Master of Science Thesis in Computer Science
Department of Electrical Engineering, Linköping University, 2023

Industrial 3D Anomaly Detection and Localization Using Unsupervised Machine Learning

Kevin Bärudde and Marcus Gandal

Master of Science Thesis in Computer Science

Industrial 3D Anomaly Detection and Localization Using Unsupervised Machine Learning

Kevin Bärudde and Marcus Gandal

Supervisor: **Manh Cuong Le**
ISY, Linköpings universitet
Oliver Andlid
Combitech

Examiner: **Bastian Wandt**
ISY, Linköpings universitet

Computer Vision Laboratory
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

Detecting defects in industrially manufactured products is crucial to ensure their safety and quality. This process can be both expensive and error-prone if done manually, making automated solutions desirable. There is extensive research on industrial anomaly detection in images, but recent studies have shown that adding 3D information can increase the performance. This thesis aims to extend the 2D anomaly detection framework, PaDiM, to incorporate 3D information. The proposed methods combine RGB with depth maps or point clouds and the effects of using PointNet++ and vision transformers to extract features are investigated. The methods are evaluated on the MVTec 3D-AD public dataset using the metrics image AUROC, pixel AUROC and AUPRO, and on a small dataset collected with a Time-of-Flight sensor. This thesis concludes that the addition of 3D information improves the performance of PaDiM and vision transformers achieve the best results, scoring an average image AUROC of 86.2 ± 0.2 on MVTec 3D-AD.

Keywords *- Machine Learning, 3D anomaly detection, feature extraction, manufacturing, computer vision, vision transformer, PointNet*

Acknowledgments

We would like to thank the people at Combitech and ISY for making this thesis possible. We wish to express our gratitude to Oliver Andlid for being our supervisor at Combitech and Cuong Le for being our supervisor at Linköping University. Finally, we wish to thank Bastian Wandt for being our examiner at Linköping University.

Linköping, June 2023
Kevin Bärudde och Marcus Gandal

Contents

Notation **xi**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Aim . . . 2
1.3 Research Questions . . . 3
1.4 Delimitation . . . 3
1.5 Contributions . . . 3

2 Related Work **5**
2.1 Unsupervised 2D Anomaly Detection and Localization . . . 5
2.1.1 Representation-based Methods . . . 5
2.1.2 Reconstruction-based Methods . . . 6
2.1.3 Patch Distribution Modeling Framework for Anomaly Detection . . . 7
2.2 Unsupervised 3D Anomaly Detection and Localization . . . 8
2.2.1 3D Variational Auto-Encoder . . . 8
2.2.2 3D Fast Unsupervised Anomaly Detection with Generative Adversarial Networks . . . 8
2.2.3 3D Student-Teacher . . . 9
2.2.4 Asymmetric Student-Teacher . . . 10

3 Theory **11**
3.1 Anomaly Detection and Localization . . . 11
3.2 Machine Learning . . . 11
3.3 Neural Network . . . 11
3.3.1 Activation Function . . . 12
3.3.2 Learning . . . 13
3.4 Convolutional Neural Network . . . 13
3.4.1 Convolutional Layers . . . 13
3.4.2 Residual Networks . . . 14
3.5 Time-of-Flight Sensor . . . 14
3.6 3D Data Representations . . . 15

3.6.1 Depth Map . . . 15
3.6.2 Point Cloud . . . 15
3.7 Feature Extraction . . . 15
3.7.1 RGB Images . . . 15
3.7.2 Depth Map . . . 15
3.7.3 Point Cloud . . . 16
3.8 Probabilistic Concepts . . . 16
3.8.1 Probability Density Function . . . 16
3.8.2 Density Estimation . . . 16
3.8.3 Multivariate Gaussian Distribution . . . 16
3.8.4 Mahalanobis Distance . . . 17
3.8.5 Mixture Model . . . 17
3.8.6 Mixture Density Network . . . 17
3.9 Vision Transformer . . . 18
3.9.1 Patch Embedding . . . 19
3.9.2 Class Embedding . . . 19
3.9.3 Position Embedding . . . 19
3.9.4 Transformer Encoder . . . 19
3.9.5 Multi-Layer Perceptron Head . . . 21
3.9.6 Data-efficient image Transformer . . . 21
3.9.7 Class-attention image Transformer . . . 21
3.10 PointNet++ . . . 21
3.10.1 Farthest Point Sampling . . . 21
3.10.2 Layers . . . 22
3.10.3 Centroid Grouping . . . 22
3.10.4 Architectures . . . 23

4 Method **25**
4.1 Data . . . 25
4.1.1 Aluminium Can Dataset . . . 25
4.1.2 Depth Map Preprocessing . . . 25
4.1.3 Point Cloud Preprocessing . . . 26
4.1.4 RGB Preprocessing . . . 26
4.2 Implementation . . . 27
4.2.1 PaDiM . . . 27
4.2.2 PaDiM Baseline . . . 27
4.2.3 PaDiM-3D . . . 27
4.2.4 PaDiM-ViT . . . 29
4.2.5 Investigating Possible Improvements to the Depth Methods 31
4.2.6 Mixture Density Network for Density Estimation . . . 32
4.2.7 Point-PaDiM . . . 33
4.2.8 Investigation of Point Cloud Improvements . . . 34

5 Evaluation **37**
5.1 Metrics . . . 37
5.2 Experiments . . . 39

5.2.1 Accuracy . . . 39
5.2.2 Inference Time . . . 39
5.2.3 Resolution . . . 39

6 Results **41**
6.1 PaDiM Baseline . . . 41
6.2 PaDiM-3D . . . 41
6.3 PaDiM-ViT . . . 42
6.4 Comparison Between PaDiM-3D and PaDiM-ViT . . . 42
6.5 Visual Inspection of the Depth Methods . . . 43
6.6 Investigating Possible Improvements for the Depth Methods . . . 43
6.7 Mixture Density Network . . . 45
6.7.1 DeiT . . . 45
6.7.2 CaiT . . . 46
6.7.3 Mixture Density Network Visual Inspection . . . 47
6.8 Point-PaDiM results . . . 48
6.8.1 Sampling Method . . . 48
6.8.2 Layers . . . 48
6.8.3 Removing Background Points . . . 49
6.8.4 Combining With RGB . . . 51
6.8.5 Channels . . . 51
6.8.6 Downsampling . . . 51
6.9 Final PaDiM Results . . . 52
6.10 Inference Times . . . 53
6.11 Resolution . . . 53
6.12 Aluminium Can Dataset . . . 54

7 Discussion **57**
7.1 Results . . . 57
7.1.1 Vision Transformers vs. ResNets . . . 57
7.1.2 Mixture Density Network . . . 58
7.1.3 Point-PaDiM . . . 58
7.2 Method . . . 60
7.2.1 PaDiM-3D . . . 60
7.2.2 Point-PaDiM . . . 60
7.2.3 Aluminium Can Dataset . . . 60
7.2.4 Reliability . . . 61
7.2.5 Mixture Density Network . . . 61
7.2.6 Evaluation . . . 61
7.2.7 Heat Map Visualization . . . 62
7.3 Limitations With 3D Information . . . 62
7.4 The work in a wider context . . . 62

8 Conclusion **63**
8.1 Future Work . . . 64

Bibliography **65**

A Detailed Results **73**
A.1 PaDiM 2D 73
A.2 Inference Times Details 74
A.3 Downsampling Details 75
A.4 Extended Final PaDiM Results 76

B Extended Visual Results **77**
B.1 PaDiM-ViT 77
B.2 Point-PaDiM 77
B.2.1 Networks 77
B.2.2 Background Removal 78
B.3 Aluminium Can Dataset 79

Notation

Abbreviations

Abbreviations	Explanation
ST	Student-Teacher
NN	Neural Network
ToF	Time-of-Flight
RGB	Red, Green and Blue
MVG	Multivariate Gaussian Distribution
GMM	Gaussian Mixture Model
MDN	Mixture Density Network
HoG	Histogram of Oriented Gradients
SIFT	Scale Invariant Feature Transform
FPFH	Fast Point Feature Histogram
FPS	Farthest Point Sampling
ReLU	Rectified Linear Unit
CNN	Convolutional Neural Network
MLP	Multi-Layer Perception
MSA	Multi-head Self-Attention
SSG	Single-Scale Grouping
MSG	Multi-Scale Grouping
MRG	Multi-Resolution Grouping
ViT	Vision Transformer
CaiT	Class-attention Image Transformer
DeiT	Data-efficient Image Transformer

AD Methods

Abbreviations	Explanation
PaDiM	Patch Distribution Modeling Framework
AST	Asymmetric Student-Teacher Network
ResNet	Residual Neural Network
f-AnoGAN	Fast Unsupervised Anomaly Detection with Generative Adversarial Network

Evaluation Metrics

Abbreviations	Explanation
TPR	True Positive Rate
FPR	False Positive Rate
PRO	Per Region Overlap
AUROC	Area Under the Receiver Operating Characteristic

Organisations

Abbreviations	Explanation
CVL	Computer Vision Laboratory
ISY	Instutionen för systemteknik (Department of Electrical Engineering)
LiU	Linköping University

Equations

Notation	Explanation
$A \cap B$	Intersection
$\|x\|$	Absolute value
$\mathbb{R}$	Set of all real numbers
$a \in A$	a is an element of the Set A

1 Introduction

1.1 Motivation

Detecting anomalies in industrial manufacturing using machine learning can be challenging as the types of anomalies are often unknown beforehand. Moreover, anomalies are typically rare in practice, making it difficult to gather enough anomalous data for training machine learning models. These factors make supervised methods unsuited for the task. To address these challenges, unsupervised methods are often used instead, which seek to learn the distribution of the normal data. Anomalies are then detected when a sample that deviates from the norm is encountered. Anomaly detection is an area of study that goes far beyond industrial manufacturing; finance, cyber security and medicine are just a few examples of fields in which anomaly detection is used to detect inconsistencies in data [32].

In practice, detecting anomalies is both costly and error-prone to do manually. One solution is to automate the process by letting a machine perform the task instead of a human. There is extensive research on industrial anomaly detection on images, however, not much attention has been put on 3D anomaly detection. Modern technology allows for high-resolution 3D sensors and since the release of the MVTec 3D-AD dataset [7], the research on 3D anomaly detection has begun to increase. Some examples of data from the MVTec 3D-AD dataset are given in Figure 1.1. Recent studies have shown that 3D representations provide relevant information useful for detecting anomalies [4, 47], especially when combined with RGB data [7, 40]. Furthermore, with the rapid advances in computer vision, more modern architectures have been proposed that achieve state of the art performance on staple vision tasks, such as image classification [16].

Combitech is a consulting company whose main area of focus is to provide technical solutions to its customers. Recently, the company has seen a growing in-

terest amongst its customers to automate the process of detecting defective items in industrial manufacturing. The Patch Distribution Modeling Framework for Anomaly Detection and Localization (PaDiM) is an example of an unsupervised anomaly detection framework designed to detect and localize anomalies in industrial images.

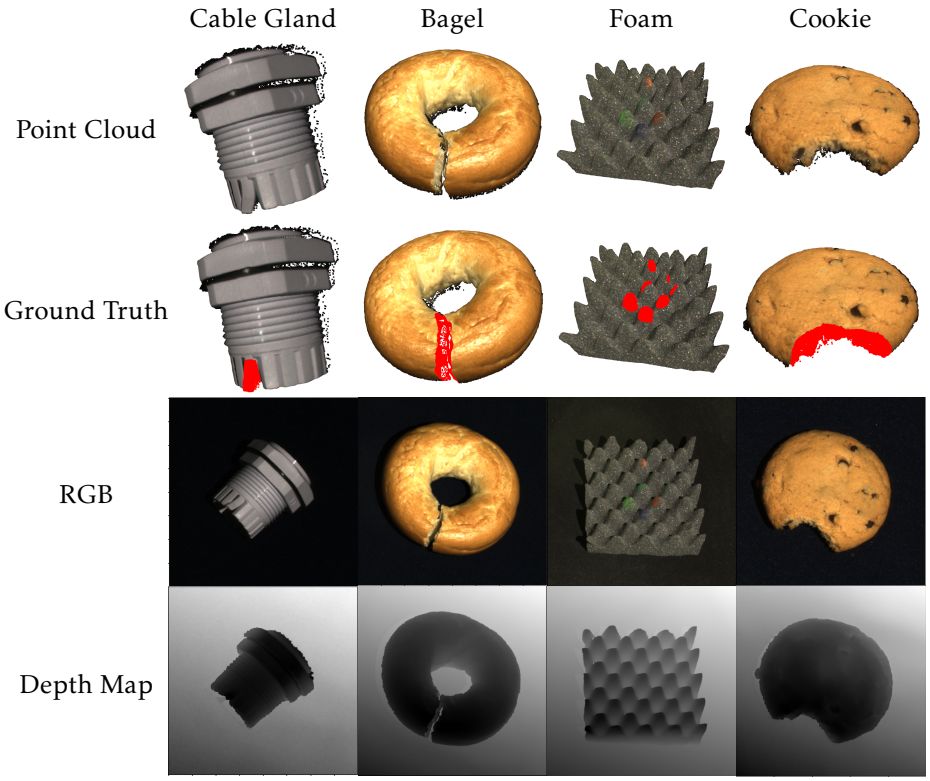

Figure 1.1: *Examples of data in the MVTec 3D-AD dataset. The dataset includes both RGB images and 3D information. Ground truth anomalies indicated in red are also provided.*

1.2 Aim

The aim of this master thesis was to investigate if 3D information can be integrated into PaDiM to increase its performance. PaDiM has already been investigated on 2D data by Combitech and showed promising results on real-life images, it was therefore interesting to investigate if the performance could be further improved by adding 3D information. The end goal was to produce a working framework based on PaDiM that detects and localizes anomalies in real objects using both 2D data and 3D data, and that would be beneficial to Combitech.

1.3 Research Questions

- What performance can be achieved when testing the proposed methods on the MVTec-3D AD dataset?
- What performance can be achieved when testing the proposed methods on a small dataset collected with a time-of-flight camera?
- Can the proposed 3D anomaly detection methods achieve better performance than the original PaDiM?
- How does the resolution of the sensor affect the performance of the proposed methods?
- Can the proposed methods achieve real-time inference times to be used in a real industrial environment?

1.4 Delimitation

All models were trained and tested on the MVTec 3D-AD dataset or a small real-life dataset collected with a Helios2 time-of-flight[1] sensor during the project. Determining the optimal threshold for predicting anomalies was not part of the investigation and the methods instead generated scalar-valued anomaly heat maps which could be visually interpreted. The hardware used for training and testing the models were: Intel Core i7-8700K, NVIDIA GeForce GTX 1080 Ti and i9-9880H, NVIDIA GeForce RTX 2070 (mobile).

1.5 Contributions

Marcus was responsible for the depth methods and Kevin for the point cloud methods. Each member was responsible for writing the chapters in the report specifically relating to their work, and the rest was written together.

[1] https://thinklucid.com/product/helios2-time-of-flight-imx556/

2

Related Work

2.1 Unsupervised 2D Anomaly Detection and Localization

Unsupervised 2D industrial anomaly detection is a well-established field of research and the methods can broadly be divided into two categories: Representation-based methods and Reconstruction-based methods [32]. Currently, representation-based methods achieve the best results [32].

2.1.1 Representation-based Methods

Representation-based methods involve two main components: *feature extraction* and *density estimation* [32]. The feature extraction aims to extract visual features from images, usually using a pre-trained CNN backbone trained on a large dataset such as ImageNet [42]. Examples of commonly used CNN architectures used are ResNet [19] and EfficientNet [48]. After the feature extraction, some form of density estimation is performed to learn the distribution of the features in normal images. During inference, anomalies are indicated by a low density. PaDiM [13] falls under this category.

Sub-image anomaly detection with deep pyramid correspondences (SPADE) [11] and PatchCore [38] are two methods that resemble PaDiM. Both methods utilize a memory bank which stores features seen during training. SPADE uses a two-step approach where anomaly localization is only performed if an anomaly is first detected on an image level. SPADE introduces a very resource-demanding anomaly localization process that PatchCore attempts to solve by using greedy coreset subsampling to reduce the size of the bank, while still keeping the most relevant information. For density estimation, k-nearest neighbours (kNN) is used, where a large distance to the neighbours indicates a high probability of an anomaly.

One downside of the memory bank approach is that it scales linearly with the size of the dataset [11]. In contrast, PaDiM is independent of the size of the training set.

More recent works propose a type of parametric neural density estimator called normalizing flows to estimate the density of the features. This solution relaxes the strong assumption of the underlying distribution and can approximate more complex distributions. DifferNet, proposed by Rudolph et at. [39] concatenates features extracted from images at different scales before feeding them into a normalizing flow network for density estimation. Moreover, different transformations are applied on the images to improve the robustness of the model. Following the work of Rudolph et al., CFLOW-AD [18] seeks to improve the localization performance by adding positional encoding to the feature representations, which achieves excellent localization performance on the MVTec 2D-AD dataset. Rudolph et al. extend DifferNet [39] by incorporating feature maps at different scales. This is shown to improve the anomaly detection and localization performance.

The Mixture Density Network (MDN) [8] is another type of neural density estimator that has been applied to anomaly detection and localization. VT-ADL proposed by Mishra et al. [31] implements a Gaussian Mixture Density Network to estimate the density of the features extracted from a vision transformer to localize anomalies. VT-ADL trains a vision transformer to learn both the parameters of the MDN and to reconstruct the feature vectors from the input images, hence it is a combination of a representation-based and reconstruction-based method. A possible drawback with VT-ADL is that the vision transformer is trained from scratch on a small dataset with $\sim$ 300 samples per class. It has been stated in previous research [16] that vision transformers typically require large amounts of data to reach their full potential. Moreover, estimating the feature distribution and reconstruction parameters simultaneously may slow down the convergence speed of the model.

2.1.2 Reconstruction-based Methods

In Reconstruction-based methods, a model is trained to reconstruct the normal images which it receives as inputs. When presented with an image that contains an anomaly, the model is unable to properly reconstruct the anomalous region which is indicated by a large reconstruction error. Commonly used architectures for reconstruction-based methods are generative models such as generative adversarial networks (GANs) and auto-encoders (AEs). Schlegl et al. [44] developed the framework f-AnoGAN to detect anomalies in medical images. The method consists of first training a variant of the GAN called Wasserstein Generative Adversarial Network (WGAN) to generate non-anomalous images from a normal distributed latent space. A WGAN measures the Wasserstein distance between the distribution of the input data and the generated data. Secondly, an encoder network is trained to map input images to the latent space to be decoded by the generator module in the WGAN. The anomaly score of a test sample is computed as the weighted sum between the reconstruction error and the feature error which

is the squared-error between the discriminator output of the original image and the discriminator output of the generated image. An illustration of f-AnoGAN is given in Figure 2.4. The f-AnoGAN has also been evaluated in an industrial setting on the MVTec 2D-AD dataset by Bergmann et al. [6], but achieved poor performance. Baur et al. [3] train an AE network to first map input brain MR images to a low dimensional latent space which can be sampled from to reconstruct the inputs. The reconstruction error is used to compute the anomaly score for each test sample. A similar approach has also been studied on industrial images by Youkachen et al. [56].

2.1.3 Patch Distribution Modeling Framework for Anomaly Detection

Patch distribution modelling framework for anomaly detection (PaDiM) is a method proposed in 2020 by Defard et al. [13] to solve the problem of anomaly detection and localization in industrial images. PaDiM concatenates features from different layers in a ResNet [19] backbone to obtain patch embeddings from different semantic levels. These embeddings are used to learn the parameters of a multivariate Gaussian distribution for each patch. Anomalies are detected and localized by computing the Mahalanobis distance [28] between the patch embeddings of the test image and the learnt Gaussian distributions of the patches where large distances indicate anomalous patches. An illustration of the PaDiM framework is given in Figure 2.1 and 2.2. PaDiM was originally designed for 2D anomaly detection and localization, and was tested on the MVTec 2D-AD dataset [6] achieving state-of-the-art performance in 2020 [13].

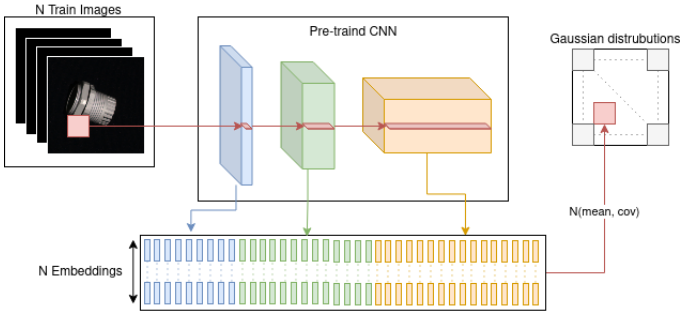

Figure 2.1: *Learning the Gaussian distribution*

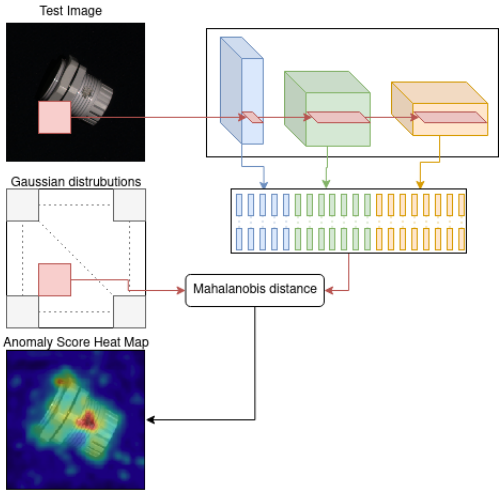

Figure 2.2: *Locating anomalies*

2.2 Unsupervised 3D Anomaly Detection and Localization

2.2.1 3D Variational Auto-Encoder

Bengs et al. [4] use a variational auto-encoder (VAE) to detect anomalies in Magnetic Resonance Imaging (MRI) scans of the human brain. The MRI scans are represented as 3D voxel grids of the brain volumes. To work with the voxel data, 3D convolutional layers were used to build the VAE's encoder and decoder. Figure 2.3 shows an illustration of the architecture. The goal is to train the VAE to reconstruct healthy scans so that it fails to reconstruct abnormal MRI scans. The method was further optimized by erasing regions of the input image to force the method to learn the in-painting of data. The method was compared to an identical VAE structure, but created with a 2D convolutional layer and trained on slices of MRI scans instead. This method was evaluated on the MVTec 3D-AD dataset by Bergmann et al. [7]. The reported area under the PRO curve (AUPRO) with an integration limit of 0.3 was 49.2% using only XYZ data and a score of 47.1% when using both RGB and XYZ data.

2.2.2 3D Fast Unsupervised Anomaly Detection with Generative Adversarial Networks

Simarro et al. [47] and Presa et al. [33] both presented fast unsupervised anomaly detection with generative adversarial networks methods to detect anomalies in brain data represented by voxel grids. These methods are 3D extensions of the 2D f-AnoGAN by Schlegl et al. [44] described in Section 2.1.2. They extend the method by using 3D WGAN instead of a 2D WGAN. The 3D f-AnoGAN proposed

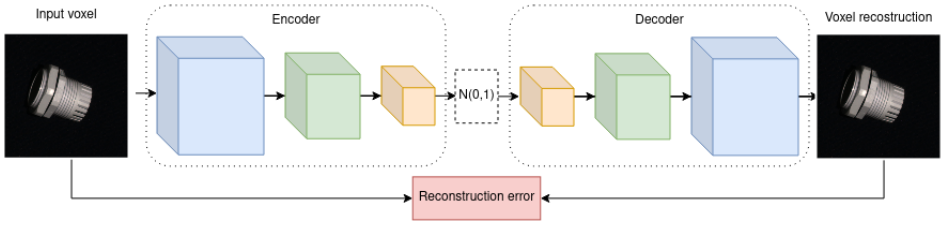

Figure 2.3: *Illustration of an 3D Variational Auto-Encoder*

by Simarro et al. [47] was evaluated by Bergmann et al. [7] on the MVTec 3D-AD dataset. They reported an AUPRO score of 58.3% with an integration limit of 0.3 using only XYZ voxel data and 63.0% using both RGB and XYZ voxel data.

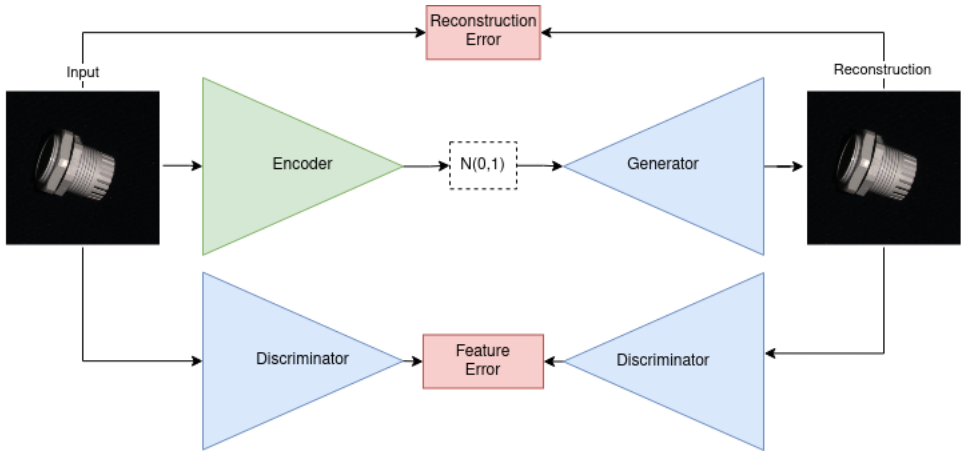

Figure 2.4: *Illustration of the f-AnoGAN structure*

2.2.3 3D Student-Teacher

Bergmann et al. [5] propose a method called 3D Student-Teacher (*3D-ST*) which is designed to detect and localize anomalies in point cloud data. The method has been evaluated on the MVTec 3D-AD dataset and is the first published anomaly detection and localization method that operates directly on 3D point clouds. A teacher network is pre-trained on the pretext task of outputting local descriptors for each point in a 3D point cloud and was pre-trained in a self-supervised manner on synthetic point clouds from ModelNet10 [55]. They show that training a teacher on a synthetic dataset yields descriptors that are useful for 3D anomaly detection and localization. A student network that has the same architecture as the teacher is then trained on the MVTec 3D-AD dataset to match the outputs of the teacher network. Anomalous regions are localized by measuring the difference between the output of the student and the output of teacher where a big

difference indicates anomalies. 3D-ST achieves an average AUPRO score of 83.3% on MVTec 3D-AD, with an integration limit of 0.3. 3D-ST is proven to be effective for detecting and localizing anomalies in 3D point clouds achieving above 90% AUPRO on most classes in MVTec 3D-AD, however, the method performs poorly on certain classes where it scores around 50% AUPRO.

2.2.4 Asymmetric Student-Teacher

The Asymmetric Student-Teacher (AST) architecture for industrial 3D anomaly detection was proposed by Rudolph et al. [40] in 2023 and is one of the few methods that have been implemented for both 2D anomaly detection and 3D anomaly detection. The method concatenates RGB features extracted from the pre-trained backbone network, EfficientNet-B5 [48], with depth images to incorporate 3D information. Moreover, a positional encoding is added to include information about the position of the pixels. In contrast to the traditional student-teacher architecture; the student and teacher networks are purposely chosen to be different from each other to combat over-generalization that often leads to low anomaly scores for certain anomalies. The teacher network is a 2D normalizing flow that is trained on the pretext task of transforming the input to a normal distribution, thereby creating targets for the student network. The student network is a conventional feed-forward neural network that is trained to match the outputs of the teacher network. The anomaly score is given by the difference between the teacher network and the student network. The AST method reports an image area under the receiver operating characteristic (AUROC) score of $93.7 \pm 0.2\%$ and a pixel AUROC score of $97.6 \pm 0.02\%$. It is shown that a combination of RGB and depth data improves the performance over using a single input type. In their work, only the effect of using raw depth map information is studied and the effect of using features extracted from depth maps is excluded.

3

Theory

3.1 Anomaly Detection and Localization

An anomaly can be defined as a rare event that deviates from the expected pattern of a system. Anomaly detection refers to the binary classification problem of finding such events and correctly classify them as anomalous. In anomaly localization, the location of the anomaly is also of interest [49]. For example, determining which pixels in an image constitute the anomaly.

3.2 Machine Learning

Machine Learning is a subdomain of artificial intelligence that describes a set of methods that learn how to solve problems without explicit instructions, given that enough training data is provided [29]. The end-goal is that the models are able to generalize to previously unseen data. Two common machine learning paradigms are *Supervised Learning* and *Unsupervised Learning* [29]. In supervised learning, a model is given training data which has been (manually) labelled with the expected outputs, and the model then attempts to learn a mapping from the inputs to the true labels. In contrast, unsupervised learning discovers patterns in data without the requirement of any labels.

3.3 Neural Network

A Neural Network (NN) [30] models a typically nonlinear function and is inspired by the neurons in the brain. The model is composed of a collection of neurons that are ordered in layers. The first layer is often referred to as the *input layer* and the last layer is usually called the *output layer*. In-between these layers,

a chosen number of *hidden layers* are typically added, which allows the neural network to solve more complex tasks. An illustration of a simple neural network is given in Figure 3.1. The output of a neuron is dependent on the output of the neurons in the previous layer. The output of a single neuron is given by:

$$z = (\mathbf{w}^T \mathbf{x} + b) \in \mathbb{R}^{1\times 1} \tag{3.1}$$

$\mathbf{w} \in \mathbb{R}^{N\times 1}$ are called the weights, $\mathbf{x} \in \mathbb{R}^{N\times 1}$ the inputs and $b \in \mathbb{R}^{1\times 1}$ is called the bias term. The output of a neuron is the weighted sum of all of its inputs plus the bias term. When there are multiple neurons in a layer, matrix notations become useful. The weights are now stored in the matrix $\mathbf{W} \in \mathbb{R}^{N\times M}$ and the biases in the vector $\mathbf{b} \in \mathbb{R}^{M\times 1}$, where N is the input dimension and M is the output dimension. The equation now becomes:

$$\mathbf{z} = (\mathbf{W}^T \mathbf{x} + \mathbf{b}) \in \mathbb{R}^{M\times 1} \tag{3.2}$$

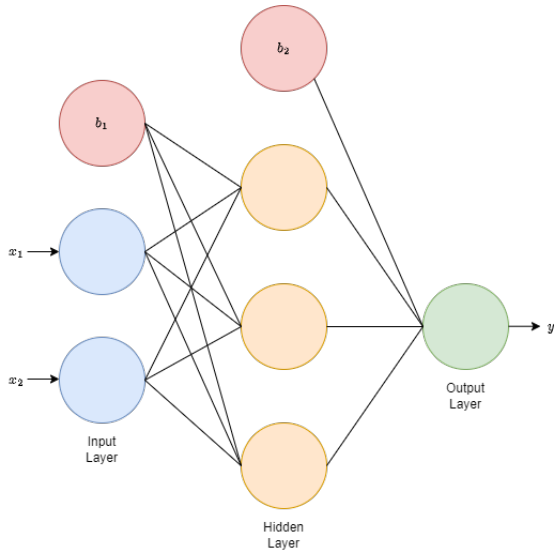

Figure 3.1: *A neural network consisting of one input layer (with two inputs), one hidden layer and one output layer (with one output).*

3.3.1 Activation Function

To introduce non-linearity to a neural network, a (non-linear) activation function $\sigma(z) = \sigma(\mathbf{w}^T \mathbf{x} + b)$ is applied to the output of a neuron. The final output of a single neuron then becomes:

$$y = \sigma(z) \in \mathbb{R}^{1\times 1} \tag{3.3}$$

Figure 3.2 illustrates a neuron with three inputs $x_{0,1,2}$ and output y with the activation function σ.

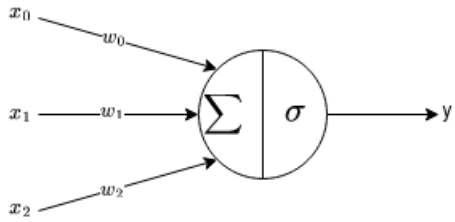

Figure 3.2: *A single neuron of a network*

3.3.2 Learning

The weights and biases of a neural network are updated during training where a loss function computes the error between the expected output and the predicted output. Common loss functions are the mean-square error (MSE) for regression problems and the cross-entropy (CE) for classification problems. The fitting of a neural network is typically done in an iterative manner by first computing the gradient of the loss function with respect to the weights, then updating the weights in the opposite direction of the gradient. Backpropagation [41] is commonly used as an efficient way to compute the gradients. The updating of the weights is performed with an optimization algorithm such as Adam [22].

3.4 Convolutional Neural Network

A convolutional neural network (CNN) [24] is a type of neural network that usually operates on image data. A common application for CNNs is object recognition where the goal is to correctly classify the categories of the objects in images. Such a CNN usually consist of two main parts: A *feature extractor* and a *classifier*. The feature extractor transforms the data using the convolution operator resulting in a set of feature maps. The classifier is a traditional neural network (see Section 3.3) that consists of a set of fully-connected layers that return the class scores.

3.4.1 Convolutional Layers

The convolutional layers of feature extraction consist of learned kernels that are used to perform convolution operations on the data. The layer contains L kernels with the shape of $K_l \in R^{C\times S\times S}$ for each kernel $l = \{1, ..., L\}$ where C represents the number of channels and S represents the size of the kernel. The operation is performed on the input X with the height and width of M and N. An illustration of a convolution with one channel is displayed in Figure 3.3.

$$(X * K_l)[m, n] = \sum_{z=1}^{C} \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[z, i, j] \cdot K_l[z, m - i, n - j] \tag{3.4}$$

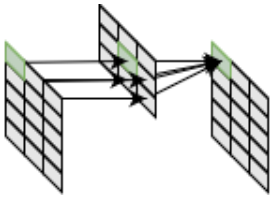

Figure 3.3: *2D Convolution on an image with one channel.*

The convolution operator can also be defined for data of different dimensionality than two. A 1D convolution can be used for signal data and a 3D convolution for voxel grids.

3.4.2 Residual Networks

The residual networks (ResNets) by He et al. [19] are CNN architectures that introduce residual layers to ease the training of deeper networks. Created in 2015 to classify images in the CIFAR-10[1] and object detection/localization on the ImageNet [14] datasets. The method achieved state-of-the-art performance for both datasets. The network managed to receive this performance by using residual layers to mitigate the vanishing/exploding gradient problem, allowing the network to extend the number of layers used compared to other networks. The residual layer that builds the ResNet architecture is illustrated in Figure 3.4.

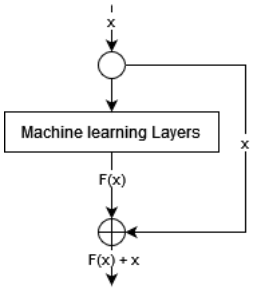

Figure 3.4: *A Residual Layer*

3.5 Time-of-Flight Sensor

Time-of-Flight (ToF) [23] is the measurement of the time it takes for light to travel from a point A to a point B. The method is used to calculate the distance to a target by using the time it takes for light to travel from the sensor and back. Figure 3.5 shows an example of ToF when a light beam moves to the target and back. Using the measured time and the universal constant of the speed of light, the distance to the target can be calculated.

[1] https://www.cs.toronto.edu/~kriz/cifar.html

Figure 3.5: *Time-of-Flight concept.*

3.6 3D Data Representations

3.6.1 Depth Map

Depth maps are typically grayscale images where the intensity of each pixel is determined by the distance from the viewpoint to the surface covered by that pixel.

3.6.2 Point Cloud

Point Clouds can be used to represent 3D shapes and are typically generated with a 3D scanner. Point clouds are either organized or unorganized [21]. An unorganized point cloud is represented as a list, where every entry contains x, y and z values. In contrast, an organized point cloud is represented as a 2D matrix where the points are stored according to the spatial relationship between the points [21]. The MVTec-3D AD dataset [7] uses organized point clouds to represent the 3D data.

3.7 Feature Extraction

3.7.1 RGB Images

Feature extraction is effectively a transformation of data that conserves the most valuable information used to solve a task. An example of a feature extractor is the Convolutional Neural Network (CNN) [24] such as ResNet-50 [19], which makes use of the convolution operator in order to compress the data into a smaller feature space, while still preserving the most vital information about the data. In contrast to a CNN, which is a learned feature extractor, there are also handcrafted feature extractors designed by experts and theorized to give effective representations of data. Examples of handcrafted features are: Histogram-of-Oriented-Gradients (HOG) [12] and Scale-Invariant-Feature-Transform (SIFT) [27].

3.7.2 Depth Map

Feature extraction on depth maps using a pre-trained CNN has been investigated in previous works relating to other research fields, for example pose estimation [45]. Handcrafted features such as HOG [12] and SIFT [27] have also been investigated for head pose estimation using depth images [54].

3.7.3 Point Cloud

Extracting features from a point cloud can be done using either handcrafted features like Fast Point Feature Histogram (FPFH) [43] or learnt features that are either tree-based or point-based methods [26] like PointNet [34]. The FPFH method finds features by using the geometric relation between a point and its k nearest neighbours. The features are obtained by calculating the angular variations between estimated normals between the neighbouring points. The PointNet algorithm learns the features by applying one-dimensional convolution on the point cloud.

3.8 Probabilistic Concepts

3.8.1 Probability Density Function

Let X be a continuous random variable. The probability that X falls between the limits a and b can be determined by integrating the probability density function f over the limits. This relation can be mathematically expressed as [46]:

$$P(a < X < b) = \int_a^b f(x)dx \tag{3.5}$$

Any point along the probability function is commonly referred to as the *likelihood*. An example of a probability density function is that of the univariate Gaussian distribution:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{3.6}$$

3.8.2 Density Estimation

Estimating the probability density function of the data given a set of observed samples is called density estimation [46]. One of the oldest techniques for density estimation is to group points that share the same value into bins that together form a histogram. Given a set of n observed points $\mathbf{X} = (X_1, ..., X_n)$, the histogram with bin width h can then be defined as [46]:

$$\hat{f}(x) = \frac{1}{nh}k \tag{3.7}$$

where k is the number of X_i in the same bin as x. More advanced density estimators exist, but the histogram provides an intuitive idea to the concept.

3.8.3 Multivariate Gaussian Distribution

The Multivariate Gaussian Distribution (MVG) is a generalization of the continuous univariate Gaussian distribution. The MVG of a k-dimensional random vec-

tor $\mathbf{X} = (X_1, X_2, ..., X_k)^T$ can be written as [50]:

$$X \sim \mathcal{N}_k(\boldsymbol{\mu}, \Sigma) \tag{3.8}$$

where $\boldsymbol{\mu}$ is the k-dimensional mean vector

$$\boldsymbol{\mu} = E[\mathbf{X}] = (E[X_1], E[X_2], ..., E[X_k])^T \tag{3.9}$$

and Σ is the $k \times k$ dimensional covariance matrix

$$\Sigma_{ij} = E[(X_i - \mu_i)(X_j - \mu_j)] = Cov[X_i, X_j] \tag{3.10}$$

3.8.4 Mahalanobis Distance

The Mahalanobis distance [28] can be used to determine the distance from a test point $\mathbf{x}$ to a distribution and is computed by the equation:

$$\sqrt{(\mathbf{x} - \boldsymbol{\mu}^T)\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})} \tag{3.11}$$

where $\boldsymbol{\mu}$ is the mean vector of the distribution and Σ^{-1} is the inverse of the covariance matrix.

3.8.5 Mixture Model

A mixture model combines m probability density functions called *components*, i, to form a more complex probability density function [9]. Typically, a mixture consists of multiple components originating from the same type of distribution, where each distribution is associated with a weight α_i [9]. An example is the univariate Gaussian mixture model (GMM) given by:

$$p(x) = \sum_{i=1}^{m} \alpha_i \mathcal{N}(\mu_i, \sigma_i) \tag{3.12}$$

3.8.6 Mixture Density Network

The Mixture Density Network (MDN) proposed by Bishop [8] is a neural density estimator that combines a Neural Network with a Mixture Model. The model seeks to learn the conditional probability:

$$p(\mathbf{t}|\mathbf{x}) = \sum_{i=1}^{m} \alpha_i(\mathbf{x})\phi_i(\mathbf{t}|\mathbf{x})) \tag{3.13}$$

where m is the number of mixture components, $\alpha_i(\mathbf{x})$ is the mixing coefficient and $\phi_i(\mathbf{t}|\mathbf{x})$ is the conditional density of the target vector $\mathbf{t}$ for the ith kernel. For the purpose of this project, let $\phi_i(\mathbf{t}|\mathbf{x})$ be the Gaussian function, this gives:

$$p(\mathbf{t}|\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \mathcal{N}(\mathbf{t}|\boldsymbol{\mu}_i(\mathbf{x}), \sigma_i^2(\mathbf{x})) \tag{3.14}$$

The mixing coefficients $\alpha_i(\mathbf{x})$ must satisfy the constraint:

$$\sum_{i=1}^{m} \alpha_i(\mathbf{x}) = 1 \tag{3.15}$$

where $0 \leq \alpha_i(\mathbf{x}) \leq 1$, which is achieved by the *softmax* function:

$$\alpha_i = \frac{exp(z_i^{\alpha})}{\sum_{j=1}^{M} exp(z_j^{\alpha})} \tag{3.16}$$

The variances σ_i are constrained to be positive using the exponential function:

$$\sigma_i = exp(z_i^{\sigma}) \tag{3.17}$$

No constraint is imposed on the mean vector:

$$\mu_{ik} = z_{ik}^{\mu} \tag{3.18}$$

Finally, the loss function that is minimized is given by the negative log-likelihood [8, 9]:

$$L(\mathbf{w}) = -\sum_{n=1}^{N} ln(\sum_{i=1}^{m} \alpha_i(\mathbf{x}_n, \mathbf{w}), \mathcal{N}(\mathbf{t}|\boldsymbol{\mu}_i(\mathbf{x}_n, \mathbf{w}), \sigma_i^2(\mathbf{x}_n, \mathbf{w}))) \tag{3.19}$$

where $\mathbf{w}$ are the weights and biases of the MDN and N is the number of samples.

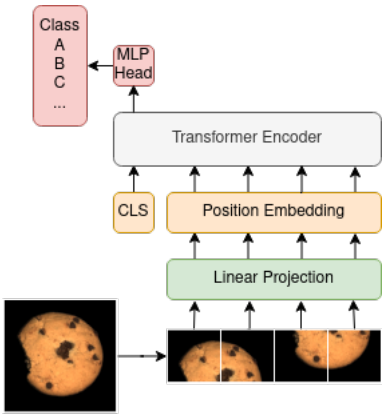

Figure 3.6: *Overview of the architecture of the vision transformer.*

3.9 Vision Transformer

The Vision Transformer (ViT) was first proposed by Dostrovitskiy et al. [16] in 2021 and is inspired by the Transformer originally proposed by Vaswani et

al. [53] in 2017 that achieves state-of-the-art performance for many natural language processing (NLP) tasks [15]. The ViT was the first vision model relying exclusively on self-attention that was proven to outperform the long standing ruler, the CNN, in many vision tasks [16]. One of the advantages of the ViT is that it can be trained using fewer resources than an equivalent CNN, however at the cost of requiring large amounts of data [16]. A Vision Transformer does not come with the same inductive biases as CNN, for example, translation equivariance and locality and thus has to learn it from scratch by observing more data [16]. An architetural overview of the ViT is given in Figure 3.6.

3.9.1 Patch Embedding

The ViT receives as input an image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$, where H and W are the height and width of the image and C is the number of input channels. The image is divided into patches of size $P \times P$, resulting in a total of $N = HW/P^2$ patches. These patches are flattened resulting in the sequence of flattened patches $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$. The flattened patches are linearly projected using the learnable matrix, $\mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}$, to the embedding dimension D that is constant throughout the network.

3.9.2 Class Embedding

A learnable class embedding (CLS), $\mathbf{x}_{cls} \in \mathbb{R}^{1 \times D}$ that has the objective to capture relevant information for classification, is prepended to the patch embeddings. The class embedding interacts with the patch embeddings through self-attention.

3.9.3 Position Embedding

A learnable position embedding $\mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D}$ is added to the class and patch embeddings to inject positional information to the embeddings as a transformer does not have any pre-knowledge of the sequence of the input patches.

3.9.4 Transformer Encoder

The transformer encoder takes as input the position enhanced embeddings and outputs contextualized embeddings that are learnt through self-attention. In addition to the two components described below, the transformer encoder employs two residual connections [19] which are applied after the multi-head self-attention (MSA) and after the multi-layer perceptron (MLP) inside the encoder block. Moreover, Layer normalization [2] is applied before the MSA and MLP blocks respectively. Most ViT architectures stack multiple transformer encoders sequentially [16, 51, 52]. The architecture of the encoder block is illustrated in Figure 3.7.

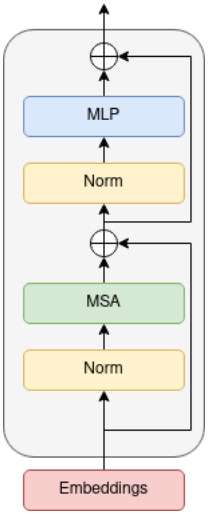

Figure 3.7: *The encoder block of the vision transformer.*

Multi-Head Self-Attention

The MSA layer takes as input the layer-normalized position encoded class and patch embeddings, $\mathbf{z} \in \mathbb{R}^{(N+1)\times D}$. The input is linearly transformed using the transformation matrix $\mathbf{U}_{qkv} \in \mathbb{R}^{D\times 3D_h}$, resulting in $[\mathbf{q}, \mathbf{k}, \mathbf{v}] = \mathbf{z}\mathbf{U}_{qkv} \in \mathbb{R}^{N\times 3D_h}$. The (scaled dot product) attention is computed as:

$$A = softmax(\mathbf{q}\mathbf{k}^T/\sqrt{D_h}) \in \mathbb{R}^{N\times N} \tag{3.20}$$

The output of one self-attention operation is $SA(\mathbf{z}) = A\mathbf{v} \in \mathbb{R}^{N\times D_h}$. In multi-headed self-attention, k self-attention operations are computed in parallel where each block is called a head. Typically, D_h is set to D/k [16]. The k outputs are concatenated, and a linear transformation with matrix $\mathbf{U}_{msa} \in \mathbb{R}^{kD_h\times D}$ is applied which yields the result:

$$MSA(\mathbf{z}) = [SA_1(z); SA_2(z); ...; SA_k(z)]\mathbf{U}_{msa} \in \mathbb{R}^{N\times D} \tag{3.21}$$

The result of the multi-head self-attention layer has the same dimensionality as the input $\mathbf{z}$.

Multi-Layer Perceptron

The MLP layer consists of two linear layers with GELU [20] non-linearity and receives as input the layer-normalized output from the MSA block. The first linear layer expands the dimensions, and the second linear layer shrinks it back to D.

3.9.5 Multi-Layer Perceptron Head

The feature vector of the CLS embedding is fed into a MLP classification head on top of the final transformer encoder block.

3.9.6 Data-efficient image Transformer

The Data-efficient image transformer (DeiT) was proposed by Touvron et al. [51] and closely follows the architecture of the original ViT proposed by Dosovitskiy et al. [16]. In addition to the CLS token that is prepended to the patch embeddings in the ViT, DeiT also appends a distillation token. The distillation token works similarly to the class token, but attempts to reconstruct the output of a teacher network instead of the true labels. The distillation token also interacts with the class and patch tokens through self-attention. The best performance was achieved when letting the teacher be a convolution neural network, RegNetY-16GF [36], for which the DeiT performs on par with the best convolution neural networks on the image classification task on ImageNet-1k [42] in terms of efficiency vs. accuracy. It is theorized by the authors that one of the reasons for why a convolution neural network is more effective than a vision transformer as the teacher, is because the inductive biases are inherited.

3.9.7 Class-attention image Transformer

In contrast to convolution neural networks, vision transformers have not shown to benefit from deep architectures [52]. The Class-attention image transformer (CaiT) was proposed by Touvron et al. [52] and scales the ViT to deeper architectures. It features two main improvements over the ViT, *LayerScale* and *Class-Attention*. LayerScale adds a learnable diagonal matrix to the output of each residual block in the encoder block to improve the training dynamic. Class-attention is a mechanism that separates the self-attention between patches and the self-attention between the patches and the class token. This is achieved by inserting the CLS token later in the network while freezing the weights learnt through self-attention, allowing the network to focus on one task at a time.

3.10 PointNet++

In 2017 Qi et al. [35] introduced PointNet++ as a model for different 3D point cloud tasks. The model is an improvement on the PointNet [34] model. The method is divided into two main parts: classification and segmentation network; where it achieved state-of-the-art results on challenging benchmarks in classification and segmentation tasks.

3.10.1 Farthest Point Sampling

Farthest point sampling (FPS) [17] is a sampling method for point clouds, that takes the set N consisting of n points and samples them down to a subset N'

consisting of m points. Given the point cloud $N = \{x_1, x_2, ..., x_n\}$ a point x_i is appended to the N' subset. Then until the subset N' is full the farthest point from all points in N' is appended to the subset. The resulting subset $N' = \{x_1, x_2, ..., x_m\}$ where the point x_i is the farthest point from the subset $\{x_1, x_2, ..., x_{i-1}\}$.

3.10.2 Layers

PointNet++ consists of two different types of layers: the set abstraction layer to learn the features of the data and the feature propagation layer to interpolate the features back to the input size.

Set Abstraction

The set abstraction layer consists of three parts: the sampling, grouping, and PointNet layers. The inputs to a set abstraction layer are N points with d dimensions and C channels, represented as $N \times C$ and $N \times d$ matrices. The sampling layer takes these points and uses FPS to sample N' points as centroids from the point cloud into a matrix of $N' \times C$. Then, the grouping layer uses the ball query method to collect all points within a radius around the centroids into a $N' \times K \times C$ matrix. Where K is the maximum number of neighbours and has a flexible amount of points. The last part is the PointNet layer from the PointNet architecture [34] consists of multi-layer perceptions (MLP) that extract local features and return a $N' \times C'$ matrix.

Feature Propagation

The feature propagation layer interpolates the feature vectors using the centroid grouping from the set abstraction layers. The features from the centroids are propagated to the neighbouring points; this is done by first using the inverse distance to weigh the features and then using a one-by-one convolutional layer and multiple shared fully connected layers and rectified linear unit (ReLU) layers over the features.

3.10.3 Centroid Grouping

The set abstraction layer samples a set of centroids and groups the data using the ball query method using a set radius; called Single-scale grouping (SSG). This can lead to a situation where there is a large variance in the density of the centroid grouping. To avoid this, two methods were developed: Multi-scale grouping (MSG) and Multi-resolution grouping (MRG).

Multi-Scale Grouping

The MSG variant concatenates multiple ball query groupings using different radii. This allows the layer to group data from a region while still containing useful information. This method is computationally expensive as it has to run the set

abstraction layer multiple times for different resolutions. An illustration of this method is shown in Figure 3.8.

Figure 3.8: *Multi-scale grouping.*

Multi-Resolution Grouping

MRG was developed to solve the problem of low-density regions while avoiding the cost of the MSG method. MRG consists of two parts that are concatenated. First, a region is sampled using SSG. Then, the region is divided into sub-regions that are put through the set abstraction layer, then these sampled sub-regions are fed through the set abstraction layer resulting in a large grouping of points. This is computationally more efficient as grouping over large areas is costly. An illustration of this method is shown in Figure 3.9.

Figure 3.9: *Multi-resolution grouping.*

3.10.4 Architectures

The original PointNet++ paper created different architectures for different tasks[2]. The three different tasks are classification, semantic segmentation and part segmentation. All tasks can use the three different groping methods (SSG, MSG, MRG) in the set abstraction layers. The general structure of a classification and segmentation network is shown in Figure 3.11 and Figure 3.10 respectively.

Classification

The classification network has three set abstraction layers that feed into fully connected layers. The first layer samples 512 points with a radius of 0.2, the second with 128 points with a 0.4 radius and the last layer samples down the points to 1 point with 1024 channels. These 1024 channels are fed into three fully connected layers that classify the point cloud.

[2] `https://github.com/charlesq34/pointnet2`

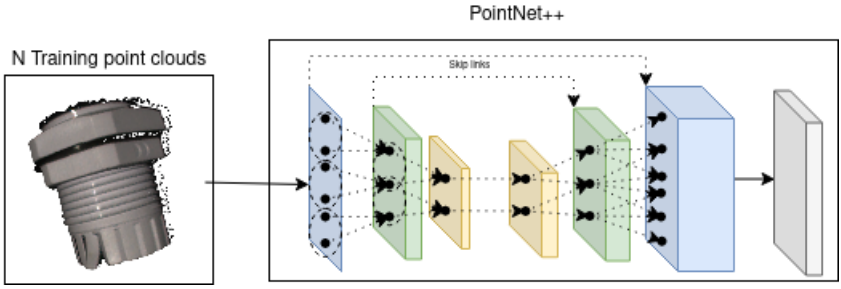

Figure 3.10: *PointNet++ Segmentation network.*

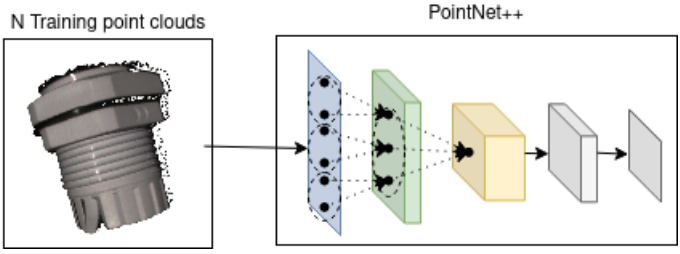

Figure 3.11: *PointNet++ Classification network.*

Part segmentation

The part segmentation network uses three set abstraction layers followed by three feature propagation layers. The three set abstraction layers use the same parameters as the classification network. The feature propagation layers use the groupings from the set abstraction layers to reconstruct the point cloud.

Semantic segmentation

The semantic segmentation network has a similar structure to the part segmentation network, the main difference is that it uses four set abstraction and feature propagation layers. The first set abstraction layer starts with sampling 1024 points with a radius of 0.1, the second layer samples 256 points with a radius of 0.2, the third layer samples 64 points with a radius of 0.4 and the last layer, samples 16 points with a radius of 0.8.

4

Method

4.1 Data

Most experiments were conducted on the MVTec 3D-AD public dataset [7]. The dataset was downloaded from MVTec's official website [1].

4.1.1 Aluminium Can Dataset

A real-world dataset consisting of aluminium cans was used to validate the methods. The data was collected using a Helios2 time-of-flight[2] sensor. The cans were placed approximately 1.2 meters from the sensor. 10 scans were taken per object at different rotations to get a sufficiently large dataset. The collected dataset consisted of a total of 120 samples of which 105 were used during training and 15 were used during testing. The training set included only normal samples and the test set included both normal and anomalous samples. The dataset included only XYZ-information.

4.1.2 Depth Map Preprocessing

The depth maps were created using the XYZ-information included in the MVTec 3D-AD dataset. Due to the low information content, the x and y values were scrapped and only the z-values were stored as an image. Since many of the depth maps had irregular backgrounds which could potentially result in false positives, a background removal inspired by Rudolph et al. [40] was used. First, the average depths of the 5×5 corner regions were calculated. With these four values, a background plane was interpolated and each pixel located less than 5mm away

[1] https://www.mvtec.com/company/research/datasets/mvtec-3d-ad
[2] https://thinklucid.com/product/helios2-time-of-flight-imx556/

from this plane was considered to be part of the background and therefore set to 0. This threshold was conservative as a higher threshold sometimes resulted in parts of the objects being removed. However, this solution instead caused some parts of the background remaining after the background plane removal. These parts were detected using connected-component labelling in the binary mask using the scikit-image library[3]. Only the largest connected component was kept in the resulting binary mask, therefore the assumption was made that the largest connected region was the object of interest. An illustration of the depth map preprocessing pipeline is given in Figure 4.1.

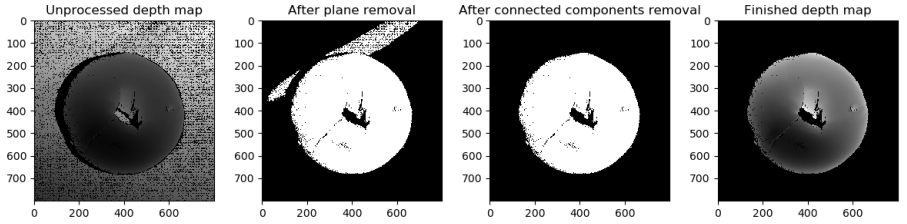

Figure 4.1: *The two-step preprocessing pipeline for depth maps.*

4.1.3 Point Cloud Preprocessing

In the first preprocessing step, a background plane was created by interpolating the points between the average position of the non-zero 5x5 corner points of the XYZ image. The difference compared to the depth map preprocessing is that the interpolated background contained XYZ information. All null values (stored as (0,0,0) values) were moved to this interpolated background. A segmentation mask was also created where any point within a 7mm distance from the interpolated background was considered as the background. Then, the point cloud was fed through a 2D adaptive pooling layer to decrease its size. Lastly, all points were normalized by setting the mean point to (0,0,0) and the maximum distance from the mean to 1. A step-by-step illustration of the preprocessing is given in Figure 4.2.

4.1.4 RGB Preprocessing

Unless specified otherwise, the RGB input to the backbone networks was resized to 224×224 using bilinear interpolation and normalized using ImageNet's normalization constants: mean = [0.485, 0.456, 0.406] and standard deviation = [0.229, 0.224, 0.225].

[3] `https://scikit-image.org/`

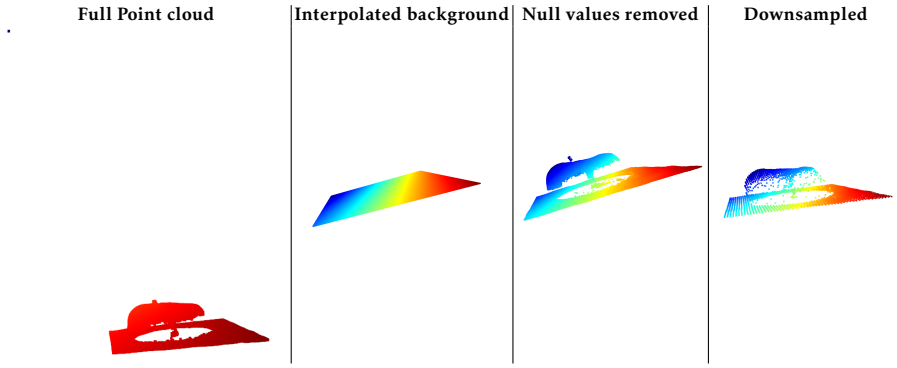

Figure 4.2: The step-by-step preprocessing pipeline for point clouds

4.2 Implementation

4.2.1 PaDiM

A lightweight Python implementation of PaDiM using PyTorch[4] was downloaded from GitHub[5] and was used as a baseline for this project. The MVTec 2D-AD dataset [6] was downloaded from the official website of MVTec[6] and a quick evaluation of the baseline was performed on the dataset. It was determined that the implementation achieved satisfactory results that were near the ones reported by the original authors of PaDiM [13]. The baseline was extended in the various ways described in the following sections to investigate the effects.

4.2.2 PaDiM Baseline

The baseline implementation of PaDiM was evaluated on the RGB data in the MVTec 3D-AD dataset using the backbones ResNet-18 and Wide ResNet-50 as proposed by the original authors [13]. These results were used to compare with the proposed methods.

4.2.3 PaDiM-3D

The baseline PaDiM implementation was extended to work on depth maps in the two different ways that are described below. The depth maps were repeated 3 times to create a 3-channel input required for the ResNet backbones.

Method A

In the first approach, a multivariate Gaussian distribution was learnt for each patch in the RGB images and depth maps independently and during inference

[4] https://pytorch.org/
[5] https://github.com/xiahaifeng1995/PaDiM-Anomaly-Detection-Localization-master
[6] https://www.mvtec.com/company/research/datasets/mvtec-ad

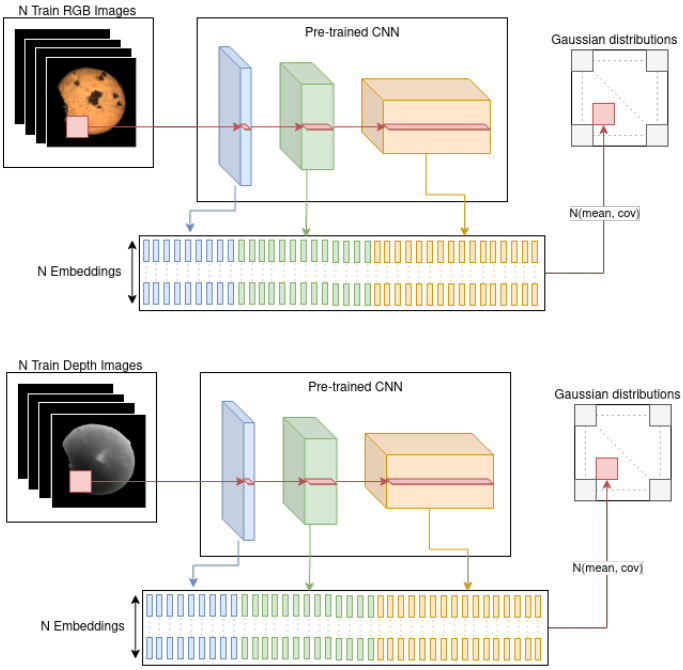

Figure 4.3: *A multivariate Gaussian distribution is learnt for the RGB images and the depth maps separately.*

the two anomaly score maps were summed to yield a final score map. The idea behind this method is that the RGB model would be more capable of detecting RGB-related anomalies such as discolouration, while the depth model would be more effective at finding anomalies such as cracks or holes. By summing their respective anomaly maps, both types of anomalies could be detected and the models would complement each other. An illustration of this architecture is given in Figure 4.3. Features were extracted from the RGB images and the depth maps using the same backbone model. As proposed by the original authors of PaDiM [13], 100 feature channels from ResNet-18 were sampled randomly from the first three convolutional layers and the spatial dimensions (56×56) of the first layer were used for the multivariate Gaussian matrix. The same spatial dimensions were used for Wide ResNet-50, but 550 randomly sampled channels from the first three convolutional layers were used instead.

Method B

In the second approach, the embedding vectors of the patches in the RGB images and depth maps were concatenated and only one model was trained on these new embedding vectors. An illustration of the architecture is given in Figure 4.4. The anomaly score maps were computed in the same way as in the traditional PaDiM. For ResNet-18, 100 feature channels were sampled randomly for RGB and depth

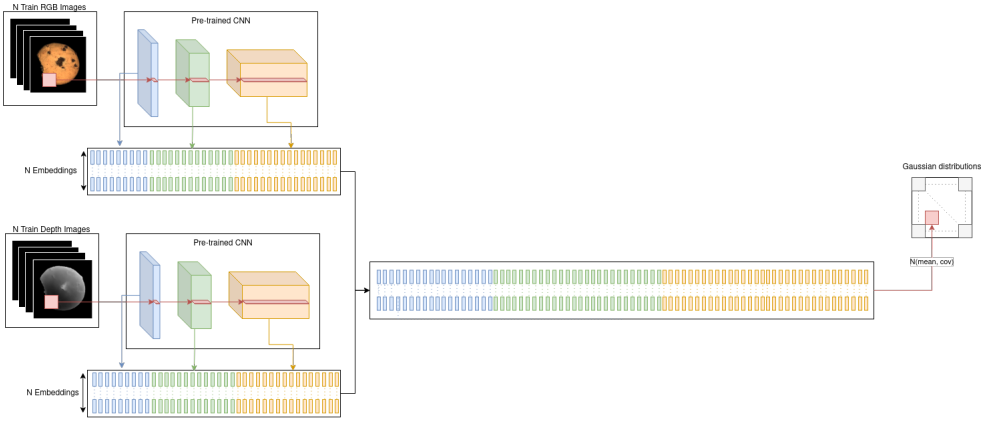

Figure 4.4: *The embedding vectors of the RGB images and depth images are concatenated.*

respectively resulting in a total of 200 channels after concatenation. For Wide ResNet-50, 550/2 = 275 channels were randomly sampled per input resulting in a total of 550 channels after concatenation.

4.2.4 PaDiM-ViT

PaDiM-ViT is a variant of PaDiM-3D in which the ResNet backbone is replaced with a vision transformer (ViT). Three different vision transformers were investigated, all of which share similar architectures. The vision transformer implementations used were borrowed from the timm[7] library and had pre-trained weights associated with them. The ViT model was pre-trained on ImageNet-21K [37] at an image resolution of 224 × 224 and fine-tuned on the smaller ImageNet-1K [42] using a resolution of 384 × 384. DeiT and CaiT were both pre-trained and fine-tuned on ImageNet-1K at resolutions 224 × 224 and 384 × 384 respectively. The vision transformers were used exclusively as feature extractors and were not further trained. PaDiM-ViT uses the same method A and method B as described in Section 4.2.3. PaDiM-ViT was tested on two different image resolutions, 224×224 and 384 × 384. The fine-tuned models were only used for the higher input resolutions. The same vision transformer was used to extract features from the RGB images and the depth maps. An illustration of PaDiM-ViT for method B is given in Figure 4.5.

ViT

The ViT used had a patch size of 16 × 16 and a feature dimension of 768. The CLS token was ignored, resulting in a total of 196 patches for 224 × 224 images and 576 patches for 384 × 384 images, which were reshaped into 14 × 14 and

[7] `https://huggingface.co/docs/timm`

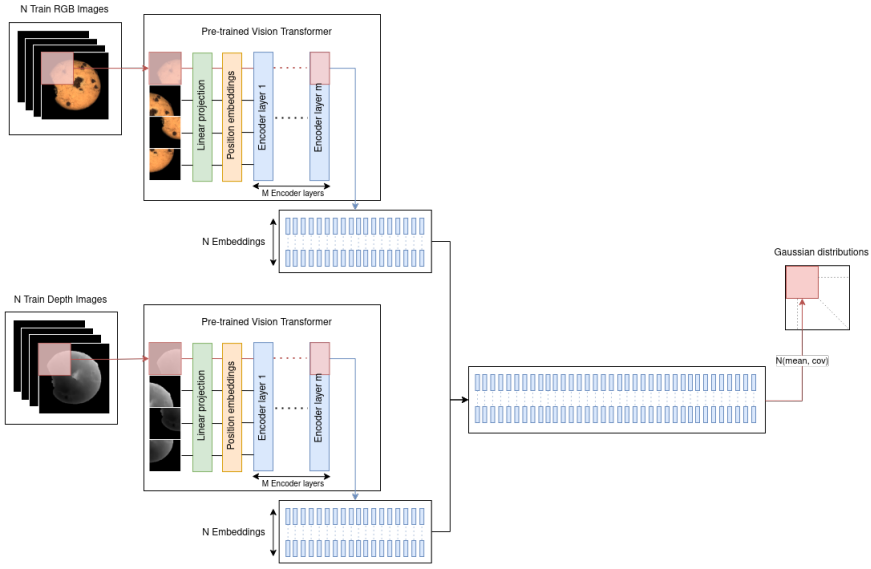

Figure 4.5: *A vision transformer is used to extract features instead of a ResNet.*

24 × 24 sized feature maps respectively. For method A, the full feature vectors were used and for method B, 768/2 = 384 channels were randomly sampled from the RGB and depth features respectively. This was done to reduce the feature dimensionality and saved both memory and time.

DeiT

The DeiT backbone had a patch size of 16 × 16 resulting in 196 patches for an input resolution of 224 × 224 and 576 patches for an input resolution of 384 × 384. Each patch had 768 feature channels. The class and distillation tokens were ignored. For method B, 768/2 = 384 channels were randomly sampled from each of the RGB features and depth features, which resulted in a total of 768 features after concatenation.

CaiT

The CaiT architecture was composed of 24 transformer encoder layers and only the features from the patch self-attention part of the network were used. The image was divided into 16 × 16 patches yielding a total of 196 patches for 224 × 224 images and 576 patches for 384 × 384 images. The full-sized feature vectors were used in both method A and method B as the feature dimension for CaiT is lower than for the other vision transformers. This resulted in the feature dimension 384 for RGB and depth respectively.

4.2.5 Investigating Possible Improvements to the Depth Methods

Object Alignment

The effect of first translating the object to the center of the image and then rotating it to the x-axis was studied. Since PaDiM learns what is normal for each patch, it was hypothesized that this would improve the ability to learn. To perform this operation, the resulting foreground mask obtained by the depth map preprocessing described in Section 4.1.2 was used to compute the appropriate transformation. The scikit-image library[8] was used to find the center of mass of the object that could then be aligned with the center of the image using translation. To determine the angle of rotation, scikit-image computes the covariance matrix of the image intensity along the image axes. The eigenvector of the covariance matrix with the largest eigenvalue points in the direction along the largest covariance. The angle between this vector and the x-axis was picked as the rotation angle. This resulted in an object effectively having two possible orientations instead of an infinite number. This was due to the axis sometimes facing 180 degrees in the opposite direction.

Gap Filling on the Depth Maps

A common issue with the depth maps were the missing values that could result in false positives, these pixels always had the value 0. Gap filling was investigated as a method to address these. The foreground mask acquired from the foreground extraction described in section 4.1.2 was used as a starting point. Next, binary closing[9] was applied on the mask to fill the holes inside the foreground. By taking the pixels that were 0 in the original mask and 1 in the closed mask, the missing values inside the object were obtained. These values were repeatedly filled over 3 iterations using the mean of the surrounding valid pixel intensities [40].

Zeroing the Background in the RGB Images

In the same way as described in the previous section, binary closing was applied to the foreground mask obtained from the preprocessing. Element-wise multiplication between the RGB images in the MVTec 3D-AD dataset and the new mask was done to zero the background in the RGB images. This removed the inconsistent lighting that some classes had.

Histogram Equalization on the Depth Maps

It was hypothesized that increasing the contrast in the depth maps could help the model with detecting anomalies. Adaptive equalization from the Scikit-image library[10] was added to the depth maps. This made small surface variations more apparent.

[8] https://scikit-image.org/
[9] https://scikit-image.org/docs/stable/api/skimage.morphology.html
[10] https://scikit-image.org/docs/stable/api/skimage.exposure.html

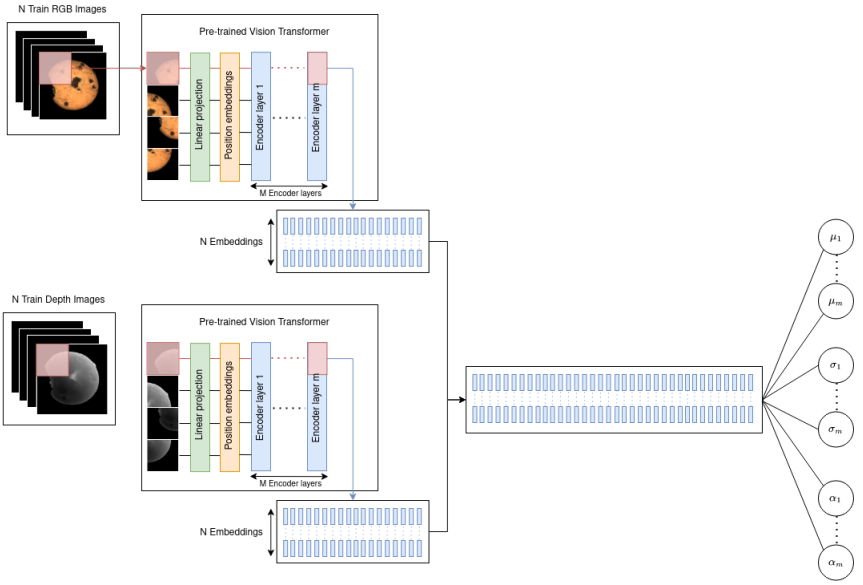

Figure 4.6: *The extracted features are fed into an MDN that learns the parameters of multiple Gaussian distributions.*

4.2.6 Mixture Density Network for Density Estimation

The outputs of the feature extractor were fed into a MDN consisting of three linear layers that replaced the MVG as the density estimator. Only the parameters of the Gaussian mixture model were fit during training. Let N be the number of patches and D be the dimensionality of the feature vector for each patch. The input vectors stored in a matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ were linearly transformed using the three weight matrices: $\mathbf{W}_\mu \in \mathbb{R}^{D \times (D \cdot m)}$, $\mathbf{W}_\alpha \in \mathbb{R}^{D \times (D \cdot m)}$ and $\mathbf{W}_\sigma \in \mathbb{R}^{D \times (D \cdot m)}$, where m is the total number of mixture components. The matrices $\boldsymbol{\mu} = \mathbf{X}\mathbf{W}_\mu \in \mathbb{R}^{N \times (D \cdot m)}$ and $\boldsymbol{\sigma} = \mathbf{X}\mathbf{W}_\sigma \in \mathbb{R}^{N \times (D \cdot m)}$ contained the D-dimensional mean and standard deviation for each mixture Gaussian distribution. $\boldsymbol{\alpha} = \mathbf{X}\mathbf{W}_\alpha \in \mathbb{R}^{N \times (D \cdot m)}$ contained the mixture coefficient for each Gaussian distribution. The models were trained using the Adam optimizer [22] with a learning rate of 0.0001 and a weight decay of 0.0001. A batch size of 2 was used.

During inference, the log-likelihood of the patches belonging to the estimated distribution was used. A low log-likelihood indicated a high probability of an anomaly. In contrast to the MVG approach, a distribution per patch was not learnt, as it was practically infeasible to train a separate model per patch. Instead, a single distribution over all the patches was learnt. An illustration of PaDiM-ViT-B with the MDN is given i Figure 4.6. The covariance between the features was assumed to be zero, as the computational cost would become too high otherwise. Furthermore, as pointed out by Bishop [8], given that the correct number of mixture components is chosen, it is not necessary to have full covariance to estimate

the density of any distribution.

4.2.7 Point-PaDiM

The Point-PaDiM method replaces the CNN backbone of the PaDiM method with a PointNet++ backbone. This is so that PaDiM may work on point clouds instead of images. Pre-trained PointNet++ networks were acquired from ModelZoo[11] where three networks were used: classification, part segmentation and semantic segmentation. For all three networks, the points were preprocessed according to Section 4.1.3 and then fed to the network; the abstracted positions and features are extracted from the layers. An illustration of this process is presented in Figure 4.7. Each point extracted from the largest layer was defined as a patch. After a list of patches and features were collected, the mean and covariance matrix of a multivariate Gaussian was calculated for each patch.

To calculate the anomaly scores using point cloud data, the point cloud went through the same preprocessing steps as during training, but before the distribution was calculated. Then, the Mahalanobis distance between the features and the multivariate Gaussian was first calculated to get the anomaly score for each patch. Then, two approaches were used depending on the structure of the data: reshaping or point placement. Reshaping was done if the output had a square size and kept the image structure then reshaped the scores into a map. This was possible as the order of the points was the same between the input and output. Point placement instead approximates which index corresponds to the position of a point. If the maximum x and y values are known, the 0-1 normalized values for x and y coordinates were used to calculate the x and y indices in the anomaly map, by multiplying the x or y value with the size of the anomaly map. The point placement method was only used if the reshape method could not be used, as the reshape method correctly maps the indices to the scores in the anomaly map.

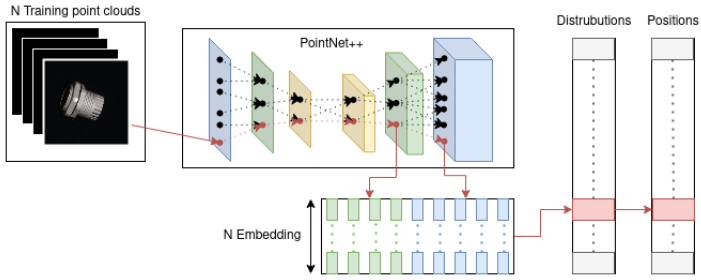

Figure 4.7: *Learning distribution with PointNet++ backbone (part segmentation)*

[11] https://modelzoo.co/model/pointnet-pointnet2-pytorch

PointNet++ Classification MSG

The classification network was pre-trained to classify the point clouds in the ModelNet40 dataset [55] and uses the MSG method to sample points in the point abstraction layers. None of the set abstraction layers has a square sampling size, thereby breaking the image structure. Therefore the point placement method had to be used to create the anomaly map.

PointNet++ Part Segmentation

The part segmentation network was pre-trained on the ShapeNet dataset [10] consisting of 16 different classes of objects. Each object is divided into 50 parts. The output of the part segmentation network is the same size as the input. If the image structure is maintained for the input by using 2D downsampling, then the anomaly map can be created using reshaping. Otherwise, if the image structure is broken for the input then point placement has to approximate the anomaly map.

PointNet++ Semantic Segmentation

The semantic segmentation network was pre-trained on the S3DIS dataset [1] to separate 13 different objects in an environment. The reshape method was used when the input had a square size, otherwise the point placement method was used.

4.2.8 Investigation of Point Cloud Improvements

Downsampling the Input

The final layer in the segmentation network reconstructs the size of the input with 128 features per point. For example, the Peach class in the MVTec 3D-AD dataset has 361 samples, each consisting of 360000 (600x600) points and 128 features. This requires a lot of memory to store, therefore the point cloud needed to be downsampled to reduce the memory consumption. Four different downsampling methods were tested:

- **IDX**: Selecting random points by the indices
- **FPS**: Farthest point sampling
- **BQAVG**: FPS with the mean Ball Query point
- **BILIN**: Bilinear downsizing
- **AVG**: 2D Adaptive average pooling

The methods are divided into two categories: IDX and FPS which select individual points, and BILIN, BQAVG and AVG which combine groups of points. The point selection sampling methods must use the point placement method when creating the anomaly maps. Although the sampling size may be square-shaped, the sampling breaks the image structure of the data.

Layer Concatenation

Similar to the original PaDiM, multiple layers in the PointNet++ backbone were concatenated to include more channels. The output from a layer in PointNet++ does not contain spatial information like CNNs, as the positions of the points in the point cloud are kept in a separate list. To concatenate the layers, the Euclidean distance between the points in each layer was calculated, where the channels of the closest points in each layer were concatenated together.

Segmentated Background

By using the segmentation mask created during the preprocessing of the point cloud, the points in the background were removed. As PointNet++ does not need a square-shaped input, the data in the background could be removed. This method is intended to reduce the distraction of the background points. As the number of points that an object consists of is inconsistent between samples, 1D average pooling was used to get a manageable and consistent size. To create the anomaly map, the point placement method was used because the original data structure was lost.

Combine with RGB

The PointNet++ backbone only uses XYZ data which may result in the model missing colour-related anomalies such as discolouration. To include colour information in model, PointNet++ and ResNet were combined. The backbones were combined using the methods A and B from PaDiM-3D, described in Section 4.2.3. Because a different backbone was used for each input type, the two anomaly maps in method A were normalized with a minimum value of 0 and a mean of 1 before being combined. When using the B method the output shape from the PointNet++ network has to be $56 \times 56 \times C$, to be able to concatenate the channels with the ResNet18 output.

Channel selection

In the original PaDiM framework when using the ResNet-18 backbone, 100 channels are selected at random to lower the complexity of the model. The last layer in the PointNet++ segmentation network has 128 channels, using all channels instead of 100 channels may lead to a performance increase without a large increase in the complexity of the model.

5

Evaluation

5.1 Metrics

The three evaluation metrics used to measure the performance of the models were: Image AUROC, pixel AUROC and AUPRO. Image AUROC measures the anomaly detection performance on an image level, while the remaining two metrics focus on anomaly localization on a pixel level. Anomaly detection and localization can be viewed as a binary classification task where a *positive* corresponds to an anomaly and a *negative* represents a non-anomaly. In the MVTec 3D-AD dataset, each sample is associated with a binary ground truth mask where the value of 1 indicates an anomaly and the value of 0 indicates a non-anomaly. Therefore, a true positive (TP) on an image level is when the model predicts the sample as being anomalous and the associated ground truth mask for that sample has at least one pixel value that is equal to 1. A TP on a pixel level is when a pixel is predicted as anomalous and the corresponding pixel value in the ground truth mask is equal to 1. Based on this logic, the definitions for false positive (FP), false negative (FN) and true negative (TN) can also be deduced. The false positive rate (FPR) can then be defined as:

$$FPR = \frac{FP}{FP + TN} \tag{5.1}$$

and the true positive rate (TPR) by:

$$TPR = \frac{TP}{TP + FN} \tag{5.2}$$

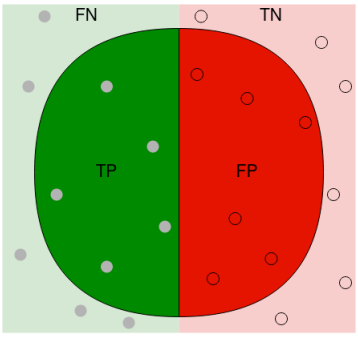

Figure 5.1: *Illustrates the FN, TP, TN and FP relation to each other*

Area Under the Receiver Operating Characteristic

The receiver operating characteristic (ROC) measure compares the true positive rate (TPR) and the false positive rate (FPR) for varying thresholds. The area under the ROC (AUROC) measures how these thresholds influence the model performance [49].

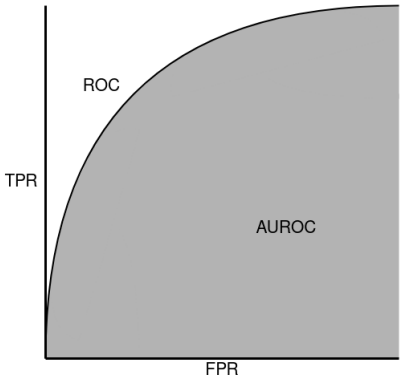

Figure 5.2: *Illustrates the AUROC curve*

Area under the Per Region Overlap

As pixel AUROC favours large anomalies, the AUPRO score was suggested by Bergmann et al. [6] as an alternative measure that scores different-sized anomalous regions equally. The PRO curve plots the PRO against the FPR and is also measured over varying thresholds. PRO considers $C_{i,k}$ as the set of all pixels in each connected component k in the ground truth mask i. The set of all pixels that are predicted as anomalous for a threshold t is denoted by P_i. Furthermore, let N denote the total number of connected components in the evaluation dataset. The PRO is computed with the following equation [6]:

$$PRO = \frac{1}{N} \sum_{i} \sum_{k} \frac{|P_i \cap C_{i,k}|}{|C_{i,k}|} \tag{5.3}$$

Similarly to the AUROC measure, the area under the PRO curve (AUPRO) is calculated and an integration limit is often set at a decided FPR. In this project, this limit was set to 0.3.

Visual Inspection

To make it easier to interpret the number results obtained from the previous metrics, an anomaly heat map containing the normalized anomaly scores was created for each test sample.

5.2 Experiments

5.2.1 Accuracy

Image AUROC was used to evaluate anomaly detection and pixel AUROC and AUPRO were used to measure anomaly localization. AUPRO was used as the main localization metric, but the pixel AUROC metric made it possible to compare to previous studies. The best performing methods were measured over five runs and the mean and standard deviations were reported. Some possible improvements were also investigated for the depth and point cloud methods respectively.

5.2.2 Inference Time

The average inference time in seconds across all 10 classes in MVTec 3D-AD was measured on the hardware-specific equipment mentioned in Section 1.4. The time of loading the weights from disk into memory was not included in the calculation. The time in-between the point when the weights had been filly loaded into memory to the point when the anomaly score map generation was complete was measured. The *time*[1] module in Python was used.

5.2.3 Resolution

To simulate a lower-resolution sensor, downsizing using nearest neighbour interpolation was done on the input data. The models were tested on the four resolution settings: 400x400, 300x300, 200x200 and 100x100.

[1] `https://docs.python.org/3/library/time.html`

6
Results

6.1 PaDiM Baseline

The results of evaluating PaDiM on the RGB images in the MVTec-3D-AD dataset are presented in Table 6.1, and serve as baselines to the 3D methods. Wide ResNet-50 significantly outperformed the smaller ResNet-18 in all categories, except for in image AUROC on Tire. Moreover, the class that received the greatest relative improvement compared to ResNet-18 was Rope.

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Mean
I-ROC	Res18	96.1	74.8	55.1	54.2	85.4	57.2	67.8	46.6	73.8	**77.8**	68.9
	WideRes50	**97.7**	**79.7**	**70.7**	**60.2**	**95.6**	**69.9**	**85.3**	**56.0**	**94.6**	77.4	**78.7**
P-ROC	Res18	99.1	96.8	97.0	96.3	98.3	87.6	98.2	96.5	88.8	93.6	95.2
	WideRes50	**99.3**	**97.9**	**98.2**	**97.6**	**99.0**	**92.7**	**99.0**	**97.9**	**99.1**	**97.5**	**97.8**
PRO	Res18	95.9	90.3	90.0	88.7	92.8	68.6	93.7	88.0	58.7	74.9	84.2
	WideRes50	**97.2**	**93.9**	**93.8**	**92.0**	**95.4**	**77.3**	**96.4**	**92.8**	**93.1**	**89.6**	**92.2**

Table 6.1: *Results of PaDiM with ResNet-18 and Wide ResNet-50 as feature extractors.*

6.2 PaDiM-3D

The results of PaDiM-3D are presented in Table 6.2. The depth information was more beneficial for ResNet-18, than for Wide ResNet-50 and the larger model significantly outperformed the smaller one. The classes Cookie, Potato and Peach received the greatest relative improvement compared to the RGB baselines, while Cable Gland, Dowel and Tire benefited the least from the added depth information. Method A marginally outperformed method B for Wide ResNet-50, but the

opposite effect was observed for ResNet-18. Three of the most difficult classes were Tire, Cable Gland and Foam, all of which received AUPRO scores below 90%. With the exception of average AUPRO for Res18-224-A, the PaDiM-3D methods outperform the baselines in all three average metrics. Moreover, WideRes50-224-A scored the highest out of the PaDiM-3D methods.

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Mean
I-AUROC	Res18-224-A	99.3	73.4	63.2	91.7	87.0	68.6	86.8	64.5	89.7	63.9	78.8
	Res18-224-B	99.1	**76.5**	64.3	94.2	89.1	63.4	88.8	68.5	88.9	68.5	80.1
	WideRes50-224-A	**99.8**	75.6	70.0	**96.7**	**94.1**	77.2	**94.0**	**72.1**	94.9	**71.9**	**84.6**
	WideRes50-224-B	**99.8**	74.6	**72.6**	94.2	93.5	**78.1**	88.7	71.9	**95.9**	71.2	84.0
P-AUROC	Res18-224-A	99.4	94.8	97.8	99.0	98.2	94.1	99.3	98.7	98.6	94.8	97.5
	Res18-224-B	99.4	95.7	98.1	99.1	98.6	94.8	99.4	99.0	98.6	95.9	97.8
	WideRes50-224-A	**99.5**	96.0	**98.7**	**99.2**	98.9	**95.8**	**99.6**	99.2	**99.3**	**97.2**	**98.3**
	WideRes50-224-B	**99.5**	**96.4**	98.6	**99.2**	**99.0**	95.2	99.5	**99.3**	**99.3**	97.1	**98.3**
AUPRO	Res18-224-A	98.1	84.6	92.7	97.2	92.3	81.8	97.6	95.9	92.1	79.7	91.2
	Res18-224-B	98.1	87.1	93.5	97.5	93.9	83.9	98.0	96.8	91.3	83.6	92.4
	WideRes50-224-A	98.2	87.7	**95.2**	97.8	95.0	**86.2**	**98.4**	97.4	95.6	**88.8**	**94.0**
	WideRes50-224-B	**98.4**	**88.5**	94.9	**97.9**	**95.4**	84.4	98.1	**97.6**	**95.8**	88.0	93.9

Table 6.2: *Results of PaDiM-3D.*

6.3 PaDiM-ViT

The results of PaDiM-ViT are presented in Table 6.3. This includes the results for all vision transformer architectures that were investigated. Out of the two methods investigated, method B generally performed the best. At a high resolution, DeiT and CaiT surpassed both of the baselines in anomaly detection and localization by a significant margin. ViT achieved higher anomaly detection than Wide ResNet-50 however, was not able to beat the baseline in anomaly localization. Bagel, Cookie and Dowel achieved some of the highest image AUROC scores, while Tire, Potato and Carrot received some of the lowest. Despite their low anomaly detection scores, Carrot and Potato had relatively high anomaly localization scores. DeiT-384-B achieved the best performance in average image AUROC, pixel AUROC and AUPRO out of the vision transformer methods.

6.4 Comparison Between PaDiM-3D and PaDiM-ViT

PaDiM-3D outperformed the vision transformers in anomaly localization at an input resolution of 224×224. Furthermore, the Wide ResNet-50 methods also scored higher than the vision transformer methods in anomaly detection at this input resolution. However, at a higher resolution, DeiT and CaiT outperformed all PaDiM-3D methods in both anomaly detection and localization. PaDiM-3D did not scale up to higher input resolutions, therefore could not be tested at resolution 384×384.

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Mean
I-AUROC	ViT-224-A	96.7	77.7	67.6	95.4	89.9	76.6	87.0	63.7	77.9	67.4	80.0
	ViT-224-B	96.8	77.3	68.3	93.8	89.9	77.6	88.6	67.9	67.5	66.1	79.4
	ViT-384-A	96.1	84.6	74.2	94.5	86.6	77.3	90.4	71.1	70.4	62.9	80.8
	ViT-384-B	96.7	86.6	74.9	93.5	87.2	80.8	**91.8**	**74.4**	70.9	60.6	81.7
	CaiT-S24-224-A	94.4	79.9	68.3	86.6	92.6	74.8	77.6	53.7	97.2	70.8	79.6
	CaiT-S24-224-B	93.8	78.7	71.2	90.1	93.5	73.8	85.3	58.1	95.9	69.7	81.0
	CaiT-S24-384-A	96.8	87.2	71.1	86.5	94.5	81.7	83.2	61.2	97.5	**79.6**	83.9
	CaiT-S24-384-B	96.4	**88.9**	74.4	90.6	**94.6**	**82.3**	91.4	65.1	**98.2**	76.6	85.9
	DeiT-224-A	97.1	82.4	69.7	96.5	94.5	76.3	87.5	58.9	97.1	70.3	83.0
	DeiT-224-B	97.6	82.0	71.4	**96.8**	94.2	78.9	89.3	63.8	95.9	69.9	84.0
	DeiT-384-A	99.4	85.7	73.1	96.3	93.4	79.0	90.8	71.5	96.0	69.4	85.5
	DeiT-384-B	**99.6**	85.4	**75.6**	96.0	93.4	79.6	91.5	71.8	96.1	72.2	**86.0**
P-AUROC	ViT-224-A	98.6	94.3	94.3	97.3	97.4	86.2	98.0	97.1	96.8	90.2	95.0
	ViT-224-B	98.4	94.7	94.2	97.1	97.5	85.9	98.1	97.4	96.8	89.8	95.0
	ViT-384-A	99.4	96.4	96.7	98.1	98.1	92.4	98.7	98.6	96.3	93.1	96.8
	ViT-384-B	99.3	96.7	96.9	97.8	98.3	92.6	98.8	98.8	96.5	93.1	96.9
	CaiT-S24-224-A	99.1	95.6	96.6	98.4	97.5	90.7	98.6	97.7	98.4	95.1	96.8
	CaiT-S24-224-B	99.1	96.1	97.3	98.4	98.0	91.2	98.9	98.3	98.4	95.5	97.1
	CaiT-S24-384-A	99.6	98.2	98.7	99.0	98.6	93.4	99.5	98.9	98.3	**98.1**	98.2
	CaiT-S24-384-B	99.6	**98.5**	**99.0**	99.0	**98.9**	93.5	**99.7**	99.2	98.8	97.5	98.4
	DeiT-224-A	99.3	95.2	97.5	98.9	98.2	93.9	99.1	98.3	99.0	96.1	97.6
	DeiT-224-B	99.3	95.6	97.7	98.8	98.4	94.1	99.2	98.6	99.1	96.4	97.7
	DeiT-384-A	**99.7**	97.2	98.9	**99.2**	98.8	**95.8**	**99.7**	99.3	**99.2**	97.4	98.5
	DeiT-384-B	**99.7**	97.6	**99.0**	**99.2**	**98.9**	**95.8**	**99.7**	**99.4**	**99.2**	97.4	**98.6**
AUPRO	ViT-224-A	91.9	82.0	81.0	90.9	89.0	64.8	91.8	89.1	80.3	66.0	82.7
	ViT-224-B	91.2	83.2	80.8	90.3	89.1	64.4	92.4	90.0	79.8	64.6	82.6
	ViT-384-A	97.0	88.6	89.0	94.9	92.4	76.4	95.6	94.6	82.4	73.8	88.5
	ViT-384-B	96.8	89.3	89.4	94.3	92.8	76.8	95.7	95.4	83.0	73.6	88.7
	CaiT-S24-224-A	95.6	86.6	88.4	93.9	88.6	74.5	94.2	90.3	89.1	81.0	88.2
	CaiT-S24-224-B	95.7	88.0	90.7	94.4	90.4	75.3	95.7	92.7	89.1	82.2	89.4
	CaiT-S24-384-A	98.3	94.5	95.5	96.9	93.9	80.4	98.1	94.9	90.7	**92.1**	93.5
	CaiT-S24-384-B	98.3	**95.6**	**96.6**	97.1	95.1	80.4	98.6	96.3	95.2	89.2	94.2
	DeiT-224-A	96.3	85.2	91.3	96.0	92.1	80.2	96.2	92.7	92.9	86.1	90.8
	DeiT-224-B	96.3	86.4	92.2	96.0	92.7	81.3	96.6	94.1	93.4	86.2	91.5
	DeiT-384-A	**98.9**	91.4	95.9	**97.8**	95.1	**86.4**	**98.8**	96.9	95.7	89.5	94.6
	DeiT-384-B	98.8	92.6	96.2	**97.8**	**95.5**	86.3	**98.8**	**97.3**	**96.0**	89.5	**94.9**

Table 6.3: *Transformer results.*

6.5 Visual Inspection of the Depth Methods

Figure 6.1 shows three examples of heat maps created by Wide ResNet-50 baseline, WideRes50-224-B and DeiT-384-B. Some of these anomalies are difficult to spot by simply looking at the RGB images, but are more evident in the depth images. This is reflected in the results where the Wide ResNet-50 baseline does not successfully detect all the anomalies in examples one and two, and does not locate the anomaly in the third example.

6.6 Investigating Possible Improvements for the Depth Methods

The results of the possible improvements described in Section 4.2.5 are presented below. The effects were only tested on DeiT-384-B as this method achieved the best performance across Tables 6.1, 6.2 and 6.3. Object alignment had the biggest

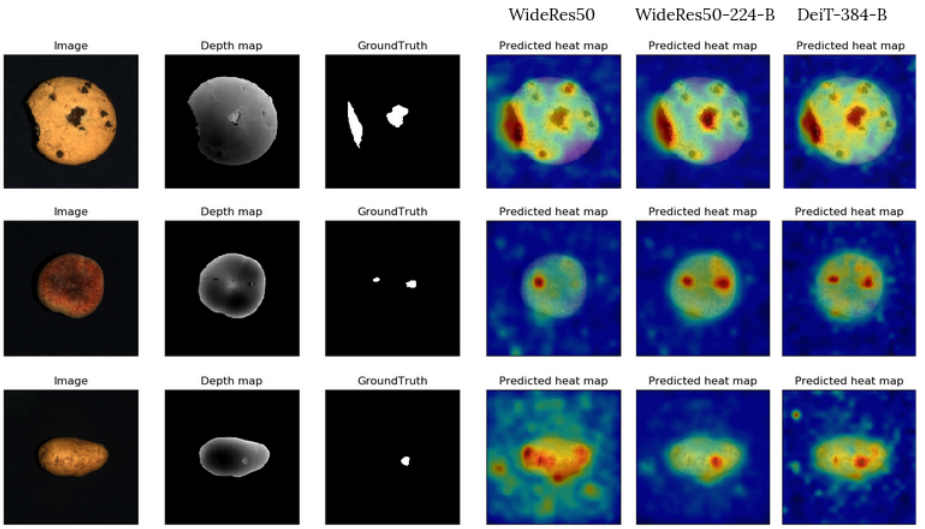

Figure 6.1: *Examples of the heat maps generated by the Wide ResNet-50 baseline and two depth methods.*

impact on the results out of the improvements considered. Cable Gland, Carrot, Dowel and Tire achieved large improvements in both anomaly detection and localization compared to the standard model. Foam, Bagel and Cookie did not benefit from the object alignment. Setting the background to zero had little effect on the performance. Gap filling and contrast enhancing both increased the ability to detect anomalies and the latter also displayed a significant improvement to localizing them.

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Mean
I-AUROC	DeiT-384-B	**99.6**	85.4	75.6	96.0	93.4	79.6	91.5	71.8	96.1	72.2	86.0
	DeiT-384-B-Align	98.8	**87.4**	**82.4**	96.7	**96.9**	73.5	94.8	73.4	94.2	**82.0**	**88.0**
	DeiT-384-B-ZeroBG	99.4	85.8	74.7	96.4	93.2	80.9	91.0	73.5	95.1	73.8	86.4
	DeiT-384-B-GF	98.9	81.0	81.4	**98.3**	95.7	83.6	**96.2**	70.0	**96.7**	71.3	87.3
	DeiT-384-B-HEq	98.0	83.5	77.8	96.0	95.3	81.6	94.2	**75.8**	96.1	74.8	87.3
	DeiT-384-B-ZeroBG-GF-HEq	98.4	81.1	80.2	96.6	93.8	**85.3**	95.3	71.9	95.9	76.3	87.5
P-AUROC	DeiT-384-B	**99.7**	97.6	99.0	**99.2**	98.9	95.8	99.7	99.4	99.2	97.4	98.6
	DeiT-384-B-Align	99.6	**98.9**	**99.3**	**99.2**	**99.6**	92.9	99.7	99.4	**99.3**	**99.6**	**98.7**
	DeiT-384-B-ZeroBG	99.6	97.6	99.0	99.0	98.6	96.2	99.7	99.4	99.1	99.1	98.6
	DeiT-384-B-GF	**99.7**	97.6	99.2	99.1	98.9	95.1	**99.8**	99.4	**99.3**	97.4	98.6
	DeiT-384-B-HEq	99.6	97.6	99.1	99.0	99.0	96.4	99.7	**99.5**	99.2	98.0	**98.7**
	DeiT-384-B-ZeroBG-GF-HEq	99.6	97.8	**99.3**	98.4	98.7	**96.7**	**99.8**	**99.5**	99.2	98.4	**98.7**
AUPRO	DeiT-384-B	**98.8**	92.6	96.2	**97.8**	95.5	86.3	98.8	97.3	96.0	89.5	94.9
	DeiT-384-B-Align	98.6	**96.5**	**97.7**	**97.8**	**98.0**	78.3	98.9	97.4	96.4	**98.0**	**95.8**
	DeiT-384-B-ZeroBG	98.6	92.8	96.4	97.4	94.8	87.3	98.9	97.2	94.6	91.6	95.0
	DeiT-384-B-GF	98.7	92.7	97.3	97.9	95.7	83.5	**99.3**	97.5	**96.9**	89.4	94.9
	DeiT-384-B-HEq	**98.8**	92.8	96.6	97.5	95.8	87.4	99.0	**97.9**	96.2	91.8	95.4
	DeiT-384-B-ZeroBG-GF-HEq	98.6	93.3	97.4	96.2	95.1	**88.5**	99.2	97.8	95.5	93.3	95.5

Table 6.4: *Results of possible improvements of the depth-based methods.*

6.7 Mixture Density Network

6.7.1 DeiT

The result of DeiT-384-B with a MDN with 50 Gaussian components, trained for 400 epochs is presented in Table 6.5. A large improvement in image AUROC was observed for Carrot and Potato, and a small improvement was obtained for Dowel and Peach. Despite the large improvement in performance on some classes, not all classes benefited from the neural density estimator which ultimately brought down the average score. Two of such classes were Tire and Foam.

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Mean
I-ROC	DeiT-384-B	**99.6**	**85.4**	75.6	**96.0**	93.4	**79.6**	91.5	71.8	**96.1**	**72.2**	86.0
	DeiT-384-B-MDN	98.3	81.9	**94.2**	91.0	**93.9**	78.3	**93.0**	**90.5**	94.9	58.5	**87.5**
P-ROC	DeiT-384-B	**99.7**	97.6	99.0	**99.2**	98.9	**95.8**	**99.7**	99.4	99.2	**97.4**	**98.6**
	DeiT-384-B-MDN	**99.7**	**97.8**	**99.5**	99.0	**99.2**	93.4	99.6	**99.7**	**99.4**	97.0	98.4
PRO	DeiT-384-B	**98.8**	92.6	96.2	**97.8**	95.5	**86.3**	**98.8**	97.3	96.0	**89.5**	**94.9**
	DeiT-384-B-MDN	98.6	**93.6**	**97.9**	97.5	**96.6**	79.4	98.6	**98.8**	**96.9**	88.3	94.6

Table 6.5: *Results for MDN with 50 Gaussian components trained for 400 epochs.*

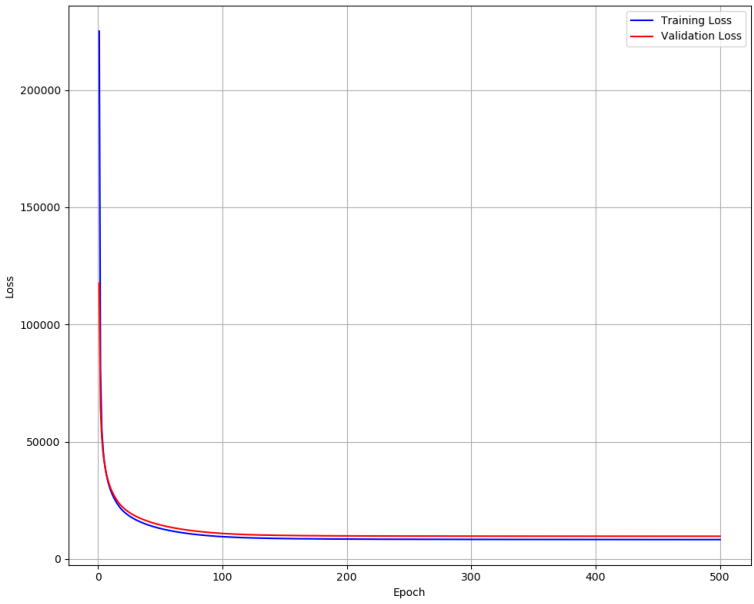

Figure 6.2: *Training and validation losses for a MDN with 150 mixture components trained for 500 epochs on Cable Gland.*

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Mean
Image AUROC	CaiT-S24-384-B	**96.4**	88.9	74.4	**90.6**	**94.6**	82.3	91.4	65.1	**98.2**	**76.6**	85.9
	CaiT-S24-384-B-MDN$_{\|100}$	94.1	88.6	**97.3**	88.1	89.9	84.7	93.2	79.2	98.1	76.0	88.9
	CaiT-S24-384-B-MDN$_{\|200}$	95.1	89.6	97.2	88.3	90.7	85.5	**93.4**	79.3	98.1	**76.6**	**89.4**
	CaiT-S24-384-B-MDN$_{\|300}$	95.6	89.5	97.2	88.1	90.3	85.7	**93.4**	79.1	98.1	76.1	89.3
	CaiT-S24-384-B-MDN$_{\|400}$	95.5	**89.8**	97.0	87.9	89.9	85.3	93.1	**79.4**	98.1	**76.6**	89.3
	CaiT-S24-384-B-MDN$_{\|500}$	95.7	89.7	97.0	87.9	89.9	**85.8**	93.3	79.3	98.1	76.4	89.3
Pixel AUROC	CaiT-S24-384-B	99.6	98.5	99.0	**99.0**	98.9	**93.5**	**99.7**	99.2	98.8	97.5	**98.4**
	CaiT-S24-384-B-MDN$_{\|100}$	99.6	**99.1**	**99.4**	98.9	99.2	89.8	99.6	**99.5**	**99.2**	97.5	98.2
	CaiT-S24-384-B-MDN$_{\|200}$	99.6	**99.1**	**99.4**	98.9	**99.3**	89.6	99.6	**99.5**	**99.2**	**97.6**	98.2
	CaiT-S24-384-B-MDN$_{\|300}$	99.6	**99.1**	**99.4**	98.9	**99.3**	89.5	99.6	**99.5**	**99.2**	**97.6**	98.2
	CaiT-S24-384-B-MDN$_{\|400}$	99.6	**99.1**	**99.4**	98.9	**99.3**	89.5	99.6	**99.5**	**99.2**	**97.6**	98.2
	CaiT-S24-384-B-MDN$_{\|500}$	99.6	**99.1**	**99.4**	98.9	**99.3**	89.5	99.6	**99.5**	**99.2**	**97.6**	98.2
AUPRO	CaiT-S24-384-B	**98.3**	95.6	96.6	**97.1**	95.1	**80.4**	**98.6**	96.3	95.2	89.2	**94.2**
	CaiT-S24-384-B-MDN$_{\|100}$	97.5	97.2	**98.0**	96.3	96.3	72.6	98.4	97.3	**97.4**	89.8	94.1
	CaiT-S24-384-B-MDN$_{\|200}$	97.6	97.2	**98.0**	96.3	**96.4**	72.4	98.4	**97.4**	**97.4**	**90.0**	94.1
	CaiT-S24-384-B-MDN$_{\|300}$	97.6	**97.3**	**98.0**	96.3	**96.4**	72.3	98.4	**97.4**	**97.4**	**90.0**	94.1
	CaiT-S24-384-B-MDN$_{\|400}$	97.6	**97.3**	**98.0**	96.3	**96.4**	72.2	98.4	**97.4**	**97.4**	**90.0**	94.1
	CaiT-S24-384-B-MDN$_{\|500}$	97.6	**97.3**	**98.0**	96.3	**96.4**	72.3	98.4	**97.4**	**97.4**	**90.0**	94.1

Table 6.6: *Results for MDN with 150 Gaussian components trained for different epochs.*

6.7.2 CaiT

The results of CaiT-S24-384-B with a MDN with 150 mixture components trained for 100, 200, 300, 400 and 500 epochs respectively are presented in Table 6.6. On average, the anomaly detection improved significantly with the neural density estimator. However, a small decrease in average anomaly localization performance was observed. The loss curves for Cable Gland is presented in Figure 6.2, which displays the loss over epoch for the training and the validation data. For more examples on loss curves, please refer to the Appendix B.1. It can be observed that after $\sim$ 200 epochs the loss change is minimal and as can be seen in Table 6.6, the performance does not noticeably improve beyond this point.

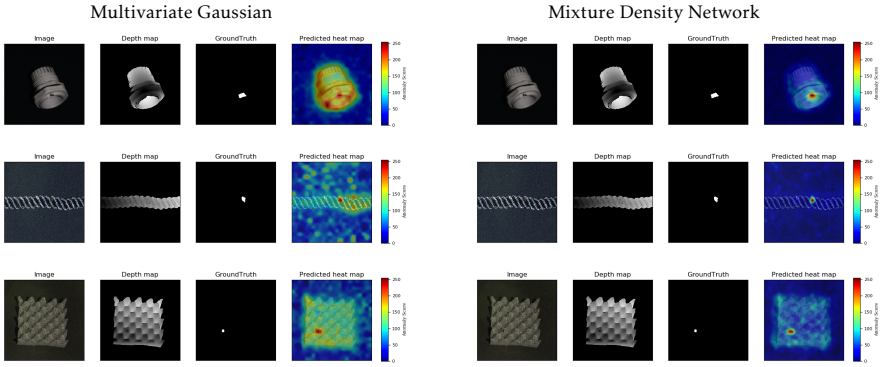

Table 6.7: *The heat maps produced by the MDN have less noise than the heat maps produced by the MVG.*

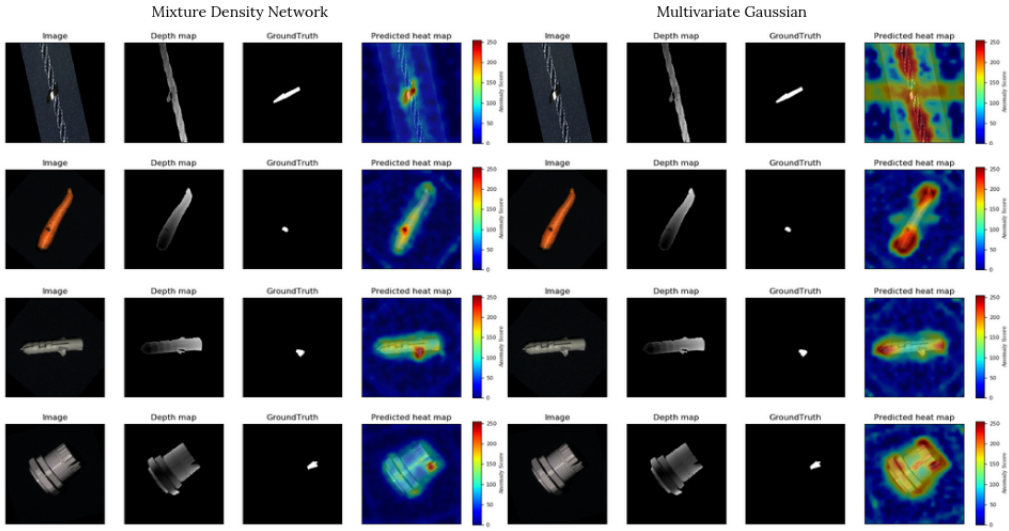

Figure 6.3: *The MDN is less sensitive to varying object orientations than the MVG.*

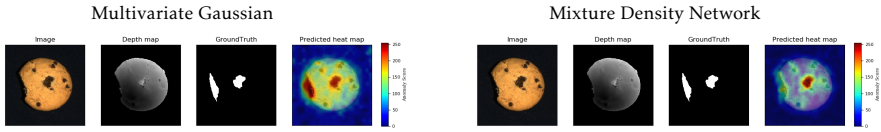

Table 6.8: *The MDN is not as effective as MVG at localizing anomalies that are part of the background.*

6.7.3 Mixture Density Network Visual Inspection

The results of applying random rotations to the images at test time are presented in Figure 6.3. It is evident that the MDN is less sensitive to rotation than the MVG and is able to both detect and localize anomalies despite the inconsistent orientations. Moreover, the MDN resulted in significantly less noise in both the foreground and the background of the anomaly heat maps. This can be seen in Table 6.7. A negative consequence of the MDN can be observed in Table 6.8, where the model was not able to detect the large bite in the cookie.

6.8 Point-PaDiM results

In Table 6.9 the image AUROC, pixel AUROC and pixel AUPRO scores are compared between the three PointNet++ backbones. The classification backbone resulted in the overall lowest scores, receiving around 10-15% points lower in mean anomaly detection and around 15% points lower in mean anomaly localization. The PointNet++ backbones trained for part segmentation and semantic segmentation received closer scores than the classification network, where the semantic segmentation backbone received a 5% higher score in anomaly detection and 3% points higher anomaly localization compared to the part segmentation network.

Figure 6.4 shows examples of resulting anomaly maps of the three backbones. The anomaly maps of the classification network show the effect of having fewer points, resulting in heat maps that are less smooth than the ones created by the other two backbones. The classification backbone uses point placement, which resulted in the anomaly scores not being placed in the correct positions, this is most clear for the Bagel class where the inner circle of the anomaly scores is not matching the shape of the Bagel. More visual results are shown in Appendix B.

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Mean
I-AUROC	CLS-56-AVG|100	60.8	50.4	54.8	48.8	55.3	**59.6**	48.2	**42.6**	54.3	65.0	54.0
	SEM-56-AVG|100	**80.2**	58.8	**58.3**	**94.7**	**73.3**	40.7	**64.1**	39.4	**80.4**	**68.7**	**65.9**
	PART-56-AVG|100	68.1	**61.2**	52.1	92.5	72.1	37.2	44.1	38.5	76.9	61.0	60.4
P-AUROC	CLS-56-AVG|100	76.0	83.1	85.8	71.8	90.1	74.0	78.5	83.7	81.8	77.8	80.3
	SEM-56-AVG|100	**81.8**	**87.3**	88.2	**93.4**	89.4	73.1	84.2	87.2	**93.4**	**89.7**	**86.7**
	PART-56-AVG|100	80.3	86.4	**90.5**	89.8	**92.5**	**78.6**	**85.2**	**88.7**	92.5	82.5	**86.7**
AUPRO	CLS-56-AVG|100	37.3	48.9	53.5	31.4	66.9	30.5	46.7	49.7	41.2	32.3	43.8
	SEM-56-AVG|100	**51.4**	**58.1**	59.3	**81.0**	63.8	27.7	**57.4**	58.1	**66.8**	**67.5**	**59.1**
	PART-56-AVG|100	41.9	56.9	**66.8**	65.6	**73.9**	**35.7**	54.4	**62.5**	64.7	45.4	56.8

Table 6.9: *The AUROC and AUPRO % scores received with PointNet++ networks.*

6.8.1 Sampling Method

Table 6.10 shows the results when using the different sampling methods. The BQAVG method is not reported as it needed over 100 GB of RAM to run with the entire point cloud. Average 2D pooling resulted in the highest anomaly detection score, followed by the BILIN method. The highest anomaly localization score was received for the FPS sampling method, and the AVG pooling method received the second localization highest score with about 3% points less.

6.8.2 Layers

Table 6.11 presents the results of concatenating the last two layers. The results indicate that using the last two layers increases the anomaly localization score by 3% points, but at the cost of a slight decrease in anomaly detection score.

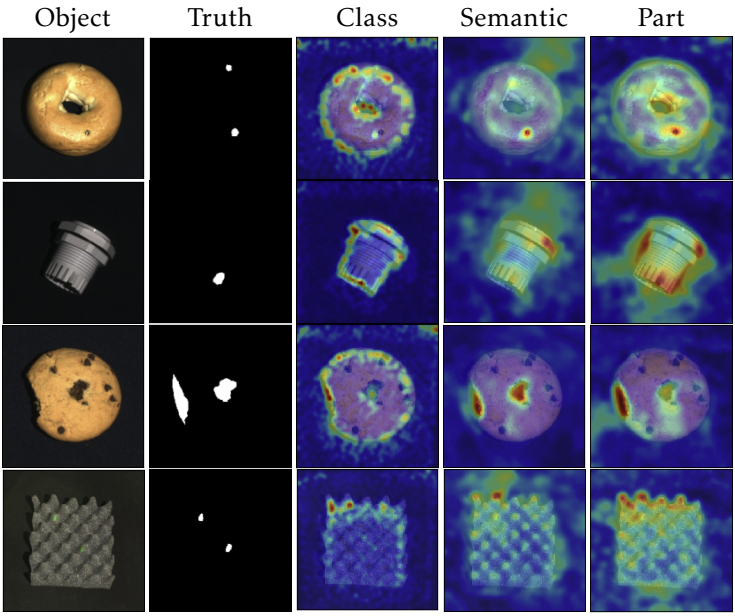

Figure 6.4: *Visual results from the different PointNet++ networks.*

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Mean
I-AUROC	SEM-56-IDX|100	54.1	56.3	51.5	84.4	69.6	**56.1**	51.7	35.8	80.2	68.1	60.8
	SEM-56-FPS|100	47.7	42.2	43.4	59.4	57.7	43.7	63.7	**43.4**	64.2	48.8	51.4
	SEM-56-BILIN|100	79.8	**63.3**	**60.4**	**95.5**	59.7	45.0	60.0	37.2	**82.7**	58.9	64.2
	SEM-56-AVG|100	**80.2**	58.8	58.3	94.7	**73.3**	40.7	**64.1**	39.4	80.4	**68.7**	**65.9**
P-AUROC	SEM-56-IDX|100	75.6	79.8	78.7	86.3	86.0	76.3	81.5	83.4	89.8	86.4	82.4
	SEM-56-FPS|100	83.5	79.5	**89.1**	85.4	**93.9**	**80.2**	**90.2**	**92.5**	93.0	86.0	**87.3**
	SEM-56-BILIN|100	**90.4**	70.3	83.7	91.9	80.4	70.2	86.7	76.9	89.4	76.1	81.6
	SEM-56-AVG|100	81.8	**87.3**	88.2	**93.4**	89.4	73.1	84.2	87.2	**93.4**	**89.7**	86.7
AUPRO	SEM-56-IDX|100	36.4	46.6	42.3	63.8	57.3	**41.1**	50.6	50.8	63.4	64.9	51.7
	SEM-56-FPS|100	49.6	44.5	**67.0**	62.6	**78.4**	38.7	**68.6**	**73.1**	**78.4**	62.1	**62.3**
	SEM-56-BILIN|100	**71.7**	36.5	49.3	80.3	40.4	29.0	65.1	37.6	56.1	33.7	50.0
	SEM-56-AVG|100	51.4	**58.1**	59.3	**81.0**	63.8	27.7	57.4	58.1	66.8	**67.5**	59.1

Table 6.10: *The AUROC and AUPRO % scores received with different down-sampling metods.*

6.8.3 Removing Background Points

Table 6.12 reports the results of removing the background points, thereby forcing the model to only consider anomaly scores in the foreground. This resulted in a higher localization score with about 10% points. However, the methods suffered for some classes where the anomalies were often part of the background, for example Bagel and Cookie.

The anomaly detection score dropped by around 5% points for all networks. This can be seen in Figure 6.5 as the cookie has a missing chunk that is part of the ground truth, but it can not be detected as the background points were removed.

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Mean
I-ROC	SEM-56-AVG-CAT|100	**79.8**	57.9	**58.2**	**94.4**	**76.0**	38.5	**63.9**	**41.1**	**80.9**	**66.2**	**65.7**
	PART-56-AVG-CAT|100	67.7	**59.2**	52.4	92.5	70.2	**45.1**	47.0	**41.1**	78.0	61.0	61.4
P-ROC	SEM-56-AVG-CAT|100	**83.4**	**89.0**	89.0	**94.4**	90.7	76.9	**87.4**	89.4	**94.0**	**91.2**	**88.5**
	PART-56-AVG-CAT|100	80.1	87.2	**90.8**	91.4	**92.4**	**81.6**	87.1	**90.8**	92.4	87.0	88.1
PRO	SEM-56-AVG-CAT|100	**55.0**	**62.6**	60.7	**84.7**	67.6	31.9	**64.2**	63.7	**69.4**	**72.2**	**63.2**
	PART-56-AVG-CAT|100	44.0	60.1	**67.6**	70.7	**74.3**	**43.3**	58.9	**68.3**	62.6	59.6	60.9

Table 6.11: *The AUROC and AUPRO % scores received with or without multiple layers.*

The images also show the negative effect of the point placement method used to insert the anomaly scores into the anomaly map, as the scores do not fill up the whole object. The lack of points in the classification backbone is less noticeable in the images compared to results in Figure 6.5 as the point are closer. Visual results for the other classes are shown in Appendix B.

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Mean
I-AUROC	CLS-56-NoBG|100	35.3	51.0	53.8	65.8	43.8	35.6	45.7	**51.5**	**82.1**	35.8	50.0
	SEM-56-NoBG|100	46.2	**64.0**	**62.6**	**85.6**	**72.6**	29.4	**61.1**	49.6	78.9	**54.4**	**60.4**
	PART-56-NoBG|100	**59.1**	43.1	56.5	73.8	68.5	**45.9**	49.9	46.6	66.6	54.3	56.5
P-AUROC	CLS-56-NoBG|100	**82.3**	91.0	**97.0**	84.1	94.2	**83.9**	91.2	96.3	**96.2**	90.0	**90.6**
	SEM-56-NoBG|100	81.5	**93.0**	95.5	**87.1**	**94.5**	82.2	**91.3**	**97.0**	91.2	**92.7**	**90.6**
	PART-56-NoBG|100	81.5	92.1	96.0	83.7	94.3	83.2	90.7	**97.0**	92.0	92.2	90.3
AUPRO	CLS-56-NoBG|100	42.8	71.2	**90.5**	55.6	81.9	**47.1**	69.9	87.1	**81.6**	66.8	**69.5**
	SEM-56-NoBG|100	41.8	**76.1**	85.1	**66.6**	**82.1**	41.2	**70.2**	**90.0**	63.0	**76.2**	69.2
	PART-56-NoBG|100	**43.0**	73.8	86.7	53.0	81.9	43.5	68.9	89.4	66.3	75.0	68.2

Table 6.12: *The AUROC and AUPRO % scores received when comparing with or without background points.*

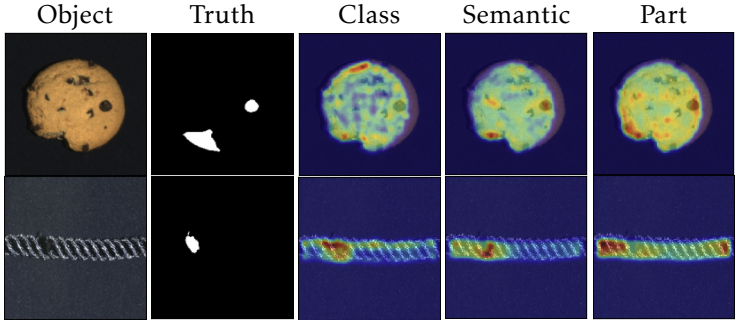

Figure 6.5: *Examples results when removing the background.*

6.8.4 Combining With RGB

Table 6.13 shows the results of combining the PointNet++ and ResNet18 backbones. Both methods A and B increase the overall scores and a significant improvement of around 25% points was received in anomaly localization. The anomaly detection performance increased by around 3-9% points. The A method received the largest increases in anomaly detection, while both methods received similar increases in anomaly localization. The scores for some of the classes declined when introducing the ResNet18 backbone, Cookie received between 91% and 55% in anomaly detection.

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Mean
I-AUROC	SEM-A-56-AVG|100	88.6	66.9	**61.2**	**91.3**	85.4	53.6	**70.9**	39.1	**87.5**	76.0	**72.1**
	SEM-B-56-AVG|100	**96.3**	**74.9**	55.2	65.7	**85.7**	**57.4**	69.6	46.4	74.3	77.7	70.3
	PART-A-56-AVG|100	85.0	68.7	60.2	85.3	84.1	**57.4**	62.3	37.7	84.7	70.1	69.6
	PART-B-56-AVG|100	96.0	74.3	55.1	55.2	85.5	57.3	68.0	**46.6**	74.0	**77.8**	69.0
P-AUROC	SEM-A-56-AVG|100	97.9	95.7	95.8	**97.8**	97.4	88.3	97.5	95.0	95.5	**95.0**	95.6
	SEM-B-56-AVG|100	**99.1**	**96.9**	**97.0**	97.5	**98.3**	88.8	**98.5**	**96.5**	90.8	94.2	**95.8**
	PART-A-56-AVG|100	97.5	95.6	96.5	97.1	97.8	**89.6**	97.2	95.1	**95.7**	92.3	95.4
	PART-B-56-AVG|100	**99.1**	96.8	**97.0**	96.5	**98.3**	87.8	98.3	**96.5**	89.2	93.7	95.3
AUPRO	SEM-A-56-AVG|100	89.2	85.9	85.6	**92.7**	89.5	66.9	91.3	83.1	75.4	**81.9**	84.2
	SEM-B-56-AVG|100	**96.2**	**90.3**	**90.2**	92.1	**93.0**	**70.3**	**94.7**	**88.1**	62.2	76.7	**85.4**
	PART-A-56-AVG|100	87.4	85.8	87.8	90.0	91.0	69.4	90.1	83.5	**77.4**	71.0	83.3
	PART-B-56-AVG|100	95.9	90.2	90.1	89.1	92.8	68.9	93.8	87.9	59.5	74.8	84.3

Table 6.13: *The AU-ROC and AU-PRO % scores received when combining PointNet++ and ResNet18.*

6.8.5 Channels

Table 6.14 shows that using all the channels instead of randomly sampling 100, can increase the anomaly detection score. The B method received a higher increase than the A method. Both the A and B methods received similar anomaly detection scores, but the B method received a higher anomaly localization score of about 2-3% points.

6.8.6 Downsampling

Table 6.15 shows the effect of using different downsampling sizes. The size 170 was the largest size that could be measured because of memory limitations. As the Part segmentation network uses more memory than the semantic segmentation one, the size 170×170 was not measured. Increasing the size to 100×100 marginally increased the anomaly detection for the semantic segmentation network, but at the cost of a decrease in anomaly localization score of 3% points. The part segmentation network decreased both scores. Increasing the size to 170×170 lowered the anomaly detection and localization performance on average.

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Mean
I-AUROC	SEM-A-56-AVG|128	91.7	68.9	**60.9**	**93.0**	85.2	56.1	**72.6**	39.3	**88.2**	73.4	**72.9**
	SEM-B-56-AVG|128	**96.7**	**77.8**	57.0	72.7	**87.4**	**59.1**	71.5	**47.4**	77.5	**78.3**	72.5
	PART-A-56-AVG|128	90.7	69.4	55.6	88.0	84.9	65.2	63.4	40.7	82.8	68.3	70.9
	PART-B-56-AVG|128	96.4	**77.8**	56.9	62.3	86.8	**59.1**	69.4	46.8	76.9	**78.3**	71.1
P-AUROC	SEM-A-56-AVG|128	98.5	95.7	95.5	**98.3**	97.3	**89.5**	97.9	94.0	**95.5**	90.4	95.3
	SEM-B-56-AVG|128	**99.2**	**97.0**	**97.2**	97.6	**98.4**	**89.5**	**98.6**	**96.7**	93.4	**95.1**	**96.2**
	PART-A-56-AVG|128	98.4	95.4	96.5	97.8	97.8	87.8	97.1	94.5	96.9	88.3	95.0
	PART-B-56-AVG|128	99.1	96.9	97.1	96.7	**98.4**	88.6	98.3	96.6	92.3	94.7	95.9
AUPRO	SEM-A-56-AVG|128	93.0	85.7	84.8	**94.6**	89.3	70.1	92.8	79.9	**74.6**	65.5	83.0
	SEM-B-56-AVG|128	**96.3**	**90.7**	**90.5**	92.5	**93.2**	**71.4**	**94.8**	**88.7**	69.7	**80.3**	**86.8**
	PART-A-56-AVG|128	91.8	85.4	88.1	92.8	91.2	65.6	89.6	81.3	81.1	58.7	82.6
	PART-B-56-AVG|128	96.0	90.6	90.4	89.8	93.1	70.2	94.1	88.5	67.8	78.9	85.9

Table 6.14: *The AUROC and AUPRO % scores received with different amount of channels.*

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Mean
I-AUROC	SEM-100-AVG|100	81.5	57.9	51.6	**98.4**	68.9	42.1	65.9	35.0	87.3	59.3	64.8
	SEM-100-AVG-A|100	**91.8**	68.2	58.4	96.7	**82.9**	53.8	**76.6**	36.8	**90.3**	**72.6**	**72.8**
	PART-100-AVG|100	62.1	57.0	47.1	90.6	68.3	50.1	46.1	**38.3**	74.1	54.4	58.8
	PART-100-AVG-A|100	88.8	68.6	53.1	86.7	79.5	**62.2**	60.1	37.6	78.9	66.1	68.2
	SEM-170-AVG|100	77.4	65.6	53.2	97.8	66.6	52.0	57.1	35.3	85.3	58.3	64.9
	SEM-170-AVG-A|100	90.8	**69.8**	**58.7**	95.9	81.3	55.2	57.3	37.0	88.9	70.1	70.5
P-AUROC	SEM-100-AVG|100	91.3	86.0	85.1	95.6	90.4	75.7	90.1	80.3	94.4	71.4	86.0
	SEM-100-AVG-A|100	98.6	**95.4**	94.9	98.2	97.5	**91.2**	**98.3**	93.6	96.1	89.5	**95.3**
	PART-100-AVG|100	85.5	86.9	94.1	92.8	92.7	79.5	84.0	85.4	95.1	65.1	86.1
	PART-100-AVG-A|100	98.2	95.0	**96.9**	97.4	**97.7**	90.0	97.0	93.7	**96.6**	84.6	94.7
	SEM-170-AVG|100	91.7	84.0	83.0	96.0	90.2	78.6	83.5	81.8	92.8	74.6	85.6
	SEM-170-AVG-A|100	**98.7**	94.9	94.6	**98.3**	97.5	90.2	96.0	**93.9**	95.4	**90.3**	95.0
AUPRO	SEM-100-AVG|100	70.6	53.8	50.6	86.9	66.5	36.5	70.9	40.4	68.5	32.8	57.8
	SEM-100-AVG-A|100	**93.5**	**84.9**	83.0	**94.1**	89.8	**72.5**	**94.0**	78.5	75.7	63.1	**82.9**
	PART-100-AVG|100	49.5	57.9	79.7	75.9	75.9	42.9	51.8	52.0	72.0	18.7	57.6
	PART-100-AVG-A|100	91.5	84.5	**89.5**	91.8	**91.0**	71.6	89.6	78.9	**78.9**	47.5	81.5
	SEM-170-AVG|100	68.6	51.7	43.0	86.1	66.3	37.9	50.2	44.9	65.0	35.0	54.9
	SEM-170-AVG-A|100	93.2	83.5	81.7	94.0	90.1	70.8	86.1	**79.6**	73.6	**65.8**	81.9

Table 6.15: *The AUROC and AUPRO % scores received with different down-sampling sizes.*

6.9 Final PaDiM Results

The performance of the baselines and the highest scoring methods were each measured across five runs and the mean and standard deviation were calculated. These results are presented in Table 6.16. The results show that the top-scoring depth method surpasses both of the baselines in anomaly detection and localization. The point cloud method achieved a higher anomaly detection score than the ResNet-18 baseline, however was not able to surpass either of the baselines in anomaly localization. DeiT-384-b received the highest average scores in both anomaly detection and localization out of the methods tested. The possible improvements to the depth methods presented in Table 6.4 were not applied in this experiment.

	Method	Mean I-AUROC	Mean P-AUROC	Mean AUPRO
Ours	WideRes50-224-A	83.6 ± 0.5	98.3 ± 1.8	93.7 ± 0.1
	DeiT-384-B	86.2 ± 0.2	**98.6** ± 0.0	**94.9** ± 0.0
	SEM-100-A|128	74.4 ± 0.4	95.7 ± 0.2	84.4 ± 0.6
Related Work	Res18	71.6 ± 0.9	95.8 ± 0.2	85.9 ± 0.5
	WideRes50	78.8 ± 0.3	97.8 ± 0.0	92.1 ± 0.2
	3D VAE	-	-	47.1
	f-AnoGAN	-	-	63.0
	3D-ST	-	-	83.3
	AST	**93.7** ± 0.2	97.6 ± 0.02	-

Table 6.16: *Final results PaDiM measured over 5 different runs.*

6.10 Inference Times

Figure 6.6 presents the inference times of the models and compares the time with the model performance. The results show an increase in inference time for all PointNet++ methods. The vision transformer methods achieved sub-second inference times that were similar to ResNet-18. Wide ResNet-50 had the longest inference time of 11.53 seconds, 45 times longer than ResNet-18. Further details about the inference times are reported in the Appendix A.2.

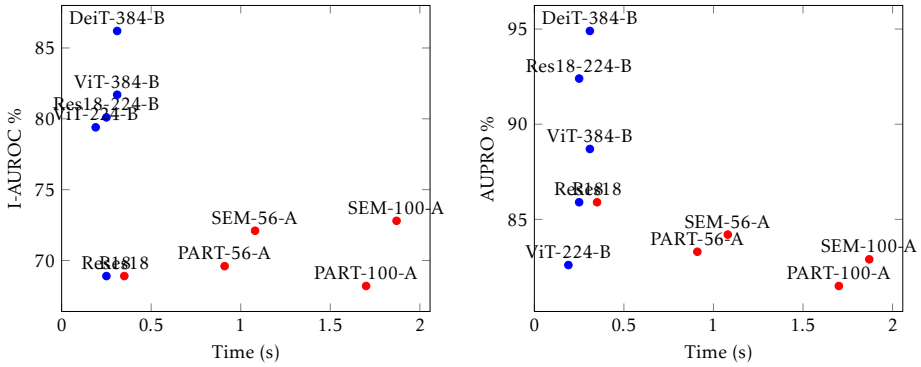

Figure 6.6: *Scatter plot comparing the performance vs inference time. Blue are model measured on the GTX 1080 machine and red is the models measured on the RTX 2070 machine.*

6.11 Resolution

The effect of using different input resolutions is shown in Figure 6.7. In general, the performance in decreases with the size of the data and a large drop in image AUROC and AUPRO can be observed between the 200×200 and the 100×100 cases. Detailed results are reported in the Appendix A.3.

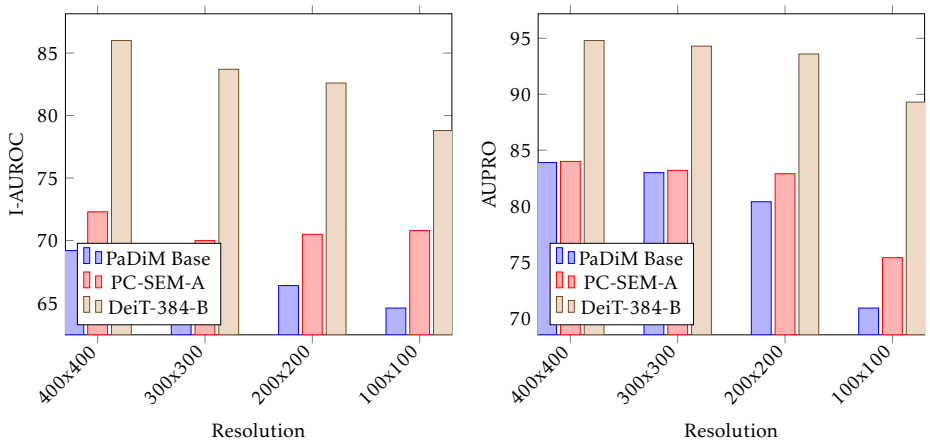

Figure 6.7: *Bar plot of the % scores on different resolutions.*

6.12 Aluminium Can Dataset

An example of an aluminium can collected with the time-of-flight sensor is shown in Figure 6.8. The resulting point cloud and depth map after cropping had a resolution of 100×100. The two leftmost images display the aluminium can from the side and the rightmost image shows the can from a top-down perspective. The sensor was not able to correctly capture the aluminium can which is indicated by the large spread of the points. Furthermore, some points are even located below the floor.

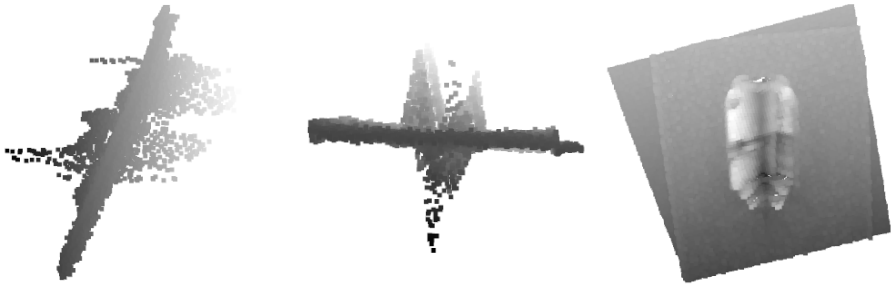

Figure 6.8: *The resulting point cloud of an aluminium can after cropping.*

The results of DeiT-384-B and SEM-100|128 on the aluminium can dataset are presented in Figure 6.9. The images show three examples of which two have anomalies and one does not. The first column displays the depth maps, the second shows the transformer results and the last column the result of the PointNet++ model. For more examples, please refer to Appendix B.3. For the transformer model, a MDN with 150 mixture components trained for 10 epochs was

used as the density estimator. The model was able to detect and localize the anomalies in many of the test images. However, the predicted regions were generally not that precise and there were numerous false positives. The model was generally good at correctly predicting the normal images and the noise in the bottom image is a result of normalizing over a narrow range of low values. The PointNet++ method fails to detect most anomalies preferring to label the bottom of the can as anomalous.

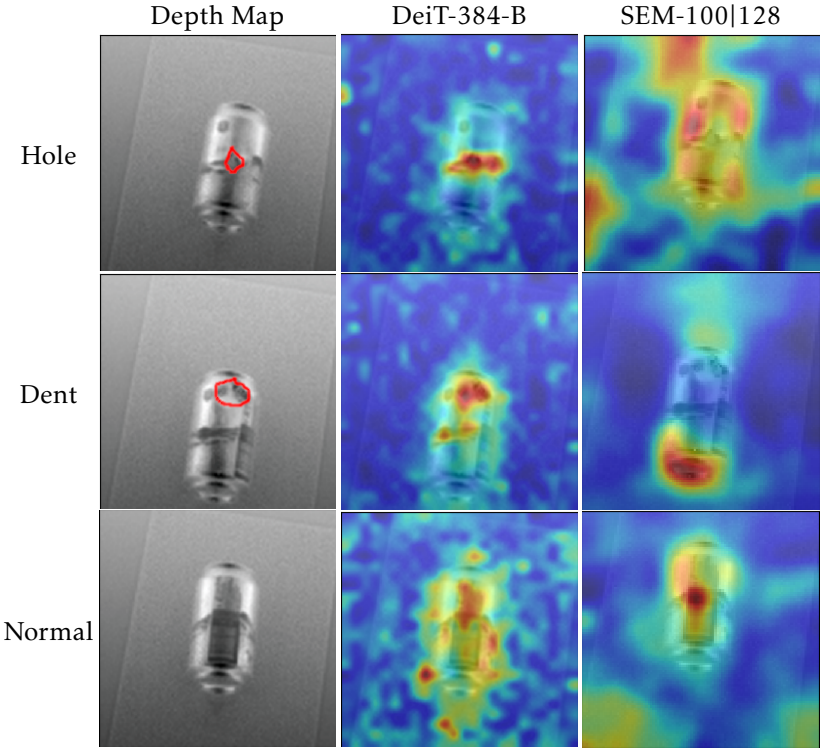

Figure 6.9: *Examples of results on the aluminium can dataset.*

7

Discussion

7.1 Results

7.1.1 Vision Transformers vs. ResNets

The results indicate that a higher performance in anomaly detection and localization can be achieved with the vision transformers than with the ResNets. However, this does not necessarily mean that vision transformers extract features with richer semantic meaning than the ResNets and in fact, Wide ResNet-50 often produced better results than both CaiT and DeiT when the input resolution was set to 224×224. The big advantage of the vision transformers is that they are more cost-efficient. PaDiM demands a lot of memory because the parameters of a Gaussian distribution is stored for each patch in an image. Having smaller feature maps like the vision transformers is thus beneficial from a memory perspective, as there are fewer patches. The trade-off is that the smaller feature maps result in lower anomaly localization accuracy, because the patches are less fine-grained. However, the vision transformers compensate for this flaw by allowing larger input resolutions than the ResNets. At an input resolution of 384×384, the vision transformers outperformed the Wide ResNet-50, while also requiring less memory.

Layer Selection

The vision transformer is architecturally different from the CNN, but has been shown to be similar in some aspects. For instance, the vision transformer learns to pay more attention to distant patches in the deeper layers than in the shallower layers [16]. Similarly, the receptive field in a CNN increases with depth, an inductive bias that is native to the architecture. The authors of PaDiM concatenate features from the three first layers in ResNets as it balances between

local and global features. They argue that the features from deeper layers may be too specific for the task for which it was trained and therefore may not be as general-purpose [13]. This is reasonable as a CNN is forced to compress information into a smaller feature space and learns to keep only what is essential to solve the task. Investigating the effect of combining different layers in the vision transformer would have been interesting. In contrast to a CNN, a vision transformer represents global information through the CLS embedding, which has a similar purpose as the feature vector obtained at the end of a CNN. However, a vision transformer is also able to preserve local information through the patch embeddings even in the deep layers. It is therefore likely that the effect would not be as significant for a vision transformer as for a CNN.

7.1.2 Mixture Density Network

One of the main advantages with the MDN is that it is less sensitive to different object orientations than the MVG. As the MVG learns one distribution per patch, the model considers it anomalous if the position of the object during testing is different from the one during training. This theory is backed up by the empirical results from this thesis where aligning the objects significantly improved the performance of the MVG on elongated classes.

The MDN learns a global distribution which makes the model less susceptible to orientation shifts. Given that the features are also invariant to transformations and kept consistent regardless of placement, MDN should not depend on the object position at all. Another effect that was observed when using the MDN was that the noise significantly decreased in both the foreground and the background, indicating that the MDN more accurately learns the appearance of the object than the MVG and also learns that the background is normal. A drawback of this however, is that it can not as effectively detect anomalies that are part of the background. This was apparent on Cookie where the MDN did not consider the large missing chunks as anomalous. Another drawback of the MDN is that although it discovers the anomalous regions, the localization was not as accurate as for the MVG.

The poor performance achieved by the MDN on Foam and Tire can be explained by the difficulty of estimating the distribution of irregular surfaces; Foam has peaks and valleys and Tire has tracks. The MDN likely learned that inconsistent surfaces are part of the normality and could therefore not appropriately discover anomalies in these classes. For classes such as carrot where MDN performed the best, the surfaces are smooth and fairly consistent across the images.

From a practical point of view, the robustness of the MDN is desirable as objects are rarely placed at consistent orientations in real-life applications. Moreover, an approximate localization of anomalies is often sufficient in practice.

7.1.3 Point-PaDiM

The results from the Point-PaDiM method indicate that only using the PointNet++ backbone can not beat the ResNet-18 backbone in overall anomaly de-

tection and localization performance. Combining a segmentation PointNet++ backbone and the ResNet-18 backbones using the A or B method results in a higher anomaly detection score than when only using the ResNet-18 backbone. The method that received the highest performance was SEM-100-A|128, achieving an AUPRO score of 84.4 ± 0.6, which is well above the 3D VAE and f-AnoGAN methods discussed in Related Work. Point-PaDiM also scored 1.1% points higher in AUPRO than 3D-ST that also operates directly on point clouds.

Networks

Out of all the three networks, the classification network received the lowest scores. A possible reason for this outcome is that the classification network does not have feature propagation layers; which results in the features being extracted from the set abstraction layers, that have at most 512 patches. Comparing the number of patches between the classification- and segmentation network shows that the amount of patches is an order of magnitude higher for the segmentation networks. This lack of patches may lower the performance, or that the layer does not sample the same point each time.

The semantic segmentation network received the highest scores in all the tests. There are two likely reasons why: the data structure and the network depth. The structure of both datasets is a top-down scan of the object including its surroundings, this is closer to a semantic segmentation task where there are multiple objects in the point cloud compared to a part segmentation task where there is only one object. The semantic segmentation network is deeper than the part segmentation network respectively having four set abstractions and four feature propagation layers compared to three set abstractions and three feature propagation layers.

Variations

The sampling methods that combined points resulted in higher detection scores for most classes. For localization, there is no clear winner between point sampling and point combination methods. The BQAVG method may have been able to increase the detection score of the FPS method, as the other combining sampling methods receive higher scores. This may have succeeded in improving the scores in general over the AVG sampling method as the FPS sampling method received the highest anomaly localisation score. But as the ball query implementation required too much memory when applying it on a large point cloud, this could not be evaluated.

Removing the background increased the localization scores for all networks as most anomalies were part of the foreground. However, since anomaly scores could only be placed in the foreground, some anomalies such as the large bites in Cookie could not be detected. There is also the problem that the patches between different objects are inconsistent as the position and rotation of the object effect with paces are segmented. The average 1D pooling also causes a problem as it may take the average over points that are far away from each other.

Concatenating different layers did not have a large effect on the performance of the model. This may be because the difference between the feature values in different propagation layers is small because the features are propagated and may not change much. There is also the possibility of an implementation error due to the non-conventional concatenation method.

When increasing the size of the point cloud fed into the PointNet++ segmentation network the number of patches increases equally. Interestingly, increasing the number of patches did not equal an increase in performance. This may be because the downsampling of the data removes local variations in the normal samples; this can result in the Gaussian distribution of normal samples having a lower variance, leading to a larger Mahalanobis distance between normal and abnormal data.

7.2 Method

7.2.1 PaDiM-3D

Although the pre-trained ResNets and vision transformers were shown to extract semantically meaningful features from depth maps, they were originally trained on RGB images from ImageNet. To simulate RGB images, the grayscale depth maps were repeated three times and concatenated which may have resulted in images that did not accurately resemble the original images in ImageNet.

7.2.2 Point-PaDiM

The downsampling method that performed the best for the point cloud methods is designed for images, so the possible preprocessing steps were limited as the image structure of the data could not be broken. If a point cloud downsampling method was found that returned consistent sampling size, there would be fewer limitations on the preprocessing stages. For example, null points could be removed from the data entirely instead of moving them into the background.

After removing the background, the number of points remaining varies, so 1D adaptive average pooling was used. However, there is no guarantee that points that are next to each other in the point cloud are also next to each other in the point list.

The PointNet++ model used was not the original implementation and the network is from 2017, using a newer model or the original implementation may have resulted in better performance. The original model was implemented in C++ and may have increased the inference time.

7.2.3 Aluminium Can Dataset

The sensor had to be placed at a relatively large distance from the aluminium cans or else they were not captured correctly. This resulted in a relatively small portion of the points in the point cloud being part of the object of interest. To work around this, the small region of points containing the aluminium cans were

cropped from the entire point cloud, resulting in a large waste of information. A higher performance could have been achieved had the resolution of the points on the aluminium cans been higher, this hypothesis is backed up by the experimental results conducted in this thesis.

When creating the anomalies, many of them turned out rather ambiguous and some would argue that whole can itself could be considered an anomaly, please refer to the Appendix B.3 for examples. For this reason, it was difficult to correctly assess the performance of the models. A better choice of anomalies would have been small and more local regions, ideally also difficult to spot in intensity images. These types of anomalies are likely more probable in practice too.

7.2.4 Reliability

Measuring the results over multiple runs and taking the mean and standard deviation is useful to describe how sensitive a method is to factors such as randomness. This was done in one of the experiments, but could have been applied more often to increase the credibility of the results.

7.2.5 Mixture Density Network

The hyperparameters for the MDN should have been selected more systematically, instead of picking them arbitrarily. Moreover, investigating how the number of Gaussian mixture components affects the performance of the model would have been interesting from a scientific point of view. In theory, having too many components is preferred over having too few, as this would simply mean that the weights for some mixtures would become 0 [8]. Analyzing the weights could therefore have been used when deciding upon the optimal number of mixtures.

Investigating the MDN for the ResNets would have also been interesting as they were heavily constrained by the computational bottleneck of computing a Gaussian distribution per patch in large feature maps. The MDN is a way to reduce the high memory cost which would have likely benefited the ResNet methods.

7.2.6 Evaluation

Although most research on anomaly localization on the MVTec 3D-AD dataset use Pixel AUROC and AUPRO to measure anomaly localization, it is evident that these metrics produce optimistic results for this particular dataset. This is due to the large portion of background pixels which make up the majority of the true negatives and will in most cases sum up to a large number, resulting in low false positive rates. In turn, this results in high AUROC scores and AUPRO scores that do not properly reflect the actual localization ability of the methods. A metric that is less sensitive to the pixel imbalance is the area under the precision-recall curve (AUPRC), which could have been used as a complimentary metric. However, it would have been difficult to compare these results to previous research.

7.2.7 Heat Map Visualization

Since the values of each heat map were normalized per image, the results are sometimes misleading. A high heat map value does not necessarily point to a high absolute anomaly score and likewise, a low heat map value does not implicate a low absolute score. Rather, it displays the scores relative to each other. For example, in a heat map where all the values are low, the normalization is still going to spread the values across the full intensity range, resulting in the background appearing as though it had high values, when in reality they were just lower relative to the maximum value in that heat map. The consequence is that the heat maps are most accurate in cases where the model is highly certain of an anomaly, indicated by a maximum value that is relatively much larger than the other values. In contrast, the anomaly maps are not as accurate in cases where no anomalies are present, as all the absolute scores are expected to be low. An alternative to this normalization method is to instead normalize across all images. However, this would instead hide the anomalies that have low maximum values relative to the maximum value in the dataset.

7.3 Limitations With 3D Information

As can be seen in the Appendix A.3, the 3D methods resulted in poorer performance than the baselines on some classes and this can be explained by the type of data and also its quality. Notably, classes such as Tire received significantly lower image AUROC when 3D information was added, compared to when only RGB data was used. The irregular surfaces caused by the tracks, likely resulted in a large variance in the 3D data and may have reduced the model's ability to learn the distribution. Moreover, Tire had many missing values in both the foreground and the background that were caused by shadows and this is a consequence of the data acquisition. In contrast, the RGB data for Tire had significantly lower variance in the pixel intensity and did not include any missing values.

7.4 The work in a wider context

Although the use of automated programs may be a cheap solution compared to manually detecting defects in items, they are by no means foolproof. Industries need to be aware of that automated solutions can make mistakes that can lead to disastrous consequences. For critical systems, it is therefore encouraged to view these programs as assisting entities, rather than relying exclusively on them.

8

Conclusion

This thesis investigated industrial 3D anomaly detection with the PaDiM framework. The methods PaDiM-3D and PaDiM-ViT combine RGB and depth data with a ResNet or vision transformer to extract features. Point-PaDiM combines RGB with point cloud data and uses PointNet++ to extract XYZ features and ResNet to extract RGB features. PaDiM-ViT with the DeiT backbone achieved the highest average image AUROC, pixel AUROC and AUPRO on the MVTec 3D-AD dataset, while also achieving sub-second inference times. Vision transformers allowed for larger input resolutions, which was required to outperform ResNet-50. PaDiM-3D with the Wide ResNet-50 backbone achieved similar results to PaDiM-ViT, but required approximately 40 times more time to compute. Point-PaDiM achieved reasonably fast inference times and was able to outperform the ResNet-18 baseline on average, but performed poorly on some individual classes. A mixture density network was investigated as an alternative density estimator for PaDiM-ViT, which improved the anomaly detection performance and robustness of the model. It was however shown that the model had difficulties with detecting anomalies in objects that have irregular surfaces.

The proposed 3D methods improved the performance of PaDiM on MVTec 3D-AD and were also able to detect certain anomalies on the aluminium can dataset. However, these results were in some cases inaccurate, which can partially be explained by the poor quality of the data. The resolution of the input data was shown to have a large impact on the performance and a significant drop was observed when reducing the resolution from 200×200 to 100×100 pixels.

It was shown that the performance of the models depends on the object type and classes that have a large variance in the 3D data may benefit from using only RGB data. A larger dataset or data augmentation could have helped the models learn the distributions of the 3D data better. Another factor that greatly affects the performance is the quality of the data and missing values are common in

3D scans. The causes might vary depending on the sensor, but reflective objects like the aluminum cans were difficult to capture with the time-of-flight sensor. Choosing the appropriate sensor and setup for the problem helps reducing the frequency of error values.

8.1 Future Work

This work has shown that pre-trained vision transformers extract useful features for anomaly detection and localization on industrial data. This should not be limited to only PaDiM, but other 2D representation-based methods such as PatchCore [38], could also benefit from this architecture. Furthermore, such methods may also be extendable to 3D using a similar method to the one proposed in this thesis. Transformer architectures have also recently been adopted to point clouds, such as Point-M2AE by Zhang et al. [57]. Combining feature extraction with a point cloud transformer with a vision transformer to extract image features would certainly be interesting future work. A better pooling method such as AFPS [25] could mitigate the time bottleneck of the FPS sampling in the point clouds. Finally, more relevant depth features can be obtained by using models pre-trained on depth maps, instead of models pre-trained on RGB data like the ones investigated in this thesis. Alternatively, a model could be trained from scratch on depth maps to learn to extract depth features.

Bibliography

[1] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[3] Christoph Baur, Benedikt Wiestler, Shadi Albarqouni, and Nassir Navab. Deep autoencoding models for unsupervised anomaly segmentation in brain mr images. In Alessandro Crimi, Spyridon Bakas, Hugo Kuijf, Farahani Keyvan, Mauricio Reyes, and Theo van Walsum, editors, *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 161–169, Cham, 2019. Springer International Publishing. ISBN 978-3-030-11723-8.

[4] Marcel Bengs, Finn Behrendt, Julia Krüger, Roland Opfer, and Alexander Schlaefer. Three-dimensional deep learning with spatial erasing for unsupervised anomaly segmentation in brain mri. *International journal of computer assisted radiology and surgery*, 16(9):1413–1423, 2021.

[5] Paul Bergmann and David Sattlegger. Anomaly detection in 3d point clouds using deep geometric descriptors. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2613–2623, January 2023.

[6] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9592–9600, 2019.

[7] Paul Bergmann., Xin Jin., David Sattlegger., and Carsten Steger. The mvtec 3d-ad dataset for unsupervised 3d anomaly detection and localization. In *Proceedings of the 17th International Joint Conference on Computer Vision,*

Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP,, pages 202–213. INSTICC, SciTePress, 2022. ISBN 978-989-758-555-5. doi: 10.5220/0010865000003124.

[8] Christopher M. Bishop. Mixture density networks. 1994. URL `https://publications.aston.ac.uk/id/eprint/373/`.

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

[10] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[11] Niv Cohen and Yedid Hoshen. Sub-image anomaly detection with deep pyramid correspondences. *arXiv preprint arXiv:2005.02357*, 2020.

[12] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[13] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization. In *International Conference on Pattern Recognition*, pages 475–489. Springer, 2021.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[17] Y. Eldar, M. Lindenbaum, M. Porat, and Y.Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997. doi: 10.1109/83.623193.

[18] Denis Gudovskiy, Shun Ishizaka, and Kazuki Kozuka. Cflow-ad: Real-time unsupervised anomaly detection with localization via conditional normalizing flows. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 98–107, 2022.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016. URL http://arxiv.org/abs/1606.08415.

[21] CMathWorks Inc. What are organized and unorganized point clouds?, 2023. URL https://www.mathworks.com/help/lidar/ug/organized-and-unorganized-point-clouds.html.

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[23] A. Kolb, E. Barth, R. Koch, and R. Larsen. Time-of-flight cameras in computer graphics. *Computer Graphics Forum*, 29(1): 141–159, 2010. doi: https://doi.org/10.1111/j.1467-8659.2009.01583.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2009.01583.x.

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[25] Jingtao Li, Jian Zhou, Yan Xiong, Xing Chen, and Chaitali Chakrabarti. An adjustable farthest point sampling method for approximately-sorted point cloud data. In *2022 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 1–6, 2022. doi: 10.1109/SiPS55645.2022.9919246.

[26] Weiping Liu, Jia Sun, Wanyi Li, Ting Hu, and Peng Wang. Deep learning on point clouds and its application: A survey. *Sensors*, 19(19), 2019. ISSN 1424-8220. doi: 10.3390/s19194188. URL https://www.mdpi.com/1424-8220/19/19/4188.

[27] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[28] Prasanta Chandra Mahalanobis. On the generalised distance in statistics. In *Proceedings of the national Institute of Science of India*, volume 12, pages 49–55, 1936.

[29] Batta Mahesh. Machine learning algorithms -a review, 01 2019.

[30] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4): 115–133, 1943.

[31] Pankaj Mishra, Riccardo Verk, Daniele Fornasier, Claudio Piciarelli, and Gian Luca Foresti. VT-ADL: A vision transformer network for image anomaly detection and localization. In *30th IEEE/IES International Symposium on Industrial Electronics (ISIE)*, June 2021.

[32] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.

[33] Sergio Presa, Fátima A Saiz, and Iñigo Barandiaran. A fast deep learning based approach for unsupervised anomaly detection in 3d data. In *2022 7th International Conference on Frontiers of Signal Processing (ICFSP)*, pages 6–13. IEEE, 2022.

[34] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[35] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf`.

[36] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollar. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[37] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021.

[38] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14318–14328, 2022.

[39] Marco Rudolph, Bastian Wandt, and Bodo Rosenhahn. Same same but differnet: Semi-supervised defect detection with normalizing flows. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1907–1916, 2021.

[40] Marco Rudolph, Tom Wehrbein, Bodo Rosenhahn, and Bastian Wandt. Asymmetric student-teacher networks for industrial anomaly detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2592–2602, 2023.

[41] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3): 211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[43] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009. doi: 10.1109/ROBOT.2009.5152473.

[44] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Georg Langs, and Ursula Schmidt-Erfurth. f-anogan: Fast unsupervised anomaly detection with generative adversarial networks. *Medical image analysis*, 54:30–44, 2019.

[45] Max Schwarz, Hannes Schulz, and Sven Behnke. Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1329–1335, 2015. doi: 10.1109/ICRA.2015.7139363.

[46] Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[47] Jaime Simarro Viana, Ezequiel de la Rosa, Thijs Vande Vyvere, et al. Unsupervised 3d brain anomaly detection. In *International MICCAI Brainlesion Workshop*, pages 133–142. Springer, 2020.

[48] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/tan19a.html`.

[49] Xian Tao, Xinyi Gong, Xin Zhang, Shaohua Yan, and Chandranath Adak. Deep learning for unsupervised anomaly localization in industrial images: A survey. *IEEE Transactions on Instrumentation and Measurement*, 71:1–21, 2022. doi: 10.1109/TIM.2022.3196436.

[50] Yung Liang Tong. *The multivariate normal distribution*. Springer Science & Business Media, 2012.

[51] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/touvron21a.html.

[52] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 32–42, October 2021.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[54] Bingjie Wang, Wei Liang, Yucheng Wang, and Yan Liang. Head pose estimation with combined 2d sift and 3d hog features. In *2013 Seventh International Conference on Image and Graphics*, pages 650–655, 2013. doi: 10.1109/ICIG.2013.133.

[55] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[56] Sanyapong Youkachen, Miti Ruchanurucks, Teera Phatrapomnant, and Hirohiko Kaneko. Defect segmentation of hot-rolled steel strip surface by using convolutional auto-encoder and conventional image processing. In *2019 10th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*, pages 1–5, 2019. doi: 10.1109/ICTEmSys.2019.8695928.

[57] Renrui Zhang, Ziyu Guo, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, Hongsheng Li, and Peng Gao. Point-m2ae: Multi-scale masked autoencoders for hierarchical point cloud pre-training, 2022.

Appendix

A

Detailed Results

A.1 PaDiM 2D

The results in Table A.1 were used to confirm that the implementation of PaDiM was correct.

Class	Image AUROC	Pixel AUROC	AUPRO (FPR limit: 0.3)
Bottle	99.6%	98.1%	94.7%
Cable	85.5%	94.9%	86.1%
Capsule	87.0%	98.2%	90.5%
Carpet	98.4%	98.8%	95.6%
Grid	89.8%	93.6%	84.4%
Hazelnut	84.1%	97.9%	92.7%
Leather	98.8%	99.0%	97.1%
Metal nut	97.4%	96.7%	90.6%
Pill	86.9%	94.6%	90.7%
Screw	74.5%	97.2%	90.7%
Tile	95.9%	91.7%	75.0%
Toothbrush	94.7%	98.6%	91.6%
Transistor	92.5%	96.8%	90.2%
Wood	99.0%	94.0%	90.5%
Zipper	74.1%	97.6%	92.2%
Average	90.3%	96.8%	90.7%

Table A.1: *PaDiM results on the MVTec 2D-AD dataset.*

A.2 Inference Times Details

Table A.1 and A.2 report the inference times of the methods.

Method	Inference Time (s)
Res18	0.35
PART-56|100	0.7
SEM-56|100	0.9
PART-56-CAT|100	1.27
SEM-56-CAT|100	1.47
PART-56-A|100	0.91
PART-56-A|128	1.18
SEM-56-A|100	1.08
SEM-56-A|128	1.31
PART-56-B|100	1.12
PART-56-B|128	1.86
SEM-56-B|100	1.25
SEM-56-B|128	1.91
PART-100|100	1.47
SEM-100|100	1.58
PART-100-A|100	1.70
SEM-100-A|100	1.87

Figure A.1: *The inference time using the i9-9880H and RTX 2070 (mobile).*

Method	Inference Time (s)
Res18	0.25
WideRes50	11.53
Res18-224-A	0.45
Res18-224-B	0.25
ViT-224-A	0.23
ViT-224-B	0.19
ViT-384-A	0.36
ViT-384-B	0.31
CaiT-S24-224-A	0.21
CaiT-S24-224-B	0.18
CaiT-S24-384-A	0.37
CaiT-S24-384-B	0.34
DeiT-224-A	0.24
DeiT-224-B	0.23
DeiT-384-A	0.42
DeiT-384-B	0.31

Figure A.2: *The inference time using the i7-8700K and GTX 1080.*

A.3 Downsampling Details

Table A.2 contains the detailed results of downsampling the data.

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire	Mean
I-AUROC	ResNet18 400x400	95.3	74.8	54.2	56.9	85.4	59.8	72.1	47.5	68.4	77.9	69.2
	ResNet18 300x300	95.2	71.6	58.9	60.0	83.3	46.9	68.5	45.0	61.8	80.0	67.1
	ResNet18 200x200	95.7	75.2	61.7	56.0	85.9	48.9	75.0	43.8	57.7	63.6	66.4
	ResNet18 100x100	92.7	62.3	47.7	60.4	74.6	74.1	64.3	35.7	59.0	75.3	64.6
	SEM-56-A|100 400x400	92.5	71.3	61.3	91.9	83.7	52.4	72.4	38.9	87.3	71.8	72.3
	SEM-56-A|100 300x300	90.2	69.5	62.5	92.8	81.8	36.6	67.4	38.0	85.5	75.5	70.0
	SEM-56-A|100 200x200	91.3	68.6	63.2	91.8	84.5	38.9	71.2	37.1	85.0	73.1	70.5
	SEM-56-A|100 100x100	90.1	68.8	61.6	90.5	81.6	60.1	60.4	36.5	86.3	72.4	70.8
	DeiT-384-B 400x400	99.6	84.5	75.2	96.7	93.4	80.4	91.7	70.5	97.1	97.1	86.0
	DeiT-384-B 300x300	98.4	76.6	76.2	94.5	92.3	80.9	91.3	70.4	95.9	61.0	83.7
	DeiT-384-B 200x200	99.0	77.3	74.1	94.9	93.3	71.5	86.4	70.7	94.9	64.0	82.6
	DeiT-384-B 100x100	99.8	75.1	72.5	95.7	91.5	66.0	79.6	71.1	83.1	53.9	78.8
P-AUROC	ResNet18 400x400	99.0	96.8	97.0	96.3	98.3	86.3	98.3	96.6	88.8	93.4	95.1
	ResNet18 300x300	99.0	96.5	96.9	96.3	98.2	85.5	98.2	96.5	87.4	92.6	94.7
	ResNet18 200x200	99.0	96.7	94.6	96.3	98.2	81.7	98.1	96.4	79.2	92.4	93.3
	ResNet18 100x100	98.2	94.0	91.5	95.3	97.4	72.7	97.2	94.9	48.0	85.0	87.4
	SEM-56-A|100 400x400	98.2	95.7	95.9	97.8	97.5	87.4	97.6	94.8	95.8	94.5	95.5
	SEM-56-A|100 300x300	90.2	69.5	62.5	92.8	81.8	36.6	67.4	38.0	85.5	75.5	70.0
	SEM-56-A|100 200x200	97.9	95.7	95.0	97.7	97.5	85.2	97.5	94.2	93.6	94.4	94.9
	SEM-56-A|100 100x100	97.1	93.0	93.1	97.2	96.6	78.9	96.3	93.5	87.9	92.0	92.6
	DeiT-384-B 400x400	99.7	97.6	99.0	99.2	99.0	95.5	99.7	99.4	99.3	97.3	98.5
	DeiT-384-B 300x300	99.6	97.1	98.9	99.3	98.8	95.1	99.7	99.4	99.2	97.3	98.4
	DeiT-384-B 200x200	99.6	96.7	98.8	99.4	98.9	95.2	99.6	99.3	99.0	96.3	98.3
	DeiT-384-B 100x100	99.5	94.7	98.1	99.3	98.5	92.3	99.2	99.2	98.5	91.9	97.1
AUPRO	ResNet18 400x400	95.8	90.3	90.0	88.4	92.8	66.1	93.8	88.2	58.7	74.7	83.9
	ResNet18 300x300	95.5	89.3	89.3	88.5	92.2	65.0	93.6	88.1	56.3	72.5	83.0
	ResNet18 200x200	95.8	90.0	82.5	88.3	92.4	58.3	93.3	87.2	44.1	72.2	80.4
	ResNet18 100x100	92.3	81.4	73.8	85.8	88.6	43.3	90.2	82.8	14.9	55.8	70.9
	SEM-56-A|100 400x400	90.7	86.0	86.1	92.3	89.8	65.0	91.4	82.4	76.4	80.2	84.0
	SEM-56-A|100 300x300	88.6	84.8	85.3	92.0	89.6	64.2	90.9	82.5	74.3	79.8	83.2
	SEM-56-A|100 200x200	89.4	86.0	83.0	91.9	89.7	58.4	90.9	80.5	68.3	80.4	81.9
	SEM-56-A|100 100x100	85.4	77.9	76.4	90.7	85.9	46.9	87.1	78.9	52.2	72.4	75.4
	DeiT-384-B 400x400	98.8	92.6	96.2	97.9	95.5	85.1	98.9	97.3	96.4	89.1	94.8
	DeiT-384-B 300x300	98.5	91.0	96.0	98.1	95.1	83.7	98.7	97.3	95.7	89.2	94.3
	DeiT-384-B 200x200	98.4	90.0	95.8	98.1	95.5	84.3	98.3	96.9	93.3	85.2	93.6
	DeiT-384-B 100x100	97.9	82.9	93.4	97.6	93.9	74.7	96.8	96.5	91.1	68.2	89.3

Table A.2: *The AUROC and AUPRO scores of the methods at different input resolutions.*

A.4 Extended Final PaDiM Results

Table A.3 contains the results of the individual classes when comparing the best methods across five runs.

	Method	Bagel	Cable Gland	Carrot	Cookie	Dowel	Foam	Peach	Potato	Rope	Tire
I-AUROC	Res18	94.5 ± 1.4	73.5 ± 3.5	59.6 ± 1.6	57.8 ± 5.9	88.4 ± 0.6	60.4 ± 4.5	73.4 ± 2.4	49.2 ± 1.6	80.8 ± 2.3	78.4 ± 1.0
	WideRes50	97.6 ± 0.4	80.0 ± 1.4	70.4 ± 1.1	62.9 ± 2.1	95.9 ± 0.2	69.0 ± 1.4	85.0 ± 2.1	53.7 ± 1.6	94.2 ± 0.6	79.3 ± 1.3
	WideRes50-224-A	99.6 ± 0.2	74.9 ± 1.2	68.6 ± 1.2	96.1 ± 0.8	93.7 ± 0.5	77.1 ± 1.1	90.3 ± 1.4	70.8 ± 1.9	94.4 ± 0.5	71.1 ± 1.2
	DeiT-384-B	99.5 ± 0.1	84.9 ± 0.5	75.6 ± 0.1	96.6 ± 0.4	93.1 ± 0.4	81.0 ± 0.7	91.3 ± 0.4	71.9 ± 0.9	96.0 ± 0.5	71.7 ± 0.8
	SEM-100-A|128	92.4 ± 0.5	71.1 ± 1.7	60.3 ± 0.8	96.9 ± 0.5	85.1 ± 1.3	57.5 ± 4.6	78.3 ± 1.2	43.6 ± 1.3	92.3 ± 0.8	66.7 ± 1.8
P-AUROC	Res18	98.9 ± 0.1	96.5 ± 0.4	97.1 ± 0.1	96.5 ± 0.2	98.2 ± 0.1	89.7 ± 0.5	98.2 ± 0.3	96.7 ± 0.2	93.5 ± 2.3	93.0 ± 1.0
	WideRes50	99.4 ± 0.0	97.9 ± 0.1	98.2 ± 0.0	97.6 ± 0.1	99.0 ± 0.0	92.9 ± 0.1	98.8 ± 0.1	97.8 ± 0.1	99.1 ± 0.0	97.3 ± 0.2
	WideRes50-224-A	99.5 ± 0.0	95.9 ± 0.0	98.6 ± 0.0	99.2 ± 0.0	98.8 ± 0.0	95.6 ± 0.1	99.5 ± 0.0	99.2 ± 0.0	99.3 ± 0.0	97.2 ± 0.2
	DeiT-384-B	99.7 ± 0.0	97.6 ± 0.0	99.0 ± 0.0	99.1 ± 0.0	98.9 ± 0.0	95.9 ± 0.0	99.7 ± 0.0	99.4 ± 0.0	99.2 ± 0.0	97.4 ± 0.0
	SEM-100-A|128	98.7 ± 0.1	95.9 ± 0.3	95.3 ± 0.3	98.5 ± 0.0	97.7 ± 0.1	91.9 ± 0.8	98.1 ± 0.2	94.0 ± 0.1	97.1 ± 0.3	89.4 ± 0.6
AUPRO	Res18	95.6 ± 0.6	89.4 ± 1.3	90.5 ± 0.6	89.4 ± 0.4	92.4 ± 0.5	72.4 ± 1.2	93.9 ± 1.1	88.8 ± 0.7	73.0 ± 5.1	74.0 ± 2.5
	WideRes50	97.2 ± 0.2	93.7 ± 0.2	93.9 ± 0.2	92.2 ± 0.2	95.5 ± 0.1	77.9 ± 0.3	95.7 ± 0.4	92.6 ± 0.4	93.7 ± 0.3	88.7 ± 0.9
	WideRes50-224-A	98.2 ± 0.1	87.3 ± 0.2	95.0 ± 0.2	97.7 ± 0.0	95.0 ± 0.1	85.8 ± 0.3	98.1 ± 0.2	97.2 ± 0.2	95.4 ± 0.1	87.6 ± 0.8
	DeiT-384-B	98.8 ± 0.0	92.6 ± 0.1	96.3 ± 0.1	97.8 ± 0.0	95.5 ± 0.0	86.3 ± 0.2	98.8 ± 0.0	97.3 ± 0.1	96.1 ± 0.0	89.7 ± 0.1
	SEM-100-A|128	94.8 ± 0.4	87.6 ± 1.2	84.9 ± 1.1	94.7 ± 0.0	90.4 ± 0.3	74.9 ± 2.0	93.9 ± 0.5	79.9 ± 0.5	80.6 ± 1.7	62.4 ± 2.1

Table A.3: *The individual class results of evaluating the methods over five runs.*

B

Extended Visual Results

B.1 PaDiM-ViT

B.2 Point-PaDiM

B.2.1 Networks

Figure B.2 shows anomaly map examples of classes not shown when comparing PointNet++ backbones in the result chapter.

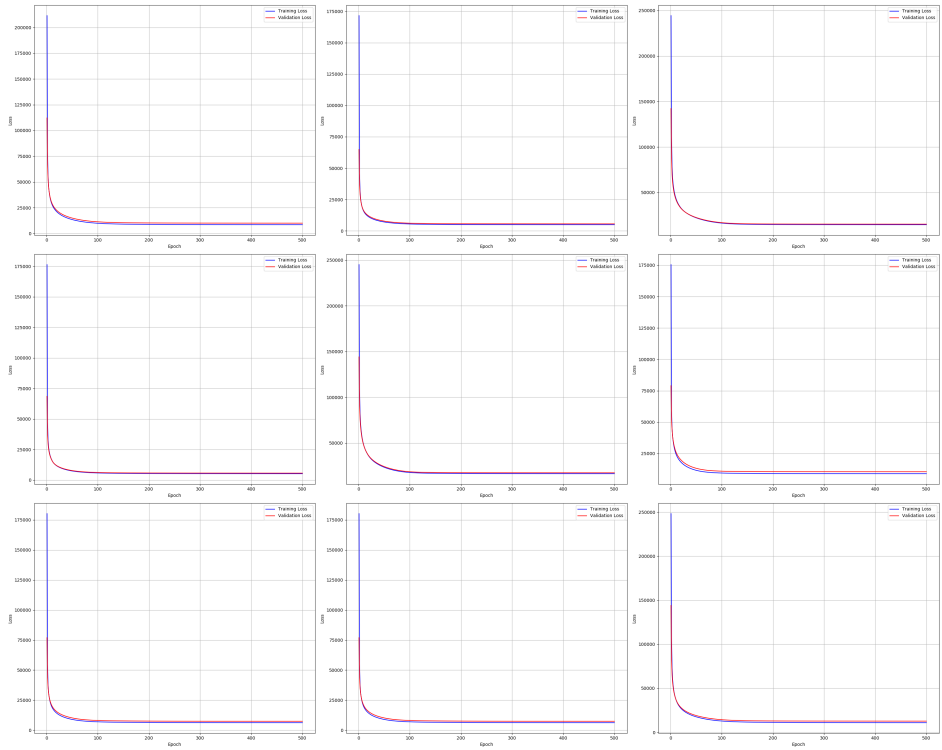

Figure B.1: *The loss curves from MDN with 150 Gaussian components trained for 500 epochs.*

B.2.2 Background Removal

Figure B.3 shows the anomaly maps of the objects not shown in the result chapter.

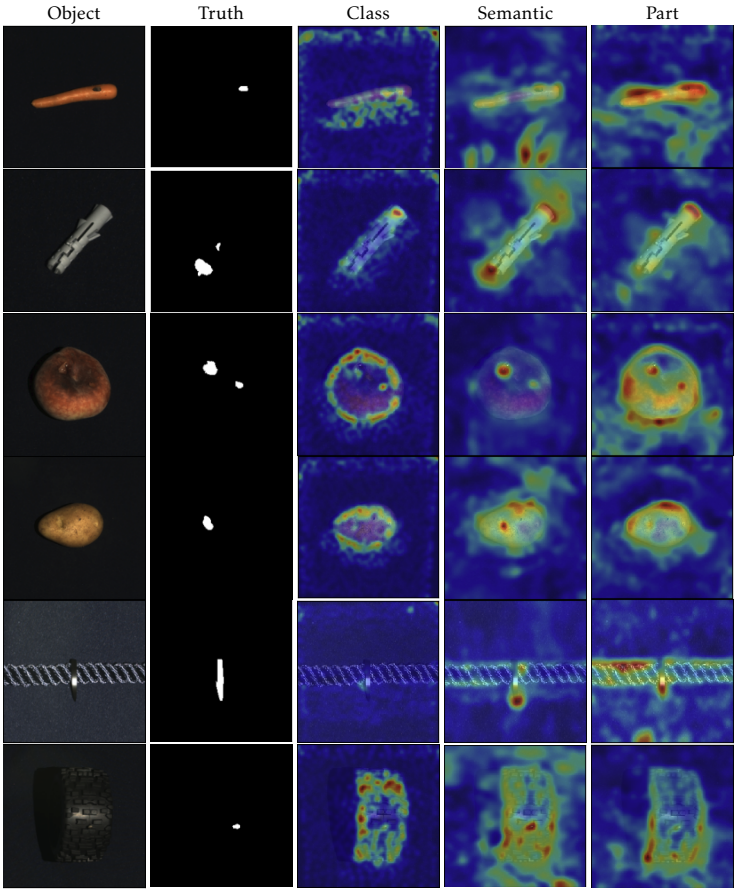

Figure B.2: *Resulting anomaly maps using different PoinNet++ backbones.*

B.3 Aluminium Can Dataset

Figure B.4 shows all the results from the aluminium can dataset. The depth maps were preprocessed using the steps described in 4.1.

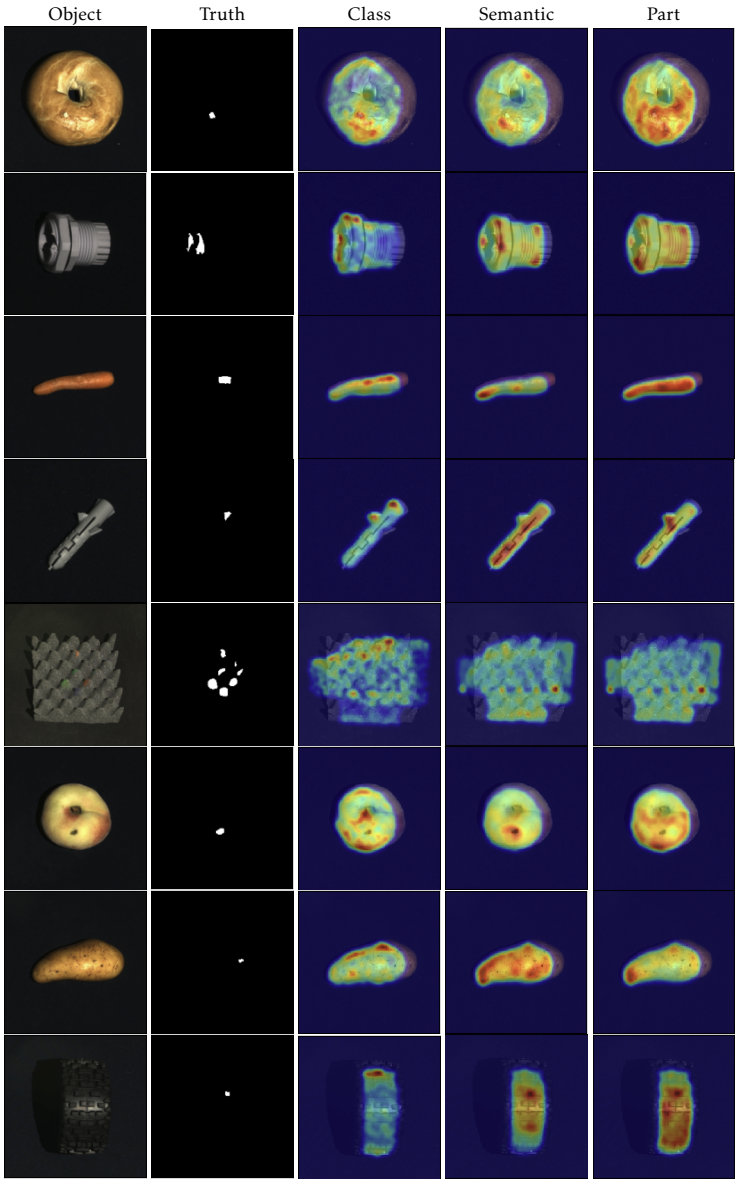

Figure B.3: *Resulting anomaly maps using no background for PointNet++.*

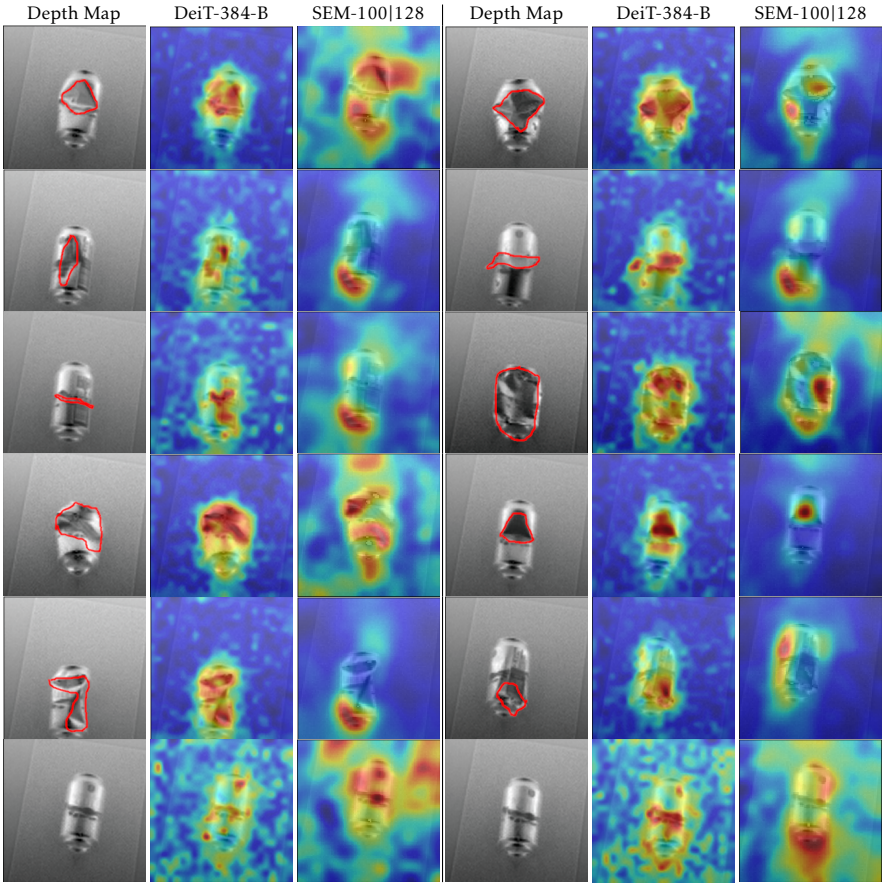

Figure B.4: *Resulting heat maps from DeiT-384-B and SEM-100|128 on the aluminium can dataset.*