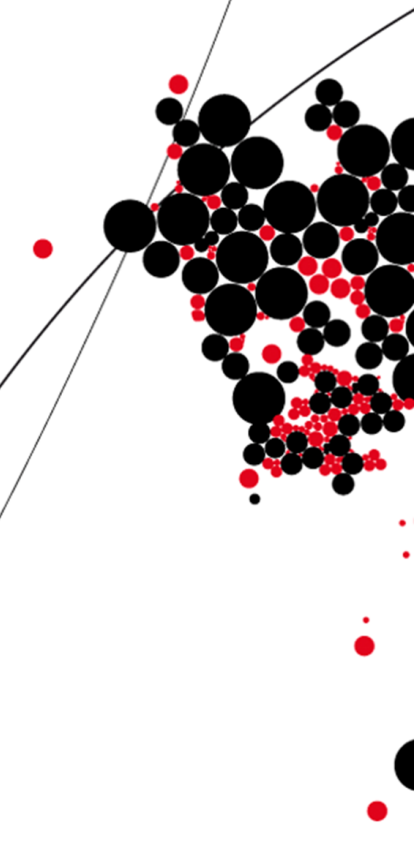
UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

A Comparison of Anomaly Detection Algorithms with applications on Recoater Streaking in an Additive Manufacturing Process.

Reinier H. Stribos

M.Sc. Thesis
March 2023



Supervisors:
prof. dr. M.I.A. Stoelinga
prof. dr. T.M. Heskes
M.Sc. M.C. Slot
M.Sc. L.A. Jimenez
M.Sc. R.C. Bouman

Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Abstract

Additive manufacturing, the process of producing parts from 3D models in a layer-to-layer process, has seen an enormous growth in recent years. However, consistently producing high quality part remains challenging. One possible anomaly affecting the output quality during the printing process is recoater streaking. Different detection models have been proposed in literature with varying levels of proficiency. However, a thorough comparison of these models is lacking. Moreover, every model is only tested and tailored to their own specific datasets. In this research, these different detection models have been implemented and compared to get a better overview of the advantages and disadvantages of each model. Furthermore, an existing method has been improved to make it more general applicable and a tried and tested pre-processing step has been introduced to this application. All tested models score >96% accuracy, but three models outperformed the others and these three all exceed in a single metric. Therefore, it depends on which metrics are deemed most important which model is regarded as the highest performing model.

List of acronyms

AM	Additive manufacturing
RCS	Recoater streaking
LB-PBF	Laser beam powder bed fusion
LBP	Local binary pattern
FF	Fitted function
MA	Moving average
LBP	Local binary pattern
BoW	Bag-of-Words
CNN	Convolutional neural network
AUC	Area under the curve
ROC	Receiver operator characteristic
AP	Average precision
TP	True positives
TN	True negatives
FP	False positives
FN	False negatives

Contents

Chapter 1

Introduction

1.1 Problem description

Additive manufacturing (AM), the process of producing parts from 3D models, has seen an enormous growth in recent years. This is mainly because, when compared to traditional manufacturing, designing a new model is substantially faster [1]. Additionally, more complex geometries are possible, and it needs significantly fewer materials. These properties make it promising for aerospace and medical applications [2], as well as rapid prototyping and producing critical parts on-site [3].

However, these industries require a high degree of quality assurance and process reliability, which are difficult to achieve [1]. Analysing the quality of the printed part afterwards is possible but very costly, and it does not improve the quality nor does it guarantee it. Early warning systems that tell about the presence of anomalies during the printing process are necessary to save resources as a small deviation during the printing process can have a big impact on the final part quality [4].

Some AM processes work by fusing powder together and printing a part layer by layer. For each layer, powder is spread over a build plate and fused together with a laser into a two-dimensional part cross-section, this is repeated until the complete part is printed. More information about these processes can be found in Section

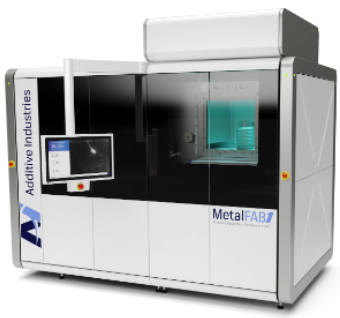


Figure 1.1: The printer studied in this research, the MetalFab1 from [5].

2.1 and Figure 1.1 shows a printer that works according to this process. Regarding such AM processes, defects can be categorized into three classes:

- *Powder related*: defects regarding the spreading of the new powder layer.
- *Process related*: defects regarding the interaction between the laser, powder, and the metal.
- *Post-process related*: defects occurring during the post-processing afterwards [6].

While not related to the AM process itself, deformation can also be caused by equipment or the design of the to-be-manufactured part [7].

The powder related anomalies can have a big effect on the final part quality [8]. In a powder-AM process, every layer of the print starts with coating the build plate with a new layer of metal powder. The distribution of the new powder layer is done by the recoater. The recoater must ensure that the powder is distributed as evenly as possible, every layer is identical, and must not disturb the previous layers [9]. The recoater works by carrying a cartridge across the build plate dispensing powder along the way which is then pushed by either a roller or a blade into a uniformly spread layer [6], [9]. During the coating of a new layer, distortions can occur in the powder layer and can be divided in the following defects; *recoater hopping*, caused by the recoater blade striking a part and 'hopping', *debris*, debris located in the powder bed, *incomplete spreading*, an insufficient amount of powder is fetched from the powder dispenser, *elevation of printed parts*, a part curling upwards out of the powder layer, and the subject of this study, *recoater streaking* [6], [10], [11].

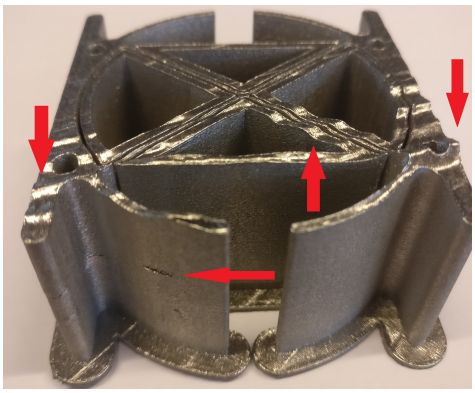


Figure 1.2: 3D printed part with recoater streaking defect. The red arrows indicate the surface roughness.

This research focuses on Recoater streaking (RCS) because it is the most occurring defect regarding the powder-related defects [11]. RCS is caused either by the recoater dragging an elevated part across the powder bed or by a damaged blade or

roller [3], resulting in an increase in surface roughness [12] and rendering the final part unusable. Figure 1.2 shows a printed part with increased surface roughness.

RCS is characterized by stripes in the powder layer parallel to the recoater direction [3]. Figure 1.3 shows a powder bed with instances of recoater streaking. The two dark stripes between the red lines are occurrences of recoater streaking.

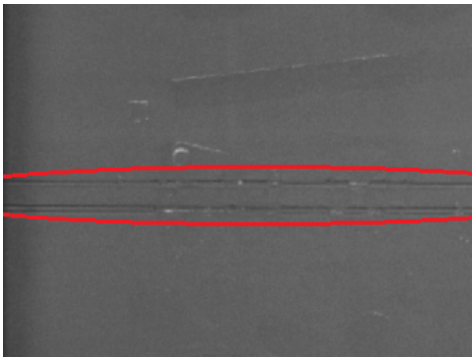


Figure 1.3: Powder bed with instances of recoater streaking (two dark stripes).

Detection methods that can detect RCS early on, could save many resources. Such detection methods, specialized for RCS, have been developed. Some of these methods incorporate the use of Line Profiles [12], Clusters, [3], Neural Networks [13]–[16], and Support Vector Machines [17]. A more detailed overview of the methods can be found in Table 1.1. These methods are all designed to detect RCS based on images and perform quite accurately in general. However, these methods have all been tested on, and tailored to, their own specific data sets. To really be able to compare different models, they should be tested on the same data sets, which has not yet been done [10]. Furthermore, as each model has only been tested on its own data, it is currently unknown how each model reacts to a different data set [20].

This research has the following objectives; (1) exploring and implementing image-based anomaly detection algorithms to improve early-warning systems on the detection of RCS, and (2) thoroughly compare the performance of state-of-the-art algorithms in a unifying benchmark. In addition, (3) focus has been laid on the case of MetalFab1 machines and to tailor algorithms with its requirements and conditions.

This study showed that multiple algorithms can detect RCS and perform accurately. However, different algorithms excel in different performance metrics. Furthermore, the effectiveness of different pre-processing steps differs per detection algorithm. Therefore, which algorithms to use should be decided on a case by case basis.

Ref.	Year	Method	Reported Acc.	Benefit	Shortcomings	Sec
[12]	2011	Line profiles	-	Very quick	Needs specific lighting and camera set-up	4.4.1
[3]	2018	Bag of Words + filter features	83.4%	Detects multiple anomalies + can do segmentation	Slow + low accuracy for RCS	4.4.2
[13]	2021	Neural network	98.74%	Detects multiple anomalies + high acc.	Needs large training set	4.4.3
[14], [15]	2018	Neural network	93%	Detects multiple anomalies + can do segmentation	Very slow	4.4.4
[16]	2021	Neural network	90.3%	Detects multiple anomalies	Tested on small dataset	4.4.5
[17]	2021	Local binary patterns	98.3%	Detects multiple anomalies + high acc.	Only tested for recoater hopping detection	4.4.6
			Not tested in this study			
[18]	2021	Pixel intensity	-	Works on small training set	Designed for Electron Beam Melting	✗
[19]	2014	Low coherence interferometry	-	Deformities clearly visible in scans	Needs specific optical sensor + untested	✗

Table 1.1: Overview of existing anomaly detection algorithms.

1.2 Research objectives and questions

As mentioned previously, literature provides multiple possible methods to detect RCS. However, what is missing is a clear comparison of the different methods, distinctly showing the advantage and disadvantages of each model, so that when developing a new machine, or implementing a new detection model, an informed decision can be made.

The goal of this research is to create such a comparison. For this comparison quantitative and qualitative metrics have been selected and are explained in more detail in Section 4.2. In addition, all RCS detection models have been implemented to, and benchmarked on the MetalFab1 machine. To be able to reach this goal, the following research questions have been established.

1. Which detection algorithm has the best overall performance when detecting recoater streaking?
    (a) Which detection algorithm has the best performance across all metrics when detecting recoater streaking?
    (b) Which detection algorithm has the best performance when tailored to detecting recoater streaking on the MetalFab1?

The rest of this thesis is structured as follows: first, Section 2 provides some background information about the topics discussed during this research, while Section 3 provides a detailed explanation on the dataset used to benchmark the different models on RCS. Section 4 provides details on the steps taken to execute these benchmarks. Section 5 presents the results of the different models and Section 6 discusses these results. Finally, Section 7 presents the conclusion of this research, and Section 8 discusses interesting research directions for future work.

Chapter 2

Background

This chapter presents some background information on a few aspects of this research. Section 2.1 describes the specific additive manufacturing process studied in this research and Section 2.2 describes the specific machine. Section 2.3 gives some details about the research field of anomaly detection. Finally, Section 2.4 gives some details about neural networks and their use.

2.1 Laser Beam Powder Bed Fusion

LB-PBF is an additive manufacturing process that uses a laser to melt and fuse metal powders together to print a part. Figure 2.1 shows a schematic overview of the printing process. This process starts with (1) the creation of a 3D model, called a CAD-model, which is sliced into a number of very thin (20-100 μm thick [21]) layers. (2) this model is loaded into the machine and the printer then lowers the oxygen level in the build chamber to around 50 ppm O_2 to ensure that the metal does not oxidize during printing. (3) for each layer, a dispenser (i.e. recoater) coats the build plate

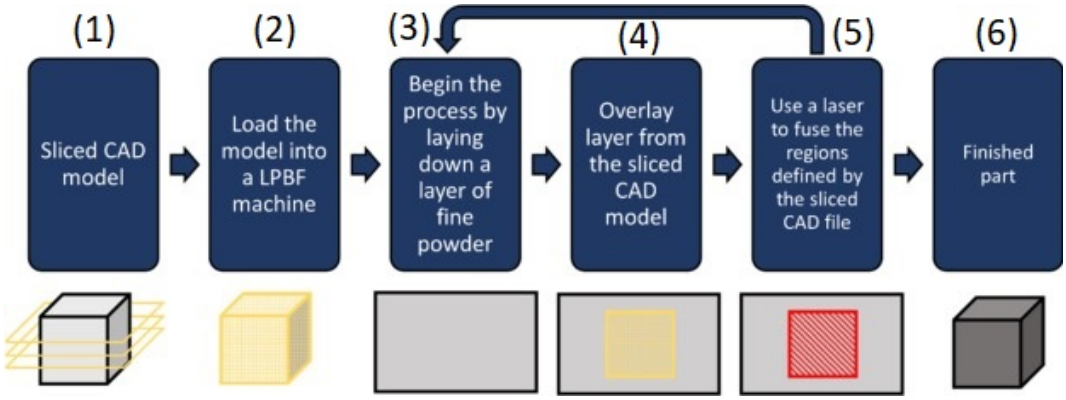


Figure 2.1: Build process for an LB-PBF machine. Rectangular gray drawings represent a powder bed. Picture taken from [6].

with a thin, uniformly spread layer of metal powder. (4) the layer from the sliced CAD model is projected and (5) the powder is then melted together using a laser beam to recreate the layer of the sliced model. Surrounding powder remains loose and serves as support for subsequent layers. After one layer is finished, the build plate lowers slightly and steps 3-5 are repeated, until all layers have been printed. (6) the melted particles fuse together and solidify to eventually form the original 3D model, layer by layer.

After the part is printed, the powder is automatically removed and stored for later use, and the build plate is moved to the heat treatment furnace for post-processing. During post-processing the part is heated and subdued to pressure evenly across its surface to remove residual stresses that have build up in the metal.

2.2 MetalFab1

The printer studied in this research, the MetalFab1 in Figure 1.1, is developed by Additive Industries and was released to the public in 2017 [5]. The MetalFab1 is a Laser Beam Powder Bed Fusion metal 3D printer. It has a build chamber of 420x420x400 mm, 4 lasers, and can print up to 150cm^3/h [21]. It can print a broad variety of metals, but this research focuses on titanium.

2.3 Anomaly detection

Anomaly detection refers to the problem of detecting patterns in data that conflict with normal behaviour. Detecting anomalies in data has been studied as far back as the 19th century [22] and is used in a wide variety of different fields. E.g., it has been adopted to detect, among many other applications, fraud detection for credit cards, intrusion detection for cyber-security, and fault detection in safety critical systems [23]. Over the years many different anomaly detection models have been developed, ranging from very specific for certain applications to more generic models that can be applied extensively.

Anomaly detection has been widely applied to AM [24], [25]. E.g., research has been done to detect, among other things, lack of fusion [26]–[28], porosity [29]–[32], irregular heating [33]–[35], or recoater streaking. Some anomaly detection models have been specifically designed for the field of AM. E.g., Grasso et al. proposed a detection model for RCS for an electron beam powder bed fusion process [18], which is closely related to LB-PBF . They analysed images by first comparing an image post-meltering with a post-recoating image of the same layer and highlighting the difference between these two images. The author's reasoning was that this will result in extreme values for defects in the powder bed. Secondly, they applied a

transfer function to the newly obtained image to enhance the isolation of the extreme values. As can be seen in Figure 2.2, this function will return a high value for the extreme values and a low value for the central values in between. Finally, a pixel is classified as anomaly if its new value is higher than a pre-defined threshold.

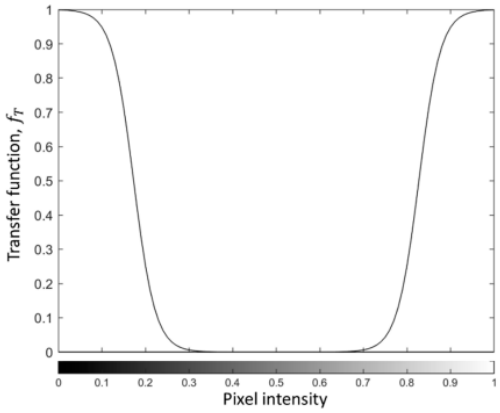


Figure 2.2: Transfer function to highlight extreme values. Image taken from [18].

One field of anomaly detection, is visual anomaly detection, the field of interpreting digital images or videos [36]. Filter response features have been found to be very effective. They can be used to highlight, filter-out, or extract certain features of images and various filters for various situations have been created [37]–[39]. Figure 2.3 shows an example of different filters being applied to the same image. Filter features have been used frequently in literature for feature extraction. E.g., Koblar et al. [40] used filter response features to extract features from fingerprint images. In this research, various methods use such filters to detect RCS.

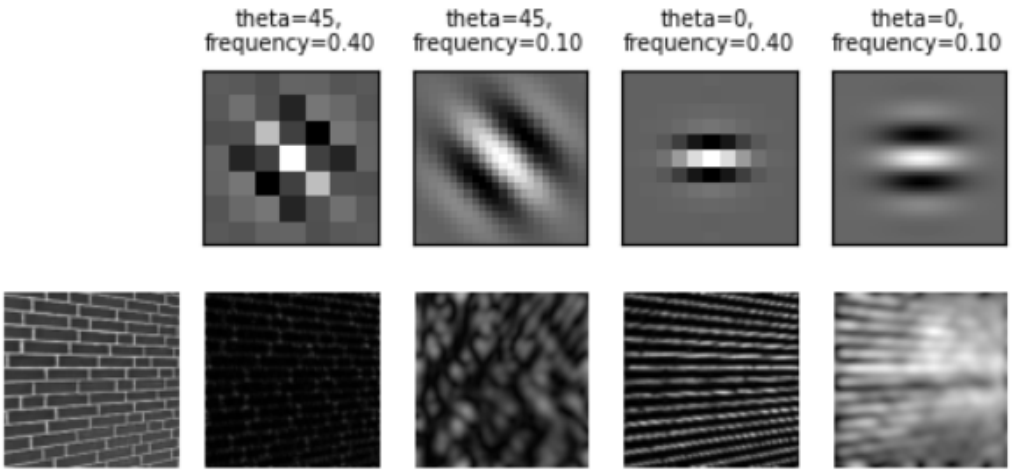


Figure 2.3: Effect of different Gabor filters on an image. Image taken from [41].

2.4 Neural networks

In this study, multiple neural networks are employed. A neural network consists of multiple layers filled with nodes called neurons [42]. A neuron receives signals from the neurons in the previous layer and processes them to signal neurons in the subsequent layer. These signals travel from the input layer to the output layer. This section describes various aspects of neural networks used in this research.

Traditional neural networks consist of only fully connected layers, where each neuron has a connection with every neuron in the next layer. These layers are a large computational burden. Convolutional neural networks make use of convolutional layers next to fully connected layers. A convolutional layer contains a set of feature maps [43]. The height and weight of the feature maps are smaller than those of the input image. Each feature map is convolved with the input image to compute an activation map made of neurons. In other words, the feature map is moved across the input image and the dot products between the image and feature map are computed at every position. All neurons within a feature map share the same weight, resulting in fewer parameters and computations [44].

Neural networks may include pooling or normalization layers in between two convolutional layers. Pooling layers reduce the dimensions of the data and by doing so, also the amount of parameters and computations of the network. They do this by combining the outputs of a cluster of neurons into a single neuron for the next layer. The two most common types of pooling are average and max pooling. Average pooling uses the average value of these clusters, while max pooling takes the maximum value [45]. A normalization layer normalizes the input of a layer over all of the summed inputs to the neurons for a layer to significantly reduce training time [46].

Every node not in a normalization or pooling layer, has an activation function. This function defines the output of that node given an input. Furthermore, an activation function introduces non-linearity in a neural network, which allows a network to calculate complex and abstract equations across many layers. Multiple different activation functions exist, but in this study only the ReLu and the Softmax functions are used. The rectified linear unit, or ReLu, will, as shown in Figure 2.4, output the input directly if it is positive, otherwise, it will output zero and is defined as $ReLu(z) = max\{0, z\}$ [47]. ReLu activation functions offer a faster computation and a better back-propagation than other activation functions [48]. Thanks to its simplicity and effectiveness, ReLU has become the default activation function used when designing neural networks.

The other activation function used in this research, is the *softmax function*. The softmax function converts a vector of n numbers into a probability distributions of n outcomes [49]. This function is often used as the activation function of the last

layer of a neural network to normalize the output to a probability distribution over the possible classes, and is defined as:

$$\sigma(z) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}, \text{ for } i = \{0, ..., n\} \text{ and } z = (z_1, .., z_n)$$

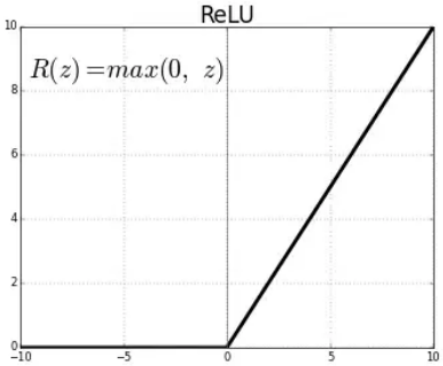


Figure 2.4: The ReLu activation function.

Chapter 3

Dataset description

Since all described algorithms are image recognition based, pictures are needed to compare them. As mentioned in Section 2.1, a print is printed layer by layer. The machine takes two pictures of the build chamber during every layer. One picture is taken after the build plate has been coated with a new layer of metal powder, while the other is taken after the layer has been exposed to the laser and the powder has been fused together. Because RCS occurs while spreading the new layer of powder, only the pictures after recoating are kept. Figure 3.1 shows two images from the dataset, one with RCS and one without.

The dataset consists of five different prints, totalling 9.136 images of printed layers. Of these five prints, two prints display RCS, totalling 3.378 images with significant RCS. All images are about 210 KB large and are stored in PNG files. Table 3.1 shows an overview of the different prints.

ID	Number of layers	Layers with RCS
F04	2542	104 - 2542
F12	913	-
F14	3435	2495 - 3435
F15	2061	-
F16	185	-

Table 3.1: Overview of the different prints in the dataset. Showing the print ID, number of layers, and the layers containing RCS.

(a) (b)

Figure 3.1: (a) shows an image of the powder bed taken after the build plate has been coated with metal powder, (b) also shows an image of the powder bed taken after the build plate has been coated with metal powder, but here RCS is visible.

Chapter 4

Methodology

This chapter will discuss the methodology used to answer the research questions as introduced in Section 1.2. First, Section 4.1 introduces the pipeline set up to test the different algorithms. Section 4.2 defines the different metrics used to compare the different algorithms while Sections 4.3 and 4.4 describe the different pre-processing techniques and the implementation of the different detection algorithms respectively. After that, Section 4.5 explains how the different types of inputs were labelled. Finally, Section 4.6 illustrates the steps taken to ensure an unbiased comparison.

4.1 Study pipeline

Figure 4.1 shows the pipeline used to evaluate and compare the different anomaly detection algorithms and the pre-processing steps. It starts (a) with an input image or set of images, to which (b) one or none pre-processing algorithm is applied. (c) If necessary for the detection algorithm, like the algorithms as discussed in Sections 4.4.2, 4.4.4, and 4.4.5, the pre-processed input is partitioned into the right sized blocks. Finally, (d) the input is classified by one of the algorithms compared in this study, and (e) the results are evaluated by the metrics as defined below.

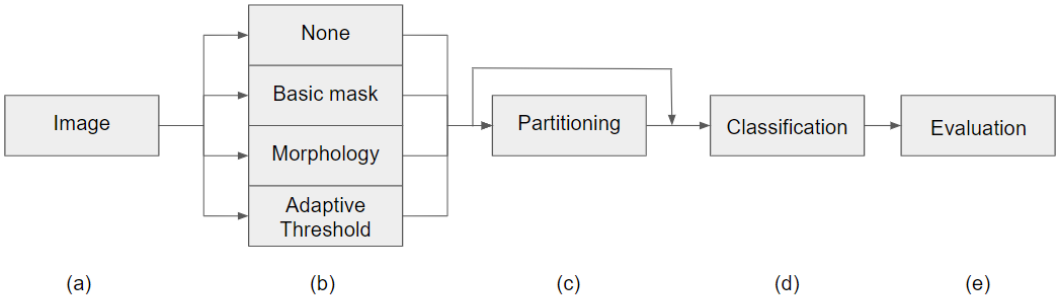


Figure 4.1: The pipeline of the study.

4.2 Comparison metrics

		Predicted	
		Pos	Neg
Actual	Pos	TP	FN
	Neg	FP	TN

Table 4.1: Confusion Matrix.

To answer research question 1.a *Which detection algorithm has the best performance across all metrics when detecting recoater streaking?*, these metrics need to be defined. This study compared the different algorithms on both quantitative metrics that can be measured and qualitative metrics that describe various capabilities.

The quantitative metrics are calculated based on confusion matrices like Table 4.1. A confusion matrix displays the predicted classes from a detection algorithm against the actual classes. It expresses this in the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

The first metric, the **accuracy**, describes the ratio of correctly classified instances and is defined in Equation 4.1.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Recall is the ratio of RCS instances that are successfully identified [50] and is defined in Equation 4.2.

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

Precision is the ratio of correctly identified RCS instances [50] and is defined in Equation 4.3. The recall and precision will be reported for the best performing accuracy only.

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

The **area under the curve** (AUC) is the area under the receiver operator characteristic (ROC) curve [51]. In a ROC-curve, the true positive rate, or recall, is plotted against the false positive rate (FPR), for different thresholds. An optimal ROC has a true positive rate of one for a false positive rate of zero, resulting in a high AUC. The AUC shows how good a model is at ranking predictions. It measures the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance. A formal definition is given in Equation 4.4.

$$\begin{aligned} FPR &= \frac{FP}{FP + TN} \\ Area\ under\ the\ curve &= \int_{FPR=0}^{1} r(FPR)\, d(FPR) \end{aligned} \tag{4.4}$$

Average precision (AP) calculates the area under the graph plotting recall against precision [52]. It is an indication whether a model correctly identifies all the positive instances without marking too many negative instances. It calculates the average of precision scores for different recall thresholds. A formal definition is given in Equation 4.5.

$$Average\ Precision = \int_{r=0}^{1} p(r)\ d(r) \tag{4.5}$$

Furthermore, the **processing time** will be measured. This measurement has been split up into the training speed and the prediction speed. This is done because these two can vary greatly and different use cases demand different requirements for the two.

To answer Research question 1.b, *Which detection algorithm has the best performance when tailored to the MetalFab1?*, these metrics and measurements were also compared solely for the MetalFab1 to find the best performing model for that machine and its requirements. One additional requirement for translating one of the detection models to a real-time detection system used in the machine, is that the time to process a single layer should be roughly equal to the time it takes to print a small layer, below one second, and ideally below 100ms [5].

Next to the quantitative metrics, qualitative metrics are compared also. The qualitative metrics cannot be measured but describe properties of the models. For example, different industries prioritise different things. Accurately detecting every mistake is important for medical applications or aerospace because failures are costly, but rapid prototyping can afford some mistakes and prioritises speed. Some algorithms can be easily adjusted to these print specific requirements before a print while others would require major changes in their design. Furthermore, some models can detect multiple defects in the powder bed simultaneously while others can solely detect recoater streaking. To describe these properties, the following metrics have been selected:

- Possibility for detection of multiple anomalies types
- Adjustability for prints specific requirements

4.3 Pre-processing

The images used in this study are taken directly from the build chamber inside the machine. These images are not perfect and contain noise. E.g., because the powder layers are very thin, and metal is reflective, printed parts from the previous layer are sometimes visible. Figure 4.2 shows an example where parts are very clearly visible. For the printing process itself, it presents no problem. However, when searching

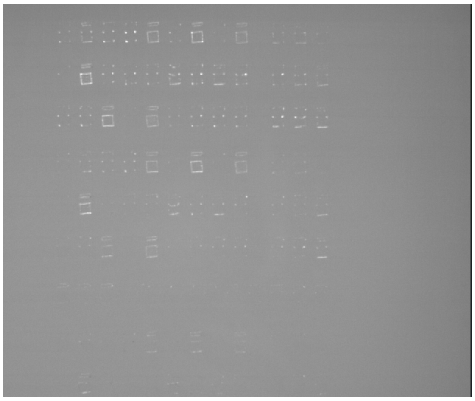


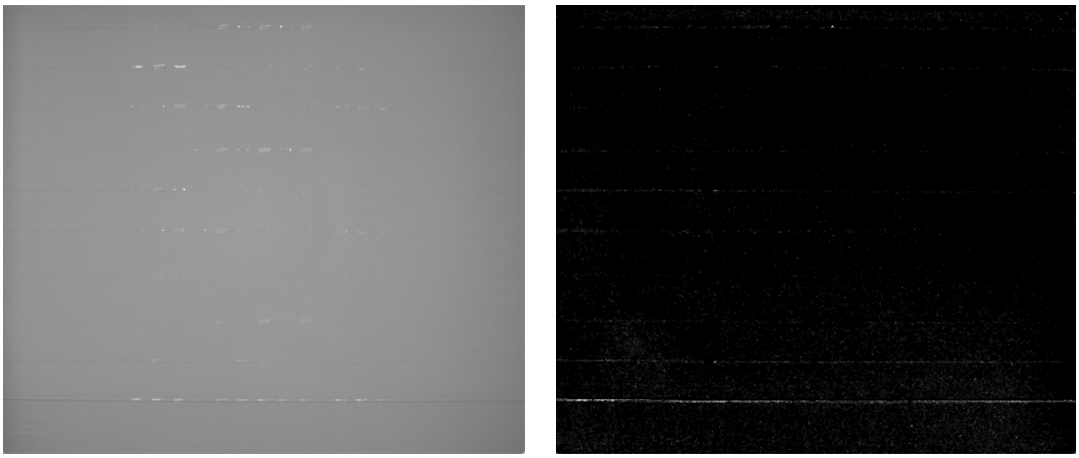
Figure 4.2: Powder bed image with part of the previous layer still visible.

for horizontal stripes in the powder bed, it can affect the final outcome. As seen in Figure 4.2, these parts can be lined up and clear lines become visible in the powder bed. If these lines are significant enough, they affect the detection models and result in false positives. Furthermore, Figure 3.1(b) in Section 3 showed an image containing RCS. As can be seen, RCS is not clearly visible and is quite small. Therefore, it can become quite difficult to detect. Pre-processing images could make RCS more clearly visible by either removing noise like reflected parts from previous layers, or by enhancing the RCS, and by doing so, increase the performance of the detection algorithms.

Method	Source	Goal	Section
Basic mask	[17]	Defect extraction	4.3.1
Image morphology	[53]	Noise reduction	4.3.2
Adaptive threshold	[13]	Defect extraction	4.3.3

Table 4.2: The different pre-processing algorithms used in this research. With their source paper, method, goal, and the section which describes them in more detail.

Two detection algorithms also included a pre-processing step. These two pre-processing steps were both designed for defect extraction. As described above, noise also presents some problems while detecting defects. Therefore, a third pre-processing step is introduced to reduce said noise. This step is used more often in computer vision to reduce noise [53] but has not been applied to additive manufacturing before. These three pre-processing steps can be found in Table 4.2 and will be implemented and combined with each detection algorithm to see whether they can improve the performance. The rest of this section explains the pre-processing steps in more detail. The pseudo code implementation of the algorithms can be found in Appendix B.1.



(a) Image with RCS (b) Basic mask processing

Figure 4.3: (a) shows a powder bed with RCS. (b) shows the same image but after basic mask processing.

4.3.1 Basic mask

The first method is the basic mask as introduced by Yin et al. in 2021 [17]. As seen in Table 4.2 this method was designed for defect extraction. Figure 4.3 shows an image with RCS and its processed image. As can be seen, RCS has been selected and extracted from the image and is now clearly visible. The basic mask pre-processing method consists of three steps which are explained in more detail below:

1. Comparison with a 'basic mask'
2. Median filtering
3. Gamma transformation

To extract anomalies in an image, first a defect-free powder bed image under the same light source was selected, called the 'basic mask', and a threshold range was set. The difference between the collected image and the base mask was calculated to obtain a new image. If the gray value of a point on the new image is within the threshold range, a possible defect is considered and its gray level is retained. If a point is not in the threshold range, it is considered defect-free and the value if set to 0. Due to this binarization, RCS becomes much more clearly visible.

Next to enhancing defects in the powder bed, this first step also introduces some noise as such a basic mask is never spotless. To remove this noise, a subsequent step is added, median filtering. Median filtering is a tried and tested method ideal for reducing random noise while not blurring image features [54]. It works by changing

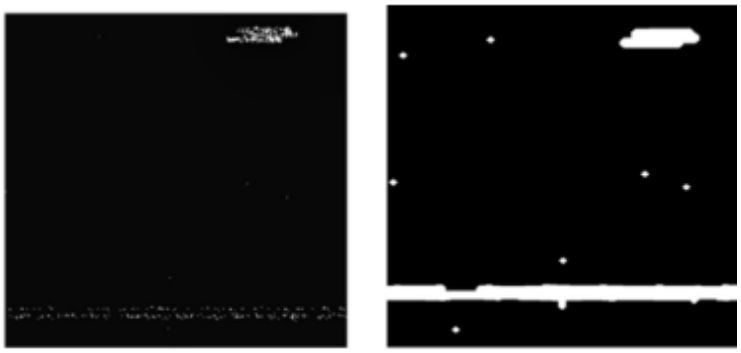
(a) Image with RCS (b) Image morphology processing

Figure 4.4: (a) shows a powder bed with RCS. (b) shows the same image but after image morphology processing.

each pixel to the median of its neighbouring pixels [54]. In this study, the eight direct neighbours are used. This way, smaller, random noise is filtered out, while larges structures like RCS remain. The last step, Gamma transformation [55] is performed to, again, enhance the anomalies in the image. During a gamma transformation, an image is raised to the power of gamma, thereby enhancing the contrast in an image.

4.3.2 Image morphology

Image morphology has been around quite some time and has been used before to remove noise from images [53]. This study introduces this method to the field of Additive Manufacturing. Figure 4.4 shows an image with RCS and its processed image. As can be seen, the background noise has been filtered out while the RCS remains visible. The pre-processing method consists of two steps. First the image is probed with a small horizontal line, 30 pixels wide by 1 pixel high, and eroded [56], meaning that every pixel is changed to the darkest pixel in that small line. This way, the light pixels of the reflecting parts are eroded out while the darker pixels of the RCS are maintained. The second step is the inverse, dilation [56]. Here, again the image is probed with the same horizontal line. But in this case, every pixel is changed to the lightest pixel in that line. This sequence of erosion and dilation is also called opening and is more often used to remove noise from an image while preserving the shape and size of larger objects [53].



(a) (b)

Figure 4.5: (a) shows an image after adaptive thresholding and (b) shows the same image but after image morphology. Both images are taken from [13].

4.3.3 Adaptive threshold segmentation

The adaptive multi-defect threshold segmentation algorithm relies on the difference in gray values of different types of powder bed defects [13]. A histogram of the gray values in an image is constructed and a function is fitted to this histogram. The adaptive thresholds are calculated based on the set of local extrema of this function. The image is then segmented by highlighting both the lowest and highest pixels and dimming all pixels in between. This is done by setting each pixel to 0, black, if it is in between the two thresholds, or to 1, white, if it lies outside.

The segmented image now consists of many small areas of white pixels. These small areas will be connected based on image morphology by dilating, opening, and closing the image with pre-defined structures as defined in [13]. After the morphology operations there still are a small number of discrete defect areas and noise. To diminish this, an area and distance threshold were set. If an area is smaller than the area threshold, it is considered noise and is removed. If the distance between two areas is smaller than the distance threshold, the two are merged together [13]. This pre-processing algorithm proposed by Shi et al. highlights defects in the powder bed. Figure 4.5 shows an image before and after these image morphology modifications.

4.4 Detection algorithms

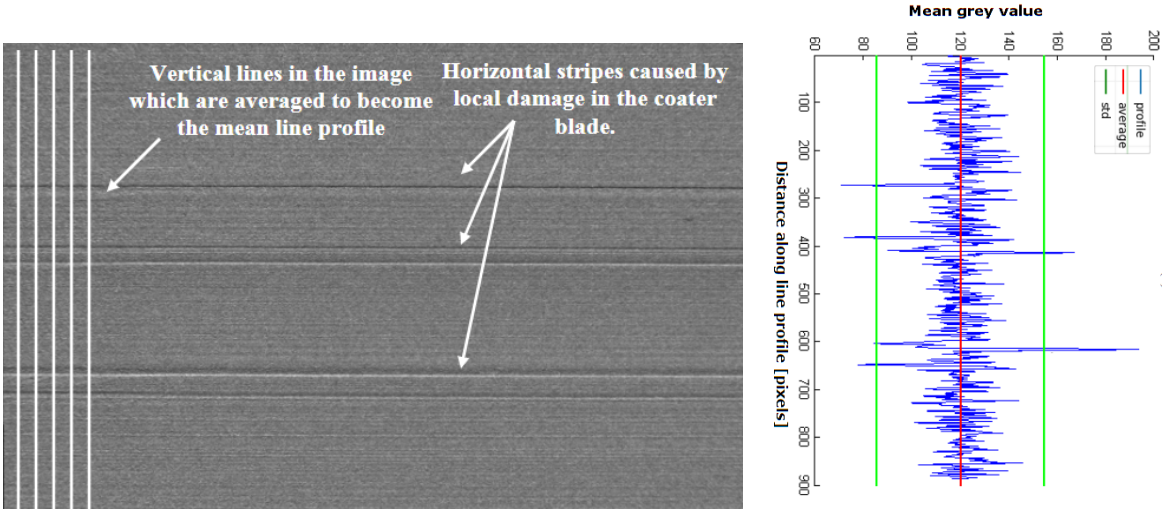
A small overview of the different anomaly detection models can be found in Table 4.3. All models are explained in more detail in the rest of this section and their pseudo code can be found in Appendix B.2.

Name	Source	Section
Fitted function line Profiles	This work	4.4.1
Moving average line Profiles	This work	4.4.1
Bag of Words	[3]	4.4.2
AlexNet	[13]	4.4.3
Multi-scale CNN	[14]	4.4.4
Triple-scale CNN	[16]	4.4.5
Local Binary Patterns	[17]	4.4.6

Table 4.3: Overview of all compared detection algorithms

4.4.1 Line Profiles

The first model for recoater streaking detection was introduced by Craeghs et al. in 2011 [12]. This model extracts so called 'line profiles' from each image to detect RCS. A line profile is defined as the average pixel gray value of five points of that horizontal line in the image of a powder bed. Figure 4.6 shows a powder bed image with its corresponding line profiles. These five points were taken at the far left as not to be disturbed by other defects in the powder bed. The standard deviation of the line profiles is then subsequently used as indicator for RCS, and streaking was detected if a single line profile is six times the standard deviation higher or lower than the mean [12].



(a) Construction of line profiles

(b) Constructed line profiles

Figure 4.6: (a) shows a powder bed with recoater streaking. (b) shows a rotated graph of the constructed line profiles from (a) with clear spikes at the heights of the recoater streaking. Images taken from [12].

However, experimental results showed that the images used by Craeghs et al. were lit uniformly while the images from the MetalFab1 are not. Line profiles from a uniformly lit image behave linearly when no RCS occurs. When images are not uniformly lit, this behaviour changes. Figure 4.7 shows line profiles that are constructed from an image with an uneven lighting. As can be seen, the line profiles no longer behave linearly, but obtain more of a parabola shape. Detection based on the standard deviation is therefore impractical.

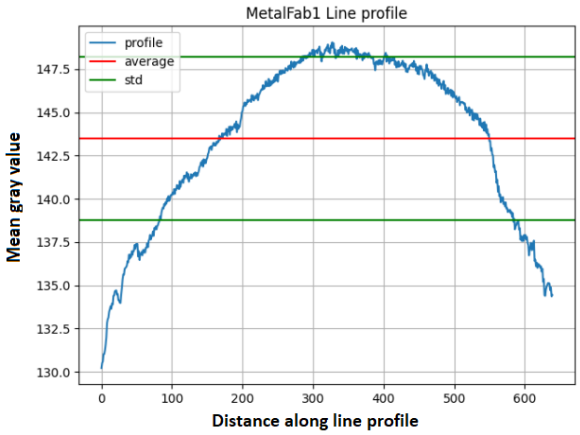


Figure 4.7: Line profiles of an image from the MetalFab1.

To create a detection model capable of detecting RCS independent of the lighting of images, two novel models are proposed in this study and compared to the other detection models. Both models calculate magnitudes of distortion based on the line profiles of an image and detect RCS if, for any point on the image, this distortion is higher than a certain threshold. The first model calculates this distortion by fitting a function (FF) to the line profile's values as a function of the distance along the image and calculating the deviation between the two. A second-degree polynomial is fitted to the line profiles using the non-linear least squares fit as implemented by the curve_fit function from SciPy [57]. The second model calculates this magnitude of distortion by looking at the moving average (MA) of the line profiles. For both models, again, the optimal threshold to detect RCS was selected by maximizing the accuracy.

4.4.2 Bag of words

The first model of Scime et al. utilized bag-of-keypoints [58] in combination with filter response vectors [3]. A schematic overview of the model can be found in Figure 4.8. The model starts (a) with the creation of a filter bank. This bank consists of multiple filters that all respond differently to an image. Figure 2.3 shows an example of different responses from different filters on the same image. These filters (b)

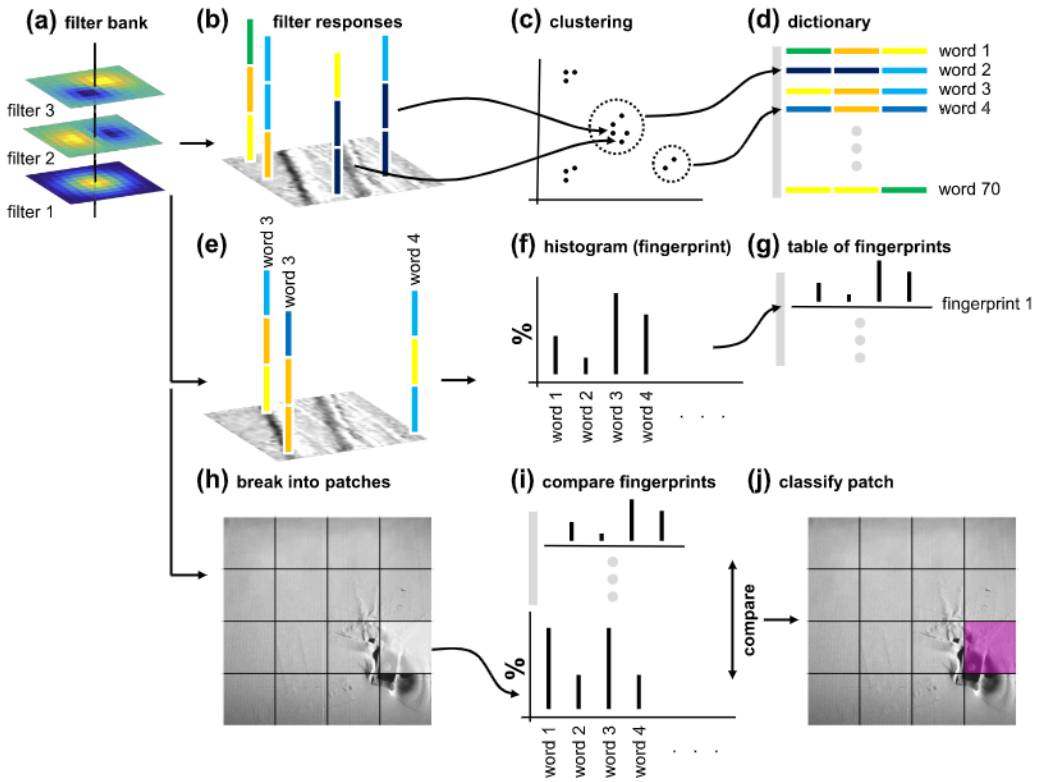


Figure 4.8: Pipeline for the Bag of Words method. Image taken from [3]

are applied on the image and output a 2D image with the same dimensions as the original image, where the value of each pixel is a vector with the responses of the applied filters to the original image. By combining multiple filters, the response vectors contain different sources of information. If pre-processing is used, the images are first pre-processed before the filters are applied.

In the original paper, this bank consisted of thirty-seven filters [3]. However, these filters were described too concisely, making it difficult to investigate which filters were used exactly. In the end, all filters, except the 'oriented line detectors', were uncovered and used for this study. The 'oriented line detector' filters were designed to detect super-elevation defects. The filters designed to detect recoater streaking were uncovered successfully, making the model suitable for this research.

The next step is dividing the images in smaller patches and creating 'fingerprints' for them. First, (c) similar response vectors are grouped together using the k-means clustering algorithm [59] and (d) each group is represented by a mean response vector, that is the mean response of each filter. (e-f) Fingerprints are then created by calculating the percentage of pixels in a patch that are matched to each mean response vector. Patches containing similar anomalies will have similar fingerprints, while their fingerprints are dissimilar from patches with different anomalies or no anomalies. (g) These fingerprint are stored in a table for later classification. In this

study, the proposed 20x20 [3] patches were used.

To classify an image, (h) it is again split up into patches to (i) create fingerprints. To (j) predict whether RCS is present in a patch, its three closest fingerprints are considered. If any off these fingerprints are labeled as defect-free, the patch is classified as such. Because these patches are very small, small irregularities are classified as defects as well. These irregularities are too small to cause any damage and should be ignored. To ensure only images showing real defects are classified as such, only images containing a certain threshold of flagged patches are classified as defects. This threshold has been established by maximizing the accuracy.

4.4.3 AlexNet

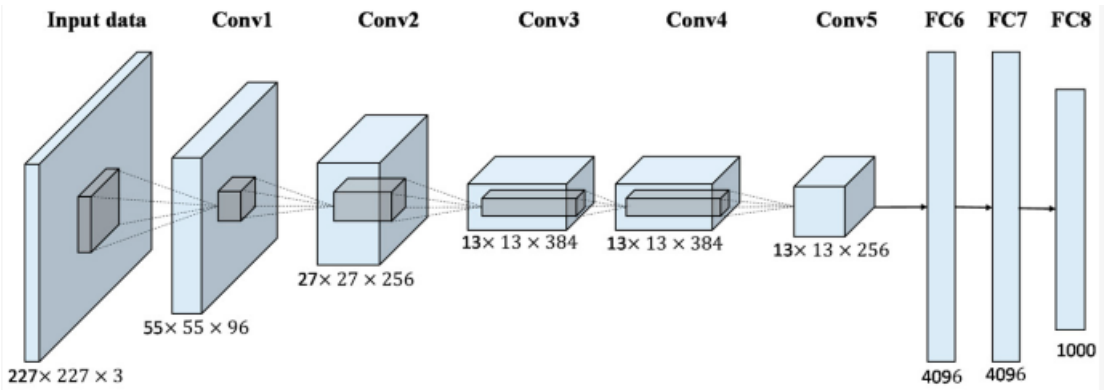


Figure 4.9: Architecture of the AlexNet, image taken from [60].

In the paper of Shi et al. [13], the authors compared three different Neural Network structures for correctly identifying defects in the powder bed; AlexNet [61], VGG-16 [62], and ResNet-50 [63]. Of these three, the AlexNet performed the best with an accuracy of 98.74% on the test set and a detection time of 0.25 seconds per layer. As the AlexNet outperformed the other neural networks by a big margin, only this model will be compared in this study.

The AlexNet consists of five convolutional layers and three fully connected layers [61], Figure 4.9 shows a schematic overview of the architecture. Originally, the last layer outputs a distribution over a 1000 classes since the network was designed for that specific task. However, this can be adjusted for specific cases and in this study a distribution over two classes in returned. Every layer but the last one uses the Rectified Linear Unit (ReLu) activation function while the last layer uses the softmax function. Furthermore, each convolutional layer is followed by a normalization layer and a maxpooling layer.

Because AlexNet was originally designed for colour images, it expects an input size of 227x227x3 pixels, where the three corresponds to the three colour channels.

The images used in the study of Shi et al. and in this study are grayscale images however. To be able to still use the AlexNet, and to stay as close to the paper of Shi et al., the gray value of every pixel was copied into the three colour channels to transfer the image to the desired dimensions [13].

The AlexNet is a rather deep and complex network. Therefore, and because the dataset is quite small, the network is at risk of overfitting [61]. The authors of the AlexNet also described two different ways to combat this, and both were implemented in this study. The first method is data augmentation, which is artificially enlarging the training dataset using label preserving transformations. In this study, of each image also its horizontal and vertical reflections are added to the training set, as well as copies with a slightly higher or lower brightness. Furthermore, random 227x227 patches were extracted from the 256x256 images. This way the training dataset was increased 180-fold without really altering the images, making overfitting less likely to occur.

For the second method, the authors added a dropout for the first two fully connected layers [61]. This dropout consists of changing the output of a hidden neuron to 0 during training with a probability of 0.5. The neurons which are dropped out in this way do not contribute to the forward pass and do not participate in back propagation. This way a different architecture is sampled for every input and single neurons cannot rely on the presence of other particular neurons. This reduces complex co-adaptations of neurons and all neurons are forced to learn more robust features that are useful with many different subsets of neurons.

Without these two methods, the neural network exhibits serious overfitting. With these two methods implemented, training takes a bit longer before convergence but overfitting is reduced significantly.

4.4.4 Multi-scale neural network

The second model proposed by Scime et al. also uses the AlexNet. As mentioned previously, while the images available in this research are grayscaled, the AlexNet expects colour images and therefore three colour channels. Shi et al. solved this problem by copying the gray value three times into the three colour channels [13]. Scime et al. found a more pragmatic way to make use of these three channels [14]. Just like in their previous model, [3], images are split into patches of various sizes to better capture the different anomalies. For this model, patches are extracted with three different sizes; 25x25 pixels, 100x100 pixels, and the whole image. The first two patches share the same center, resulting in overlapping 100x100 pixel patches. These three different patches are all resized to the expected 227x227 input shape using bilinear interpolation and inserted into one of the three colour channels [14].

As multiple patches can be extracted from a single image, more training data is available and overfitting is less of a problem. However, as overfitting was still observed, the reflections and brightness adjustments are still added to mitigate this problem. This was not done in the original paper, but was necessary to make it work.

To classify an image, the different scales are extracted and the AlexNet makes a prediction for each scale centre. If enough of these centres are classified as containing RCS, the whole image is classified as such.

4.4.5 Triple-scale neural network

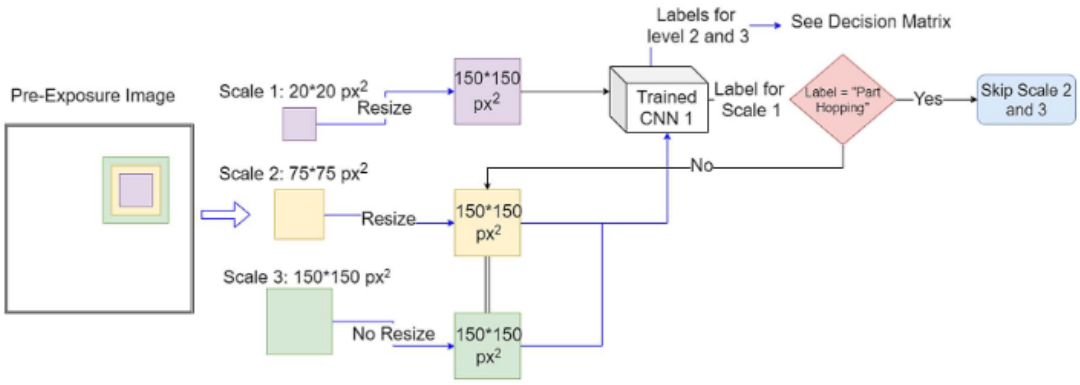


Figure 4.10: Flowchart for the labeling process of the triple-scale network, image taken from [16]

In 2021, Yadav et al. designed a new convolutional neural network (CNN) architecture to detect anomalies in the powder bed [16]. Their model was tested on four different types of anomalies. The authors argued that different anomalies do not have the same spatial detection scale. Therefore, they defined three different scales to detect all irregularities. The detection process as introduced by the original paper is shown in Figure 4.10. The first scale, 20x20 pixels, was defined to detect part hopping and overheating. The second scale, 75x75 pixels, was defined to detect recoater streaking, whereas the third scale, 150x150 pixels, was set to detect uneven powder spread. Blocks from the first scale are non-overlapping while the bigger blocks are, as the three different scales were extracted from the same center, as can be seen in Figure 4.10.

To detect anomalies, the first scale block is passed through the trained CNN and a classification is returned. If this block is classified as part hopping, the model will directly classify is as such and skip the larger scales. If the first scale is not classified as part hopping, the result is discarded and the larger scales for that specific center are passed through the CNN. A label is decided based on the outcome of scale two and three as specified by a decision matrix defined in [16].

This study focuses solely on recoater streaking, making the use of the smallest scale, 20x20 pixels, ineffective. This is due to the fact that this scale was designed to only detect part hopping and overheating, and its result is discarded when that is not present. As this study does not look at part hopping or overheating at all, this first step is skipped entirely. Both the predictions of the other two scales have influence on RCS classification and both scales are therefore consulted in this study. Again, if enough patches within an image are classified as anomaly, the image will be classified as such.

The CNN as designed by Yadav et al. consists of five layers, three convolutional and two fully connected ones [16]. Identical to the AlexNet, every layer uses the ReLu activation function while the last layer uses the softmax function. Again, each convolutional layer is followed by a normalization and a max pooling layer. In comparison to the AlexNet, this model was designed for grayscaled images and thus does not expect three colour channels, but just the one.

4.4.6 Local Binary Patterns

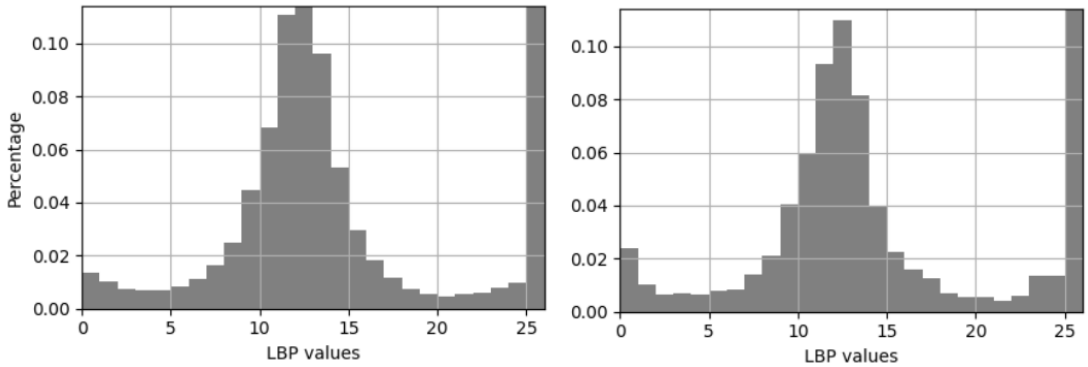


Figure 4.11: (a) shows the LBP values of an image without RCS, while (b) shows the LBP values of an image with RCS.

The model as discussed by Yin et al. was developed to detect recoater hopping [17]. Since recoater hopping is very similar to RCS, the only difference being the rotation of the defect, the algorithm can also be used in this study. This model uses Local Binary Patterns (LBP) together with a Support Vector Machine to detect the defects. A LBP is constructed for an image by, for every pixel, comparing its gray value to the gray values of its eight direct neighbours. These transformations result in 8-bit binary numbers, which decimal values are used to obtain an LBP response image [64]. The histogram containing the LBP for every pixel of an image is used as detection feature for the Support Vector Machine. Figure 4.11 shows two histograms of LBP response images, one (a) without RCS and the other (b) with. In their original

paper, Yin et al. compared several variations of LBP. They found that the unified LBP, multi-block LBP, and partition LBP [65]–[67] were positive improvements. Therefore, these improvements will also be tested in this research while improvements that were found to be less influential, like the circular neighbourhood LBP, are ignored.

The unified LBP looks to speed up the model. A LBP is considered unified when its binary number contains no more than two transitions from 0 to 1 or vice versa [65]. E.g, 10000001 is unified as it has two transitions, while 01100001 has three and is therefore non-unified. Unified LBP capture the important information in images, while non-unified LBP are often caused by noise. By discarding the non-unified LBPs and only looking at the unified LBPs, the process time increases significantly while keeping the important information by reducing the number of redundant features.

LBP can describe the local information of images precisely, but are susceptible to noise. Furthermore, it has trouble capturing the main frame of image features. Multi-block LBP was designed to solve this. It selects a block of $n \times n$ pixels and calculates the average. The LBP response image is then constructed not on single pixels, but on the averages of these blocks. Multi-block LBP captures the main structures better while filtering out smaller noise.

A LBP describes the main structure of an image. Because only one LBP response image is generated per image, local features are often missed while they contain valuable information. Partition LBP looks to capture these different local features by dividing an image into $M \times N$ partitions. A LBP response image is calculated for each partition and the histograms are concatenated to represent the entire image again.

4.5 Labeling of image patches

Multiple models compared in this study try to detect recoater streaking by zooming in on the images and classifying patches. These models are supervised learning models, meaning that they need labelled data to train themselves. Therefore, image patches need not only be constructed, but also labelled.

The line profiles model can locate RCS within an image. Furthermore, it can be applied to both complete images, and image patches. The fitted function version of this model, together with the highest performing pre-processing method, can therefore assist during the labeling of the image patches.

Labelling is done by first locating RCS within an image, dividing the images in patches and labeling each patch in accordance with the location of the RCS. After which each patch is again classified by the line profiles model and this classification is compared to the assigned label. Only patches where this comparison holds are

used for the creation of the training data set.

4.6 Cross validation

To achieve the best performance, optimal hyperparameters were searched for. Hyperparameters are parameters which are used to control the learning process and are set by the user before the training [68]. Unlike other parameters which are derived via training, like network weights. E.g. the partition size of local binary patterns or the number of training epochs of the neural networks are hyperparameters.

A possible pitfall when searching for the optimal hyperparameters is over-fitting. Over-fitting happens when all parameters are tuned too closely to a dataset and any noise in the dataset is registered as normal behaviour [69]. When this happens the model performs extremely well, almost perfect, on the dataset, but fails when confronted with new data.

Over-fitting occurs because the criteria for training the model differs from the criteria used to measure the suitability. E.g. the model is trained by maximizing its performance on a dataset but its suitability is measured by how well it performs on unseen data. Over-fitting then occurs when the model "memorizes" the training data, rather than learning the underlying pattern.

To prevent over-fitting, cross-validation was used. With cross-validation, three different datasets are created; a training set, a validation set, and a testing set [70]. All three sets should follow the same distribution of positive and negative instances.

The training set is used to train the model, after which the model classifies the observations in the validation set. This set provides an unbiased evaluation while trying out different hyperparameters. Because the data in the validation set is completely new for the model, its understanding of the underlying patterns is tested in stead of its memorization of the training data.

Finally, after selecting the best hyperparameters, the model is trained on the training set once more and its final performance is measured on the testing set. This test set is used to measure the generalizability of a fully specified model.

For this study, the dataset has been split in a training, validation, and testing set by allocating 60, 20, 20 percent of the data accordingly. By doing this, not only over-fitting is prevented, but the final performance of every model is measured on exactly the same dataset as well.

Chapter 5

Results

This chapter will discuss the results obtained during this research. Sections 5.2 until 5.5 discuss the quantitative metrics of the different anomaly detection algorithms in combination with different pre-processing methods. Section 5.6.1 compares the best performing combinations with each other and Section 5.6.2 discusses the qualitative metrics of the detection algorithms.

5.1 Hyperparameters

Table 5.1 lists all hyperparameters of this study together with their obtained optimal value. The Bag-of-Words algorithm is not listed as all its parameters were obtained during training or defined in the original paper. The optimal hyperparameters of the local binary pattern changed when different pre-processing methods were used. This table only shows the hyperparameters for the best performing pre-processing method.

5.2 Without pre-processing

Figure 5.1 and 5.2 show the quantitative metrics and the training and prediction times respectively off all the algorithms on the test set without any pre-processing. Table A.1 in Appendix A shows the results in more detail. Apart from the time, the models perform quite similar with all models scoring >90% in both accuracy and precision as can be seen in Figure 5.1. Note that the precision and recall are given for the highest accuracy. There are three models, both line profiles and the bag-of-words, that perform a bit lower than the others. These models score below 90%, but still >80% on the recall, and about 5% lower on the other metrics when compared to the other algorithms. The algorithm with the highest metrics however, is the AlexNet as discussed in Section 4.4.3 which outperforms the other algorithms on all but one metric.

Method	Hyperparameter	Value	Description
Line profile	Window size	4	The size of the window for calculating moving average.
	Fitted function	$ax^2 \cdot bx + c$	The function used to describe the line profiles.
Alexnet	Learning rate	0.001	Controls how quickly the model learns.
	Momentum	0.9	Used to remove random convergence.
	Batch size	64	The number of training samples for each epoch.
Multi-scale network	Learning rate	0.001	
	Momentum	0.9	
	Batch size	64	
Triple-scale network	Learning rate	0.001	
	Momentum	0.9	
	Batch size	64	
Local binary pattern	Block size	3x3	Multi-block LBP takes the average of these blocks.
	Partition size	9x9	Partition LBP constructs a histogram for each of these partitions.

Table 5.1: Obtained hyperparameters for each detection algorithm with a small description.

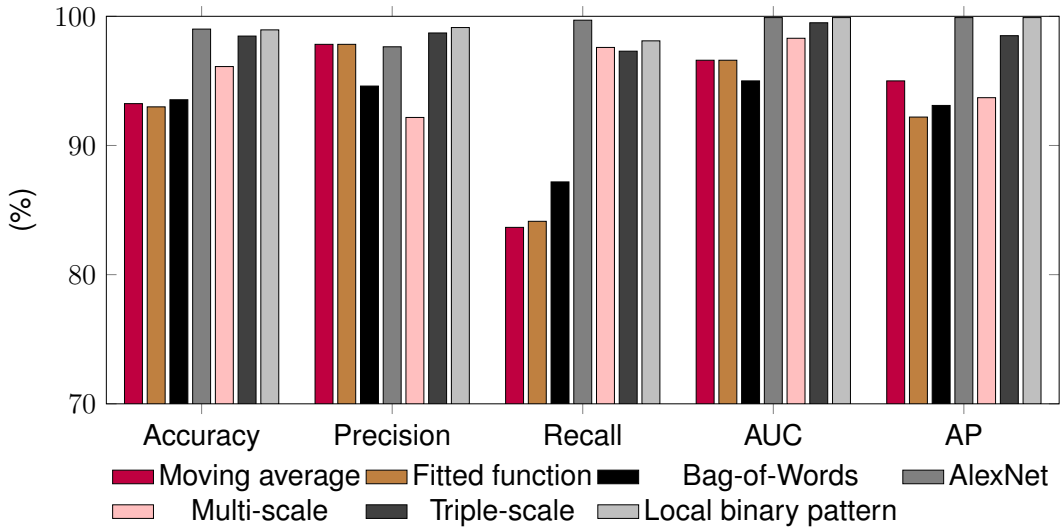


Figure 5.1: Results of all models without any pre-processing.

The biggest difference between the models is the time, as can be seen in Figure 5.2, both to train and to predict. The three neural networks take more than an hour to train while the other are in the range of a half to eight minutes. This big difference is mostly due to the many trained epochs in neural networks. The big difference in prediction speed in mainly happening between the two methods that need to divide an image in very small patches and classify each patch, and the rest. The methods splitting an image into multiple smaller patches take significantly longer than methods that do not need to do that.

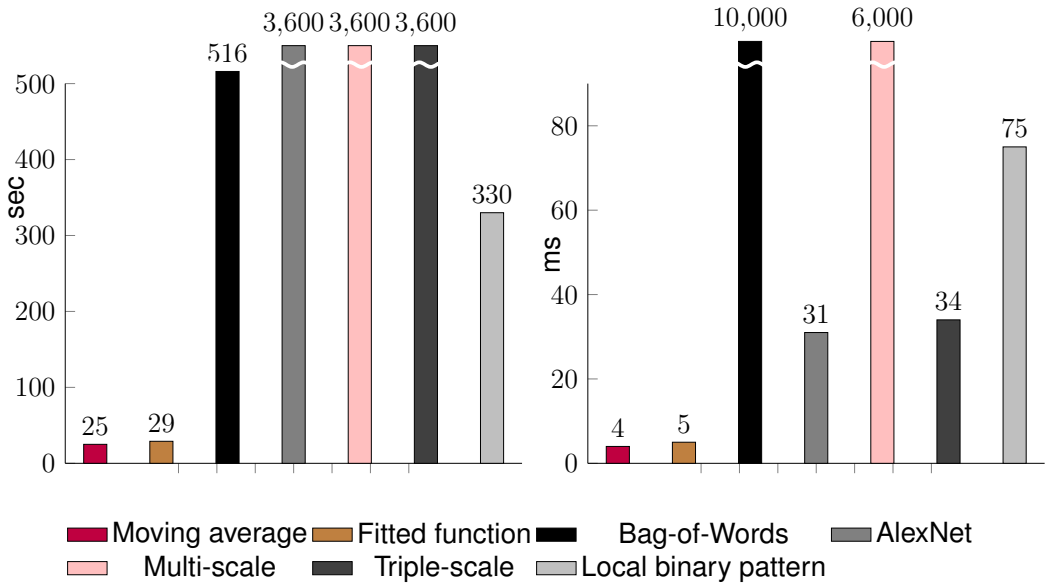


Figure 5.2: (a) Training speed in seconds and (b) the prediction speed in milliseconds of the algorithms without pre-processing.

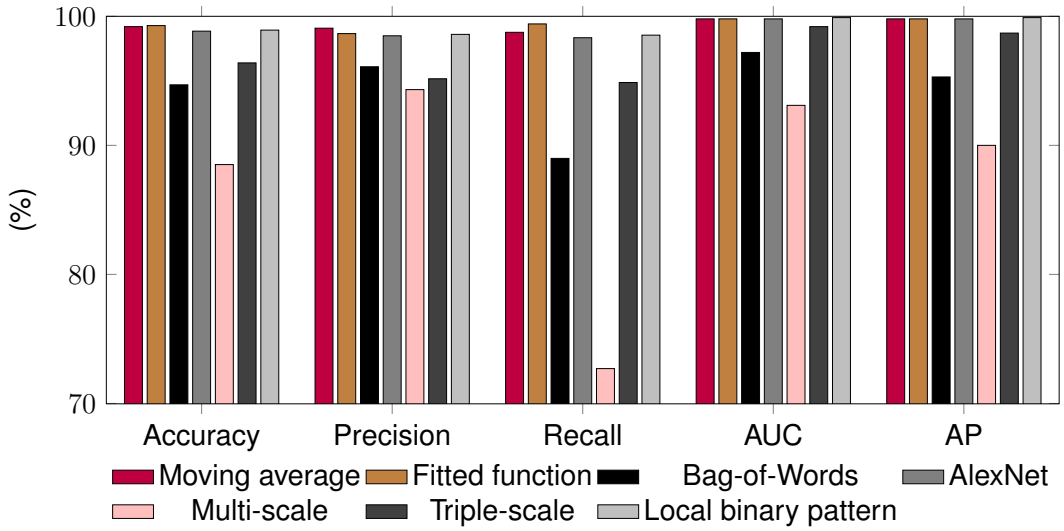


Figure 5.3: Results of all models with basic mask pre-processing.

5.3 Basic mask pre-processing

Figures 5.3 and 5.4 show the results of all the models on the test set with the basic mask pre-processing as described in Section 4.3.1. More details can again be found in Appendix A. The AlexNet performs mostly the same as without pre-processing. The two line profile methods improved significantly, about 6% in accuracy and about 15% in recall, now both outperforming the other models on all metrics. The Bag-of-Words method also improved with 2% on every metric.

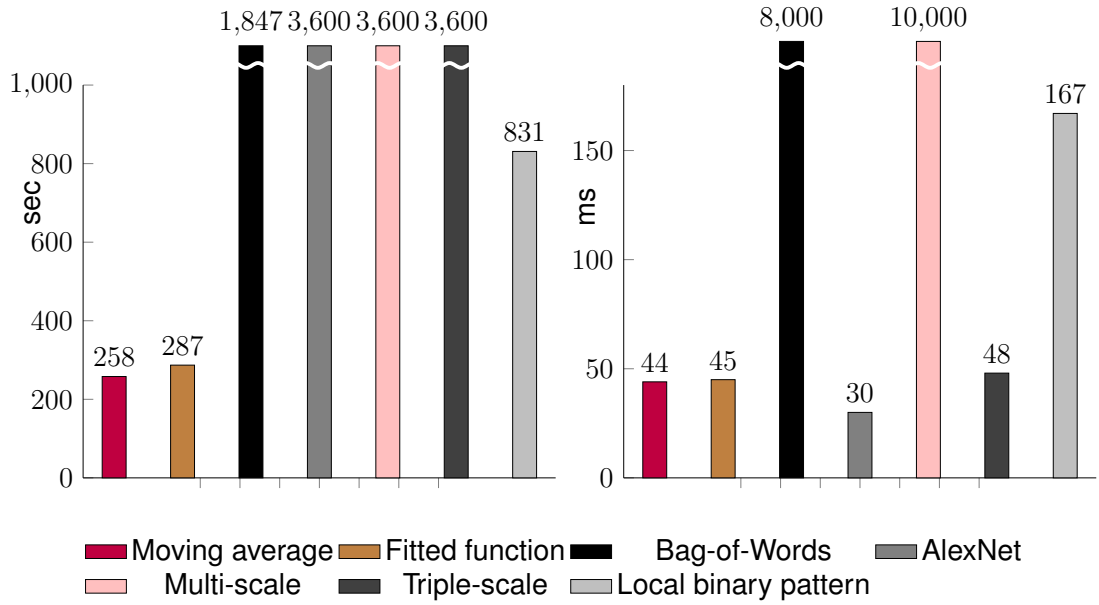


Figure 5.4: (a) Training speed in seconds and (b) the prediction speed in milliseconds of the algorithms with basic mask pre-processing.

While the performance of some algorithms improved, the multi-scale and triple-scale networks performed worse. The recall of the Multi-scale network reduced significantly with 25% and the Triple-scale network declined 2-3% on all metrics.

With the use of basic mask pre-processing, the training and prediction times became much larger for all models. Both line profiles take ten time as long to train and to predict, and Local Binary Patterns and the Bag-of-Word method take twice as long to train, as can be seen in Figure 5.4

5.4 Image morphology pre-processing

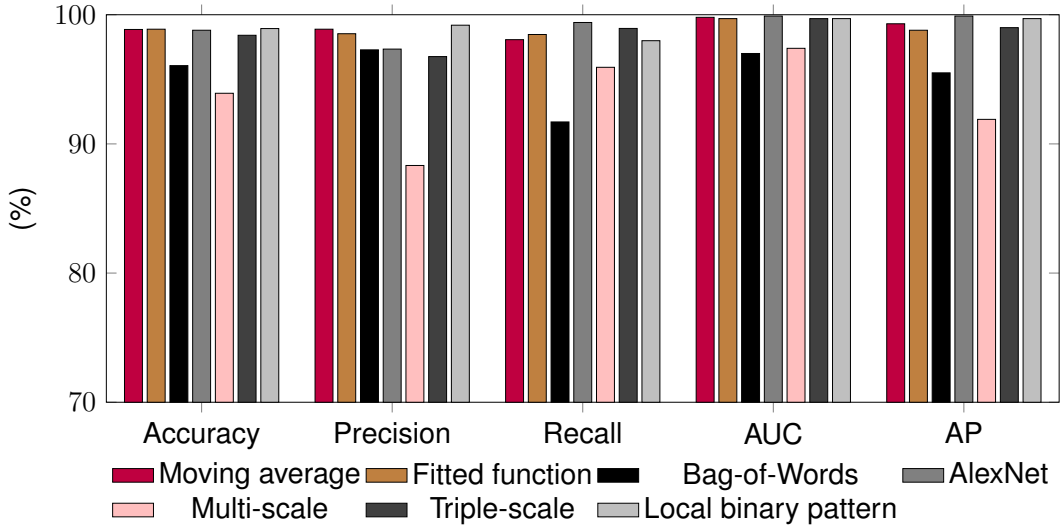


Figure 5.5: Results of all models with image morphology pre-processing

Figure 5.5 shows the results of all the models on the test set with the image morphology pre-processing as described in Section 4.3.2. The LBP, Alexnet, and Triple-scale network were mostly unaffected by this pre-processing method. However, the two line profile methods performed significantly better than without pre-processing again, however, not as much as with the basic mask method. The Bag-of-words algorithm improved more than with the basic mask, now scoring about 3-4% higher on every metric than without pre-processing. The only model that performed worse with image morphology, is the Multi-scale neural network where every metric decreased about 1-4%.

With this pre-processing step, the prediction and training time did not change significantly, as can be seen in Table A.2 in Appendix A. Only the prediction time for the Multi-scale network, which was already very slow, increased.

5.5 Adaptive threshold segmentation pre-processing

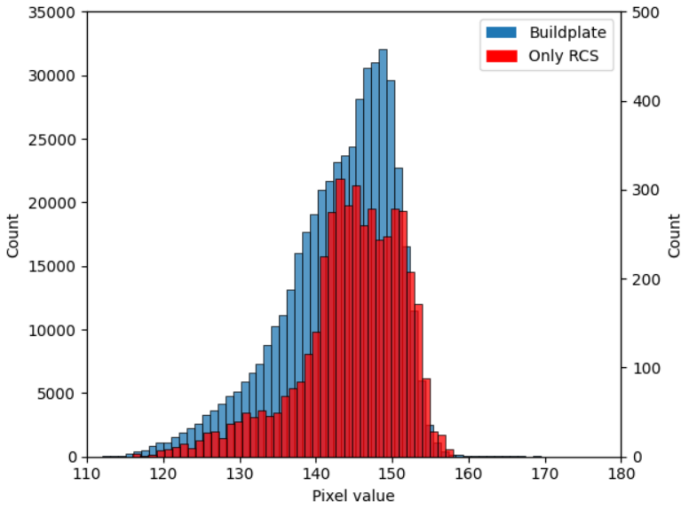


Figure 5.6: Comparison a histogram of the pixels of an entire buildplate (in blue) versus a histogram of only the RCS pixels of an image (in red).

Section 4.3.3 describes how the adaptive threshold segmentation algorithm exploits the difference in gray values between defects and normal powder beds. It separates the two by looking at the histogram of an image and filtering out all pixels within two thresholds. However, this model relies on uniformly lit images. Figure 5.6 shows two histograms, the first is computed over an entire image while the second one only looks at the pixels of RCS. As can be seen, the histogram of RCS pixel values is contained entirely by the entire image histogram. Therefore, these pixels will be filtered out based on the calculated thresholds with the bigger histogram. This pre-processing algorithm will consequently not be regarded in the rest of this study

5.6 Performance evaluation

5.6.1 Quantitative metrics

Figures 5.7 and 5.8 show a comparison of all models together with their best performing pre-processing step, again more details can be found in Appendix A. Different models improve the most from different pre-processing methods, or if already doing some sort of noise reduction or enhancement, from none. The two line profile models improved the most from the basic mask method while the Bag-of-Words gained the most from the image morphology step. As can be seen in Figure 5.7, all models score >96% accuracy and >90% in both recall and precision, with a differ-

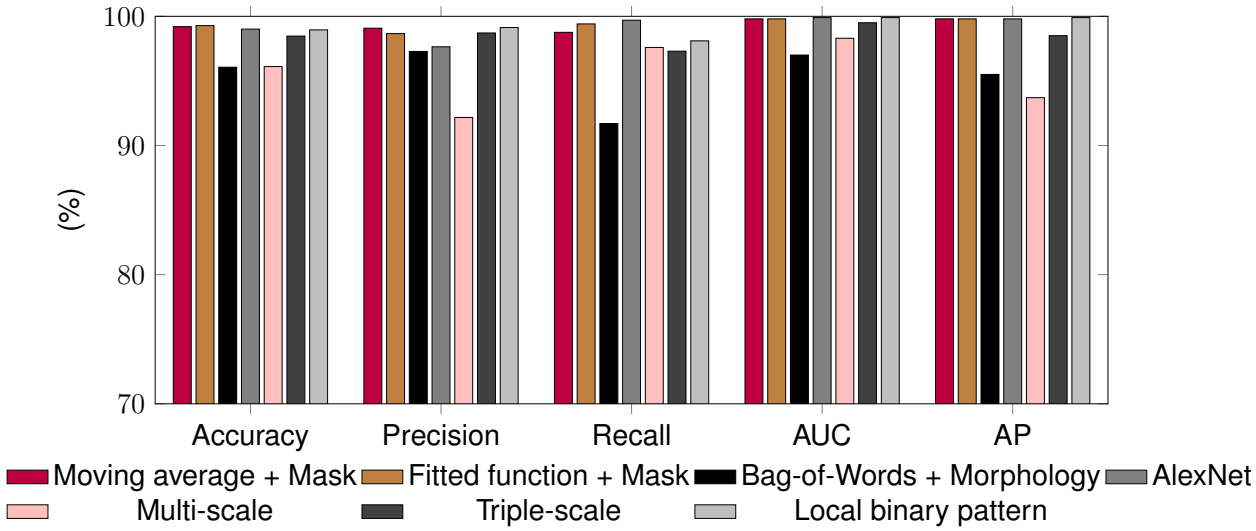


Figure 5.7: Comparison of all models with their best performing pre-processing algorithm.

ent model scoring highest for each metric. Furthermore, all models score >0.930 in AUC and AP.

The measurements where the models differ most, are the training and prediction speeds, Figure 5.8. The three neural network models take over an hour to train while the other models take around 5-6 minutes, mostly due to training for many epochs. Next to that, the two algorithms that need to divide an image into very small patches and classify each patch, take over six seconds to predict a single image while the other models all take below 100ms.

Figure 5.9 shows a heatmap with the predictions per layer for a singel print for each algorithm. Each algorithm uses the best pre-processing method as indicated above. For this print, the RCS starts at layer 104. Line profiles and Local Binary Patterns start identifying RCS right when it starts, with just a couple of false positives or negatives surrounding it. The Triple-scale network commences later, around layer 109, but also with just a couple of false positives or negatives surrounding it. While the AlexNet and Bag-of-Words algorithms observe quite some false positives before the RCS occurs and, just like the Triple-scale network, notice the real RCS only at layer 109, they quite steadily detect RCS after.

The only method that has real trouble is the Multi-scale network. As can be seen, it classified everything but one as RCS, meaning that it is very sensitive for RCS. Coherently, this model also has the lowest precision of all models as seen in Figure 5.7.

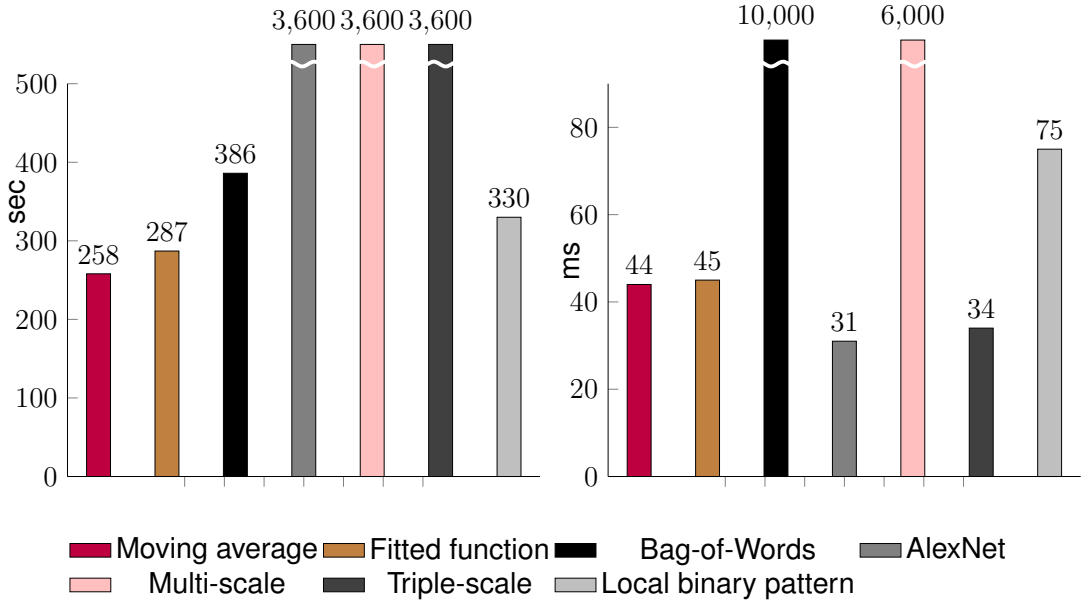


Figure 5.8: (a) Training speed in seconds and (b) the prediction speed in milliseconds of the algorithms with their best performing pre-processing algorithm.

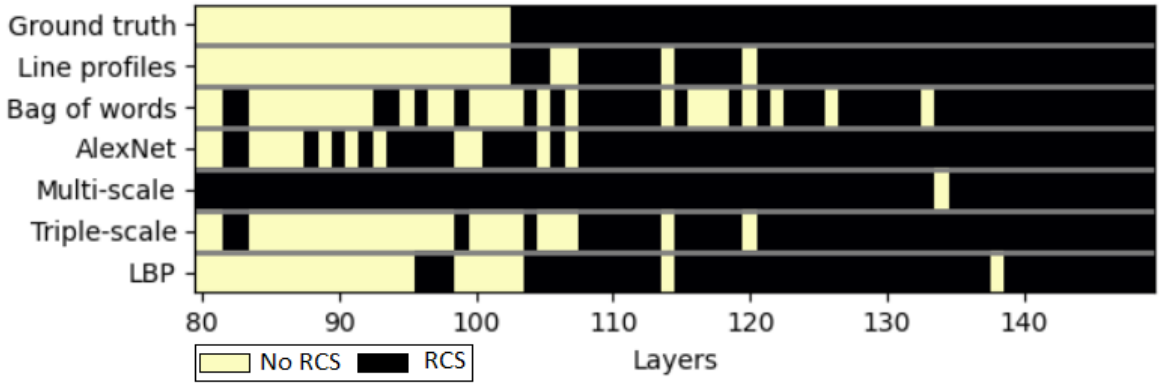


Figure 5.9: Heatmap showing the predictions per layer for each method, matched to the labels of the print. RCS starts at layer 104

5.6.2 Qualitative metrics

Line profiles	BoW	Alexnet	Multi-scale	Triple-scale	LBP
✗	✓	✓	✓	✓	✓

Table 5.2: Capability of detecting multiple anomalies simultaneously of all algorithms.

Multiple type of anomalies While RCS is the most occurring powder-bed defect [11], multiple different distortions can occur during the coating of a new layer. Some methods can detect all these different anomalies, while others cannot. E.g., the line profile methods were designed specifically for RCS, and the extracted features can only be used to detect RCS. Table 5.2 shows which algorithms can detect multiple anomalies simultaneously and which cannot.

While the dataset used in this study does not contain any anomalies next to RCS, the original papers, or literature, described this property for the other methods. These methods can, and have been to an extent evaluated on, detecting multiple different anomalies at the same time. Local Binary Patterns are a powerful tool for texture classification [67] and can detect multiple different textures. In the original paper, Local Binary Patterns were used to detect recoater hopping [17]. The performance on the other anomalies still has to be evaluated. In the original papers by Scime et al., both models were evaluated on all six different anomalies, and both models succeeded in detecting all anomalies [3], [14]. Yadav et al. originally also evaluated their method on all six anomalies and succeeded in detecting them all accurately [16]. Shi et al. applied the AlexNet on only three defects [13]. However, the AlexNet was originally designed for a dataset containing 1000 different labels [61] and should therefore be able to also work for all six anomalies.

Line profiles	BoW	Alexnet	Multi-scale	Triple-scale	LBP
✓	✓	✗	✓	✓	✗

Table 5.3: Capability of adjusting algorithm setting to adjust for print specific requirements.

Adjustability Additive manufacturing can be used for a wide degree of applications, ranging from rapid prototyping to the aeroplane industry [2], [3]. These different applications demand different requirements, aeroplanes need to be very precise while rapid prototyping needs to be rapid. Some models can be easily adapted to these print specific requirements while others would require major changes in their design. It would be possible for all algorithms to train a more or less sensitive version and to select the version suitable for the print specific requirements. However,

simply changing a setting before a print is much more practical than training and using a multitude of different sensitivities. Table 5.3 shows which algorithms can be easily adjusted before a print and which have to be re-trained.

Three models classify images based on the amount of patches containing defects, the multi-scale and the triple-scale neural networks, and the Bag-of-Words. The amount of defected patches before an image is classified as anomaly can be easily adjusted before a print to allow for print specific requirements. Likewise, the line profile models calculate the distortion in a powder bed image. The distortion threshold before an image is flagged can also be adjusted before a print.

The Alexnet was only designed for recognition [13]. To change its sensitivity before a print, a whole new model needs to be trained with different loss metrics. This takes hours and is not something to easily change before a new print. Local Binary Patterns are similar and can only be used for recognition. Both models lack the ability to alter some parameters in the trained model to alter between strictness and speed.

Chapter 6

Discussion

6.1 Label noise

As described in Section 4.5, the fitted function line profile method is used to label the extracted image patches used for training the patch-based models. This model scored an accuracy of 99.28%. As this is not perfect, some of the image patches will be mislabelled. Therefore, the models using these patches for training will be limited by the performance of the fitted function line profile method as they need to train with imperfect data. The reported metrics are not affected however, as these models are used to classify whole images just like all the other models. Thus, while the comparison with the other models is viable, the patch-based models could increase in performance with proper labelling of the training data.

6.2 Multiple type of anomalies

In this study, only recoater streaking is considered, as it is the most occurring defect regarding the powder bed [11]. The performance of the models of this study are tested when detecting only RCS also. Some models can only detect RCS while others can detect multiple anomalies simultaneously. When tested for this property in their original papers, the performance metrics for detecting RCS specifically changed quite drastically when more anomalies were added [3], [14], [16]. While the overall performance remains similar, a significant decrease is seen when only regarding the RCS metrics for these models. According to Scime et al. this is due to its small representation in the training data, the small area recoater streaking occupies on a powder layer, and its frequent co-location with an easier to classify anomaly type [3]. Thus, while detection of multiple anomalies simultaneously is possible, the performance for RCS can decrease compared to an exclusive detection.

6.3 Basic mask pre-processing

The basic mask pre-processing method compares an image with a defect-free image and highlights the differences [17]. This pre-processing method can significantly increase the performance of a detection model. Selecting the appropriate mask for the comparison is crucial for a good result. To ensure a proper comparison, the two images should be taken under the same lighting source. During this study all images were taken under the same lighting source and the same basic mask could be used for all prints. The lighting can change however, and when that happens, a different mask should be used. The lighting can change in between prints, but also during a print. Therefore, this pre-processing step is less robust as there is not one basic mask suitable for every situation. Thus, while the basic mask pre-processing method does increase the performance of certain methods, extra care should be taken into selecting the mask.

6.4 Limitations of the dataset

The dataset used in this research and described in Section 3 consists of 9.136 images taken over five different prints. While this is not too small of a dataset, it is not extensive enough to represent a wide range of characteristics of RCS and additive manufacturing. Furthermore, even though there are 3.378 images showing RCS in the dataset, they appear in only two prints. Therefore, while this is enough to construct a trustworthy comparison, all instances of RCS are quite similar and occur in the same locations. Furthermore, there will be some characteristics of RCS that are missing in the dataset. Thus, while the comparison made in this study is reliable, some detection algorithms can show artificially high metrics because all instances of RCS are quite similar. Furthermore, some edge cases are missing which could influence the final performance and lead to a better understanding of detecting RCS in a powder bed fusion process.

6.5 Bag-of-words

As mentioned in Section 4.4.2, not all filters described in the original paper for the Bag-of-Words method [3] were implemented. Thus, while this method was shown to accurately detect RCS in this study, the final performance can change when these filters are added. Not only in performance metrics but also in prediction time.

The prediction time measured in this study is quite slow, the slowest model with 10.58s per layer. When the last filters are added, this time will only increase. However, in the original paper, with all filters implemented, the model took 4s per

layer [14]. While the original paper shows a significant decrease from the time measured in this study, it is still quite slow. Thus, while there is prediction speed to be gained by an improved implementation or a better performing computer, it will not be enough to be competitive with the other models.

Chapter 7

Conclusion

The first research question as formulated in Section 1.2, inquiring which detection algorithm has the best overall performance when detecting recoater streaking, does not have a straightforward answer as multiple trade-offs have to be made. There are three models that outperform the others and they all score highest on a different metric. While the LBP algorithm has the highest precision, Alexnet scores the highest recall. At the same time, the line profile methods, in combination with the basic mask pre-processing, always performs better than one of these but never both, and have the highest accuracy of all methods. Therefore, a trade-off has to be made on a case by case basis which metric is deemed most important.

Furthermore, these three methods are all unsuccessful in one qualitative metric. E.g., the line profile methods usability diminishes when detecting multiple anomalies simultaneously has priority over high accuracy. Furthermore, the AlexNet and LBP algorithms are less effective when a printer is used for a variety of use cases and need to be adjusted for print specific requirements. In these cases, one of the other models becomes more practical. Of these models, the Triple-scale network outperforms the other two models on all metrics apart from the training speed. Therefore, again, a trade-off has to be made on a case by case basis whether qualitative or quantitative metrics are deemed most important.

To answer the second research question as formulated in Section 1.2, inquiring which detection algorithm has the best overall performance when tailored to the MetalFab1, the additional requirements as introduced in Section 4.2, stating that the prediction time for a single layer lies beneath one second, and ideally below 100ms, are taken into consideration. As shown in Figure 5.8, both the Bag-of-Words and the Multi-scale network take longer than one second to process a single layer, 10s and 6s respectively. The other methods all take less than 100ms to process a single layer. Therefore, the same trade-offs have to be made as to answer the first research question.

Chapter 8

Future Research

8.1 Extention to other materials and machines

This study is focused on the MetalFab1 printer printing titanium. However, the MetalFab1 can print a multitude of materials [21]. These various materials all have different specular reflection properties, thus the images taken for each material can differ significantly and the effectiveness of the different models from this study can alter drastically. Furthermore, there are a wide variety of additive manufacturing machines. All these machines behave differently and result in different images taken, again influencing the effectiveness of the models compared in this study. It would be interesting to see how the different detection- and pre-processing models react to images from different machines or different materials.

8.2 Adaptable basic mask

As described previously, the basic mask pre-processing method compares an image with a defect-free image with a matching lighting source. This pre-processing method can significantly increase the performance of a detection model. Selecting the mask is essential for a fitting comparison between the two images. One problem is the fact that the lighting source can change during the print, thus requiring a change in the basic mask as well. One way of solving this could be the implementation of an adaptable basic mask. This adaptable basic mask would change according to the changing lighting source to ensure an appropriate mask at all times. Future research would be interesting to look into the implementation of this adaptable basic mask as it could be beneficial for the use of this pre-processing method.

8.3 More models

All models compared in this study were designed for recoater streaking specifically, or have been used to detect RCS before. A lot more models exist in the field of image recognition that have yet to be tested on the RCS problem. For example the YOLO model series [71], which is the state of the art when it comes to image recognition and is used on an ever increasing range of problems [72]–[74]. While the models used in this study perform quite well, it would be interesting to see how these new models perform when detecting RCS.

8.4 Real time application

The comparison made in this study compares the performance of the different models over all images in the data set. From this comparison, conclusions can be drawn regarding which model performs best, based on which metrics are deemed most important. Translating this model into a real time application is non-trivial. Figure 5.9 showed that multiple models struggle with accurately identifying RCS around the root of RCS. Therefore, installing a real time application is not as trivial as pausing a print for every detected instance of RCS. It would be interesting to study the best set-up for a real time application using one of the models compared in this study.

8.5 Image Segmentation

Next to recognition of images containing RCS, detecting where RCS occurs can be of importance. Image segmentation algorithms already exist and perform accurately [71]. However, these algorithms have not yet been applied to RCS. The Bag-of-Words and Multi-scale algorithms were originally designed for just that [3], [14]. However, these models are too slow to be used in real time applications as they take too long to process a single image. Segmentation algorithms based on Alexnet or local binary patterns are currently being studied [75], [76] but have not yet been applied to RCS. Future research into image segmentation applied to RCS could be very interesting, especially in combination with detection of multiple types of anomalies.

8.6 Comparison multiple type of defects

As mentioned in Section 6, the performance of the detection models changes when they are utilized to detect multiple different anomalies simultaneously. Furthermore,

the performance of a model can vary significantly for each anomaly individually [3], [14], [16]. When other anomalies are of interest instead of RCS, or when multiple anomalies need to be detected, research has to be done in order to find out which model performs best. It would be interesting to study which model would suit which qualification best.

Bibliography

[1] G. Tapia and A. Elwany, "A review on process monitoring and control in metal-based additive manufacturing," *Journal of Manufacturing Science and Engineering, Transactions of the ASME*, vol. 136, 12 2014. [Online]. Available: https://asmedigitalcollection.asme.org/manufacturingscience/article/136/6/060801/377521/A-Review-on-Process-Monitoring-and-Control-in

[2] K. M. Taminger and R. A. Hafley, "Electron beam freeform fabrication for cost effective near-net shape manufacturing," in *NATO/RTO AVT-139 Specialists"Meeting on Cost Effective Manufacture via Net Shape Processing*, 2006.

[3] L. Scime and J. Beuth, "Anomaly detection and classification in a laser powder bed additive manufacturing process using a trained computer vision algorithm," *Additive Manufacturing*, vol. 19, pp. 114–126, 2018.

[4] K. Zeng, D. Pal, and B. Stucker, "A review of thermal analysis methods in laser sintering and selective laser melting," in *2012 International Solid Freeform Fabrication Symposium*. University of Texas at Austin, 2012.

[5] A. Industries. Additive industries presents metalfab1 process & application development tool. https://www.additiveindustries.com/news/news-and-press/additive-industries-presents-metalfab1-process-application-development-too.

[6] A. Mostafaei, C. Zhao, Y. He, S. R. Ghiaasiaan, B. Shi, S. Shao, N. Shamsaei, Z. Wu, N. Kouraytem, T. Sun, J. Pauza, J. V. Gordon, B. Webler, N. D. Parab, M. Asherloo, Q. Guo, L. Chen, and A. D. Rollett, "Defects and anomalies in powder bed fusion metal additive manufacturing," *Current Opinion in Solid State and Materials Science*, vol. 26, p. 100974, 4 2022.

[7] M. Grasso and B. M. Colosimo, "Process defects and in situ monitoring methods in metal powder bed fusion: a review," *Measurement Science and Technology*, vol. 28, p. 044005, 2 2017. [Online]. Available: https://iopscience.iop.org/article/10.1088/1361-6501/aa5c4fhttps://iopscience.iop.org/article/10.1088/1361-6501/aa5c4f/meta

[8] R. McCann, M. A. Obeidi, C. Hughes, Éanna McCarthy, D. S. Egan, R. K. Vijayaraghavan, A. M. Joshi, V. A. Garzon, D. P. Dowling, P. J. McNally, and D. Brabazon, "In-situ sensing, process monitoring and machine control in laser powder bed fusion: A review," *Additive Manufacturing*, vol. 45, p. 102058, 9 2021.

[9] I. Gibson, D. Rosen, B. Stucker, and M. Khorasani, "Powder bed fusion," *Additive Manufacturing Technologies*, pp. 125–170, 2021. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-56127-7_5

[10] M. Grasso, A. Remani, A. Dickins, B. M. Colosimo, and R. K. Leach, "In-situ measurement and monitoring methods for metal powder bed fusion: an updated review," *Measurement Science and Technology*, vol. 32, p. 112001, 7 2021. [Online]. Available: https://iopscience.iop.org/article/10.1088/1361-6501/ac0b6bhttps://iopscience.iop.org/article/10.1088/1361-6501/ac0b6b/meta

[11] P. Yadav, O. Rigo, C. Arvieu, V. K. Singh, and E. Lacoste, "Data processing techniques for in-situ monitoring in l-pbf process," *Journal of Manufacturing Processes*, vol. 81, pp. 155–165, 9 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1526612522004509

[12] T. Craeghs, S. Clijsters, E. Yasa, and J.-P. Kruth, "Online quality control of selective laser melting," in *2011 International Solid Freeform Fabrication Symposium*. University of Texas at Austin, 2011.

[13] B. Shi and Z. Chen, "A layer-wise multi-defect detection system for powder bed monitoring: Lighting strategy for imaging, adaptive segmentation and classification," *Materials & Design*, vol. 210, p. 110035, 2021.

[14] L. Scime and J. Beuth, "A multi-scale convolutional neural network for autonomous anomaly detection and classification in a laser powder bed fusion additive manufacturing process," *Additive Manufacturing*, vol. 24, pp. 273–286, 2018.

[15] L. Scime, D. Siddel, S. Baird, and V. Paquit, "Layer-wise anomaly detection and classification for powder bed additive manufacturing processes: A machine-agnostic algorithm for real-time pixel-wise semantic segmentation," *Additive Manufacturing*, vol. 36, p. 101453, 12 2020.

[16] P. Yadav, O. Rigo, C. Arvieu, E. L. Guen, and E. Lacoste, "Data treatment of in situ monitoring systems in selective laser melting machines," *Advanced Engineering Materials*, vol. 23, p. 2001327, 5 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/adem.202001327https:

//onlinelibrary.wiley.com/doi/abs/10.1002/adem.202001327https://onlinelibrary.wiley.com/doi/10.1002/adem.202001327

[17] Y. Yin, G. Dali *et al.*, "Research on feature extraction of local binary pattern of slm powder bed gray image," in *Journal of Physics: Conference Series*, vol. 1885, no. 3. IOP Publishing, 2021, p. 032007.

[18] M. Grasso, "In situ monitoring of powder bed fusion homogeneity in electron beam melting," *Materials*, vol. 14, no. 22, p. 7015, 2021.

[19] A. Neef, V. Seyda, D. Herzog, C. Emmelmann, M. Schönleber, and M. Kogel-Hollacher, "Low coherence interferometry in selective laser melting," *Physics Procedia*, vol. 56, pp. 82–89, 2014.

[20] Y. Fu, A. R. Downey, L. Yuan, T. Zhang, A. Pratt, and Y. Balogun, "Machine learning algorithms for defect detection in metal laser-based additive manufacturing: A review," *Journal of Manufacturing Processes*, vol. 75, pp. 693–710, 3 2022.

[21] A. Industries, "Metalfab continuous production," https://www.additiveindustries.com/metalfabg2-continuous-production.

[22] F. Y. Edgeworth, "Xli. on discordant observations," *The london, edinburgh, and dublin philosophical magazine and journal of science*, vol. 23, no. 143, pp. 364–375, 1887.

[23] V. Chandola, "Anomaly detection: A survey varun chandola, arindam banerjee, and vipin kumar," 2007.

[24] C. Wang, X. Tan, S. B. Tor, and C. Lim, "Machine learning in additive manufacturing: State-of-the-art and perspectives," *Additive Manufacturing*, vol. 36, p. 101538, 2020.

[25] L. Meng, B. McWilliams, W. Jarosinski, H.-Y. Park, Y.-G. Jung, J. Lee, and J. Zhang, "Machine learning in additive manufacturing: a review," *Jom*, vol. 72, pp. 2363–2377, 2020.

[26] A. Gaikwad, B. Giera, G. M. Guss, J.-B. Forien, M. J. Matthews, and P. Rao, "Heterogeneous sensing and scientific machine learning for quality assurance in laser powder bed fusion–a single-track study," *Additive Manufacturing*, vol. 36, p. 101659, 2020.

[27] W. Cui, Y. Zhang, X. Zhang, L. Li, and F. Liou, "Metal additive manufacturing parts inspection using convolutional neural network," *Applied Sciences*, vol. 10, no. 2, p. 545, 2020.

[28] J. L. Bartlett, A. Jarama, J. Jones, and X. Li, "Prediction of microstructural defects in additive manufacturing from powder bed quality using digital image correlation," *Materials Science and Engineering: A*, vol. 794, p. 140002, 2020.

[29] G. Tapia, A. H. Elwany, and H. Sang, "Prediction of porosity in metal-based additive manufacturing using spatial gaussian process models," *Additive Manufacturing*, vol. 12, pp. 282–290, 2016.

[30] M. Khanzadeh, S. Chowdhury, M. Marufuzzaman, M. A. Tschopp, and L. Bian, "Porosity prediction: Supervised-learning of thermal history for direct laser deposition," *Journal of manufacturing systems*, vol. 47, pp. 69–82, 2018.

[31] F. Imani, A. Gaikwad, M. Montazeri, P. Rao, H. Yang, and E. Reutzel, "Process mapping and in-process monitoring of porosity in laser powder bed fusion using layerwise optical imaging," *Journal of Manufacturing Science and Engineering*, vol. 140, no. 10, 2018.

[32] M. Montazeri, A. R. Nassar, A. J. Dunbar, and P. Rao, "In-process monitoring of porosity in additive manufacturing using optical emission spectroscopy," *IISE Transactions*, vol. 52, no. 5, pp. 500–515, 2020.

[33] D. S. Ye, Y. Fuh, Y. Zhang, G. Hong, and K. P. Zhu, "Defects recognition in selective laser melting with acoustic signals by svm based on feature reduction," in *IOP Conference Series: Materials Science and Engineering*, vol. 436, no. 1. IOP Publishing, 2018, p. 012020.

[34] Y. Zhang, H. G. Soon, D. Ye, J. Y. H. Fuh, and K. Zhu, "Powder-bed fusion process monitoring by machine vision with hybrid convolutional neural networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 5769–5779, 2019.

[35] B. Zhang, P. Jaiswal, R. Rai, P. Guerrier, and G. Baggs, "Convolutional neural network-based inspection of metal additive manufacturing parts," *Rapid Prototyping Journal*, 2019.

[36] T. Huang, "Computer vision: Evolution and promise," 1996.

[37] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 6 2014. [Online]. Available: https://doi.org/10.7717/peerj.453

[38] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.

[39] W. Sun, B. Yao, B. Chen, Y. He, X. Cao, T. Zhou, and H. Liu, "Noncontact surface roughness estimation using 2d complex wavelet enhanced resnet for intelligent evaluation of milled metal surface quality," *Applied Sciences*, vol. 8, no. 3, p. 381, 2018.

[40] V. Koblar, M. Pecar, K. Gantar, T. Tusar, and B. Filipic, "Determining surface roughness of semifinished products using computer vision and machine learning," in *Proceedings of the 18th International Multiconference Information Society, IS*, 2015, pp. 51–54.

[41] Scikit. Gabor filter banks for texture classification. https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_gabor.html#sphx-glr-auto-examples-features-detection-plot-gabor-py.

[42] S.-C. Wang, "Artificial neural network," in *Interdisciplinary computing in java programming*. Springer, 2003, pp. 81–100.

[43] Q. Ke, J. Liu, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, "Computer vision for human–machine interaction," in *Computer Vision for Assistive Healthcare*. Elsevier, 2018, pp. 127–145.

[44] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[45] M. Sun, Z. Song, X. Jiang, J. Pan, and Y. Pang, "Learning pooling for convolutional neural network," *Neurocomputing*, vol. 224, pp. 96–104, 2017.

[46] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[47] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[48] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[49] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[50] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *arXiv preprint arXiv:2010.16061*, 2020.

[51] T. Fawcett, “An introduction to roc analysis,” *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[52] M. Zhu, “Recall, precision and average precision,” *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, vol. 2, no. 30, p. 6, 2004.

[53] R. M. Haralick, S. R. Sternberg, and X. Zhuang, “Image analysis using mathematical morphology,” *IEEE transactions on pattern analysis and machine intelligence*, no. 4, pp. 532–550, 1987.

[54] T. Huang, G. Yang, and G. Tang, “A fast two-dimensional median filtering algorithm,” *IEEE transactions on acoustics, speech, and signal processing*, vol. 27, no. 1, pp. 13–18, 1979.

[55] C. Poynton, *Digital video and HD: Algorithms and Interfaces*. Elsevier, 2012.

[56] J. Serra, “Image analysis and mathematical morphol-ogy,” *(No Title)*, 1982.

[57] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020.

[58] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, “Visual categorization with bags of keypoints,” in *Workshop on statistical learning in computer vision, ECCV*, vol. 1, no. 1-22. Prague, 2004, pp. 1–2.

[59] S. Ramaswamy, R. Rastogi, and K. Shim, “Efficient algorithms for mining outliers from large data sets,” in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 427–438.

[60] X. Han, Y. Zhong, L. Cao, and L. Zhang, “Pre-trained alexnet architecture with pyramid pooling and supervision for high spatial resolution remote sensing image scene classification,” *Remote Sensing*, vol. 9, no. 8, p. 848, 2017.

[61] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[62] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[63] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[64] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[65] G.-W. Yuan, Y. Gao, D. Xu, and M.-R. Jiang, "A new background subtraction method using texture and color information," in *International Conference on Intelligent Computing*. Springer, 2011, pp. 541–548.

[66] C. Silva, T. Bouwmans, and C. Frélicot, "An extended center-symmetric local binary pattern for background modeling and subtraction in videos," *VISAPP 2015 - 10th International Conference on Computer Vision Theory and Applications; VISIGRAPP, Proceedings*, vol. 1, pp. 395–402, 3 2015. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01227955https://hal.archives-ouvertes.fr/hal-01227955/document

[67] T. Bouwmans, C. Silva, C. Marghes, M. S. Zitouni, H. Bhaskar, and C. Frelicot, "On the role and the importance of features for background modeling and foreground detection," *Computer Science Review*, vol. 28, pp. 26–91, 5 2018.

[68] M. Claesen and B. De Moor, "Hyperparameter search in machine learning," *arXiv preprint arXiv:1502.02127*, 2015.

[69] G. Claeskens, N. L. Hjort *et al.*, "Model selection and model averaging," *Cambridge Books*, 2008.

[70] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[71] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[72] R. Huang, J. Pedoeem, and C. Chen, "Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 2503–2510.

[73] Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, and Z. Liang, "Apple detection during different growth stages in orchards using the improved yolo-v3 model," *Computers and electronics in agriculture*, vol. 157, pp. 417–426, 2019.

[74] W. Lan, J. Dang, Y. Wang, and S. Wang, "Pedestrian detection based on yolo network model," in *2018 IEEE international conference on mechatronics and automation (ICMA)*. IEEE, 2018, pp. 1547–1551.

[75] P. M. Pereira, R. Fonseca-Pinto, R. P. Paiva, P. A. Assuncao, L. M. Tavora, L. A. Thomaz, and S. M. Faria, "Dermoscopic skin lesion image segmentation based on local binary pattern clustering: Comparative study," *Biomedical Signal Processing and Control*, vol. 59, p. 101924, 2020.

[76] J. Chen, Z. Wan, J. Zhang, W. Li, Y. Chen, Y. Li, and Y. Duan, "Medical image segmentation and reconstruction of prostate tumor based on 3d alexnet," *Computer methods and programs in biomedicine*, vol. 200, p. 105878, 2021.

Appendix A

Comparison results

Method	Acc.	Precision	Recall	AUC	AP	Train speed	Pred. speed
Moving Average	93.24%	97.83%	83.66%	0.966	0.950	**25s**	**4.55ms**
Fitted function	92.99%	97.83%	84.13%	0.966	0.922	29s	5.22ms
LBP	98.95%	**99.13%**	98.10%	**0.999**	**0.999**	330s	75ms
Bag of Words	93.54%	94.60%	87.18%	0.950	0.931	516s	10.47s
Multi-scale	96.11%	92.17%	97.59%	0.983	0.937	>1 hour	6.7s
Triple-scale	98.47%	98.71%	97.30%	0.995	0.985	>1 hour	34.84ms
AlexNet	**99.01%**	97.64%	**99.70%**	**0.999**	**0.999**	>1 hour	31.34ms

Table A.1: Results of all models without any pre-processing

Method	Acc.	Precision	Recall	AUC	AP	Train speed	Pred. speed
Moving Average	98.86%	98.88%	98.07%	0.998	0.993	**30s**	**5.22ms**
Fitted function	98.88%	98.53%	98.47%	0.997	0.988	36s	6.63ms
LBP	**98.93%**	**99.19%**	97.99%	0.997	0.997	340s	77ms
Bag of Words	96.06%	97.28%	91.70%	0.970	0.955	386s	10.58s
Multi-scale	93.92%	88.33%	95.93%	0.974	0.919	>1 hour	20s
Triple-scale	98.41%	96.76%	98.94%	0.997	0.990	>1 hour	38.76ms
AlexNet	98.80%	97.34%	**99.40%**	**0.999**	**0.999**	>1 hour	30.22ms

Table A.2: Results of all models with image morphology pre-processing

Method	Acc.	Precision	Recall	AUC	AP	Train speed	Pred. speed
Moving Average	99.20%	**99.08%**	98.76%	0.998	0.998	**258s**	44.16ms
Fitted function	**99.28%**	98.66%	**99.41%**	0.998	0.998	287s	45.74ms
LBP	98.93%	98.60%	98.54%	**0.999**	**0.999**	831s	167ms
Bag of Words	94.69%	96.09%	88.99%	0.972	0.953	1847s	8.40s
Multi-scale	88.51%	94.32%	72.70%	0.931	0.900	>1 hour	10.38s
Triple-scale	96.39%	95.16%	94.87%	0.992	0.987	>1 hour	48.07ms
AlexNet	98.85%	98.49%	98.34%	0.998	0.998	>1 hour	**30.58ms**

Table A.3: Results of all models with basic mask pre-processing

Method	Pre	Acc.	Precision	Recall	AUC	AP	Training	Prediction
Line profiles	Mask	**99.28%**	98.66%	99.41%	0.998	0.998	**287s**	45.74ms
LBP	None	98.95%	**99.13%**	98.10%	**0.999**	**0.999**	330s	75ms
Bag of Words	Morph.	96.06%	97.28%	91.70%	0.970	0.955	386s	10.58s
Multi-scale	None	96.11%	92.17%	97.59%	0.983	0.937	>1 hour	6.7s
Triple-scale	None	98.47%	98.71%	97.30%	0.995	0.985	>1 hour	34.84ms
AlexNet	None	99.01%	97.64%	**99.70%**	**0.999**	0.998	>1 hour	**31.34ms**

Table A.4: Comparison of all models on the MetalFab1 dataset

Appendix B

Algorithms

B.1 Pre-processing algorithms

Algorithm 1 Basic mask

function BASIC MASK(image)
processed $\leftarrow$ image - mask
threshold$_1$ $\leftarrow$ 100
threshold$_2$ $\leftarrow$ 255
for all pixel $\in$ processed **do**
if pixel $\leq$ threshold$_1$ *or* pixel $\geq$ threshold$_2$ **then**
pixel $\leftarrow$ 0
end if
end for
processed $\leftarrow$ $MedianFilter$(processed, $filterSize = (3, 3)$)
processed $\leftarrow$ $GammaTransformation$(processed, $gamma = 2$)
Return processed
end function

Algorithm 2 Image morphology

function IMAGE MORPHOLOGY(image)
structure $\leftarrow$ $Rectangle(width = 30, height = 1)$
image $\leftarrow$ $Erode$(image, structure)
image $\leftarrow$ $Dilate$(image, structure)
Return image
end function

Algorithm 3 Adaptive thresholds

function ADAPTIVE THRESHOLD(image)
  histogram $\leftarrow ConstructHistogram$(image)
  threshold$_1$ $\leftarrow Min(LocalExtrema$(histogram))
  threshold$_2$ $\leftarrow Max(LocalExtrema$(histogram))
  for all pixel $\in$ image **do**
    if threshold$_1$ $\leq$ pixel $\leq$ threshold$_2$ **then**
      pixel $\leftarrow$ 0
    end if
  end for
  image $\leftarrow Close(Open(Dilation$(image, $Structure_1), Structure_2), Structure_3)$
  for all pixelArea $\in$ image **do**
    if pixelArea.$Surface()$ < AreaThreshold **then**
      pixelArea.$Delete()$
    end if
    if pixelArea.$DistanceToOthers()$ < DistanceThreshold **then**
      pixelArea.$Merge()$
    end if
  end for
  Return image
end function

B.2 Detection algorithms

Algorithm 4 Line profiles

function PREDICT(image)
  LineProfiles ← {}
  for all row ∈ image **do**
    LineProfiles.$append$(row.$mean$())
  end for
  MovingAverage ← $CalculateMovingAverage$(LineProfiles)
  Distortion ← LineProfiles - MovingAverage
  Return max(abs(Distortion)) ≥ threshold
end function

Algorithm 5 Local binary pattern

function PREDICT(image)
  LBPs ← {}
  for all pixel ∈ image **do**
    LBPs.$append$($GetLocalBinaryPattern$(pixel))
  end for
  histogram ← $ConstructHistogram$(LBPs)
  Return SVM.$Predict$(LBPs)
end function

Algorithm 6 Bag-of-Words

function FIT(images, labels)
features ← $ApplyFilterBank$(images)
clusters ← KMeans.$cluster$(features.$pixels()$)
clusters ← clusters.$mean$()
Bag-of-Words ← KNN.fit(clusters)
features ← $Partition$(features, $size = (20, 20)$)
fingerprints ← $Fingerprint$(features)
fingerprintDictionary ← KNN.fit(fingerprints, labels)
end function

function FINGERPRINT(image_features)
fingerprint ← $[0] \cdot AmountOfClusters$
for all pixel ∈ image_feature **do**
closestCluster ← Bag-of-Words.$Neighbour$(pixel, $neighbours = 1$)
fingerprint[closestCluster] += 1
end for
fingerprint.$Normalize$()
return fingerprint
end function

function PREDICT(image)
features ← $ApplyFilterBank$(images)
features ← $Partition$(features, $size = (20, 20)$)
fingerprints ← $Fingerprint$(features)
predictions ← 0
for all fingerprint ∈ fingerprints **do**
patchPrediction ← fingerprintDictionary.$Neighbour$(fingerprint, $neighbours = 3$)
if patchPrediction.$Contain$(False) **then**
predictions ← predictions
else
predictions ← predictions + 1
end if
end for
return predictions ≥ threshold
end function

Algorithm 7 AlexNet

function FIT(images)
images ← $ExtendDataset$(images, $transformations$ =(mirror, flip, brightness, randomCrop))
Alexnet.Fit(images)
end function

function PREDICT(image)
return Alexnet.$predict$(image)
end function

Algorithm 8 Multi-scale network

function FIT(images)
patches ← $GeneratePatches$(images)
patches ← $ExtendDataset$(patches, $transformations$ =(mirror, flip, brightness, randomCrop))
Alexnet.Fit(patches)
end function

function PREDICT(image)
patches ← $GeneratePatches$(images)
predictions ← Alexnet.$predict$(patches)
return predictions $\geq$ threshold
end function

Algorithm 9 Triple-scale network

function FIT(images)
patches ← $GeneratePatches$(images)
patches ← $ExtendDataset$(patches, $transformations$ =(mirror, flip, brightness, randomCrop))
CNN.Fit(patches)
end function

function PREDICT(image)
patches ← $GeneratePatches$(images)
predictions ← CNN.$predict$(patches)
return predictions $\geq$ threshold
end function