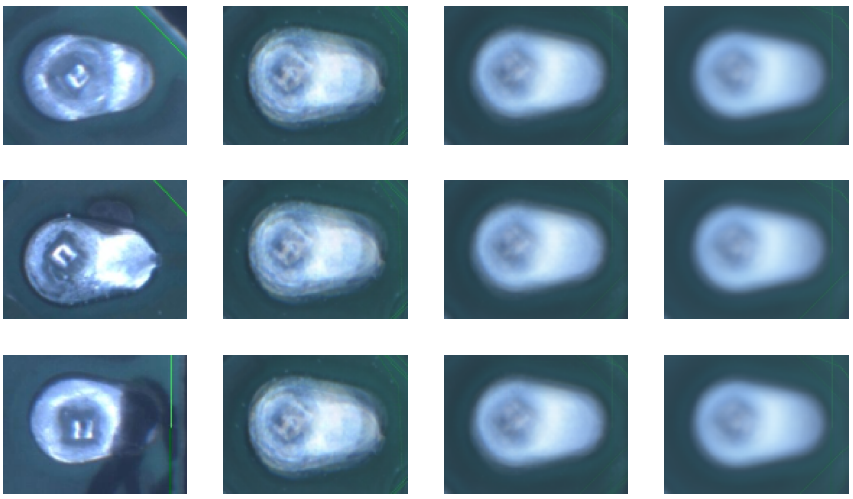
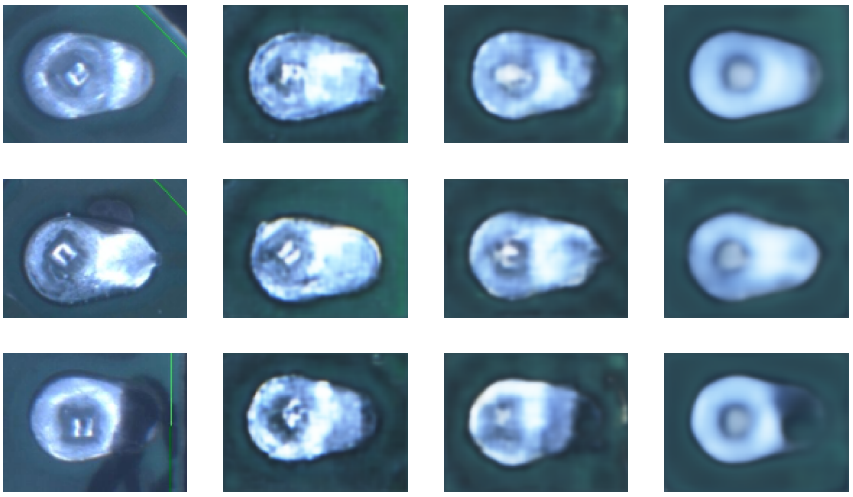


Figure 6.5: AUC values of each approach for the different sample sizes on the Burnt Soldering class

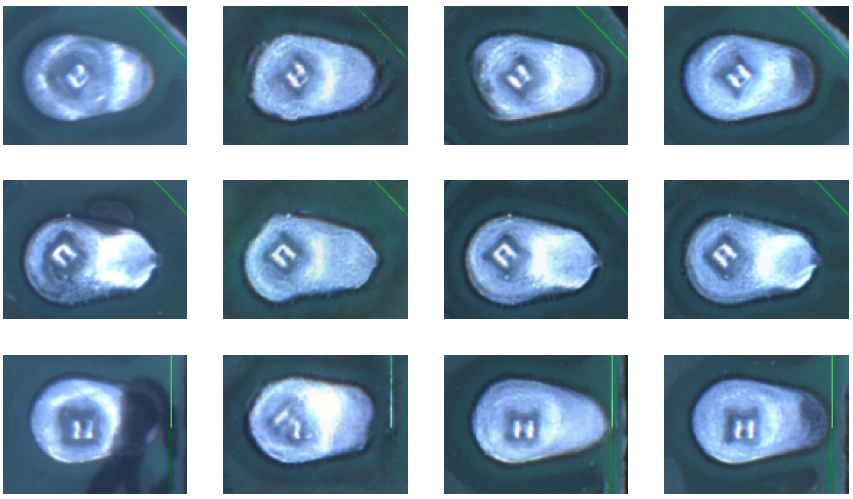
The most immediate conclusion of these results is that the DDPM approach, after performing good at smaller sample sizes, becomes the worst of the three methods, in contrast to the other classes seen before. As for the Autoencoder, it struggles to peak, highlighting the difficulty of detecting the burn anomalies, while the Average Autoencoder stays at the same level throughout the different sample sizes, proving better than the Autoencoder. To understand the causes of these results, Figure 6.6 is provided where, as with the other classes, one good and two burnt samples are shown with their respective reconstructions made by the different methods. Similarly to the previous examples, the first Burnt Sample is more subtle as the damage is small in area and has a lighter contour, while the other example is more extreme: the burn damage here is more excessive, covering even part of the soldering itself and it has an overall darker tone, making it more prominent. To aide with the comparisons, Table 6.3 is also provided, where the MSE values for each output image made by the methods are listed.



(a) Average Autoencoder



(b) Autoencoder



(c) DDPM

Figure 6.6: Example Good and Burnt Soldering samples and their reconstructions made by the different methods ordered in increasing training sample size

Methods		Training Dataset Size		
		5	50	500
Average AE	Good Sample	0.01291	0.01374	0.01123
	Subtle Sample	0.02080	0.02637	0.02400
	Extreme Sample	0.02804	0.02733	0.02385
Autoencoder	Good Sample	0.01788	0.01473	0.00919
	Subtle Sample	0.01777	0.01844	0.01201
	Extreme Sample	0.02853	0.02083	0.00953
DDPM	Good Sample	0.01103	0.01273	0.01068
	Subtle Sample	0.00999	0.00724	0.00512
	Extreme Sample	0.02304	0.02674	0.01491

Table 6.3: MSE Values of each method trained with different sample sizes and evaluated on the example good and burnt images

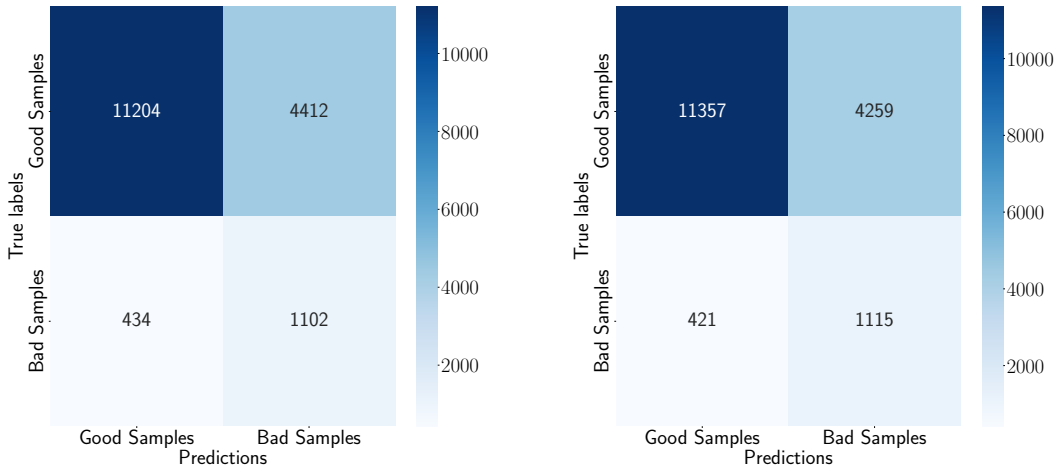
Starting with the outputs of the Average Autoencoder (Figure 6.6a), as with the other classes, the average image also holds up quite well in this case, as the subtle sample has a darker soldering color on the right, making it easier to detect due to the color difference, which is also reflected in a steady gap in the MSE values across the sample sizes, making the performance constant.

In the case of the Autoencoder (see Figure 6.6b), at 5 samples the reconstructed images have good quality and resemble the training samples which makes the detection easier. The color difference in the soldering of the subtle sample also plays a part here, as the MSE value for it is even higher than that of the extreme sample. By increasing the training sample size, this color difference still stays relevant, despite the Autoencoder trying to keep the right side of the soldering darker for both burnt samples, bringing the MSE values closer together, as the output images for the subtle sample are still too bright on the right side. It is also interesting to note here, that the Autoencoder also keeps the right side of the reconstructed extreme sample darker, which is likely due to the fact that the burn damage overlaps the soldering, affecting the output image generation.

Lastly, looking at the DDPM's output images (Figure 6.6c), we can see that at 5 and 50 samples the reconstructed images pass as good solderings with little modification to the original samples. This is also reflected in the MSE values, keeping the gap between the good and bad samples for the smaller sample sizes. As for the larger sample sizes, as the network was trained on a more diverse dataset, it keeps as much of the original images as possible (for example the board markings) with only making the burn damage disappear. Unfortunately, this affects the good sample negatively, as its MSE value becomes the highest of the three samples due to an overall brighter soldering color in the reconstruction, causing a drop in AUC values as well. This is again caused by the features that the DDPM model was trained on: the color of the example sample is probably different than that of the training samples, making the reconstruction lean to that more. Decreasing the diffusion step size might solve this issue, as it will make the model do less changes to the original image, however, in the case of the extreme samples, it will probably worsen the detection performance, as it will keep much of the larger burnt area intact, making the differences smaller. Another solution can be the supervised selection of the training samples to ensure that it will contain samples that have a lighter colored soldering.

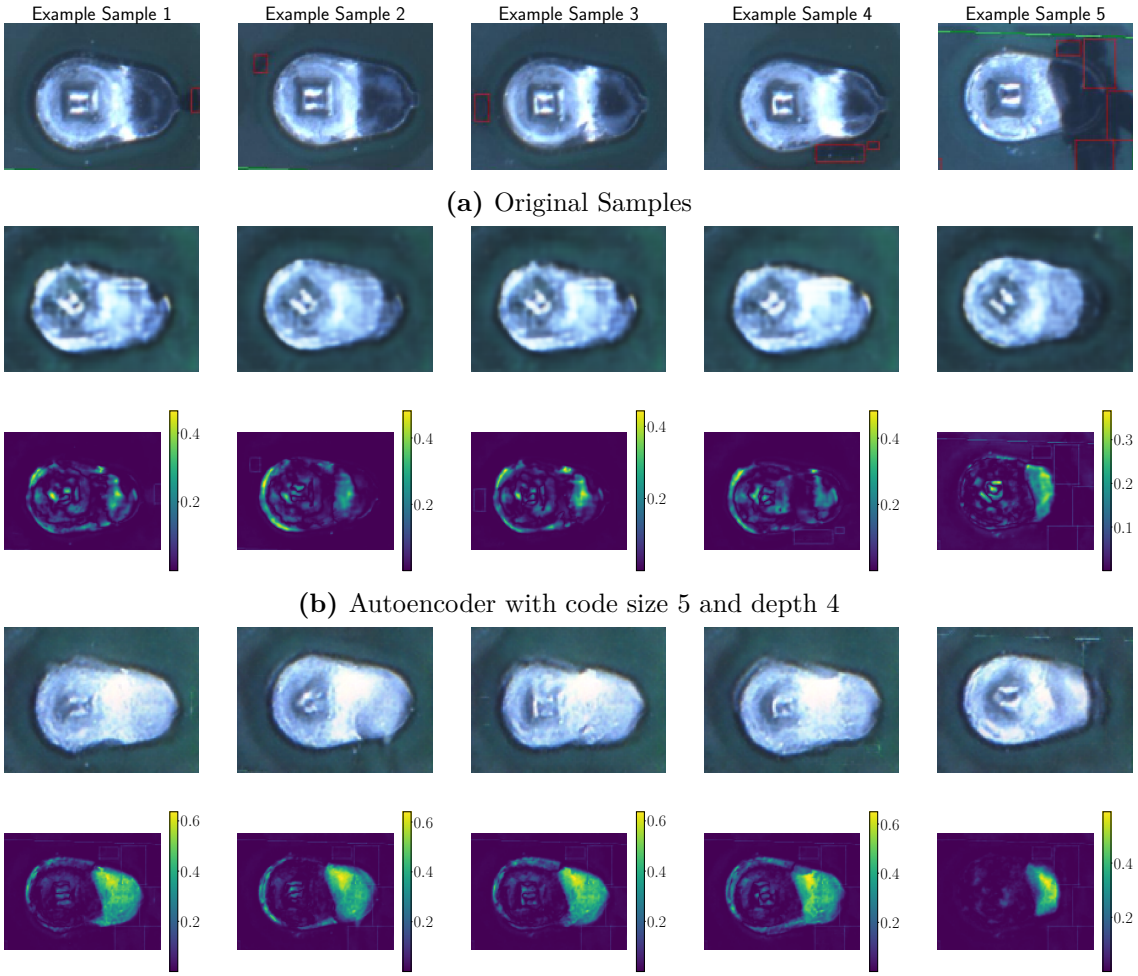
After the general analysis of the results, it seems that the DDPM approach performs slightly better than the Autoencoder at 5 and 50 samples. To provide a more in depth evaluation and to evaluate the performance of the DDPM approach for the Burnt samples

at these sample sizes in great detail, we need to compare the labeling of these two methods. To do this, first, the best classification is selected for each method, whose labeling can than be analysed together. This selection can be done using the values of the ROC curves (see Subsection 3.2.2) calculated for these methods: the best threshold value is located where the TPR value is the highest with the FPR being the lowest. In practice, by plotting the $1 - \text{FPR}$ values on the same plane as the original ROC curve, the intersection of these two curves will yield the best threshold value. After getting the best classifications for each method, their performance can be best visualized by a confusion matrix, which presents the number of true positive, false positive and true negative, false negative labels in an intuitive table format. Figure 6.7 and 6.9 show the confusion matrices for the best classification of the two approaches next to each other for easy comparison at 5 and 50 training sample sizes, respectively. Although, the difference between every field could be investigated more thoroughly, the most important, in the case of Anomaly Detection, are the false positive samples. Looking at the bad samples that were labeled incorrectly by the Autoencoder and the DDPM approach "corrected" them, we can analyze what kind of samples does the DDPM based method perform better on. Based on this, Figure 6.8 and 6.10 showcases some example samples from this group with both their reconstructions and MSE difference images below them. Moreover, Table 6.4 and Table 6.5 are also provided where the MSE values for these burnt samples are listed.



(a) Autoencoder **(b)** DDPM

Figure 6.7: Confusion matrices of the Autoencoder and DDPM methods for 5 training sample size on the Good vs. Burnt Soldering Samples



(a) Original Samples

(b) Autoencoder with code size 5 and depth 4

(c) DDPM with diffusion step count 200

Figure 6.8: Autoencoder false negative examples that are correctly detected by the DDPM model for 5 training samples

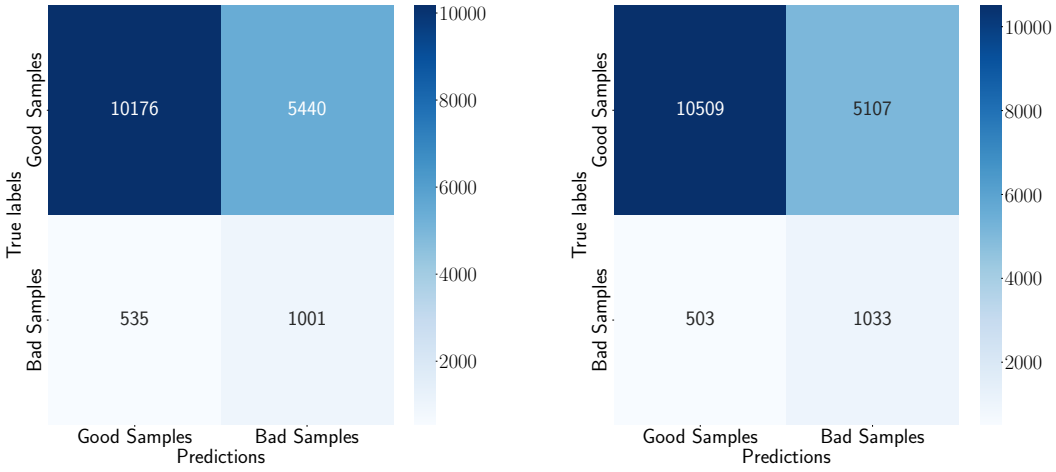
		Example Samples				
Methods	Thresholds	1	2	3	4	5
Autoencoder	0.02633	0.02477	0.02613	0.02382	0.02600	0.02506
DDPM	0.01678	0.03356	0.03303	0.03297	0.02326	0.02883

Table 6.4: MSE Values of the Autoencoder and DDPM based methods trained with 5 samples for the 5 burnt example samples

Looking at the example images' reconstructions and difference heatmaps, we can see that the Autoencoder's main alterations (see Figure 6.8b) come from the overall shape of the soldering and only in part from the different colour on the right side compared to the original soldering. Moreover, the MSE values are all very near to the threshold, meaning that these samples are barely getting mislabeled. In the case of the last sample, where there is burn damage over the soldering, the network actually kept some of the darker color, which can be seen highlighted on the MSE difference image.

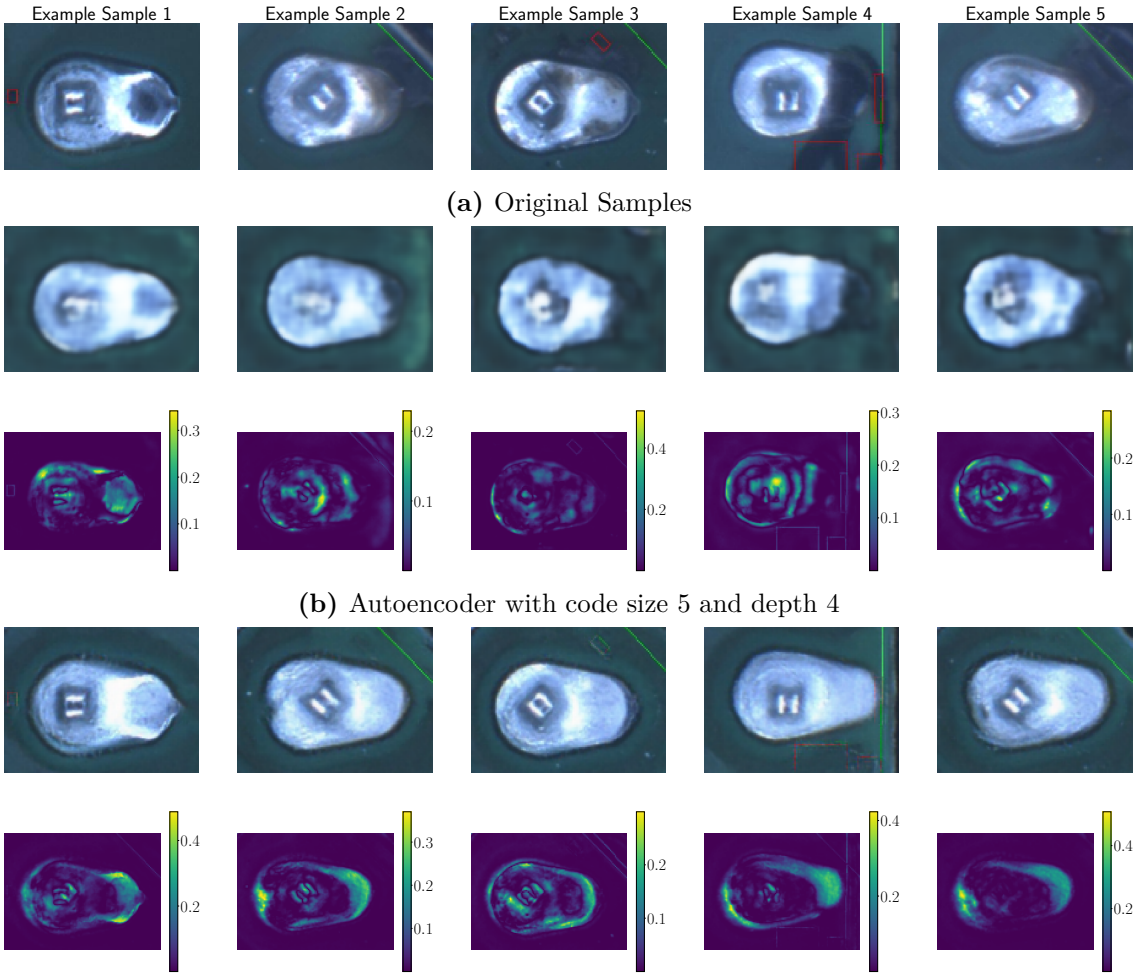
As for the DDPM approach, although the MSE values are much higher than the classification threshold, looking at the heatmaps, this difference mainly comes from the difference

in color compared to the original sample and not from correcting the burnt areas. However, in the case of the last sample, the DDPM also generated a soldering slightly darker coloured in the right side, whose major contribution to the overall MSE value can be justified here, as there was burn damage covering the soldering.



(a) Autoencoder **(b)** DDPM

Figure 6.9: Confusion matrices of the Autoencoder and DDPM methods for 50 training sample size on the Good vs. Burnt Soldering Samples



(a) Original Samples

(b) Autoencoder with code size 5 and depth 4

(c) DDPM with diffusion step count 200

Figure 6.10: Autoencoder false negative examples that are correctly detected by the DDPM model for 50 training samples

		Example Samples				
Methods	Thresholds	1	2	3	4	5
Autoencoder	0.02178	0.02148	0.01349	0.01725	0.02102	0.01764
DDPM	0.01362	0.01415	0.01629	0.01464	0.03053	0.02902

Table 6.5: MSE Values of the Autoencoder and DDPM based methods trained with 50 samples for the 5 burnt example samples

Based on the examples provided on Figure 6.10b, the Autoencoder's reconstruction is very similar to what was seen at 5 samples above: the main difference comes from the alteration of the shape of the soldering itself and not particularly from removing the burn marks. Looking at the MSE values, it is also the same situation as before, since all of them are close to the threshold value with two even being on the edge of it.

In the case of the DDPM, the main differences in the output images compared to the originals again come from the color change of the soldering, which outweigh the fact that it corrected the burn damages when it comes to calculating the MSE values. As for these

values, the two samples (Number 1 and 2) that have smaller burn marks are even very close to the threshold, making them barely detected correctly by the network.

Based on these findings, it can be said that although the DDPM approach slightly outperforms the Autoencoder at the smaller sample sizes of 5 and 50, when it comes to the ability of detecting the smaller burn damage anomalies, it simply just manages to label them correctly due to the impact of altering the color of the soldering. When looking at the larger sample size of 500, the situation changes, as the reconstructions become better and stick more to the original image, but this counters the detection of samples with smaller anomalies, as their difference will be too close to the MSE values of the good samples, making the separation of the two distributions very difficult (see Table 6.3). All in all, while the DDPM accomplishes its original task by creating output pictures with great fidelity to the originals, it cannot be used for the task of Anomaly Detection easily.

Chapter 7

Conclusions

To sum up the discussed results, Figure 7.1 is provided here, where the AUC values of each method for every training sample size are shown on the whole validation dataset. This means that these values are the joint performances of these different approaches and they signal how well each method can separate the good samples from the bad ones in general.

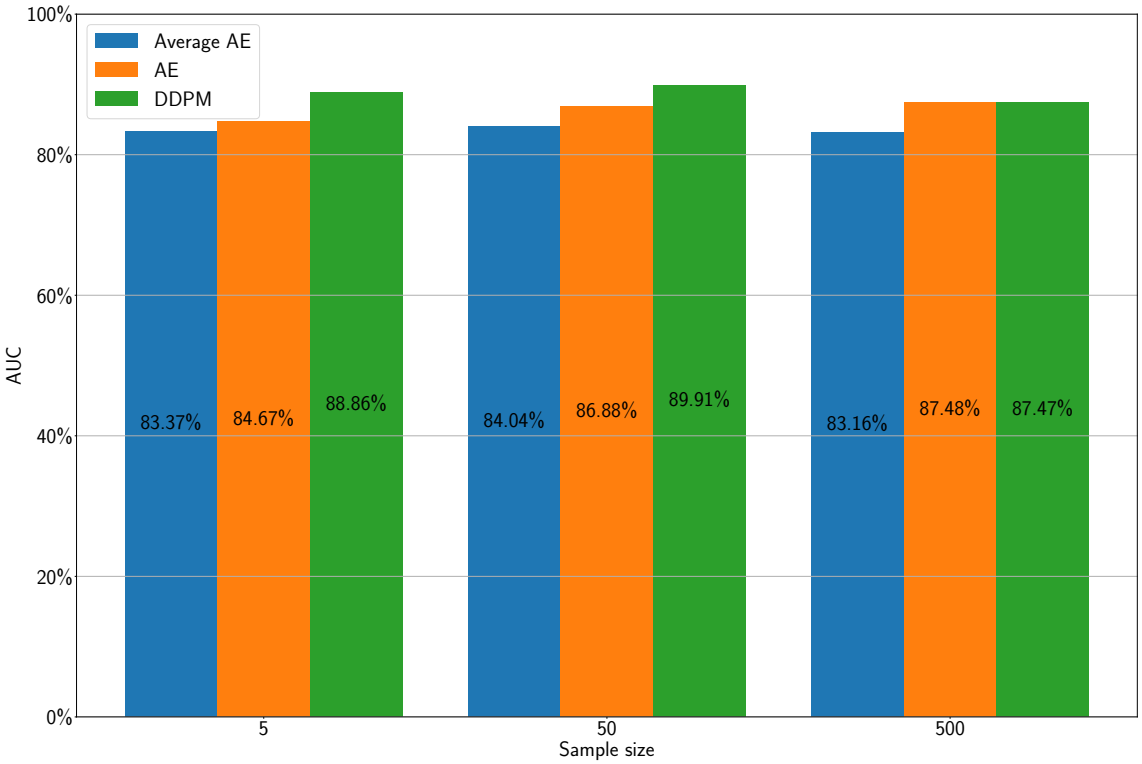


Figure 7.1: Joint AUC values across every class of the different methods for every training sample size

Looking at these values, they truly summarize the findings of the individual classes: the Average Autoencoder provides the same reliable performance across all sample sizes, the Autoencoder starts as second best, then barely peaks at 500 samples and the DDPM based approach is better at smaller sample sizes while dropping slightly at the larger ones. The main conclusion of these results is that while the DDPM provides a truly novel way of generating images with high quality and great likeliness to the training samples, it still only

performs slightly better than the Autoencoder at smaller sample sizes and when looking at the smaller to detect anomalies, it does not deliver the expected outputs. The slight edge in performance over the Autoencoder at fewer samples comes from the fact that it does alter the appearance of the samples by a large margin and not from only correcting the anomalies. Meanwhile, at the large sample sizes, it struggles to separate the distributions of the good and bad images due to its keeping true to the original sample too much thus resulting in small differences for both groups. All in all, employing the DDPM approach can certainly improve the performance of an Anomaly Detection process, however it does not overcome the Autoencoder based approach reliably at all sample sizes and especially not convincingly in the case of the smaller to detect deformities.

Despite not providing great results, it is important to note, that the DDPM network can be further optimized and improved, as these measurements were done on only a baseline model, to test out the general capabilities of the architecture. Moreover, there are also several ways that these detection performances can be improved: one of these is the selection of training data, as we have seen that what features the model learns during training can seriously affect the generation process, so leaving out the more unique training data can help the reconstruction process. Another idea for improvement can be the augmentation of the training samples, as by seeing them in different form every training iteration, the model can only learn the features always present on them, thus filtering the information from which it generates its outputs, making the output images more consistent. Finally, choosing a more complex metric to calculate the image differences that is truly able to highlight the smaller inequalities can also certainly improve the detection performance.

Bibliography

[1] Julia Wolleb, Florentin Bieder, Robin Sandkühler, and Philippe C. Cattin. Diffusion models for medical anomaly detection. In Linwei Wang, Qi Dou, P. Thomas Fletcher, Stefanie Speidel, and Shuo Li, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2022*, pages 35–45, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-16452-1.

[2] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), jul 2009. ISSN 0360-0300. DOI: 10.1145/1541880.1541882. URL https://doi.org/10.1145/1541880.1541882.

[3] Abir Smiti. A critical overview of outlier detection methods. *Computer Science Review*, 38:100306, 2020. ISSN 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2020.100306. URL https://www.sciencedirect.com/science/article/pii/S1574013720304068.

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

[5] Chong Zhou and Randy C. Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 665–674, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348874. DOI: 10.1145/3097983.3098052. URL https://doi.org/10.1145/3097983.3098052.

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[7] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *CoRR*, abs/1703.05921, 2017. URL http://arxiv.org/abs/1703.05921.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020. URL https://arxiv.org/abs/2006.11239.

[9] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[10] Xinyi Zhang, Naiqi Li, Jiawei Li, Tao Dai, Yong Jiang, and Shu-Tao Xia. Unsupervised surface anomaly detection with diffusion probabilistic model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6782–6791, October 2023.

[11] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. ISSN 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2005.10.010. URL https://www.sciencedirect.com/science/article/pii/S016786550500303X. ROC Analysis in Pattern Recognition.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

[13] Julian Wyatt, Adam Leach, Sebastian M. Schmon, and Chris G. Willcocks. Anoddpm: Anomaly detection with denoising diffusion probabilistic models using simplex noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 650–656, June 2022.