PRIFYSGOL
BANGOR
UNIVERSITY

School of Computer Science and Electronic Engineering
College of Environmental Sciences and Engineering

Using Machine learning and Computer vision approaches in automatic data analysis for advanced manufacturing.

Md Mahabubur Rahman

Submitted in partial satisfaction of the requirements for the Degree of Masters of Engineering

in Advanced Data Science

Supervisor Dr. Abdullah Al Mamun

August 2023

Acknowledgements

I want to thank my supervisor, Dr Abdullah Al Mamun, for his patient help and constructive suggestions on this thesis. I also want to thank all the teachers who taught me research skills and academic writing. Finally, I thank DONNA Ltd for supplying the samples and the Nuclear Future Institute for the lab access.

Statement of Originality

The work presented in this thesis/dissertation is entirely from the studies of the individual student, except where otherwise stated. Where derivations are presented and the origin of the work is either wholly or in part from other sources, then full reference is given to the original author. This work has not been presented previously for any degree, nor is it at present under consideration by any other degree awarding body.

Student

Md Mahabubur Rahman

Statement of Availability

I hereby acknowledge the availability of any part of this thesis/dissertation for viewing, photocopying or incorporation into future studies, providing that full reference is given to the origins of any information contained herein. I further give permission for a copy of this work to be deposited with the Bangor University Institutional Digital Repository, and/or in any other repository authorised for use by Bangor University and where necessary have gained the required permissions for the use of third party material. I acknowledge that Bangor University may make the title and a summary of this thesis/dissertation freely available.

Student:

Md Mahabubur Rahman

Abstract

Additive manufacturing (AM) of metals is quickly becoming a popular way to make industrial parts. Making mistakes during the AM process hurts the service performance of the details, so analyzing the errors is essential. However, counting and categorizing these problems by hand is often challenging. The YOLOv8 Mask RCNN, Retina Net, and Fast RCNN machine learning algorithms were used in this project to simplify the process of analyzing and classifying defects. Samples were made by the project partner DONNA Ltd. using the Selective Laser Melting (SLM) and Directed Energy Deposition (DED) AM methods. Using a Scanning Electron Microscope (SEM), the microstructures of the samples were looked at, and data sets with 60 pictures were made. The project centred on three flaws: unmelted particles, porosity, and micro-cracks. The best models of YOLOv8 and Mask RCNN can accurately recognize the three classes of porosity, unmelted particles, and microcracks, but they need more training, not less. For example, mask RCNN, Retina net, and fast RCNN can't recognize microcracks until they've been trained for a long time. So this project was successful in automatic data analysis for advanced manufacturing. Introduced the instance segmentation method and used the flexibility of the instance segmentation annotation to solve problems with the object recognition method. In the future, try to find a link between automatically found defects and machine settings to improve the AM process.

Key Words: Additive Manufacturing, Machine Learning, Object detection, Instance segmentation.

Contents

1 Introduction 8

1.1 Aim and Objectives 9

1.2 Summary and Dissertation..... 10

2 Background and Related Work 11

2.1 AM Defects Detection 11

2.1 Object Detection..... 12

2.2 Instance Segmentation..... 14

3 Methodology 15

3.1 Overview of Detectron 2 15

3.2 Overview of the Mask R-CNN 16

3.3 Overview of Faster R-CNN……...17

3.4 Overview of Retina Net……..18

3.5 Overview of Yolo v8……...19

3.6 Experiment 20

3.6.1 Image Collection..... 20

3.6.2 Object Detection..... 21

3.6.3 Instance Segmentation 22

3.6.4 Analysis Images……...23

4 Evaluation 24

4.1 Data set 24

4.2 Object Detection..... 25

4.2.1 Faster R-CNN……...25

4.2.2 Retina Net……....28

4.2.3 Yolo V8……..31

4.3 Instance segmentation 36

4.4 Analysis Images 38

4.5 Discussion of Implication……41

5 Conclusion 42

5.1 Summary and Conclusions 42

5.2 Review and Objectives..... 42

5.3 Future work……42

References 44

List of Figures

3.1 The Mask R-CNN system for segmentation [26] 15
3.2 The Mask R-CNN system for segmentation [29] 16
3.3 The Faster R-CNN System for Detection [30] 17
3.4 Retina Net system for Detection [31]. 18
3.5 Yolo v8 system for Detection [33]. 19
3.6 Yolo v8 system for Detection [32] 19
3.7 The SEM for micrographs collection 20
3.8 A collected micrograph 20
3.9 AM sample supplied by DONNA 21

4.1 The annotation data set constructed by Labelme 24
4.2 Training process fast_rcnn after 2000 epochs 25
4.3 Fast_rcnn object detection after 2000 epochs. 26
4.4 Training process fast_rcnn after 10000 epochs 27
4.5 Training process fast_rcnn after 10000 epochs object detection 27
4.6 Training process retina net after 2000 epochs 28
4.7 Retina net object detection after 2000 epochs 29
4.8 Training process Retina net after 10000 epochs 30
4.9 Retina net object detection after 10000 epochs 30
4.10 Yolo v8 object detection graph after 2000 epochs 32
4.11 Yolo v8 object detection performance after 2000 epochs- 33
4.12 Yolo v8 object detection graph after 10000 epochs. 34
4.13 Yolo v8 object detection performance after 10000 epochs. 35
4.14 Training process of Mask R-CNN for 2000 epochs 36
4.15 Training process of Mask R-CNN for 10000 epochs 37
4.16 Detection and Annotation Images 39
4.17 Output Results of Every Image 40

List of Table

4.1 Evalutaion Matrix for Fast_rcnn.....26
4.2 Evaluation box for coco format for Fast_rcnn.....26
4.3 Evaluation Matrix for Fast_rcnn for 10000 epochs 28
4.4 Evaluation Matrix for Fast_rcnn for 10000 epochs. 28
4.5 Evaluation Matrix for Retina net for 2000 epochs.....29
4.6 Evaluation box for coco format for Retina net.....29
4.7 Evaluation Matrix for Retina Net for 10000 epochs.....31
4.8 Evaluation box for coco format for Retina net for 10000 epochs.....31
4.9 Evaluation box for coco format Yolov8 for 2000 epochs.....32
4.10 Evaluation box for coco format Yolov8 for 10000 epochs.....34
4.11 Performance Analysis Both Epochs.....35
4.12 Evaluation Matrix for Mask_RCNN for 2000 epochs.....36
4.13 Evaluation box for coco format Mask Rcnn for 2000 epochs.....37
4.14 Evaluation box for coco format Mask Rcnn for 2000 epochs.....37
4.15 Evaluation box for coco format Mask Rcnn for 10000 epochs.....38
4.16 Evaluation box for coco format Mask Rcnn for 10000 epochs.....38

Chapter 1

Introduction

Additive manufacturing (3D printing) is gaining popularity as an innovative new technique in many industries, including aerospace, biomedical, and energy sectors. In additive manufacturing, layers are added to a material rather than subtracted. The AM method can make both complicated forms and highly rigid components. AM can improve production speed, automation, resource efficiency, and weight reduction [1]. Manufacturing defects occur in materials during AM, and it is difficult to analyze those defects manually, so an automatic defect detection system is highly desirable because Using data science makes it possible to find minor flaws that a manual review might miss. Machine learning algorithms possess the capability to acquire the ability to identify various forms of anomalies by discerning patterns and deviations within annotated data. This makes it possible for the system to find problems very accurately, which improves quality control in AM. During the manufacturing process, various factors can contribute to the occurrence of defects, including the intensity of the heat source, scanning velocity, spacing between hatches, layer thickness, powder materials, and environmental conditions within the chamber [2]. The quality grade could be affected by flaws, a technical barrier to AM. In this case, all parts should be carefully checked while they are being made to ensure they are of good quality. Also, machine settings need to be optimized. The common flaws in AM parts are porosity, pit, scratch, un-melted particles, and micro-crack. In this project, we worked on micro-cracks, porosity, and unmelted particles to segment, detect and analyze the images.

For traditional AM defect analysis, experts must do the study by hand. But the accuracy depends on the skills and knowledge of the operator, how tired they are, and how the environment is. In other words, the accuracy varies, and it's easy for things outside of the system to change. At the same time, the cost of human resources is very high, and much time is wasted. So, adding machine learning (ML) methods for automating defect detection in the AM area has become a big deal in solving the problems of low detection efficiency, inconsistent results, and missed inspections. Deep learning has made it possible for algorithms to process and classify images more accurately than people.

On the other hand, feature learning is a complicated process that takes thousands of epochs of training. At the same time, we need to gather enough information to make an excellent data set

because there aren't many similar data sets on the Internet. To improve the quality of our project model, getting as big a sample size as possible is essential. At the same time, if we want to improve the test result, we must find a way to train a data set with a small number of samples.

First, I used a method called "image segmentation." Object detection is a more advanced method than image classification. It has more complicated functions that can accurately find and classify multiple objects in an image. This gives a deeper understanding of the image and supports a broader range of practical applications, like face recognition and automatic driving. The objective of the object detection method is to accurately enclose each instance of a defect within a bounding box in the given image [3]. Conventional approaches to object recognition rely on manually designed features and superficial architectures that are amenable to training. However, these systems often reach a performance limit beyond which further improvements become challenging. As deep learning grows quickly, more powerful tools are available to fix problems in traditional designs. Different network designs, training strategies, and optimization functions make these models act differently.

The utilisation of object recognition facilitated the achievement of instance segmentation. The algorithm identifies several categories of object instances inside additional photos. The objective of this task is to accurately predict the class label of an object and generate a pixel-level mask that corresponds to the individual instance of that object. The proliferation of deep learning has led to a significant increase in the field of instance classification in recent years. The task of instance segmentation is inherently more intricate since it necessitates the precise identification and segmentation of all objects present within an image. So, it mixes parts of the standard computer vision tasks of finding objects and figuring out what they mean. Also, it gives more options than object recognition.

We need to add picture data to the ML model to train it. Using a scanning electron microscope (SEM), high-quality pictures of the microstructure must be taken to find and categorize defects in the samples more accurately. The quality of a photograph is affected by more than just how much it has changed. For instance, rust and scratches on the surface can be mistaken for problems with how it was made. So, the character of parts needs to be carefully polished to improve the quality of pictures of the surface taken with a microscope.

Aims and Objectives:

- The goal of the project is to use machine learning (ML) techniques such as object recognition and instance segmentation to make it easier to find and classify defects in additive manufacturing (AM) processes.
- The goal is to look at images of microstructures to improve process parameters and possibly set up real-time monitoring of the production process.
- It stands out by comparing four different machine learning (ML) models: Mask R-CNN, Retina Net, Faster RCNN, and YOLOv8. These models are used to analyse short-term defects in images quickly and efficiently.

.

It also introduces the instance segmentation method and uses the flexibility of instance segmentation annotation to solve problems with the object detection method that make it hard to find the location of objects of many scattered points. The second chapter describes the relevant literature. The third chapter describes the algorithm models and experimental methods. The fourth chapter presents the test results and how well the model works. The last chapter describes the learnings from the project and future work.

Chapter 2

Background and Related Work

Computer vision encompasses four primary tasks, namely image classification, object detection, semantic segmentation, and instance segmentation. Image classification is a machine learning method that can put different kinds of images into other groups. Still, here this project analyzes images, so image classification is not used. In advanced manufacturing, semantic segmentation is another robust method that can be used. It includes dividing an image into meaningful, separate areas to determine where different things are in space. Semantic segmentation can be used to find and separate specific areas of interest, such as defects, parts, or structural traits, but not separate particular areas here. This project aims to detect and identify numerous targets within a single image. To this end, we will benefit most from object detection and instance segmentation for analysis images.

2.1 **Additive Manufacturing Defect Detection**

Many works have been done using Machine learning and deep learning in Additive manufacturing defect analysis. Dey Satyajit et al. [4] used UNet to find fusion flaws like micro-cracks, spherical pores, and particles that hadn't melted (unmelted particles). They used the CNN(Convolutional Neural Network) based architecture to detect their analysis. Zhang et al. [5] In laser additive manufacturing, it was suggested that a CNN model could be used to predict porosity. The CNN model employed in this study exhibits a compact architecture. Despite utilising the melt pool image derived from a singular track composed of sponge titanium powder, the framework effectively identifies porosity with an accuracy of 91.2%. Furthermore, a cross-validation analysis is conducted to choose specific hyperparameters. It can predict holes that are 100 m or more minor. Choi et al. [6] used the YOLOv3 model to find defect porosity, microcrack, and lack of fusion. It has been shown that defect identification works better with more training epochs, longer training times, and more extensive data sets. T.Herzog et al. [7] observed many signal processing using In-situ monitoring such as thermal emission, visible light, Acoustic waves, and Light spectra for supervised, unsupervised and semi-supervised algorithms. However, the Machine learning Category for supervised learning accuracy is higher than in situ monitoring. Machine learning Accuracy shows 77%, but In situ, Monitoring shows 35% for AM. Liu et al. [8] Used a CNN model named VGG16, Which shows good

accuracy for a small dataset. On the other hand, the images of emissions from the on-site tracking system can be classified with reasonable accuracy with defect classification for the Public industrial benchmark dataset DAGM.

2.2 Object Detection:

Object detection is a subfield within the domain of computer vision that focuses on the identification and localization of objects inside images or videos. Over the years, many programs and methods for finding objects have been created. One of the first and most essential ways to find things was the Viola-Jones algorithm which came up with the idea of Haar-like features and used a chain of classifiers learned with the AdaBoost algorithm [9].

Girshick et al. [10] suggest an object detection algorithm called R-CNN (Regions with CNN features) that is easy to use and can be scaled up. This work investigates the issue of item detection performance approaching a plateau on the PASCAL VOC dataset. At the time, the best methods were complicated ensemble systems that combined low-level image traits with high-level context. The mean average precision (mAP) of the suggested R-CNN algorithm is 53.3%, which is better than the previous best result on VOC 2012 by more than 30%. Yingying Xu et al. [11] This study examines the enhancements in tunnel flaw identification and segmentation through the utilisation of the Mask R-CNN architecture. Particular emphasis should be placed on the path augmentation feature pyramid network (PAFPN) and the edge detection branch. The proposed approach for detecting and isolating tunnel flaws relies on the utilisation of the Mask R-CNN architecture. The PAFPN improves features by adding rich information about the environment and edges. Girshick et al.[12] The concept of Fast R-CNN was conceived. The Fast R-CNN framework employs the VGG16 network, which has a training speed that is nine times faster in comparison to the R-CNN approach. The Fast R-CNN model has a higher learning rate compared to the SSPnet model, resulting in a threefold increase in learning speed. Additionally, the Fast R-CNN model demonstrates a tenfold improvement in testing speed. Furthermore, it achieves superior accuracy in comparison to the SSPnet model.

Faster R-CNN [13] solves the problem of slow-picking candidate proposals in the selective search algorithm by training a region proposal network (RPN) instead of using particular region proposal methods to make region proposals. RPN is prepared to take the features map as an input and output region proposal. Subsequently, the aforementioned elements are introduced into the RoI pooling layer of the Fast R-CNN model, enabling the detection network to leverage the comprehensive features of the entire image. According to Jiang et al. [14] the Faster R-

CNN model is recommended for use, with training conducted on the WIDER face dataset. The individual expresses a desire to achieve cutting-edge outcomes on face detection evaluations such as FDDB and IJB-A. The performance of the models is evaluated on several datasets, such as the WIDER face dataset, in order to assess their effectiveness. They also compare how well the model works to other ways of making face proposals, like EdgeBox, Faceness, and DeepBox. The Faster R-CNN model is more accurate than these different ways while still processing faster.

Liang et al. [15] built a Faster R-CNN-based detection model and used transfer learning methods to train this model, while a transmission line defect data set called Wire10 was built. The detection approach exhibits a high level of accuracy in identifying several defect classes within the Wire10 data set. These defect classes encompass eight distinct types of component defects, as well as the presence of nests and foreign bodies. The model achieves a mean average precision (mAP) of 91.1% and a false rate of 0.68%. Furthermore, it demonstrates effective performance when applied to aerial photos characterised by intricate backdrops and varying lighting conditions. So, the suggested method can be used to find problems quickly and accurately during automated inspections of transmission lines. But there are still some leaks in this set of algorithms. This takes two steps and a lot of time.

Redmon et al. [16] focus on a new method called YOLO (You Only Look Once), which gives a single model for detecting objects and works in real-time. Compared to other real-time detection systems, YOLO has a high mean average accuracy (mAP) even though it works quickly. Experiments show that YOLO is faster and has a better mAP than other real-time detection methods like Fast R-CNN. YOLOv2 [17] model represents an enhanced iteration of the YOLO (You Only Look Once) technique. YOLOv2 demonstrates superior performance in standard detection tasks such as the PASCAL VOC and COCO datasets. YOLOv2 employs a multi-scale training approach to strike a balance between computational efficiency and detection accuracy. The YOLOv2 model demonstrates superior speed and accuracy when compared to other state-of-the-art techniques such as Faster R-CNN with ResNet and SSD. The YOLOv2 model exhibits a favourable balance between computational efficiency and precision, rendering it suitable for real-time object detection tasks. The YOLO9000 model extends the capabilities of the YOLO algorithm.

Zhou et al. [18] The study talks about the structure of the YOLOv5 network, which comprises three main parts: the backbone, the neck, and the output. The YOLOv5x model gets the best

mAP, but its inference speed is slow. Even though the R-CNN anchor box method is also meant to make mAP much better, it works better with global information and worse with information about small areas because it doesn't do regional sampling. Wei Liu et al. [19] proposed Single Shot MultiBox Detector (SSD) a deep learning-based object identification technique that offers a unified framework for both training and inference processes. SSD is much faster and gets the same level of accuracy. Different datasets, like PASCAL VOC, COCO, and ILSVRC, have been used to test it. Miao et al. [20] made the method for detecting aerial pictures by combining SSD with two-stage fine-tuning.

2.3 Instance Segmentation

Instance segmentation is a job in computer vision that involves finding and separating the pixels of each object in an image. There are many traditional instance segmentation algorithms, for example, threshold-based algorithms like models based on adaptive filtering and double threshold [21]. YOLACT [22] is a model for real-time instance segmentation that is easy to use and works well. On the MS COCO dataset, YOLACT runs 33.5 frames per second on a single GPU with a mean average precision (mAP) of 29.8. Fast NMS is a quicker alternative to non-maximum suppression (NMS) that only slightly hurts performance. YOLACT is shorter than the ways that came before it and does an excellent job segmenting cases.

L. Piyathilaka et al. [23] proposed the YOLACT instance segmentation algorithm and transfer learning; the study shows an interesting way to find cracks in concrete. ResNet-50 works better in real-time, while ResNet-101 has a slightly higher average level of accuracy. When the YOLACT algorithm is trained on a small dataset and combined with transfer learning, it can correctly find and divide concrete cracks in real-time. M.-C. Chiu et al. [24] compare the performance of the Mask R-CNN model in different configurations and look at how adding more data affects that performance. The results show that the suggested data augmentation techniques are useful for classifying single- and mixed-type defect patterns.

Y. Lei et al. [25] show a new way to automatically divide breast tumours by mixing the mask-scoring R-CNN framework with 3D ABUS imaging. The evaluation measures, such as HD95, MSD, RMSD, and CMD, also showed promising results, which suggests that the automatic and manual segmentations were close to each other. The differences in volume between the robotic and human segmentations were not too significant, which is more proof that the proposed method is correct.

Chapter 3
Methodology

I use the detectron 2 Framework and Yolov8 in this project because it is an advanced computer vision framework made by Facebook AI Research (FAIR) and Ultralytics makes computer vision tools Yolov8 that are free to use. It is used a lot for jobs like finding objects and separating groups of things [27]. Detectron 2 builds on the success of its predecessor, Detectron, and adds several significant new features and changes. It is built with PyTorch and has a modular and flexible design that makes it easy for researchers and developers to change and add to the framework to meet their needs [28].

3.1 Overview Of Detectron 2

Detectron 2's main features and parts include being built with a modular framework that makes model building flexible. It has a lot of ready-made components, like backbone networks (like ResNet and ResNeXt), feature extractors, region proposal networks (RPN), and different heads for different jobs. Also, a collection of pre-trained models that have done well on test datasets like COCO (Common Objects in Context) and LVIS (Large Vocabulary Instance Segmentation) [28]. A YAML-based configuration system makes it easy for users to describe and change model settings, such as backbone architecture, input resolution, data augmentation, optimizer parameters, and more. Scalability and spread training allow training on multiple GPUs or machines simultaneously. The powerful PyTorch Distributed Data-Parallel (DDP) technology lets computers use their resources more effectively [28]. It has many tools and services for loading, evaluating, visualizing, and drawing conclusions from data. It also has APIs that let you add custom datasets and measures for assessing them [28]. In this project, I use Mask R-CNN for Image segmentation, Faster R-CNN, and RetinaNet for object detection. Figure 3.1 below shows the general idea.

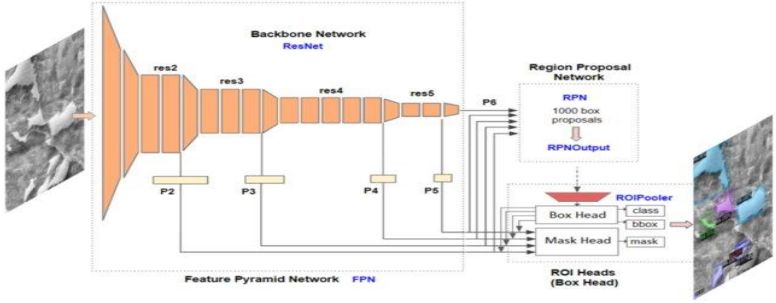

Figure 3.1: The Mask R-CNN system for segmentation [26]

3.2 Overview of Mask R-CNN

The instance segmentation method was employed to further enhance the project. I chose Mask R-CNN as the experimental algorithm because the object recognition method isn't suitable for labelling and finding defects like micro-cracks. The Mask R-CNN framework utilises the Backbone of Mask ResNet-50 or ResNet-101 as a means to generate features. The neck uses the structure of FPN to do the fusion of traits. After getting the improved features, the head uses RPN to get the plans and then uses ROI Align to combine all the ROIs into a feature map of the same size [28]. In this project, I use the ResNet-50 backbone.

The ResNet-50 architecture serves as the foundational framework for the Mask R-CNN model. ResNet-50 is a very effective deep convolutional neural network that has demonstrated exceptional performance across various computer vision applications. It's the feature extractor, giving high-level representations of the picture given as input. An Aspect The Pyramid Network (FPN) is a computational framework designed to enhance the feature extraction process in machine learning models, hence enhancing their capability to detect objects of varying sizes. The process involves the integration of feature maps derived from various hierarchical levels within the backbone network. This lets the model take in both detailed and broad information. This helps make instance separation more accurate. Here"3x" in the name of the setup file shows the training plan. Detectron 2's number before "x" shows how often the model is learned. In this case, the model is trained three times as often as the usual schedule says to do [27]. Figure 3.2 below shows the general idea.

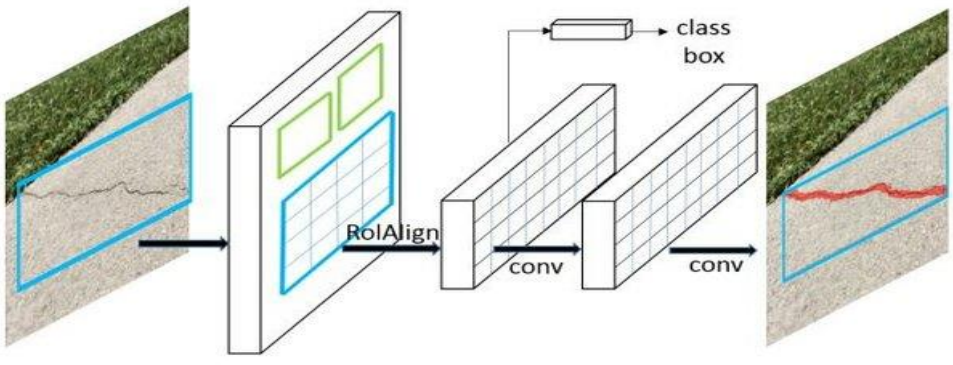

Figure 3.2: The Mask R-CNN system for segmentation [29]

3.3 Overview Of Faster R-CNN

The Faster R-CNN technology is well recognised and utilised for the purpose of object detection. Accurate findings are achieved by the utilisation of a Region Proposal Network (RPN) in conjunction with a Fast R-CNN detection network. It has two essential parts: the RPN, which makes suggestions for regions, and the Fast R-CNN network, which finds objects. Region Proposal Network (RPN) suggests places in a picture where things might be. Using a deep neural network backbone like ResNet or VGGNet, it works on the convolutional feature maps extracted from the image [13]. The RPN moves a small window, called an anchor, over the feature map and estimates, using bounding box regressions, how likely an anchor will contain an object. It makes region suggestions by choosing anchors with high scores. The Fast R-CNN network uses the proposed regions generated by the Region Proposal Network (RPN) for object detection. Region of Interest (RoI) pooling is a technique that transforms the suggested regions into feature maps of a consistent size. Subsequently, the aforementioned RoI characteristics are incorporated into a fully connected network to perform classification and bounding box regression tasks. The network assesses the probability of each object class and enhances the accuracy of the bounding box measurements [13]. Figure 3.3 below shows the general idea.

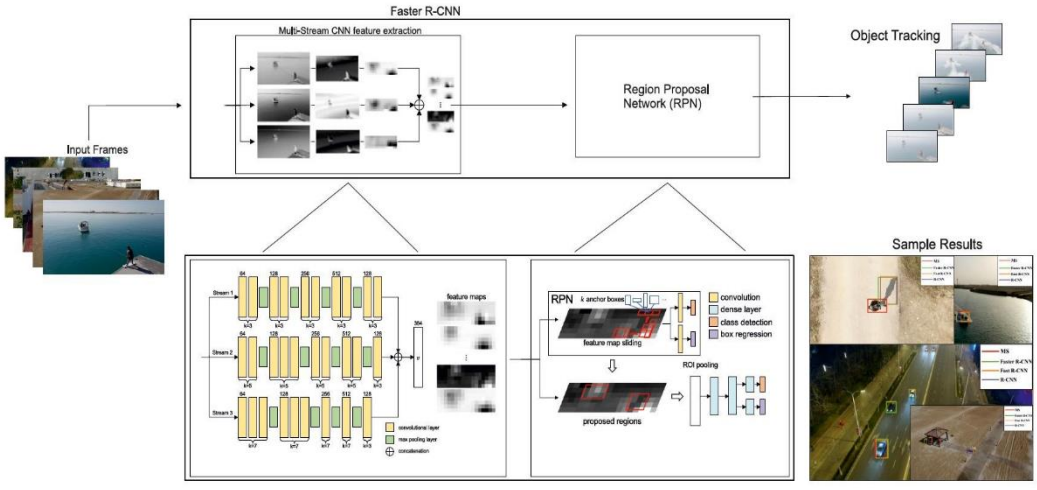

Figure 3.3: The Faster R-CNN System for Detection [30]

3.4 Overview of Retina Net

RetinaNet is a model for finding objects that solve the problem of class mismatch in anchor-based detectors and get high accuracy when finding things. It uses a new focal loss and a Feature Pyramid Network (FPN) design to make it easier to find and classify objects of different sizes.

The RetinaNet model incorporates a Feature Pyramid Network (FPN) to capture and analyse multi-scale features. The Feature Pyramid Network (FPN) is a technique that integrates feature maps obtained from various layers of a backbone network, such as ResNet. This integration results in the creation of a feature pyramid, which effectively preserves high-resolution details from earlier layers while incorporating semantic information from deeper layers. The proposed approach employs an "anchor-based" methodology, wherein a collection of predetermined anchor boxes with varying sizes and aspect ratios are distributed across the feature pyramid. These anchor boxes serve as bounding boxes to detect objects [31]. Each anchor has a class name and an offset for the enclosing box regression.RetinaNet tells you about focus loss. During training, the focus loss gives more weight to complex examples, like rare classes or misclassified samples. This makes sure that easy background examples don't take over. This makes the model pay more attention to complex pieces, which makes object recognition more accurate [31]. Figure 3.4 below shows the general idea.

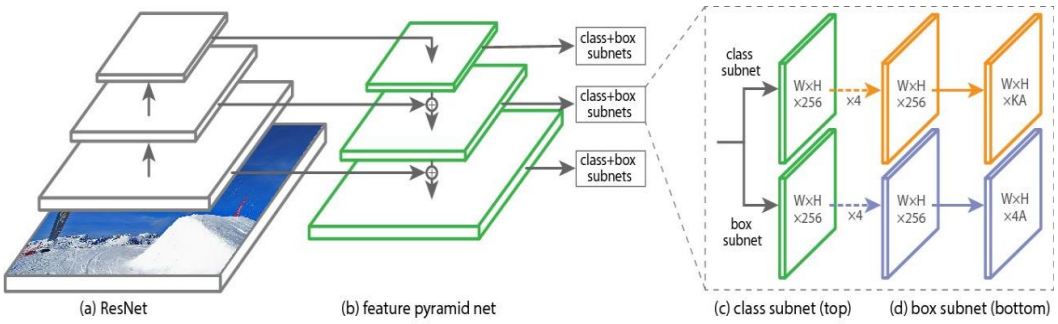

Figure 3.4: Retina Net system for Detection [31]

3.5 Overview of Yolo V8

Regarding object identification, image classification, and instance segmentation, the most recent and cutting-edge YOLO model is YOLOv8. Ultralytics, the company that developed the groundbreaking YOLOv5 concept, is responsible for YOLOv8. YOLO v8 was made with real-time speed and high classification accuracy in mind. It also takes into account the fact that computers only have so much power. It shows that YOLO v8 has hardware efficiency and architectural changes since YOLO versions with the same parameters had better throughput than YOLO v8. Also, Ultralytics says that YOLO v8 can be used on constrained edge devices, which suggests that it focuses on high-inference speed in settings with few resources [32]. Figure 3.5 below shows the general idea.

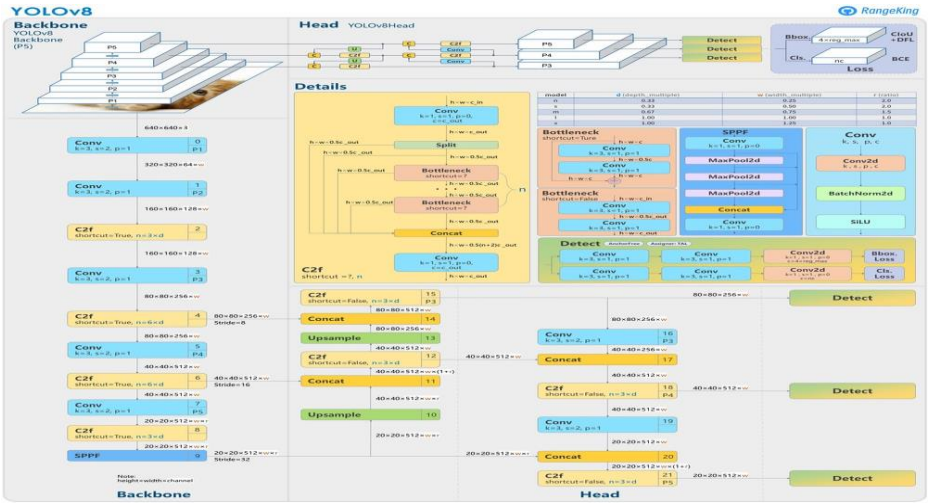

Figure 3.5: Yolo v8 system for Detection [33]

Based on Ultralytics' presentation of YOLO-v8 and YOLO-v5, it is evident that the latter demonstrates noteworthy real-time performance. Additionally, the preliminary benchmarking results provided by Ultralytics suggest that YOLO-v8 will prioritise the deployment of constrained edge devices while maintaining a high-inference speed. So, even though YOLOv8 isn't very different from YOLOv5, it does something new.

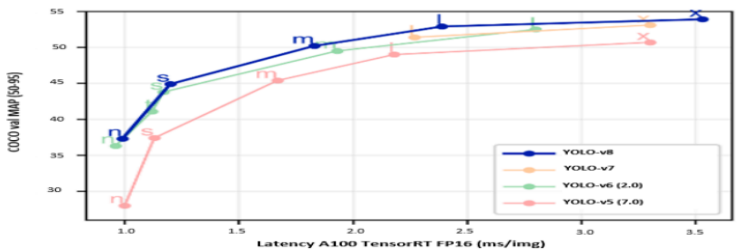

Figure 3.6: Yolo v8 system for Detection [32]

Figure 3.6 above is a comparative view of the four models in YOLO. Our project is to verify the performance of each of these four models in the data set we have constructed.

3.6 Experiment.

3.6.1 Image Collection

Figure 3.7: The SEM for micrographs collection

The first step of a computer vision project is image collection. This project will use an SEM, as Figure 3.6 above shows, to acquire images from random locations on the polished cut surface of the sample, with a magnification of 1500x, as Figure 3.7 below shows. The image size is 2560*1920 pixels after cropping the bottom information that may interfere with object detection. DONNA Ltd has provided us with the metal AM sample, as Figure 3.8 shows below. The manufactured component was cut in the middle, and then the cut surface was polished so that we could take images of the surface inside the element.

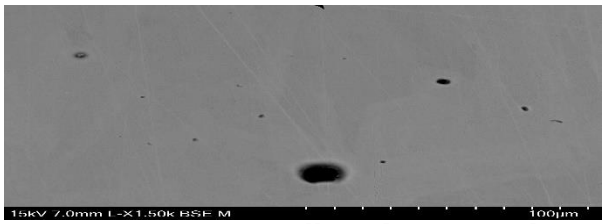

Figure 3.8: A collected micrograph

Figure 3.9: AM sample supplied by DONNA

If used correctly, object detection methods will be enough to meet the project's goals. So, this project's first choice for how to set up the experiment was to use object recognition. As the tests went on, it became clear that the object detection methods were not good enough for this project's goal of finding one class of targets, mainly because of the annotation method. This meant that an instance segmentation algorithm was needed as an improvement.

3.6.2 Object Detection

The experiment object detection stage will be divided into three phases. The first step is data set annotation by labelme software, the second is data set upload by Google Drive, and the third is code running and performance measurement. In the first step, annotate images by Labelme Software that could label targets by Polygon and build three object detection data sets: 60 Images, 42 for the training set, and 18 for test purposes. The three classes of defects are porosity, un-melted particles, and microcrack by Faster R-CNN, Retina Net. First, I converted all image sizes with a Height of 528 and a width of 960, then annotated them by labelling them and uploading them to my Google Drive. When uploading images, it has two parts one part for the training part and one part has the test dataset. Both data has image and JSON file for annotation images. Second, Set the coding part first to install the environment detectron 2.

Apart from importing standard packages such as matplotlib and numpy, the main focus is on the installation of detectron2 and the import of traditional utilities. Next, we must register the JSON format instance split data set files for the training and test sets as separate coco formatted

data sets. Another one is GPU. Using the same GPU in each training session is essential to control bias. As a result, we chose Nvidia Tesla T4 for training in every session using Google Colab. In addition, the data set choice will affect the evaluation of the models. Finally, the model is configured and trained. The batch size was 2. I use 2000 epochs, but microcrack is not detected for this dataset when I run this code. But when I increased it to 2000 to 10000 epochs, a microcrack was also found, and it defects successfully. This system Follows object detection using Faster R-CNN and Retina-Net. When I Trained the dataset total of 42 images defect was 715. Unmelted-particle 639, porosity 67, microcrack 9. In this experiment, we choose the recall, precision, mAP 50, training time, and performance on the test set to evaluate model performances.

Second I use Yolo v8 for the same dataset using roboflow. I uploaded my dataset to the roboflow account and used this link I train this dataset. For the Yolo v8 model, epoch 2000 shows good accuracy. It successfully identifies all defects.

For analyzing the test set's performance, the project will manually compare the test results to labelled data. The samples can be classified as correct detection, missing detection, and false detection, depending on the combination of the right category and the predicted category of the classifier. The YOLOv8 model demonstrates excellent combined speed and accuracy for object detection.

3.6.3 Instance Segmentation

This instance segmentation experiment was added to the object detection experiment for this project. I used the same image and data set splitting approach as in the object detection phase, with the difference that this time we used Labelme for the annotation to fit the instance segmentation model. Unlike the previous labelling with boxes, this project phase uses polygons for more detailed labelling using the Mask R-CNN model. The training and test sets are included within a single file. The next step is to install the environment. Apart from importing standard packages, Next, we need to register the JSON format instance split data set files for the training set and the test set as separate coco formatted data sets. Finally, the model is configured and trained. The architecture of Mask R-CNN was not modified, and the batch size was set to 2 while the number of iterations for training was set at 10,000. These parameter values were determined to yield the most favourable outcomes based on preliminary experimentation. It Successfully identifies all three cracks and has good accuracy.

3.6.4 Analysis Images.

In the last stage, I analyzed all defect images such as Label, Length, Height, Width, Depth, Volume, and area. Later, I calculated the average size of separate cracks. This measurement is shown in centimetres. The analysis of all images using a box plot. Here constant Depth is 0.1 cm, then I calculate the area because it is a 2D image. Height and length calculate the object's bounding box's measurements to determine the object's height and width. The enclosing box is determined by calculating the vertical size as the difference between the maximum and minimum Y-coordinates, and the horizontal width is determined by calculating the difference between the maximum and minimum X-coordinates. The shape of an item is used to figure out its surface area. The exact math formula for calculating an object's volume relies on its shape. If the thing is a rectangular prism, its volume can be found by multiplying its length, width, and height. Here, these images were made using this method.

Add up the sizes of all the items in the image that have been found. The average size is then found by dividing the total amount by the number of things.

Average area = (Total Area of All Objects) / (Number of Objects)

Divide the total area by the number of pixels in the items to get the average area.

Average area = (Total Area of Objects) / (Total Number of Pixels in Objects).

Chapter 4
Result and Evaluation

4.1 Dataset

The first achievement is our data sets. Labelme Software constructed three classes of data sets for training in the Yolo format used in Roboflow. All data sets consist of 60 images, 42 photos in the training set, and 18 in the test set. The primary image size of this data set height was 2560 pixels and width 1920 pixels. But when annotation data using labelme, the height was 528 width was 960 because of more accessible to annotate images using labelme. Microcrack, Unmelted particle, porosity three classes annotation was polygon format.

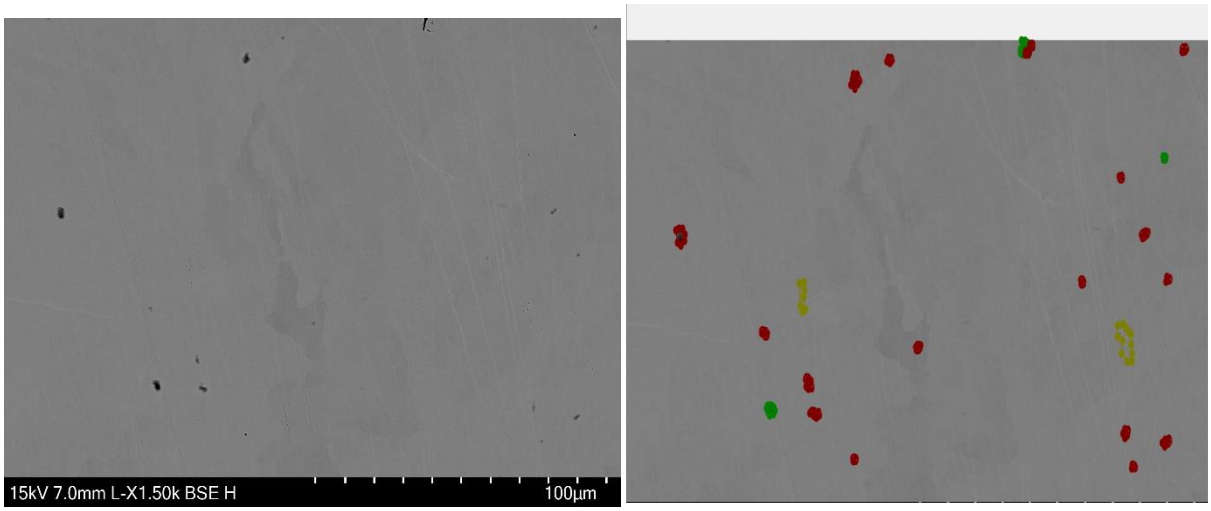

Figure 4.1: The annotation data set constructed by Labelme.

Figure 4.1 The left part of this picture is one of the images in the data set, and the right amount of this picture is one of the annotation data sets. Red is an unmelted particle; green is porosity, and yellow is microcrack. This tool is very friendly for image annotation.

4.2 Object Detection:

4.2.1 Faster R-CNN

Here compare two epochs. One is 2000, and another is 10000 epochs. 2000 epoch does not detect microcrack properly, but 10000 epochs for the training dataset detect perfectly. All dataset is detected for coco format in the faster_rcnn_R_50 model.

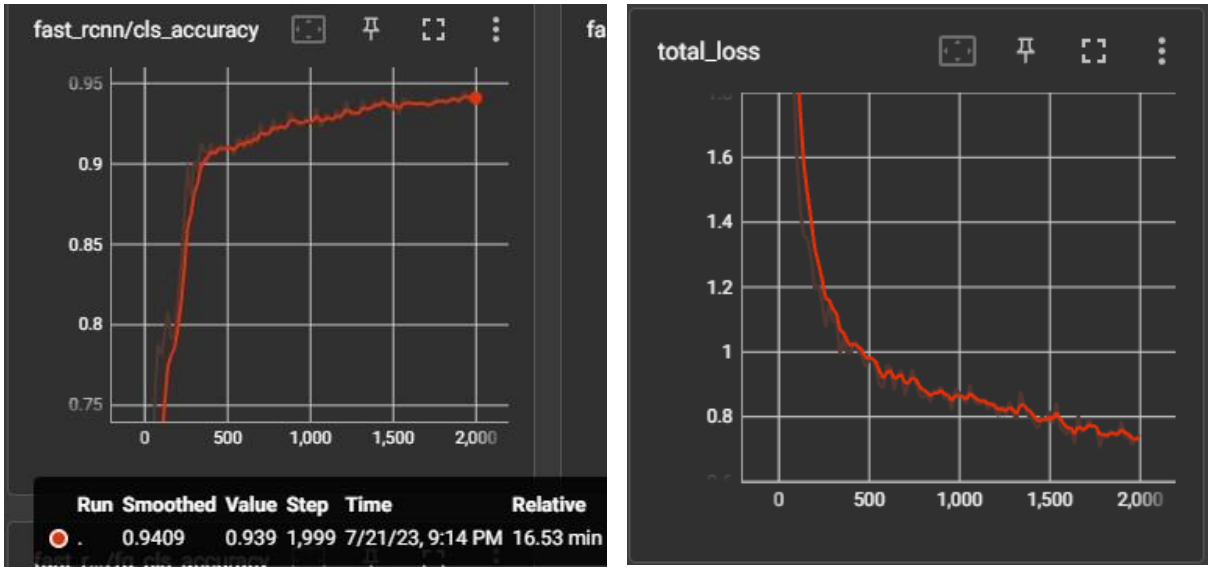

Figure 4.2: Training process fast_rcnn after 2000 epochs.

As shown in Figure 4.2, the training process appears to be smooth without significant fluctuations, with the loss value of 519 iterations dropping sharply before staying slowly afterwards to reach a deficient loss value. At the same time, the accuracy curve also rises sharply before 519 iterations and climbs gently after that, showing a high degree of consistency. Next, we see the data evaluated by the model. With medium area, IoU=0.50:0.95, maxDets=100, the AP can reach 0.33.14, and the AR can get 0.60. It takes 16.53 minutes.

The total training dataset was 42 images, and find actual defect of 715. Unmelted particle 639, porosity 67, microcrack 9.

After identifying object detection, Figure 4.3 shows the overall picture below. This picture shows the classification name and the detection percentage of every defect in this image.

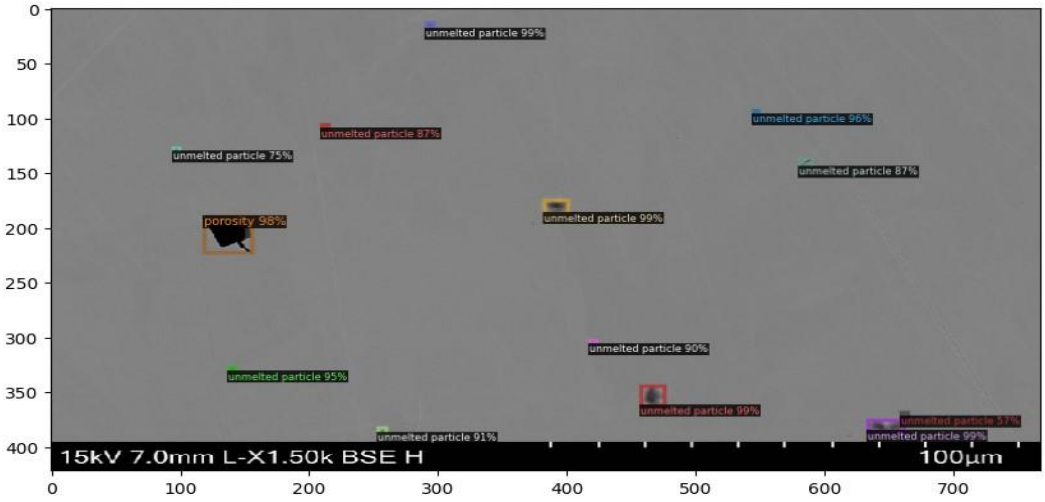

Figure 4.3: Fast_rcnn object detection after 2000 epochs.

The evaluation of the faster CNN for 2000 epochs is shown in the Table below:

Parameters	Small	Medium	Large
Average Precision	0.31	0.59	-1.0
Average Recall	0.36	0.60	-1.0

Table 4.1: Evalutaion Matrix for Fast_rcnn

Three defects of the evaluation.coco_evaluation, Per-category box faster CNN for 2000 epochs results in the table below:

Category	Percentage
Unmelted particle	42.747
porosity	56.675
microcrack	Nan

Table 4.2: Evaluation box for coco format for Fast_rcnn

This model does not identify microcracks for 2000 epochs.

Now For 10000 Epochs in the exact model figure below :

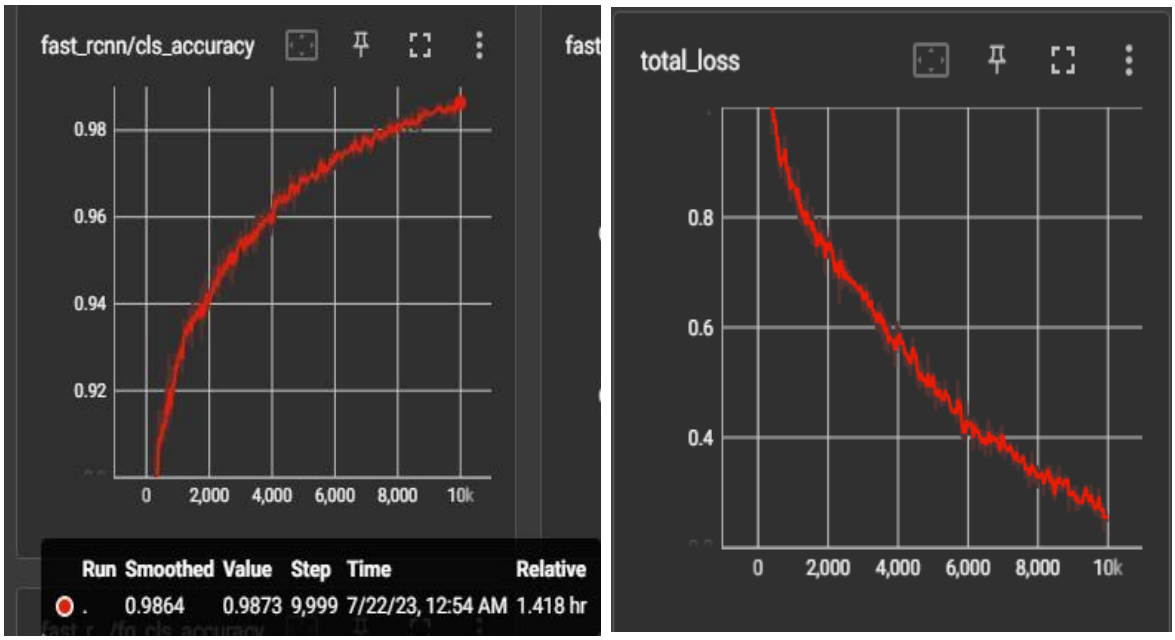

Figure 4.4: Training process fast_rcnn after 10000 epochs.

As shown in Figure 4.4, the training process appears to be smooth without significant fluctuations, with the loss value of 2000 iterations dropping sharply before staying slowly afterwards to reach a deficient loss value. Next, we see the data evaluated by the model. With medium area, IoU=0.50:0.95, maxDets=100, the AP can reach 73.44, and the AR can run 95.8. It Takes Times 1.418 hr. After identifying object detection, Figure 4.5 shows the overall picture below. This picture shows the classification name and the detection percentage of every defect in this image.

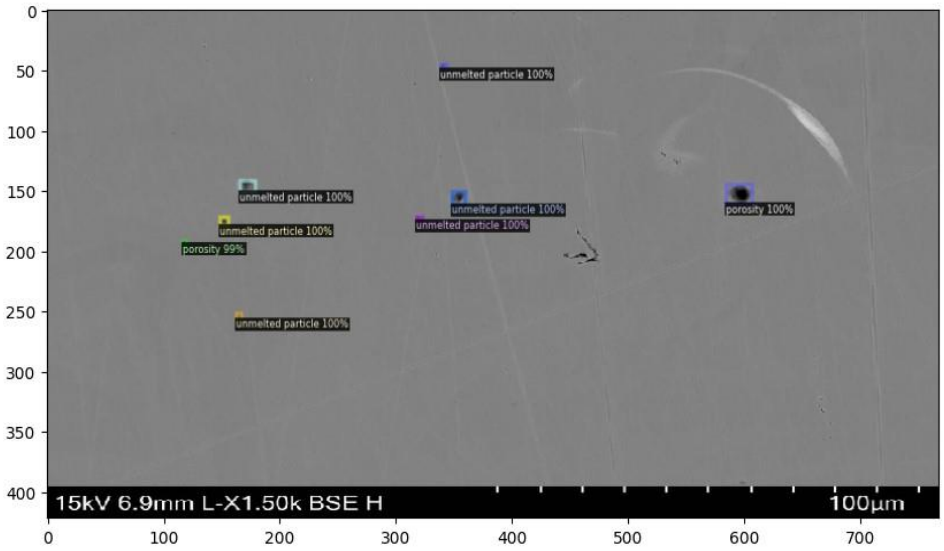

Figure 4.5: Training process fast_rcnn after 10000 epochs object detection

The evaluation of the faster CNN for 10000 epochs is shown in the Table below:

Parameters	Small	Medium	Large
Average Precision	0.71	0.953	-1.0
Average Recall	0.74	0.958	-1.0

Table 4.3: Evaluation Matrix for Fast_rcnn for 10000 epochs

Three defects of the evaluation.coco_evaluation, Per-category box faster CNN for 2000 epochs results in the table below:

Category	Percentage
Unmelted particle	83.15
Porosity	87.75
Microcrack	49.40

Table 4.4: Evaluation Matrix for Fast_rcnn for 10000 epochs

This model identifies microcracks for 10,000 epochs. If the epoch number increases, the accuracy and detection are perfect.

4.2.2 RetinaNet

Retina Net shows better accuracy to comparing Faster R_cnn. This model identifies microcracks after 2000 epochs, but increasing epochs after 10000 deliver better performance and high accuracy. The Training model was retinanet_R_101_FPN_3x. ResNet-101 was used as the backbone, and it was trained three times as many times as the usual configuration called for.

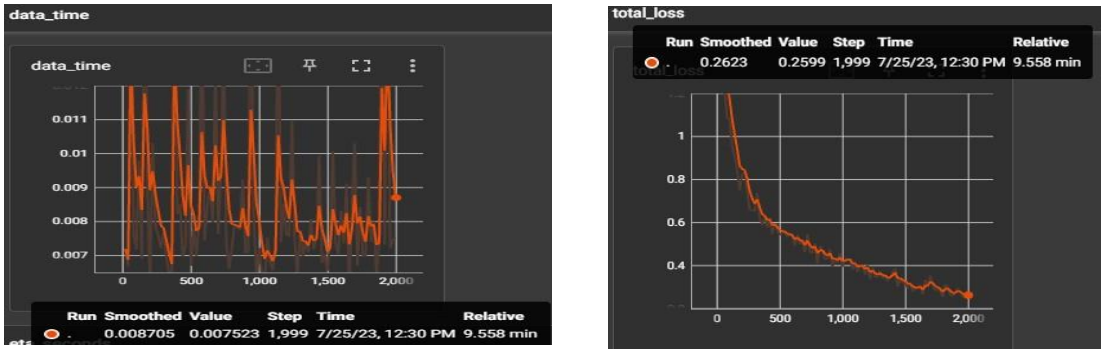

Figure 4.6: Training process retina net after 2000 epochs.

As shown in Figure 4.6, the training process appears smooth without significant fluctuations. At the same time, the data_time curve shows a high degree of consistency. Next, we see the data evaluated by the model. With medium area, IoU=0.50:0.95, maxDets=100, the AP can reach 0.53.18, and the AR can reach 0.63. It takes 9.46 minutes. After identifying object detection, Figure 4.7 shows the overall picture below. This picture shows the classification name and the detection percentage of every defect in this image.

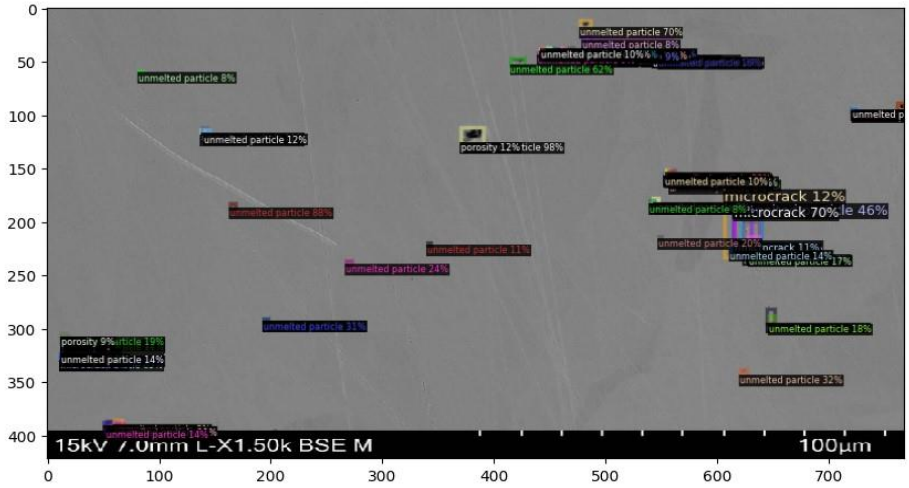

Figure 4.7: Retina net object detection after 2000 epochs.

The evaluation of the retina net for 2000 epochs is shown in the Table below:

Parameters	Small	Medium	Large
Average Precision	0.51	0.92	-1.0
Average Recall	0.60	0.98	-1.0

Table 4.5: Evaluation Matrix for Retina net for 2000 epochs.

Three defects of the evaluation.coco_evaluation, Per-category box retina net for 2000 epochs results in the table below:

Category	Percentage
Unmelted particle	45.44
porosity	66.40
microcrack	49.78

Table 4.6: Evaluation box for coco format for Retina net.

Now For 10000 Epochs in the exact model figure below:

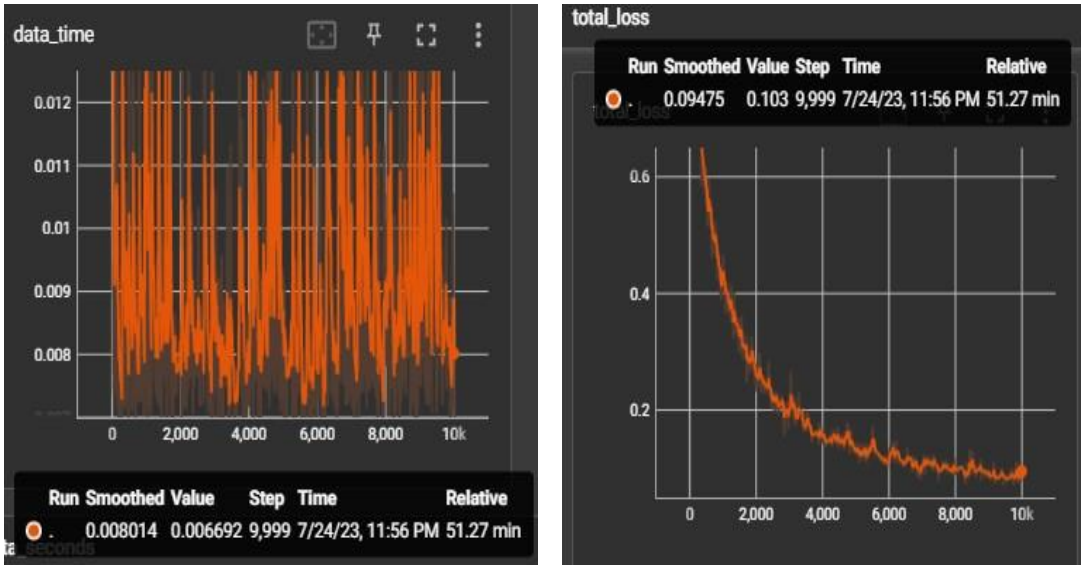

Figure 4.8: Training process Retina net after 10000 epochs

As shown in Figure 4.8, the training process appears to be smooth without significant fluctuations. At the same time, the data_time curve shows a high degree of consistency. Next, we see the data evaluated by the model. With medium area, IoU=0.50:0.95, maxDets=100, the AP can reach 79.63, and the AR can run 84.1. It takes 51.14 minutes. After identifying object detection, Figure 4.9 shows the overall picture below. This picture shows the classification name and the detection percentage of every defect in this image.

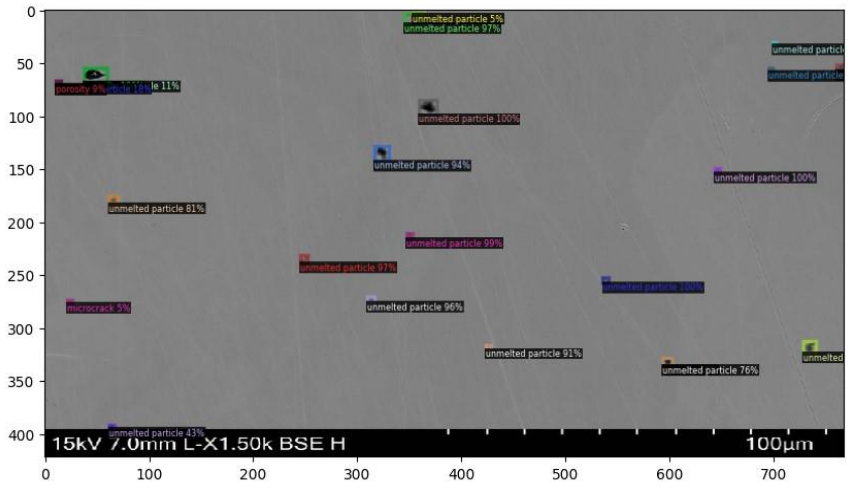

Figure 4.9: Retina net object detection after 10000 epochs

The evaluation of the retina net for 10,000 epochs is shown in the Table below:

Parameters	Small	Medium	Large
Average Precision	0.78	0.99	-1.0
Average Recall	0.82	0.99	-1.0

Table 4.7: Evaluation Matrix for Retina Net for 10000 epochs.

Three defects of the evaluation.coco_evaluation, Per-category box retina net for 10000 epochs results in the table below:

Category	Percentage
Unmelted particle	72.33
porosity	85.29
microcrack	81.26

Table 4.8: Evaluation box for coco format for Retina net for 10000 epochs.

Retina net for 10000 epochs performances is high. All three objects identify high accuracy for the same dataset.

The Retina net model works well for this kind of job because it is accurate and quick for object detection for this dataset. When increasing epochs, the Retina net accuracy is double for detecting microcracks compared to the fast_rcnn model. The performance of the models is judged by how well they handle small, medium, and enormous objects. The RetinaNet models always do better than Fast R-CNN in all size categories for 2000 and 10000 epochs. In short, the RetinaNet model trained for 10000 epochs is the best model based on the review results. It gets higher average precision and recall scores, meaning it can find objects more accurately at different IoU levels and sizes.

4.2.3 YOLOv8

The success of the YOLOv8 object detection model on 2000 epochs for this dataset is analyzed in detail. A Tesla V100-SXM2-16GB GPU was used to train and test the model.YOLOv8 is a famous and influential model for finding objects. It is known for being able to process data in

real-time. In this study, we test the YOLOv8 model on a set of 6 images with a total of 100 items from three different classes: "microcrack," "porosity," and "unmelted particle." In this study, the YOLOv8 model was used. It has 168 levels and 11,126,745 parameters. A Tesla V100-SXM2-16GB GPU was used to train it for 160 iterations on the given dataset. Optimizing the box, class, and focus loss functions was part of the training process to learn how to find and classify objects. The learning rate, batch size, and other hyperparameters were changed to ensure the system would converge safely. The validation sample is used to test the trained YOLOv8 model. Table 4.9 shows the result below:

Class	Images	Instances	Precision	Recall	mAP 50	mAP(50-95)
All	6	100	0.902	0.384	0.827	0.522
Microcrack	6	1	1.00	0.00	0.995	0.697
Porosity	6	8	0.794	0.625	0.771	0.540
Unmelted Particle	6	91	0.913	0.527	0.716	0.328

Table 4.9: Evaluation box for coco format Yolov8 for 2000 epochs

The accuracy (Box P) is 0.902; remember (R): 0.384 mAP at IoU=0.50 (mAP50): 0.827 mAP at IoU=0.50:0.95 (mAP50-95): 0.522. Each class looked at how well the model worked on its own. For the "microcrack" class, the model's accuracy was high (1.0), but its recall was low (0.0), which means that some "microcrack" cases were not found. The "porosity" class had a good mix of how accurate it was (0.794) and how well it remembered things (0.625). However, recall information was not given for the "unmelted particle" class, making it hard to review thoroughly. A relatively small set of 6 images and 100 instances was used for the study. Such a data set may not show how well the model can generalize to data it hasn't seen before.

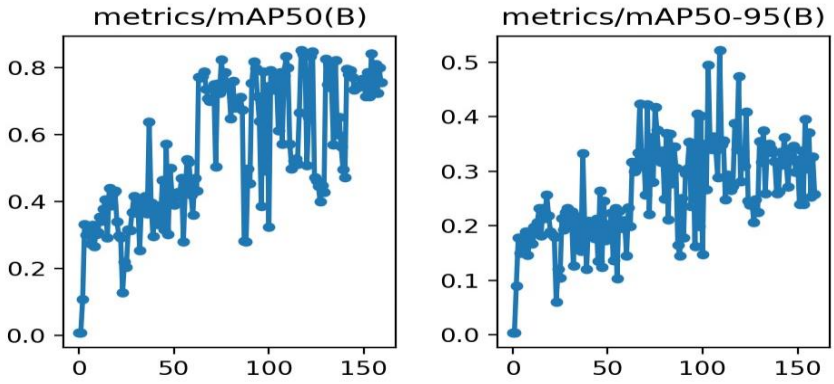

Figure 4.10: Yolo v8 object detection graph after 2000 epochs.

The results of Figure 4.10 show that the model worked best at epoch 110 when it had a high mAP50-95 of 0.827. After this point, the model's performance didn't improve, and the EarlyStopping algorithm kicked in because no gain had been seen in the last 50 epochs. However, more research is needed to determine if raising the number of epochs would make the model more general. When it came to processing speed, the YOLOv8 model was very effective. With 0.2ms for pre-processing, 2.7ms for inference, and 1.5ms for post-processing per picture, the model could be used in real time. After the detection object, Figure 4.11 shows the performances of the Yolo v8 model.

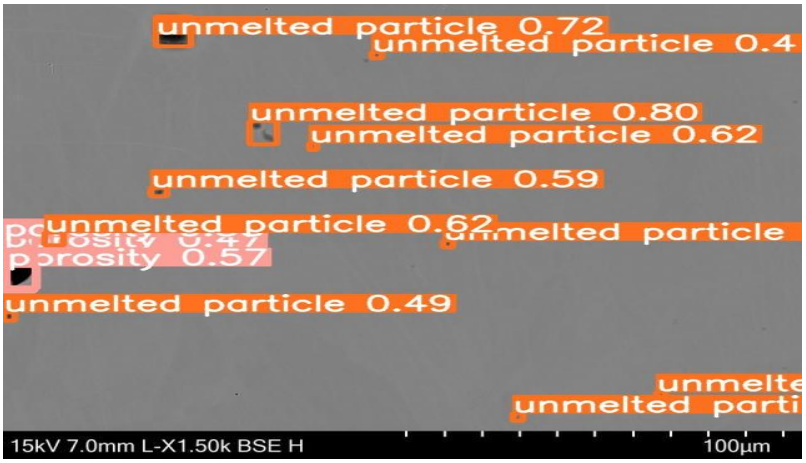

Figure 4.11: Yolo v8 object detection performance after 2000 epochs

10000 epoch for this model:

The training was stopped early because, in the last 50 epochs, there was no sign of growth. This early-stopping approach helps avoid overfitting and saves computational resources when the model's performance hits a plateau. At point 102, the best results were found, and the model's state at that time was saved as "best.pt." This model image shows when the model performed best on the validation dataset while being trained. The training process was done for 152 epochs, which took about 0.387 hours (about 23 minutes). The length of training time says that the model is light and efficient, which makes it suitable for real-time use. Version 0.20 of the YOLOv8 model that was used for evaluation was built with Python 3.10.6, Torch 2.0.1+cu118, and CUDA acceleration on a Tesla V100-SXM2-16GB GPU with 16151MiB of RAM. The model design comprises 168 layers and 11,126,745 parameters in total. The evaluation measures give information about how well the model works in Table 4.10 below:

Class	Images	Instances	Precision	Recall	mAP 50	mAP(50-95)
All	6	100	0.479	0.821	0.823	0.474
Microcrack	6	1	0.501	1.00	0.995	0.597
Porosity	6	8	0.337	0.750	0.768	0.508
Unmelted Particle	6	91	0.598	0.714	0.705	0.316

Table 4.10: Evaluation box for coco format Yolov8 for 10000 epochs

The model got a recall of 0.821 and a box accuracy of 0.479. The mean Average Precision at IoU=0.50 is 0.823, which shows that the model can correctly find and classify objects with an IoU threshold of 0.50. The "microcrack" class had the best recall of 1.000, which means that the model correctly found all instances of this class. The "microcrack" box precision is 0.501, which is pretty good. The " porosity " class got a recall of 0.750 and a box accuracy of 0.337. The model's "porosity" mAP-50 is 0.768, which shows it works well. The "unmelted particle" class got 0.714 recall and 0.598 box accuracy. The model's "unmelted particle" mAP- 50 value is 0.705. The results of Figure 4.12 show that the model worked best at epoch 102.

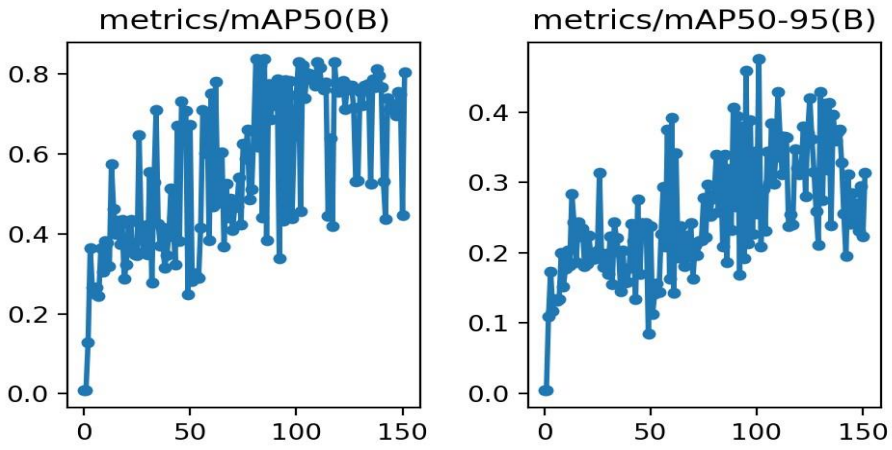

Figure 4.12: Yolo v8 object detection graph after 10000 epochs

The speed of model inference is excellent: pre-processing takes 0.2ms per image, inference takes 2.3ms per image, calculating loss takes 0ms per image, and post-processing takes 1.1ms per image. This means the model can handle images well and be used in real-time applications. After the detection object, Figure 4.13 shows the performances of the Yolo v8 model for 10,000 epochs.

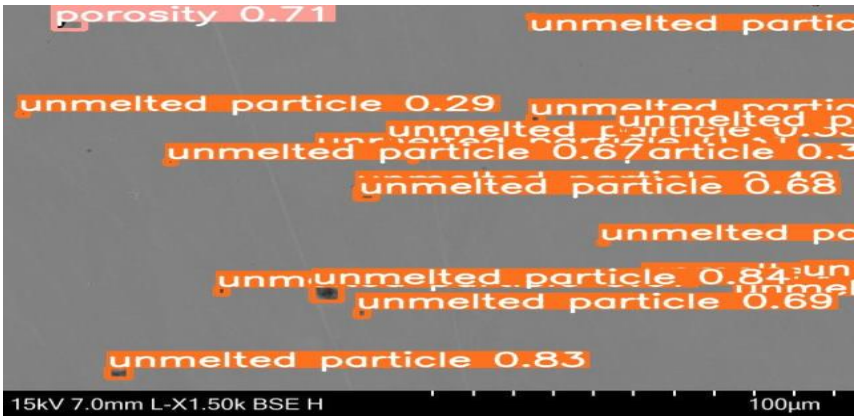

Figure 4.13: Yolo v8 object detection performance after 10000 epochs

A Summary and Analysis for 2000 and 10000 epochs:

Metric	2000 epochs	1000 epochs	Improvement
Best Epoch	102	N/A	N/A
Precision	0.902	0.479	No
Recall	0.384	0.821	Yes
mAP 50	0.827	0.823	Slight
mAP(50-95)	0.522	0.474	No
Inference Speed	2.7ms	2.3ms	Slight

Table 4.11: Performance Analysis Both Epochs

Regarding performance, the model with 10000 epochs had a higher recall (R) than the model with 2000 epochs. This means that it is better at finding instances of the target classes. However, the box precision (P) and mAP-50 scores were better for 2000 epochs, which suggests that the model's ability to locate objects properly was better during training. The YOLOv8 model did well in both situations, with good inference speed and competitive mAP-50 scores.

4.3 Instance segmentation

First, we look at the loss and accuracy curves to see how the model was trained. The x-axis in the initial image denotes the number of training epochs, while the y-axis indicates the measure of accuracy. The second image displays the training epochs on the x-axis and the overall loss on the y-axis. We are training Epoch for 2000 Figure 4.14**.**

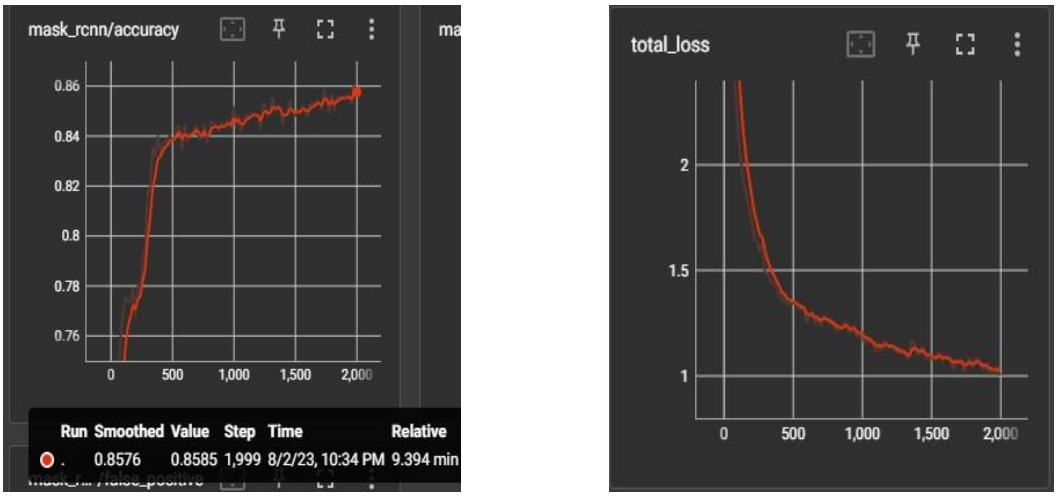

Figure 4.14: Training process of Mask R-CNN for 2000 epochs.

According to the findings depicted in Figure 4.14, it can be observed that the training process exhibits a consistent and steady progression, devoid of any notable variations, with the loss value of 500 iterations dropping sharply before staying slowly afterwards to reach a very low loss value. At the same time, the accuracy curve also rises sharply before 500 iterations and climbs gently afterwards, showing a high degree of consistency. Next, we see the data evaluated by the model. With medium area, IoU=0.50:0.95, maxDets=100, the AP can reach 35.06, and the AR can reach 38.90; the model completes the training of 2000 iterations in 9 min 34s.

The evaluation of the Mask_Rcnn for 2000 epochs is shown in the Table below:

Parameters	Small	Medium	Large
Average Precision	0.33	0.58	-1.0
Average Recall	0.37	0.61	-1.0

Table 4.12: Evaluation Matrix for Mask_RCNN for 2000 epochs.

Three defects of the evaluation.coco_evaluation, Per-category box Mask_RCnn for 2000 epochs results in the table below:

Category	Percentage
Unmelted particle	45.68
porosity	59.52
microcrack	0.00

Table 4.13: Evaluation box for coco format Mask Rcnn for 2000 epochs

Now For 10000 Epochs Figure 4.15 shows the performances below:

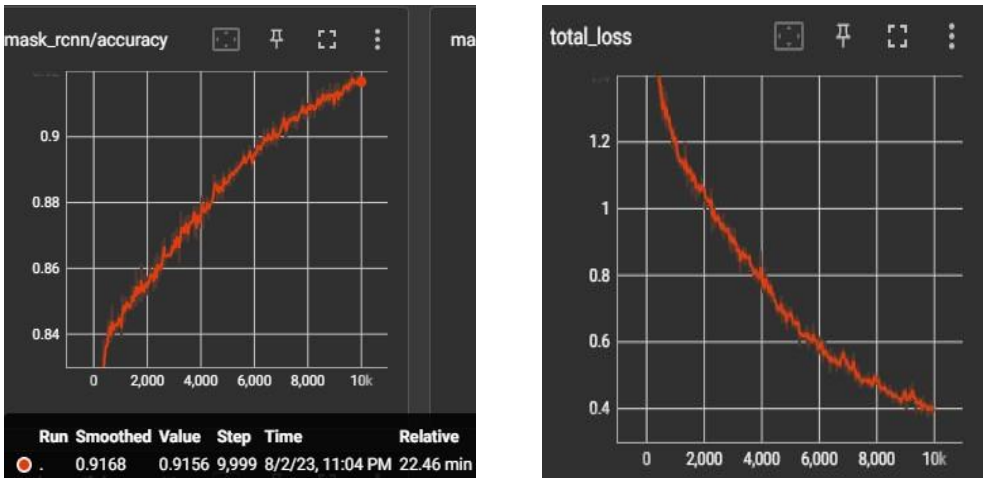

Figure 4.15: Training process of Mask R-CNN for 10000 epochs

As shown in Figure 4.15, the training process appears smooth without significant fluctuations. At the same time, the accuracy curve also rises sharply. Next, we see the data evaluated by the model. With medium area, IoU=0.50:0.95, maxDets=100, the AP can reach 82.47, and the AR can reach 84.8; the model completes the training of 10000 iterations in 22 min 46s. The evaluation of the Mask_Rcnn for 10000 epochs is shown in the Table below:

Parameters	Small	Medium	Large
Average Precision	0.81	0.96	-1.0
Average Recall	0.83	0.96	-1.0

Table 4.14: Evaluation box for coco format Mask Rcnn for 2000 epochs
Three defects of the evaluation.coco_evaluation, Per-category box Mask_RCnn for 10000 epochs results in the table below:

Category	Percentage
Unmelted particle	87.57
porosity	93.42
microcrack	66.42

Table 4.15: Evaluation box for coco format Mask Rcnn for 10000 epochs

A Summary and Analysis for 2000 and 10000 epochs, for instance, segmentation :

Category	Epoch 2000	Epoch 10000
Unmelted Particle	45.68	87.57
Porosity	59.52	93.42
Microcrack	0.0	66.42

Table 4.16: Evaluation box for coco format Mask Rcnn for 10000 epochs

Using the rating criteria given, it is clear that the 10000-iteration epoch is the most accurate. At this epoch, the model has much higher Average Precision (AP) and Average Recall (AR) values than it did at the 2000 epoch across all IoU levels and area scales. At 10000 epochs, the model gets an AP of 82.475%, while the model at 2000 epochs only gets 35.069%. In the same way, the AR at 10000 epochs is 0.848, but at 2000 epochs, it is only 0.389. Based on these results, it looks like the model trained for 10,000 epochs does a much better job of finding things and figuring out where they are.

4.4 Analysis Images:

Detection and annotation is the process of finding and labelling different things in a picture, with a focus on "microcracks," "porosity," and "unmelted particles." The image is loaded, and the annotations in the JSON file, which tell where these features are and what they are called, tell us what they are. Figure 4.16 shows below:

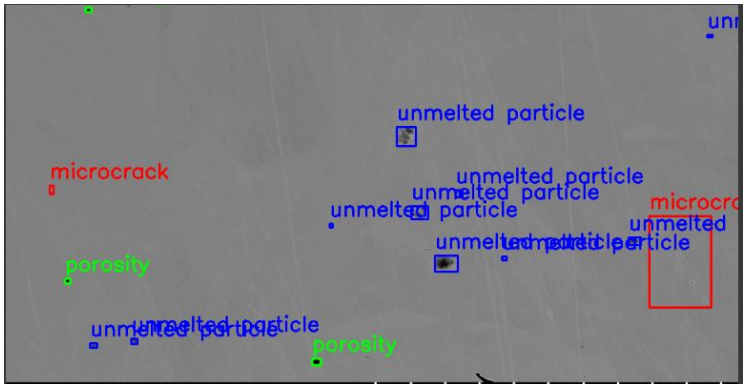

Figure 4.16: Detection and Annotation Images

Boxes and Labels at the Edges: Make a box around the found object for each annotation and show the object's label. Different colours on the labels offer different kinds of traits. Green means "porosity," red means "microcrack," and blue means "unmelted particle." Other things are shown with a white box around them.

Size and Shape Estimation: It figures out the size and shape of each trait found. Using a made-up conversion factor (pixels to cm), it figures out the length and width of each containing box in centimetres. In each case, it is assumed that the depth of each element is 0.1 cm. Based on these numbers, determine how many cubic centimetres each feature takes up.

Length, width, and depth: The coordinates of the bounding box made around each feature are used to figure out the height and width of the feature. It uses the made-up conversion factor to change the size of the bounding box from pixels to centimetres. In each case, it is assumed that the depth of each element is 0.1 cm.

Volume Identification: The volume of each detected feature is found by multiplying its length, width, and estimated depth (0.1 cm) together. This gives you a rough idea of the amount in centimetres.

Area Calculation: Figures out in square pixels how much space each identified feature takes up. All found things, like microcracks, pores, and unmelted bits, have their areas calculated. Each feature's areas are also added to see how much space each type of feature takes up in the whole picture.

Average Area Calculation: This function figures out the average size of each feature type in the picture. To do this, the total area of each feature is divided by the number of times it appears in the image. The same hypothetical conversion factor turns the average size into square centimetres.

Storage of Results:

The answers are saved in a list of dictionaries called the "cracks list." Each dictionary is a single crack found in the picture, and it has the following information:

"Label" is the name given to the crack, such as "microcrack," "porosity," or "unmelted particle."

"length_cm" is the crack's length in centimetres.

"width_cm" is the crack's width in centimetres.

depth_cm: The depth of the crack, which is believed to be 0.1 cm.

"volume_cc" is the number of cubic centimetres of the crack, which is the length * width * depth.

"area" is the crack's size regarding how many pixels are in its segmentation mask.

Also, it figures out the average area of microcracks, porosity, and unmelted particles and saves them in variables called average_microcrack_area, average_porosity_area, and average_unmelted_particle_area Figure 4.17 shown below details:

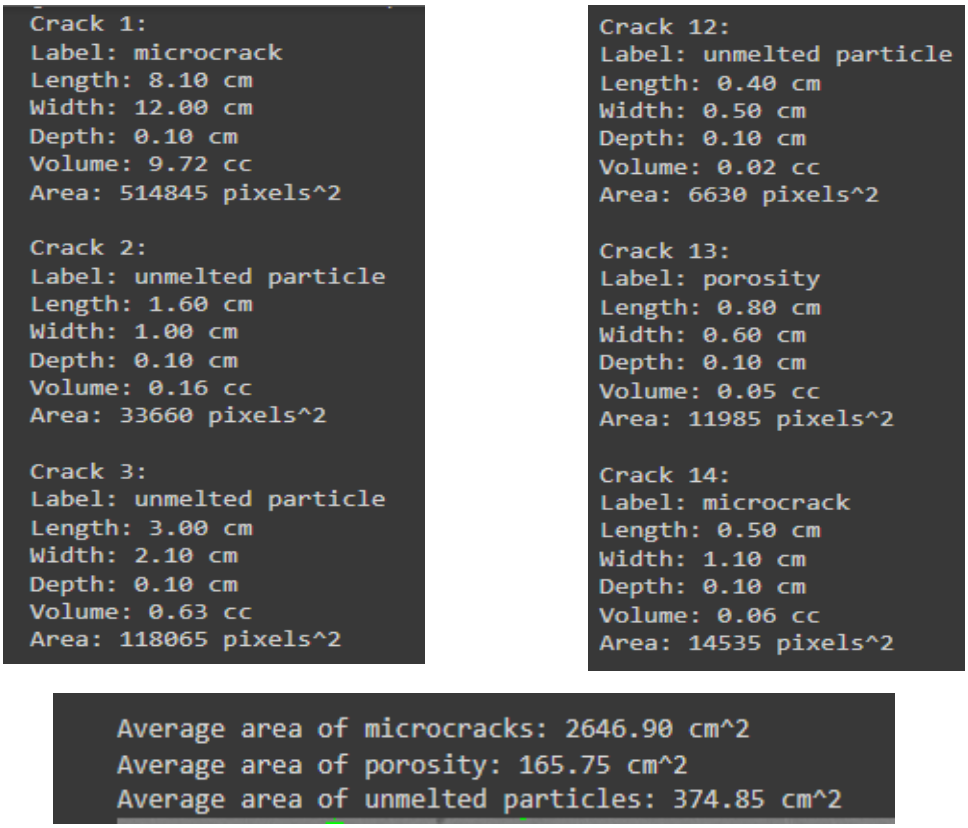

Figure 4.17: Output Results of Every Image.

4.5 Discussion of Implications:

The project uses computer vision and machine learning approaches and analyses the data for advanced manufacturing. Zhang et al. [5] use the CNN model to predict porosity. Still, this project successfully analyses the data and reaches a state-of-the-art level of detection accuracy, comparing performances of different Mask R-CNN, retina net, fast rcnn, and yolo v8 models, and found that model performance results on data set mask rcnn and yolo v8 performance good result for 10000 epochs but not a good result for 2000 epochs. Zhou et al. [18] discuss the Yolo v5 model as the best accuracy, but it is slower; In short, Mask R-CNN and YOLOv8 are the models with better overall performance in terms of size, and it is recommended to compare and choose between these two models. Then the project introduced the instance segmentation method and used the flexibility of instance segmentation annotation tools like Labelme to overcome the challenges of the object identification method and, last, analysis of image length, width, depth, volume, area, and Average area for every crack.

Chapter 5
Conclusion and Future Work

This project evaluates whether popular target detection models and instance segmentation algorithms can analyze data (SEM) through computer vision approaches in automatic data analysis for advanced manufacturing. This study focuses on three types of defects: un-melted particles, porosity, and micro-crack. The object detection and instance segmentation algorithms used are YOLOv8, Retina Net, Fast RCNN, and Mask R-CNN. All algorithms show good recognition of unmelted particles and porosity. However, if low epochs like 2000, it does not identify microcrack for Mask R-CNN, Retina net, and Fast R Cnn, but Yolo v8 identifies microcrack. When epochs 10000, it automatically identifies everything, and all algorithms detect microcracks, while YOLOv8 shows much better recognition of microcracks than Mask RCNN. Last, calculate the image area and average area of every crack successfully in each image.

Review of Aim and Objectives: A computer vision algorithm was used to look at pictures with microstructures and figure out how to find and analyze them. Compared to how the four models in the detectron 2 framework affected the recognition quality and the time it took to train. The micrographs were examined to measure porosity, unmelted particles, and microcracks. The manufacturing factors the manufacturer gave matched up with the analyzed data.

Future Work:

Trial Regrets and Lessons Learned:

- Acknowledge that approached certain aspects of the trial differently.
- Despite these misgivings, there is a significant chance that the incidence of micro-cracks could be reduced by decreasing the number of training epochs.

Precision and Results:

- Emphasize that the finest precision and results in the defect detection system are contingent on several factors.
- Emphasize the significance of a large dataset consisting of annotated images to enhance model precision and robustness.

- Mention the crucial function of a high-performance GPU in accurately performing the intensive computations required for defect detection.

In-Situ Monitoring System:

- Describe the plan to integrate an in-situ monitoring system into the extant infrastructure.
- Describe how this system will interface with the core system to facilitate real-time signal processing monitoring.
- The primary objective of this monitoring system is to detect defects in real time, thereby augmenting the overall quality control process.

.

References:

[1] F. H. Kim, F. H. Kim and S. P. Moylan, Literature review of metal additive manufacturing defects. US Department of Commerce, National Institute of Standards and Technology . . ., 2018 (p. 1).

[2] F. Han, S. Liu, J. Zou, Y. Ai and C. Xu, 'Defect detection: Defect classification and localization for additive manufacturing using deep learning method,' in 2020 21st International Conference on Electronic Packaging Technology (ICEPT), IEEE, 2020, pp. 1–4 (p. 1).

[3] M. K. Ferguson, A. Ronay, Y.-T. T. Lee and K. H. Law, 'Detection and segmentation of manufacturing defects with convolutional neural networks and transfer learning,' Smart and sustainable manufacturing systems, vol. 2, 2018 (p. 3).

[4] Dey, Satyajit, Zhijin Lyu, Gauri Mahalle, Anas Achouri, and Abdullah Al Mamun. "Application of Deep Learning models to characterize manufacturing defects in additive manufactured components." *Procedia Structural Integrity* 42 (2022): 943-951.

[5] Zhang, Bin, Shunyu Liu, and Yung C. Shin. "In-process monitoring of porosity during laser additive manufacturing process." *Additive Manufacturing* 28 (2019): 497-505.

[6] Choi, Byungjoo, Yongjun Choi, Moon Gu Lee, Jung Sub Kim, Sang Won Lee, and Yongho Jeon. "Defect Detection Using Deep Learning-Based YOLOv3 in Cross-Sectional Image of Additive Manufacturing." *Archives of Metallurgy and Materials* (2021): 1037-1041.

[7] Herzog, T., M. Brandt, A. Trinchi, A. Sola, and A. Molotnikov. "Process monitoring and machine learning for defect detection in laser-based metal additive manufacturing." *Journal of Intelligent Manufacturing* (2023): 1-31.

[8] Liu, X., and A. Mileo. "A Deep Learning Approach to Defect Detection in Additive Manufacturing of Titanium Alloys." In *2021 International Solid Freeform Fabrication Symposium*. University of Texas at Austin, 2021

[9] Tikoo, Smriti, and Nitin Malik. "Detection of face using Viola Jones and recognition using back propagation neural network." *arXiv preprint arXiv:1701.08257* (2017).

[10] Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 580-587. 2014.

[11] Xu, Yingying, Dawei Li, Qian Xie, Qiaoyun Wu, and Jun Wang. "Automatic defect detection and segmentation of tunnel surface using modified Mask R-CNN." *Measurement* 178 (2021): 109316.

[12] Girshick, Ross. "Fast r-cnn." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440-1448. 2015.

[13] S. Ren, K. He, R. Girshick and J. Sun, 'Faster r-cnn: Towards realtime object detection with region proposal networks,' Advances in neural information processing systems, vol. 28, 2015 (pp. 6, 17).

[14] Jiang, Huaizu, and Erik Learned-Miller. "Face detection with the faster R-CNN." In *2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017)*, pp. 650-657. IEEE, 2017

[15] Liang, Huagang, Chao Zuo, and Wangmin Wei. "Detection and evaluation method of transmission line defects based on deep learning." *IEEE Access* 8 (2020): 38448-38458.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, 'You only look once: Unified, realtime object detection,' in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788 (pp. v, 7, 15, 16)

[17] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263-7271. 2017.

[18] Zhou, Fangbo, Huailin Zhao, and Zhen Nie. "Safety helmet detection based on YOLOv5." In *2021 IEEE International conference on power electronics, computer applications (ICPECA)*, pp. 6-11. IEEE, 2021

[19] W. Liu et al., 'Ssd: Single shot multibox detector,' in European conference on computer vision, Springer, 2016, pp. 21–37 (p. 8).

[20] X. Miao, X. Liu, J. Chen, S. Zhuang, J. Fan and H. Jiang, 'Insulator detection in aerial images for transmission line inspection using single shot multibox detector,' IEEE Access, vol. 7, pp. 9945–9956, 2019 (p. 9).

[21] H. Ye and S. Yan, 'Double threshold image segmentation algorithm based on adaptive filtering,' in 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), IEEE, 2017, pp. 1008–1011 (p. 10).

[22] D. Bolya, C. Zhou, F. Xiao and Y. J. Lee, 'Yolact: Realtime instance segmentation,' in Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 9157–9166 (p. 10)

[23] L. Piyathilaka, D. Preethichandra, U. Izhar and G. Kahandawa, 'Realtime concrete crack detection and instance segmentation using deep transfer learning,' in Engineering Proceedings, Multidisciplinary Digital Publishing Institute, vol. 2, 2020, p. 91 (p. 11).

[24] M.-C. Chiu and T.-M. Chen, 'Applying data augmentation and mask r-cnn-based instance segmentation method for mixed-type wafer maps defect patterns classification,' IEEE Transactions on Semiconductor Manufacturing, vol. 34, no. 4, pp. 455–463, 2021 (pp. 3, 11).

[25] Y. Lei et al., 'Breast tumor segmentation in 3d automatic breast ultrasound using mask scoring r-cnn,' Medical physics, vol. 48, no. 1, pp. 204–214, 2021 (p. 13).

[26] Ackermann, Marc, Deniz Iren, Sebastian Wesselmecking, Deekshith Shetty, and Ulrich Krupp. "Automated segmentation of martensite-austenite islands in bainitic steel." *Materials Characterization* 191 (2022): 112091.

[27] Wu, Y., Kirillov, A., Massa, F., Lo, W. H., & Girshick, R. (2019). Detectron2. Retrieved from https://github.com/facebookresearch/detectron2.

[28] Wu, Y., & Kirillov, A. (2019). Detectron: Object detection with PyTorch. Facebook AI Research Blog. https://ai.facebook.com/blog/-detectron-object-detection-with-pytorch.

[29] Tan, Chenjun, Nasim Uddin, and Yahya M. Mohammed. "Deep learning-based crack detection using mask R-CNN technique." In *9 th International Conference on Structural Health Monitoring of Intelligent Infrastructure*. 2019.

[30] Avola, Danilo, Luigi Cinque, Anxhelo Diko, Alessio Fagioli, Gian Luca Foresti, Alessio Mecca, Daniele Pannone, and Claudio Piciarelli. "MS-Faster R-CNN: Multi-stream backbone for improved Faster R-CNN object detection and aerial tracking from UAV images." *Remote Sensing* 13, no. 9 (2021): 1670.

[31] Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal loss for dense object detection." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980-2988. 2017.

[32] Hussain, Muhammad. "YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection." *Machines* 11, no. 7 (2023): 677.

[33] https://blog.roboflow.com/whats-new-in-yolov8/.