

OBJECT TRACING FROM SYNTHETIC FLUID SPRAY THROUGH  
INSTANCE SEGMENTATION

MD REFAT KHAN PATHAN

DISSERTATION SUBMITTED IN FULFILMENT  
OF THE REQUIREMENTS FOR THE DEGREE OF  
MASTER OF SCIENCE IN  
COMPUTER SCIENCE (BY RESEARCH)

SCHOOL OF ENGINEERING AND TECHNOLOGY  
SUNWAY UNIVERSITY  
MALAYSIA

2024

# ORIGINAL LITERARY WORK DECLARATION

Name of Candidate : MD REFAT KHAN PATHAN

IC /Passport No : A02235456

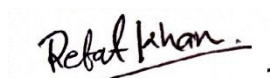
Student ID : 22020176

Name of Degree : MSc in Computer Science (By Research)

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"): **Object tracing from synthetic fluid spray through instance segmentation**

## I do solemnly and sincerely declare that:

1. I am the sole author/writer of this Work;
2. This Work is original;
3. Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
4. I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
5. I hereby assign all and every right in the copyright to this Work to the Sunway University (SunU), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of Sunway University having been first had and obtained;
6. I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by Sunway University.



Candidate's Signature

Date: 18/03/2024

## Subscribed and solemnly declared before,



Witness's Signature

Date: 19/03/2024

Name : Lau Sian Lun

Designation : Professor

# OBJECT TRACING FROM SYNTHETIC FLUID SPRAY THROUGH INSTANCE SEGMENTATION

## ABSTRACT

Since the advent of rapid transportation systems, combustion engines have been a significant technological advancement. These engines require fuel to be atomized for efficient combustion. Atomization involves breaking down liquid sprays into smaller ligaments and droplets, which is crucial for optimal combustion in liquid-fueled propulsion devices. However, accurately measuring and analyzing the size and distribution of objects in a spray can be challenging, especially near the nozzle, due to the complexity of the spray and limited digital detectors. To address this, a method using two region-based R-CNN models named Mask R-CNN and BMask R-CNN and two transformer-based models named PatchDCT and FastInst has been developed to detect and segment individual droplets from synthetic liquid spray images. This approach involved training the model on augmented images and testing it on both augmented and original images. A total of 5791 epochs for Mask R-CNN, 3000 epochs for BMask R-CNN, 20k epochs for PatchDCT and FastInst have been used. Initially, the object detection rate was not good due to the dense objects all over the images. A divide and conquer technique using cropping window extraction was employed with nine window sizes to reduce object density, resulting in a significant increase in object count. The total number of objects on the first test dataset increased from 517 to 1231 using (height, width/3) and 2887 using (height/3, width/3) window with Mask R-CNN. A similar increment happened with BMask R-CNN, PatchDCT and FastInst model testing. After filtering all of the crop windows, it is shown that the models are able to identify significantly more droplets while maintaining a good mask prediction when the width is decreased by three while maintaining the height as the fluid flow is horizontal. The resulting objects were then analyzed using a customized nearest neighbor algorithm to

calculate their correspondence between frames and a BFS algorithm was used to trace their movement path. On the training dataset, Mask R-CNN achieved 74.93 mean average precision with 75% IoU, where BMask R-CNN has only 44.68, PatchDCT shows 81 and FastInst achieved 41.49. Along with the movement path, objects' contact tracing and breakdown rate has also been calculated. The information obtained, including object size, distribution, and tracing, can be valuable for engineers designing fuel injectors, leading to improved performance and efficiency.

*Keywords- Liquid Atomization Analysis, Droplet Segmentation, Object Density Reduction, Object Breakdown, Object Contact Tracing.*



## ACKNOWLEDGEMENTS

First of all, I would like to express my humble gratitude to Sunway University for giving me the opportunity to do my master's study and waiving the tuition fees through the Funded Project Support Scheme. I would also like to appreciate and thank DeepSpray project team for appointing me as a graduate research assistant and supporting the research stipend under funding code GRTEX-INT-DCIS-DOD-001-2020.

I extend my sincere gratitude to Dr. Rahul Babu Koneru for generously providing the dataset crucial for this thesis. I am indebted to Associate Professor Prashant Khare for his invaluable expert insights on methodology and results. I am grateful to Dr. Luis Bravo for granting me access to their High-Performance Computing (HPC) resources. I wish to express my deepest appreciation to my esteemed supervisors, Professor Sian Lun Lau, Professor Mayeen Uddin Khandaker, and Dr. Chiung Ching Ho, for their unwavering support, invaluable guidance, and relentless motivation throughout this research endeavor. I am also indebted to my dedicated lab members, particularly Md. Humayun Kabir Biswas and Wei Lun Lim, for their constant spiritual and technical support, which significantly contributed to the success of this work. I would also like to thank Liang Kah Juin for helping me achieve some results. A special acknowledgment is due to Ms. Foong Yee Ling, whose timely assistance and unwavering support were instrumental in overcoming various challenges encountered during the course of my master's program. Her contribution was pivotal in ensuring the smooth progress of my academic journey.

Lastly, I would like to acknowledge the support provided by my family throughout my studies. I would also like to thank my friends that have helped me whenever I am in a stressful situation.

## TABLE OF CONTENTS

OBJECT TRACING FROM SYNTHETIC FLUID SPRAY THROUGH INSTANCE SEGMENTATION .....	i
ABSTRACT .....	ii
ACKNOWLEDGEMENTS .....	iv
TABLE OF CONTENTS .....	v
LIST OF FIGURES .....	vii
LIST OF TABLES .....	x
LIST OF SYMBOLS AND ABBREVIATIONS .....	xii
LIST OF APPENDICES .....	xiii
CHAPTER 1: INTRODUCTION .....	1
1.1 Overview .....	1
1.2 Problem Statement .....	5
1.3 Research Gap and Questions .....	5
1.4 Research Objectives .....	6
1.5 Scope of Study .....	8
1.6 Technical Challenges .....	8
1.7 Contribution .....	10
1.8 Chapter Outline .....	10
CHAPTER 2: LITERATURE REVIEW .....	12
2.1 Droplet Detection in Liquid Spray .....	12
2.2 Object Segmentation Models .....	17
2.2.1 General Segmentation Types .....	17
2.2.2 Information Extraction-Based Segmentation Types .....	20
2.2.3 Models for Instance Segmentation .....	22
2.2.4 Models with Transformer .....	28
2.2.5 Backbone Network .....	32
2.3 Object Tracing .....	34
2.4 Summary .....	35
CHAPTER 3: METHODOLOGY .....	37
3.1 Dataset .....	43
3.2 Data Augmentation .....	46
3.3 Training with Mask R-CNN .....	49
3.4 Training with BMask R-CNN .....	51
3.5 Training Transformers .....	51
3.6 Image Post Processing .....	52
3.6.1 Extract object mask using contour .....	53

3.6.2	Realign objects coordinates.....	54
3.7	Object Tracing .....	55
3.8	Evaluation Matrix Used .....	58
3.9	Summary .....	60
CHAPTER 4: RESULTS AND DISCUSSION.....		61
4.1	Results of Mask R-CNN .....	62
4.1.1	Object detection on the augmented and preliminary dataset.....	63
4.1.2	Object detection after divide and conquer.....	68
4.2	Results of BMask R-CNN.....	72
4.2.1	Object detection on the augmented and preliminary dataset.....	72
4.2.2	Object detection after divide and conquer.....	79
4.3	Result of Transformer Models.....	85
4.3.1	Object detection on the augmented and preliminary dataset.....	85
4.3.2	Object detection after divide and conquer.....	93
4.4	Result of the final dataset .....	97
4.5	Results of Object Tracing .....	100
4.5.1	DeepSort Tracing .....	102
4.5.2	Object Hot Zone Tracing – with Mask R-CNN .....	102
4.5.3	Object Hot Zone Tracing – with BMask R-CNN .....	105
4.6	Discussion .....	107
CHAPTER 5: CONCLUSION.....		115
5.1	Limitations of Study.....	116
5.2	Future Direction .....	117

## LIST OF FIGURES

Figure 1.1 Left: Sketch of the test rig and Right: Close-up of the primary dense spray (Lieber et al., 2019). .....	4
Figure 2.1 Sample dataset used by different recent literature, which are almost similar simple kind of droplet detection. ....	16
Figure 2.2 Visual differences between semantic, instance and panoptic segmentation. Images are taken from (C. Chen et al., 2020). ....	22
Figure 2.3 Architecture of Fast R-CNN. Image taken from (Girshick, 2015) .....	24
Figure 2.4 Faster R-CNN detection pipeline (RPN + Fast R-CNN). Image taken from (Ananth, 2021). ....	26
Figure 2.5 Simplified principal architecture of Mask R-CNN. Image taken from (K. He et al., 2017). ....	28
Figure 2.6 VGG-16 network architecture. Image taken from (Hassan, 2021). ....	33
Figure 2.7 ResNet architecture. Image taken from (Rastogi, 2022). ....	34
Figure 3.1 (a) Steps of working procedure and, (b) flowchart explaining the unified methodology focusing on these steps. ....	42
Figure 3.2 (a) Showing all seven images and (b) Classes are pointed out in one image. ....	46
Figure 3.3 Image augmentation procedure in (a), (b), (c), some sample images are shown in (d) from augmented image dataset, and finally, visual verification has been done in (e). ....	48
Figure 3.4 Simplified architecture of Mask R-CNN .....	50
Figure 3.5 Image cropping and testing procedure. Initiated with (a) cropping with given window size, (b) separating based on window size and (c) testing procedures of each individual section .....	53
Figure 4.1 Comparison between mAR, mAP, F1 score among different weight testing. ....	63
Figure 4.2 Detection result for multiple weight files on augmented test images. Original test image is shown in (a), 1000th weight test is shown in (b), 2791st weight test is shown in (c) and 5791st weight test is shown in (d). ....	64
Figure 4.3 Comparison of detected objects on different weights in one image. ....	65
Figure 4.4 Summary of detected objects on original images using all three weights. ....	66
Figure 4.5 One original image test result shown for (a) 1000 <sup>th</sup> , (b) 2791 <sup>st</sup> , and (c) 5791 <sup>st</sup> weight (missed object detection areas are marked in red boxes) .....	67

Figure 4.6 One sample image using (height, width/3) crop window from 5791st weight, showing increment in the object detection. Notable object detection increased on smaller droplets; this image is comparable with Figure 4.5 (c). Yellow boxes are to point out region that previous had undetected objects, but now it has a lot of object detected after using divide and conquer method.....	71
Figure 4.7 Comparison of object count between ground truth and selected weights for all configurations and selected weights.....	73
Figure 4.8 Detection result for multiple configurations with different weights on one selected augmented test image. FPN based result with 1000 epoch is shown in (a) and 3000 epoch result is shown in (b). Base model result with 1000 epoch is shown in (c) and 3000 epoch result is shown in (d). Finally, Cascade model result with 1000 epoch is shown in (e) and 3000 epoch result is shown in (f).....	75
Figure 4.9 Summary of detected objects on original seven images with three configuration models on 3000 <sup>th</sup> weight. ....	76
Figure 4.10 Original image testing with different model configurations and epochs. FCN based result for 1000 <sup>th</sup> weight is shown in (a), 3000 <sup>th</sup> weight is shown in (b). Base model result with 100 <sup>th</sup> weight is shown in (c) and 3000 <sup>th</sup> weight is shown in (d). Finally, Cascade model result with 1000 <sup>th</sup> weight is shown in (e) and 3000 <sup>th</sup> weight is shown in (f).....	78
Figure 4.11 One sample image using (height, width/3) window with FPN based BMask R-CNN and 3000 <sup>th</sup> weight, showing the increment in the object detection. This image is comparable with Figure 4.10 (b). ....	80
Figure 4.12 One sample image using (height, width/3) window with Base BMask R-CNN and 3000 <sup>th</sup> weight, showing the increment in the object detection. This image is comparable with Figure 4.10 (d) .....	82
Figure 4.13 One sample image using (height, width/3) window with Cascade BMask R-CNN and 3000 <sup>th</sup> weight, showing the increment in the object detection. This image is comparable with Figure 4.10 (f).....	84
Figure 4.14 Result of object segmentation on different epochs for PatchDCT. Result for 5k epochs is shown in (a), 10k epochs is shown in (b), 15k epoch result is shown in (c) and 20k epoch result is shown in (d).....	86
Figure 4.15 Comparison of detected objects on different weights in one image for PatchDCT. ....	86
Figure 4.16 Summary of detected objects on original seven images with different weights for PatchDCT.....	87
Figure 4.17 One original image test result shown for (a) 5k epochs, (b) 10k epochs, (c) 15k epochs, (d) 20k epochs using PatchDCT model. ....	89
Figure 4.18 Result of object segmentation on (a) 5k epochs, (b) 10 epochs, (c) 15k epochs and (d) 20k epochs for FastInst. ....	90
Figure 4.19 Comparison of detected objects on different weights in one image for FastInst. ....	90

Figure 4.20 Summary of detected objects on original seven images with different weights for FastInst.....	91
Figure 4.21 One original image test result shown for (a) 5k epochs, (b) 10k epochs, (c) 15k epochs, (d) 20k epochs using FastInst model.....	93
Figure 4.22 One sample image using (height, width/3) window with PatchDCT and 20k weight, showing the increment in the object detection. This image is comparable with Figure 4.17 (d).....	95
Figure 4.23 One sample image using (height, width/3) window with PatchDCT and 20k weight, showing the increment in the object detection. This image is comparable with Figure 4.21 (d).....	97
Figure 4.24 Object detection on 1558 <sup>th</sup> image with 5791 <sup>st</sup> weight of Mask R-CNN and (height, width/3) cropping window. ....	98
Figure 4.25 Object detection on the 1558 <sup>th</sup> image with 3000 <sup>th</sup> weight of FPN base BMask R-CNN and (height, width/3) cropping window. ....	99
Figure 4.26 Object detection on the 1558th image with 10k weight of PatchDCT and (height, width/3) cropping window. ....	99
Figure 4.27 Object detection on the 1558th image with 10k weight of FastInst and (height, width/3) cropping window. ....	100
Figure 4.28 Correspondence detection of 1573 <sup>rd</sup> image with 5791st weight of Mask R-CNN. ....	101
Figure 4.29 Sample object tracing tree.....	104
Figure 4.30 Object transition rate from Mask R-CNN on three sections as (a) 0-523rd image range, (b) 524-799th image range and (c) 800-1573rd image range.....	105
Figure 4.31 Object transition rate for BMask R-CNN on three sections as (a) 0-523rd image range, (b) 524-799th image range and (c) 800-1573rd image range.....	107
Figure 4.32 One sample augmented image pointing to the object detection on overlapping scenarios (marked in red boxes).....	112

## LIST OF TABLES

Table 2.1 Summary of traditional droplet detection methods. ....	16
Table 3.1 Correspondence calculation rule. ....	55
Table 4.1 Comparison of the total detected object for 1000 <sup>th</sup> weight's test result on different crop windows for original seven images. ....	70
Table 4.2 Comparison of the total detected object for the 2791 <sup>st</sup> weight's test result on different crop windows for original seven images. ....	70
Table 4.3 Comparison of the total detected object for 5791 <sup>st</sup> weight's test result on different crop windows for original seven images. ....	70
Table 4.4 Comparison of mAP per class on different configurations with selected weights. ....	72
Table 4.5 Comparison of mAP among different configurations with selected weights. ....	73
Table 4.6 Comparison between crop windows for FPN-based BMask R-CNN with 1000 <sup>th</sup> weight. ....	79
Table 4.7 Comparison between crop windows for FPN-based BMask R-CNN with 3000 <sup>th</sup> weight. ....	80
Table 4.8 Comparison between crop windows for Base BMask R-CNN with 1000 <sup>th</sup> weight. ....	81
Table 4.9 Comparison between crop windows for Base BMask R-CNN with 3000 <sup>th</sup> weight. ....	82
Table 4.10 Comparison between crop windows for Cascade BMask R-CNN with 1000 <sup>th</sup> weight. ....	83
Table 4.11 Comparison between crop windows for Cascade BMask R-CNN with 3000 <sup>th</sup> weight. ....	84
Table 4.12 Comparison of the total detected object for 5k,10k,15k and 20k weight's test result on different crop windows for PatchDCT. ....	94
Table 4.13 Comparison of the total detected object for 5k,10k,15k and 20k weight's test result on different crop windows for FastInst. ....	96
Table 4.14 Differences between Mask R-CNN with 5791 <sup>st</sup> weight and FCN-based BMask R-CNN with 3000 <sup>th</sup> weight on different crop windows. ....	108
Table 4.15 Differences between PatchDCT and FastInst with 10k-th weight on different crop windows. ....	109
Table 4.16 Comparison of droplet detection before and after divide and conquer technique on four best model weights. ....	110

Table A.1 Summary of segmentation models.....	119
Table A.2 Experimental results for Mask R-CNN with 1000 <sup>th</sup> , 2791 <sup>st</sup> and 5791 <sup>st</sup> weight on different crop window for seven original images, notable results are highlighted with light blue color.....	122
Table A.3 Experimental results for FPN based BMask R-CNN with 1000 <sup>th</sup> and 3000 <sup>th</sup> weight on different crop window for seven original images, notable results are highlighted with light blue color.....	124
Table A.4 Experimental results for Base BMask R-CNN with 1000 <sup>th</sup> and 3000 <sup>th</sup> weight on different crop window for seven original images, notable results are highlighted with light blue color.....	125
Table A.5 Experimental results for Cascade BMask R-CNN with 1000 <sup>th</sup> weight on different crop window for seven original images, notable results are highlighted with light blue color. ....	127
Table A.6 Experimental results for PatchDCT and FastInst with 10k weight on different crop window for seven original images, notable results are highlighted with light blue color.....	129



## LIST OF SYMBOLS AND ABBREVIATIONS

CNN – Convolutional Neural Network.

R-CNN – Region based CNN.

MRCNN – Mask R-CNN

BMask R-CNN – Boundary Preserving Mask R-CNN.

FPN – Feature Pyramid Network.

CFD - Computational Fluid Dynamics.

mAP – Mean Average Precision.

mAR – Mean Average Recall.

IoU – Intersection Over Union.

RPN – Region Proposal Network.

RoI – Region of Interest.

BFS – Breadth First Search.

KNN – K-Nearest Neighbor.

DCT – Discrete Cosine Transform.

MS COCO – Microsoft Common Objects in Context.

VOC – Visual Object Classes.

CUDA – Compute Unified Device Architecture.

GPU – Graphical Processing Unit.

## LIST OF APPENDICES

PUBLICATIONS .....	118
APPENDIX A .....	119
DETAILED EXPERIMENTAL RESULTS .....	119
APPENDIX B .....	131
RESULT IMAGE AND VIDEOS .....	131

## **CHAPTER 1: INTRODUCTION**

### **1.1 Overview**

Rapid transit technologies have revolutionized our means of transportation, enabling us to move swiftly and efficiently over great distances. The combustion engine, which has emerged as one of the most important breakthroughs in contemporary history (Reitz, 2013), sits at the core of various transportation systems. Combustion engines create power by controlling the ignition of a fuel-air combination, driving vehicles and machines forward with incredible speed and force. However, for the combustion process to work ideally, the fuel must be burnt efficiently, which is difficult owing to the liquid nature of most fuels (Wallington et al., 2006). To overcome this challenge, atomization plays a critical role in the combustion process. Atomization converts liquid fuel into a tiny spray of droplets, increasing its surface area and allowing for faster and more efficient burning. It ensures that the liquid fuel properly combines with the air in the combustion chamber by converting it to a vaporized condition, resulting in greater combustion efficiency and lower pollutants.

Advances in computer simulation and computer vision have provided a powerful tool for exploring and analyzing atomization processes in combustion engines (Sonawane & Agarwal, 2022). Researchers may simulate the flow of jet fuel and examine the influence of various atomization process factors on fuel spray characteristics by using virtual models that match real-world situations. These models allow for the precise visualization of the fuel spray pattern, droplet size distribution, and the interaction of fuel droplets with the surrounding air. Previously, people used statistical and basic mathematical methods to understand the engines' combustion capabilities. The old way to check atomizer quality was first to build one and check the fuel burning and energy generation measurement, which was expensive and time-consuming (Liu et al., 2011). With the development of

deep learning algorithms, researchers are now using computer simulations and AI to simulate atomization processes (Luo et al., 2019; Wei et al., 2020). The second phase of this research area is to do the reverse engineering of this simulation so that we can understand the object breakdown process, determine the atomizer properties, and make necessary changes to get the best combustion result. To do so, first, object detection is needed to identify the droplets, along with other classes like ligaments and lobes. Secondly, object tracing throughout the combustion chamber is needed to get some valuable insights about their trajectory, dispersion, and mixing. For object detection, many advanced algorithms have already been used by researchers, such as YOLO (Chaussonnet et al., 2019; Frazão et al., 2021), Mask RCNN and UNet (Pathan et al., 2022).

In digital image processing, segmentation is a process by which we can partition an image based on some variables to extract necessary elements like objects. Three basic kinds of segmentation processes are available: semantic, instance, and panoptic. Semantic image segmentation is the task of classifying pixel-level clustering of different objects in an image to a predefined class set. It is also called dense prediction, as every pixel is being predicted. The primary goal is to take an RGB or Grayscale image and do some processing to get a segmentation map in which every individual pixel refers to a corresponding class. Instance segmentation can be considered an extended version of semantic segmentation. Unlike semantic segmentation, which includes all classes in one output layer, instance segmentation outputs multiple layers based on the number of objects. For example, if an image is sized as  $(h \times w)$ , and there are five classes with ten objects, the output shape would be  $(h \times w \times 10)$ . The difference with semantic segmentation is that it would result in shape  $(h \times w \times 5)$ , combining the object with the same class in the same layer. Instance segmentation is highly used in autonomous vehicles, medical image segmentation, video surveillance, robotics vision, etc. Many strategic methods were introduced previously for

image segmentation, but the latest method that researchers are prioritizing is the neural network-based models such as CNN, RNN, R-CNN, FCN, and their variants. These models are powered by AI, and supervised or unsupervised learning-based algorithms are used to process various situations. Many researchers have used their custom-modified neural network based on a standard shared backbone model for their segmentation work. Some common backbone networks are VGG-16 (Hassan, 2021), ResNet (Rastogi, 2022), Inception, and Xception model. Very few works are noticed regarding the application of AI in the post processing of fuel atomization.

Fuel atomization is the fundamental breakup process, which breaks down liquid fuel into droplets. It is characterized by the destabilization and disintegration of a liquid fuel column, which results in the development of ligaments and subsequent droplets. The generalization of tracing and characterization of these particles, particularly droplets, is critical for understanding the atomization process and increasing atomizer performance. Specially the generalization is important due to the vastness of atomizer's properties. So that one model can be used with minor modification in different cases. For numerous reasons, the ability to monitor and analyze the behavior of objects such as ligaments during the droplet breakdown process is critical. For starters, it sheds light on the fundamental physical principles and dynamics of primary breakdown. By understanding the behavior of ligaments, researchers can gain a deeper understanding of the factors influencing the breakup process and identify potential areas for improvement. Also, object tracing enables the creation and testing of computational models and simulations. An accurate depiction of ligament behavior in atomization models is critical for predicting and optimizing atomizer performance. Knowledge of objects' movement path is also critical for understanding the object's breakdown rate, trajectory and liquid density. If the breakdown rate is high, it means ligaments are breaking down fast, which is a good indication that atomizer is working effectively. Also, by using trajectory tracing, regional

information can be explored like, from where most of the droplets are being generated. If the breakdown rate is low and object trajectory starts late in observation, it probably means that the fluid density is high, and it is unable to break into droplets at the early stage of atomization. Researchers may modify and enhance models by comparing experimental findings to computer simulations, resulting in more accurate predictions and better atomizer designs. This knowledge can be utilized to design and optimize atomizers for various combustion engine applications, including gas turbines, compression ignition engines, and hypersonic propulsion systems.

This thesis is based on image postprocessing of liquid jet fuel spray in a combustion engine where an atomizer breaks down the fuel into multiple particles, such as droplets, as shown in Figure 1.1. This atomizer design has a significant impact on the droplet generation process. Previously, people needed to change the atomizer design and test to improve the fuel breakdown process, causing a lot of time, labor, and money. The main goal of this research is to analyze the generated droplets (along with other classes) to achieve quantitative performance results so that mechanical engineers can improve the atomizer design without doing any or less physical engine testing.

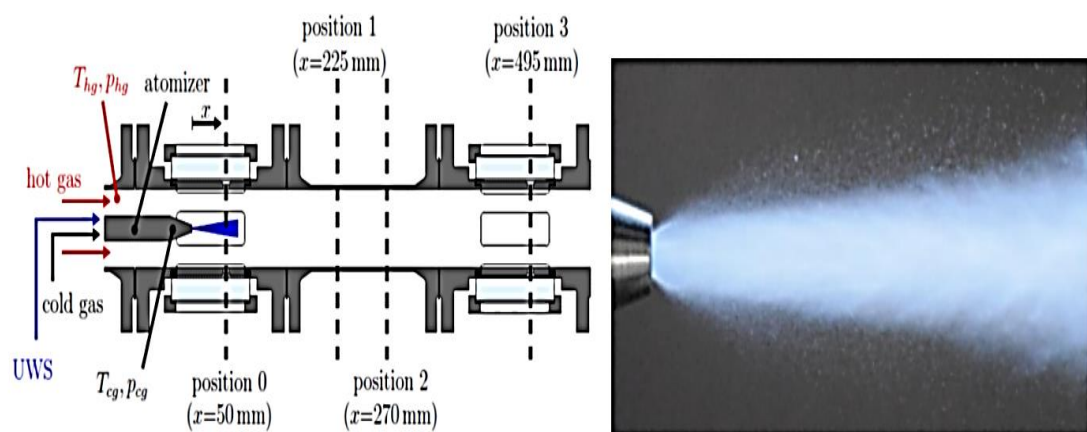


Figure 1.1 Left: Sketch of the test rig and Right: Close-up of the primary dense spray (Lieber et al., 2019).

## **1.2 Problem Statement**

There are some manual methods to analyze the fluid spray, like Phase Doppler Anemometry (PDA), Particle Size Distribution (PSD), and Phase Doppler Particle Analyzer (PDPA). However, these techniques do an estimation analysis of liquid breakdown. Another popular method is Charge Coupled Device (CCD), which is expensive and time-consuming and has thus inspired computerized atomization process simulation. There is tremendous potential in using image processing and deep learning techniques in order to perform the processing of snapshots of liquid atomization simulation experiments. When simulation is done on liquid spray, the total amount of artifacts/features is known in each experiment, but the exact number of artifacts or features is not automatically detected. Thus, it is challenging to acquire a 'ground truth' to measure the accuracy of the simulation. Without reliable ground truth, the accuracy of the simulation in relationship to other metadata generated is indeterminable.

## **1.3 Research Gap and Questions**

Researchers are trying to focus on post-processing techniques to understand the quantitative analysis of liquid breakdown. Mostly with simulation images, as it's easy to reproduce based on different variables and parameters. From the current series of post-processing attempts of liquid atomization simulation experiment, the following research gaps (RG) are identified:

- a) RG1: Previous object detection experiments were conducted in a targeted environment, making it unusable in other similar fields like atomization and spray breakdown scenarios.
- b) RG2: Segmentation of atomization features was not performed well on a generalized data, also it was limited due to the simplicity of dataset that may not have any overlaps.

- c) RG3: The dense objects' moving trajectory and breakdown throughout the combustion mechanism were not analyzed.

These research gaps generate some research questions as listed below:

- a) RG1->RQ1: How can a nonexclusive system be developed to analyze the post processing spray breakup mechanism to detect objects?
- b) RG2->RQ2: Among the identified techniques which is better to improve the segmentation of features generated in liquid atomization simulation experiments?
- c) RG3->RQ3: How can an object's movement path be traced?

#### **1.4 Research Objectives**

The main aim of this thesis is to investigate appropriate contextualization techniques of liquid spray that enable evaluation of the atomization performance. In brief, it detects the objects in spray images and traces the path of transforming dynamic objects (droplets) considering the class transformation. To achieve this from a simulated synthetic fluid dataset, the following objectives were followed:

- a) RG1->RQ1->RO1: Develop a general understanding and generalized dataset of spray breakup mechanisms in atomization processes and explore computer vision techniques to detect objects.
- b) RG2->RQ2->RO2: Trained the Mask R-CNN, BMask R-CNN, PatchDCT and FastInst models for object segmentation and extracting mask with object location and class level – evaluating the model accuracy to select the best one.
- c) RG3->RQ3->RO3: Traced objects with customized BFS and KNN algorithms. Converting the object correspondence into a numeric flow of sequential data to create a tracing path.



To study liquid atomization, researchers often resort to numerical simulations due to the complex nature of the process and the challenges associated with experimental measurements. Computational fluid dynamics (CFD) simulations offer a cost-effective and efficient way to investigate fluid behavior under different conditions. However, the effectiveness of these simulations depends heavily on the accuracy and representativeness of the input parameters and conditions. One of the primary challenges in conducting liquid atomization simulations is the limited range of operating parameters and conditions explored in existing studies. Typically, researchers focus on a narrow set of conditions that may not fully capture the complexity and variability of real-world scenarios. This limitation hinders the generalizability and applicability of the results obtained from these simulations.

The identified research gap (RG1) highlights the limitation of previous object detection experiments, which were conducted within specific environments, rendering their applicability ineffective in analogous scenarios such as atomization and spray breakdown. Addressing this gap, Research Question 1 (RQ1) seeks to devise a nonexclusive system capable of analyzing post-processing spray breakup mechanisms for object detection purposes. To achieve this, the proposed objective (RO1) aims to establish a general understanding of spray breakup mechanisms within atomization processes and to curate a generalized dataset reflective of diverse scenarios. Leveraging computer vision techniques, the objective is to develop robust algorithms capable of detecting objects within such complex environments, thereby bridging the gap between targeted and generalized applications in object detection.

Research Gap 2 (RG2) identifies challenges in accurately segmenting features in liquid atomization simulation experiments due to overlaps and occlusion. Research Question 2 (RQ2) seeks to identify deep learning techniques that can improve feature segmentation in such scenarios. Research Objective 2 (RO2) involves training multiple deep learning

models, including Mask R-CNN, BMask R-CNN, PatchDCT, and FastInst, for object segmentation and extracting masks with object location and class-level information. The objective also includes evaluating the accuracy of these models in segmenting features under challenging conditions of overlaps and occlusion, thereby addressing the segmentation issues identified in RG2.

Research Gap 3 (RG3) points out the lack of analysis on the trajectory and breakdown of dense objects throughout the combustion mechanism. Research Question 3 (RQ3) aims to address this gap by exploring methods to trace the movement path of objects in dense environments. Research Objective 3 (RO3) involves tracing objects using customized Breadth-First Search (BFS) and K-nearest neighbors (KNN) algorithms. Additionally, the objective includes converting object correspondence into a numeric flow of sequential data to create a tracing path, facilitating the analysis of object movement trajectories and breakdown throughout the combustion mechanism as identified in RG3.

### **1.5 Scope of Study**

The scope of this study focuses on segmenting dense fluid spray objects from unlabeled images through the instance segmentation process and tracing the object breakdown path to understand the primary breakdown procedure in a quantifiable way. The dataset used in this work is an unlabeled simulated fluid spray image data consisting of four major classes. The first step would be to generate labels for the dataset for training. After segmentation, object properties would be extracted. Finally, extracted objects are planned to be used in the object tracing part. The ultimate result is to give a quantifiable primary breakdown result to mechanical engineers to improve the atomizer design.

### **1.6 Technical Challenges**

The use of deep learning methods in fluid breakdown analysis is relatively new and current literature shows very few advanced object detection models like YOLO and UNet

having their own constraints and limitations. One of the major challenges regarding the image processing is the lack of ground truth which greatly limits the model training and validation abilities. Regardless of atomization process recording system (for example CCD, PDA, PDPA, Simulation), recorded images are not labelled into their object classification sections. So, an intense manual human labor is required to label each image considering there can be thousands of objects available in one CFD simulated image and thousands of frames will be needed to properly simulate one fluid breakdown process. Changing the simulation parameters would create a completely different fluid flow image set and will require to label those again. Manually labelling these kinds of data is not a sustainable idea, hence require a process to automate the annotations.

Second major challenge would be to selecting a model, train and validation. As latest popular deep learning segmentation models are well equipped and established to detect objects of regularly seen elements like human, vehicles, daily usable objects, terrines, animals and so on, it is difficult to tune the model to train on irregular objects in random shapes and locations. Atomization of fluid mostly generates tiny circular objects called droplets, Droplets can be as small as a dot in plain or a very noisy background which in both cases is very difficult for any model to detect, specially in a dense object area. Irregular shapes of droplets also make it hard to keep it in the assigned class range as bigger droplets can also be identified as another class called ligaments. Detecting and maintaining object classes and determining accurate mask are some primary challenges for deep learning models.

Finally, after detecting the objects, last technical problem would be to trace those objects' movement throughout the spray flow. Objects are not only moving in every direction but also changing their shapes (getting smaller by breakdown process) which causes the same object to be identified as different class in different frames. Keeping track of these changes requires continuous monitoring of the nodes in object tracing graph which is very

hard to debug and computationally expensive. Calculating the object breakdown rate will also require optimized graph searching algorithm to know where the breakdown occurred. Measuring all these scenarios will provide a tracing tree having multiple sub-branches and nodes. Converting this tree to a visually knowledgeable scenario like a video would also be technically challenging.

## **1.7 Contribution**

This study has the following contributions:

- a) The initial dataset had only seven unlabeled images, and custom labeling was done on these images; later, a custom image augmentation method was used to increase the number of images to 1200 for training (1000 images) and testing (200 images). The augmentation was done in a way, so that it would not directly match with original images, hence giving a generalized dataset feature.
- b) Four models were primarily used to train the augmented images, for instance segmentation task. The raw result did not give much information due to dense and uncommon object shapes. A custom post processing method named divided and conquer is proposed to reduce the object density and increase the object count.
- c) Object extraction was performed following object detection, and the object's properties were saved in a JSON file for subsequent processing. Object correspondence location was calculated using a customized KNN algorithm, and object tracing was done using a custom BFS algorithm.

## **1.8 Chapter Outline**

This report has a total of five chapters. The first chapter is the Introduction, and the second chapter explains the recent literature on fluid object detection, the object segmentation model's current development status, and object tracing. The third chapter has the methodology, including dataset explanation and experiments regarding segmentation and tracing. The fourth chapter discusses the experimental results and result comparisons.

Section 4.1, 4.2 and 4.3 explains the result of used Mask R-CNN, BMask R-CNN and Transformer models. Inside these sections, the first sub section describes results of original and augmented dataset, and second sub section describes results of divide and conquer technique. A similar flow has been maintained among these three sections for better understanding and comparison among these models. Section 4.4 has the result of final 1573 images, from which object tracing path is generated. Finally, section 4.5 explained the details results of object tracing. The last chapter concludes the work and summarizes the research findings and future work direction.

## CHAPTER 2: LITERATURE REVIEW

This chapter discusses the relevant background study of the performed research in this thesis. Section 2.1 describes the existing method for object detection on liquid spray, followed by section 2.2, which explains the object segmentation models that have the potential to be used as an object detectors. Section 2.3 discusses the extracted object tracing-related works. This chapter is summarized in section 2.4.

### 2.1 Droplet Detection in Liquid Spray

The atomization of liquid spray plays a crucial role in the fuel supply and ignition process in a chemical reacting engine (N. Sharma et al., 2020). Several parameters can influence atomization, like nozzle size and shape, spray pressure and velocity, liquid surface tension and viscosity, ambient temperature, and humidity (Ray et al., 2019). It is critical to employ a nozzle tailored for the precise type of liquid being sprayed as well as the engine's performance requirements to accomplish optimum atomization. The difficulty of correctly tracking and assessing the size and formation of droplets in a spray is one major problem in improving the atomization process (Fan et al., 2022; Poozesh et al., 2020). This is owing to the continually changing structure of the liquid interface, which makes standard methods of collecting and analyzing droplets challenging. To address this issue, researchers have devised several ways to explore the atomization of liquid spray droplets in a combustion engine. Initial approaches used sensors and other general measurement devices to measure the spray size and distribution directly from a miniature live simulation (Song et al., 2020). Another option is to capture and analyze the spray using high-speed cameras with basic image processing algorithms (Minov et al., 2016; Mlkvik et al., 2015). The latest approaches include deep learning models to analyze complex images and videos to get the targeted data.

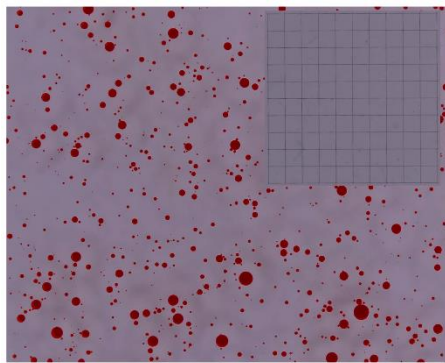
Testing with different fuels on atomization process, (Ainsalo et al., 2019) Injected the fuels from a marine-size common rail diesel injector into a constant volume spray chamber with a nitrogen atmosphere and took shadow images of the sprays. Later, an in-house MATLAB script was applied to analyze the spray penetration and opening angles from the images based on intensity levels and spray width at different distances from the nozzle and also used proprietary analysis software (Davis 10 by LaVision) to detect the droplet sizes and distributions from the images based on intensity thresholding and ellipse fitting. A Giant Nucleus Impactor (GNI) system comprising an optical microscope and two digital cameras to identify droplets has been proposed by (Jensen et al., 2020) for gigantic aerosol particle detection. They employed an autofocus technique to get the best focus and a shading threshold approach to calculate the particle perimeter. A spline fit was also utilized to rectify the difference in lighting across a picture. Using basic geometry and published water activity values, they computed the sea-salt mass in each gigantic aerosol particle. Their disadvantage in identifying droplets is that they may contain mistakes and uncertainties owing to impact splashing and disintegration, particle coalescence on the slide, contamination from handling, and fluctuations in surface tension of hydrated sea-salt aerosols. Binarization, thresholding, and edge removal have also been used to detect droplet-like particles from transmission electron microscopy (TEM) images of TiO<sub>2</sub> nano-aggregates using MATLAB image processing toolbox (Manuputty et al., 2019). MATLAB image processing tools have also been used to analyze macroscopic surface porosity. They scanned both the electron microscope (SEM) and backscattered electrons (BSE) images of the same sample, converted them to binary images, applied contrast enhancement, edge detection, dilation, filling, and adjustment functions, and calculated the fraction of area occupied by the porosities (Chourasiya et al., 2019). A hybrid algorithm based on minimum intensity projection and sharpness has been proposed to detect the droplets from the holograms (Kumar et al., 2019). At first, they subtracted

the image background from the hologram to enhance the contrast and reconstructed it with the Rayleigh-Sommerfeld diffraction kernel. They later binarized the in-focus grayscale image using an iterative thresholding algorithm and a watershed segmentation step to extract each droplet's equivalent diameter, eccentricity, and position. For detecting droplets on water-sensitive paper (WSP), (Ghiani et al., 2020) used a procedure that involved scanning the WSP images with a high-resolution scanner, isolating the WSPs and the blue stains on them using the RGB channels, binarizing the image to separate the wet and dry parts, and measuring the stain area and diameter using a mathematical formula and a spread factor. A three-dimensional imaging-based system called digital inline holography (DIH) has been used to detect droplets that are formed by the breakup of the spray lamella in crosswind conditions, which is a previously untested droplet formation mechanism that affects the drift risk of pesticide spray (Fredericks et al., 2020). Laser diagnostic techniques such as particle image analysis (PIA) were used to detect droplets from CCD camera images where they captured the double-exposure droplet images with a laser interval of  $0.4 \mu\text{s}$  and a camera resolution of  $1600 \times 1200$  (Zhan et al., 2021). They also used image post-processing methods such as the canny operator on thresholding to extract the boundary of the droplet-like regions, eliminate the non-droplet regions, and calculate the droplet diameter and velocity by pairing the corresponding droplets in two sequential frames. Another laser-based shadowgraphy method was proposed to detect droplets using a high-speed camera, and ParticleMaster software was used to analyze the images and calculate the mean droplet size (SMD) (Sikka et al., 2022). They used multivariate data analysis techniques such as PCA and PLS-R to classify and model the acoustic spectra.

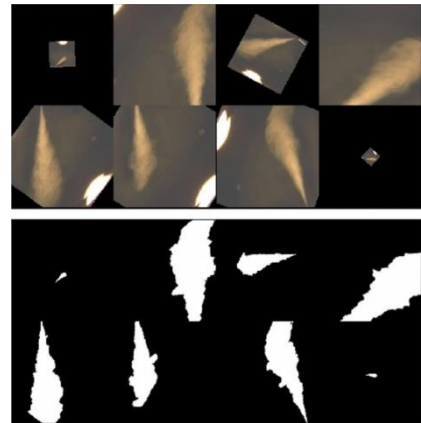
Most previous works on droplet detection depend on image binarization, edge detection, and thresholding. Thresholding is used to detect droplets from images captured from trapped silicone oil using a high-resolution camera and a macro lens (Cerruto et al., 2022),



images of spray-fame synthesis of nanomaterials (Jüngst et al., 2022), images of water droplets with three different Hypromellose polymer dispersions (Koračin et al., 2022) and so on. Though recent methods of droplet detection used deep learning-based models like YOLO (Chaussonnet et al., 2019), U-Net (Huzjan et al., 2023), Mask RCNN (Pathan et al., 2022), Faster R-CNN and Mask R-CNN (Jüngst et al., 2024) . The droplet detection method in this section either works with binarization and thresholding to some point or works with a straightforward dataset (sample shown in Figure 2.1) where object detection is not a challenge (a summary of referenced works is mentioned in Table 2.1). The research gap came to a point where there is a complex liquid spray image set, and objects are very dense, like the one discussed in this paper, where no thresholding or binarization would work. So, a sophisticated deep learning model with post-processing techniques is required to get the droplet properties like mask, size, and shape. Only the detection algorithm may not provide all the relevant information about those properties, so a model that can do detection and segmentation is needed for further analysis, which leads to section 2.2 to search for an instance segmentation model with a relevant reputation.

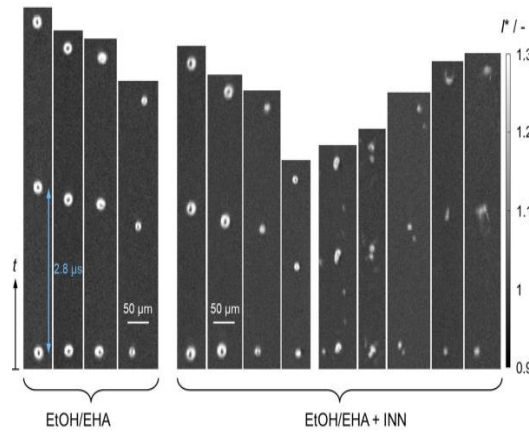


(a) Dataset used by (Cerruto et al., 2022)

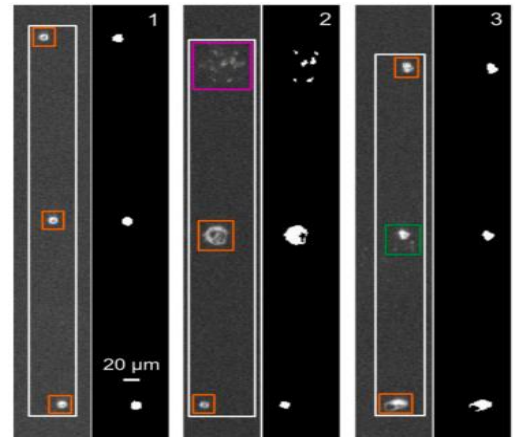


(b) Dataset used by (Huzjan et al., 2023)

Figure 2.1 Sample dataset used by different recent literature, which are almost similar to the simple kind of droplet detection.



(c) Dataset used by (Jüngst et al., 2022)



(d) Dataset used by (Jüngst et al., 2024)

Figure 2.1, continued: Sample dataset used by different recent literature, which are almost similar simple kind of droplet detection.

Table 2.1 Summary of traditional droplet detection methods.

References	Droplet Detection Method	Dataset
(Ainsalo et al., 2019)	Thresholding	Methanol and light fuel oil sprays.
(Manuputty et al., 2019)	Thresholding	TiO <sub>2</sub> nano-aggregates using transmission electron microscopy (TEM) image.
(Chourasiya et al., 2019)	Edge detection	Al-Si-Gr composite spray
(Kumar et al., 2019)	DIH, Thresholding	Monodisperse and Polydisperse droplets were generated by vibrating orifice aerosol generator (VOAG)
(Ghiani et al., 2020)	Binarization	Droplets in a Water-Sensitive Paper (WSP)
(Jensen et al., 2020)	Thresholding	Giant aerosol particles from atmosphere
(Fredericks et al., 2020)	DIH	Shadowgraphs of the lamella breakup in Wilger UR11004 nozzle on different crosswind condition
(Zhan et al., 2021)	Thresholding	Toluene injected through fuel injection system to generate droplets.
(Cerruto et al., 2022)	Thresholding	Liquid fuel spray generated for methanol and light fuel oil from a single location at the edge of the sprays.
(Jüngst et al., 2022)	Thresholding	Droplet disruption in spray-flame synthesis of metal oxides.
(Koračin et al., 2022)	Thresholding	Water and three different hypromellose (HPMC) polymer dispersions
(Huzjan et al., 2023)	Min U-Net	Dataset generated from diesel fuel spray with marine engine injector
(Jüngst et al., 2024)	Region based CNN models	Droplets of metal-oxide nanoparticles using Spray-flame synthesis (SFS)

## **2.2 Object Segmentation Models**

Object segmentation can be broadly categorized into several types, each with its own set of techniques and challenges. These types include semantic segmentation, which assigns a class label to each pixel; instance segmentation, which distinguishes between different instances of the same class; and panoptic segmentation, which aims to unify both semantic and instance segmentation. These techniques are essential for enabling machines to understand and interact with the visual world, making object segmentation a critical area of research and development in the field of computer vision. The following subsections discuss basic to advanced segmentation methods, starting with general segmentation types based on image properties and later specific segmentations like semantic, instance, and panoptic. In this section, the focus method is instance segmentation.

### **2.2.1 General Segmentation Types**

#### **2.2.1.1 Threshold Based Segmentation**

Thresholding (Bhargavi & Jyothi, 2014) is one of the simplest methods for image segmentation. Any grayscale image pixel is based on a 0-255 color intensity value. Thresholding between this range will separate the image into two parts: all the pixel values below the threshold value will be converted to 0 (black), and all the above values will be converted into 255 (white). It is often used to separate the background from its foreground objects. The threshold value has to be precisely balanced, and a constant value may not work for other similar kinds of images, which is a significant flow in this method. Some well-known thresholding methods include Global, Manual, Adaptive, and Optimal Thresholding. Global thresholding is used on a bimodal image with two intensity peaks for object and background (ROGOWSKA, 1999). A thresholding value was gained from this, which was later deduced and adjusted for the whole image. It is highly dependent on light illumination in the image regions. A random value was selected and segmented in

manual thresholding based on these two regions. After that, those two regions' average mean pixel values are calculated. Based on this average value, the image is segmented again, and this process repeats until there is a difference between the previous average value and the user-defined threshold value. In manual threshold, there are two user-defined parameters, lower threshold and upper threshold that determine the threshold interval. The pixels that are selected as being part of the particles in the image are all those with gray-level values that are either equal to or greater than the lower threshold and equal to or smaller than the upper threshold (Emerson, 2023).

Adaptive thresholding (Bradley & Roth, 2007) is an extended method of global thresholding. It divided the image into various subregions and segmented it using individual threshold values to overcome the limitation of illumination issues in global thresholding. Later, all the segmented subregions are combined to make one image. The optimal thresholding (Snyder et al., 1990) minimizes the pixel classification error by taking the derivative of the mean object and background classification error and calculating the threshold value from a quadratic equation.

#### **2.2.1.2 Edge Based Segmentation**

Edge can be defined as the discontinuous local features such as color, level, and texture that divide two objects in an image. In general, the edge can be detected by looking into the adjacent grayscale region's pixel values (Iannizzotto & Vita, 2000). Detecting edge is not the final stage of image segmentation; to have a complete segmentation, it needs further processing. But the edge can locate the object and its border, which can later be used to extract the object from a group for more processing. In a dense image, many unwanted edge groups could be generated for slight differences in regional areas and need to be avoided. Some popular algorithms for edge detection are Sobel (Gao et al., 2010), Canny edge detector (Rong et al., 2014), Kirsch (Kirsch, 1971), Prewitt (Prewitt, 1970),

and Robert's edge operator (Tippett et al., 1965). The core process of these algorithms has a kernel matrix sliding over the image pixel matrix.

#### **2.2.1.3 Region-Based Segmentation**

In the case of a noisy image, a region-based segmentation method is preferred over edge-based segmentation (Kaganami & Bei, 2009). It generally groups the connected pixels of an image that exhibit similar properties such as color, intensity, variance, multispectral features, etc. Like other methods, it also requires some predefined threshold value rules to be segmented into different regions. There are two types of region-based segmentation available: The region-growing method and the Region splitting and merging method. In the Region growing process, a predefined rule such as a threshold is used, starting with a random seed pixel; those rules are applied in a bottom-up manner to adjacent group pixels, which abide by the rules and increase the region. Finally, when no similar pixel exists, we get two regions. In the region splitting and merging method, some predefined rules exist. Initially, the image is divided into multiple regions (basically 4) and is checked to see if those regions satisfy the rules; if not, that region will be divided again till no pixel is left that satisfies the rules. Each pixel is considered a single region and is merged based on an adjacent similar region we split earlier. This method has been used in a variety of fields, such as object detection (Gould et al., 2009), skin lesion segmentation (Rajab et al., 2004), satellite image processing for segmenting large artificial objects (Mueller et al., 2004), and object segmentation in the presence of intensity variation (Mukherjee & Acton, 2014).

#### **2.2.1.4 Clustering Based Segmentation**

It is a kind of unsupervised machine learning algorithm, and the K-means algorithm is highly popular for this type of segmentation in a color image. Some random centroids are initialized, and the distance of all points from the centroid is calculated and assigned to a specific cluster based on the least distance. Then, the centroid value is recalculated based

on the mean value of that cluster, and the whole process is repeated until a suitable solution comes up. Many types of segmentation are available, serving different purposes with similar principles. For noisy images, a fuzzy c-means (FCM) (Z. Yang et al., 2009) is used for image segmentation, FCM + CNN has been used for Lung Nodule segmentation (Ganesan & Merline, 2017), and a hybrid deep autoencoder method with Bayesian fuzzy clustering has been used for brain tumor classification (Siva Raja & rani, 2020), Wide-Swath image segmentation in Synthetic aperture radar (SAR) images (Cristea et al., 2020).

## **2.2.2 Information Extraction-Based Segmentation Types**

### **2.2.2.1 Semantic Segmentation**

Semantic image segmentation is the task of classifying pixel-level clustering of different objects in an image to a predefined class set. It is also called dense prediction, as every pixel is being predicted. The primary goal is to take an RGB or Grayscale image and do some processing to get a segmentation map where every individual pixel refers to a corresponding class. For example, if an image is sized as  $(h \times w)$ , and there are five classes, the output array size should be  $h \times w \times 5$ , where each depth refers to a mask of each class.

Different backbone neural networks are being used on state-of-the-art models such as VGG-16, Unet (Ronneberger et al., 2015) and its variant, ResNet and its variant, Xception (Chollet, 2017), AlexNet, etc. In the training process, often, the models are evaluated with pixel accuracy matrix and Mean Intersection-Over-Union (Mean IoU) methods.

### **2.2.2.2 Instance Segmentation**

Instance segmentation, a subset of semantic segmentation, differs in that it doesn't group all classes into a single layer of output. This technique finds significant application in various fields, such as autonomous vehicles, medical image analysis, video surveillance, and robotics vision. Notable region-based models, for instance segmentation include

RCNN (Girshick et al., 2014), Fast RCNN (Girshick, 2015), Faster R-CNN (Ren et al., 2015), Mask RCNN (K. He et al., 2017), PANet, and TensorMask. Over time, strategic methods have been introduced for image segmentation, with a recent focus on neural network-based models and their derivatives. These models leverage AI and employ supervised or unsupervised learning algorithms to process image datasets. Many researchers customize neural network backbones using well-known models like VGG-Net, ResNet, Inception, and Xception for their segmentation tasks. In recent years, due to the advancements in CNN and their variations, researchers have been exploring various ways to customize these models to achieve more accurate pixel-level segmentation while also simplifying the process of instance segmentation. Supervised segmentation involves the preparation of well-labeled datasets to train models, enabling them to accurately segment images and even detect objects within them. Before the rise of neural networks, segmentation relied on techniques like sliding windows, mask proposals, and contour analysis. However, as neural networks have gained prominence, they have become vital for instance segmentation. In this subsection 2.2.3, we will delve into the models used for supervised instance segmentation, leading to Mask R-CNN as a popular, well-established model for complex object processing.

### **2.2.2.3 Panoptic Segmentation**

This is a combination of semantic and instance segmentation, where it assigns two labels on each predictive pixel value (Kirillov et al., 2019). The same labels are considered the same semantic group, and the instance id separates its instance. It differentiates countable and uncountable objects and separates each with a non-overlapping color. This leads to object detection, localization, and segmentation on an individual basis. It is highly used on sensitive medical images, data annotation/augmentation, remote sensing images, and sensitive separate object detection where individuality is essential. The most common backbones of different panoptic segmentation models are ResNet-50/101, Xception-71,

and Feature Pyramid Network (FPN) (T.-Y. Lin et al., 2017). A sample example of semantic, instance, and panoptic segmentation has been shown in Figure 2.2.

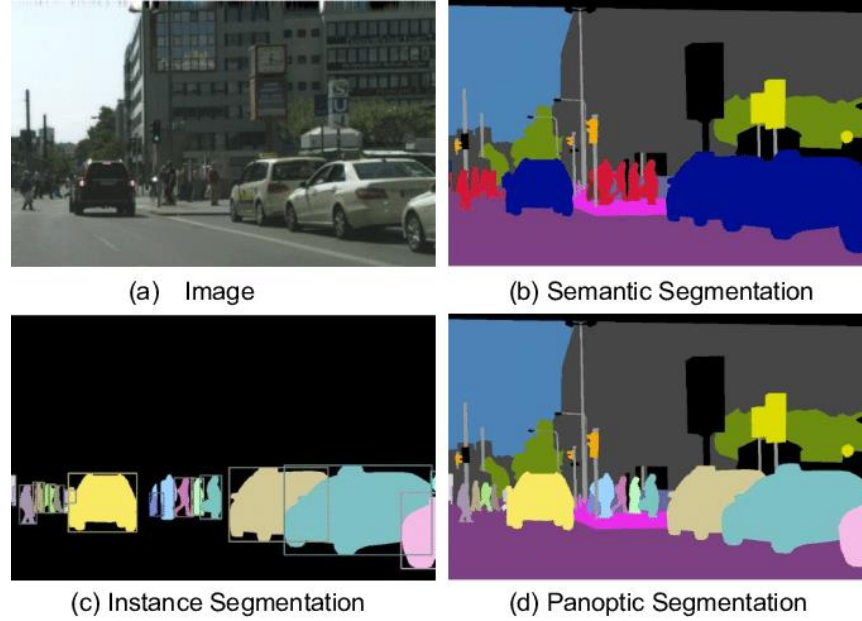


Figure 2.2 Visual differences between semantic, instance and panoptic segmentation. Images are taken from (C. Chen et al., 2020).

### 2.2.3 Models for Instance Segmentation

#### 2.2.3.1 Region Based CNN (R-CNN)

Region-based CNN, or R-CNN, is a modified version of general CNN, including a mechanism that detects the object's region in a targeted image. Previously, CNN was primarily used for object detection, and (Girshick et al., 2014) were among the first people to experiment with CNN for segmentation with regions. They used a two-step procedure to train R-CNN; in the first step, they used selective search on class agnostic region recommendations for high-quality object recognition (Van de Sande et al., 2011), later fine-tuned the CNN and pre-trained like AlexNet by using region proposals. This high-quality detection created a paradox problem, which has been addressed by a cascade R-CNN model (Z. Cai & Vasconcelos, 2019), replacing a single detector with a series of detectors, resulting in an IoU threshold increment. A similar model was also used in video



segmentation (Dong et al., 2019) by introducing an augmented temporal feature module to the network, which reduced motion blur and pose variation. Further development of cascade R-CNN has been done to detect curved and rotated objects from the axis, and a model named Rotated Cascade R-CNN (Y. Zhu et al., 2019) has been proposed and tested on the CTW1500 dataset and broke its previous evaluation record.

As a very early model development, it worked well on its goal but showed flaws like slow training, sequence in layers dependency, and computational high cost as more complex situations arrived. The temptation of improving the model leads to more advanced model development like Fast R-CNN, Faster R-CNN, and Mask R-CNN.

### **2.2.3.2 Fast R-CNN**

Microsoft improved the previous R-CNN by training it on a very deep VGG16 model and named it Fast R-CNN due to its improved 213 times speed on test-time. On its architecture, an image and multiple ROIs are considered input data for a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: SoftMax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss. The simplified model architecture of Fast R-CNN is shown in Figure 2.3, collected from the original paper. This model solved some of RCNN's and SPP-net's (Purkait et al., 2017) drawbacks like slow SVM multi-stage training and detection rate and expensive storage uses in time. Fast RCNN updated some core procedures like single-stage training, the use of multi-task loss to update network layers and removing the need for feature caching on disk storage.

A supervised instance segmentation method with Fast R-CNN has been proposed based on “hyper-columns system 2” (Hariharan et al., 2015) architecture on a VGG-16 backbone that achieved ~95% segmentation quality compared with other datasets like

VOC12 or VOC12+MS COCO (Khoreva et al., 2016). For object detection, which is constant to occlusions and deformations, Fast R-CNN has been combined with a fusion of Adversarial Spatial Dropout Network (ASDN) and Adversarial Spatial Transformer Network (ASTN) and applied to VOC12 and MS COCO dataset (X. Wang et al., 2017). Their proposed model achieved an improvement in average precision (AP) of 1.4% and 3.5%, respectively. To overcome the challenges of detecting pedestrians in natural scenes, a variant of Fast R-CNN has been proposed named Scale-Aware Fast R-CNN (SAF R-CNN) (Li et al., 2017), that uses multiple built-in sub-networks from disjoint ranges on pedestrian scenarios. Final detection results are being calculated from those sub-networks using a gate function on object suggestions, which are demonstrated to be resistant to significant variation in instance scales. Fast RCNN has also been used in a transfer learning process with a small, labeled dataset for crop disease detection to identify insects (Xin et al., 2021).

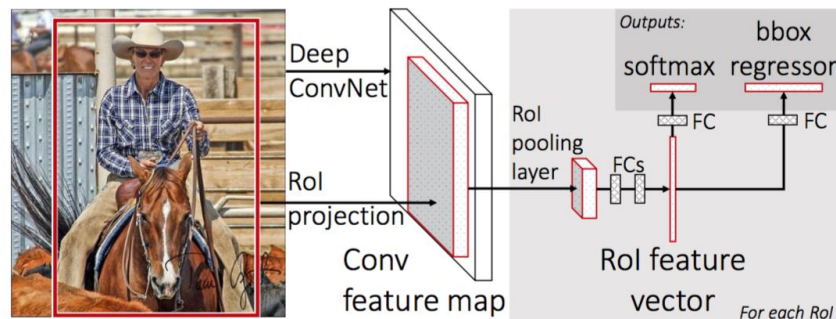


Figure 2.3 Architecture of Fast R-CNN. Image taken from (Girshick, 2015)

### 2.2.3.3 Faster R-CNN

To solve the computational drawback of Fast R-CNN, another improvement has been proposed named Faster R-CNN with a region proposal network that still shares a common convolutional layer with Fast R-CNN (shown in Figure 2.4). A simplified TensorFlow version of Faster R-CNN has been made publicly available by (X. Chen & Gupta, 2017).

This model is used in object detection, vehicle detection, pedestrian detection, image segmentation, instance search, face detection, text classification, etc. Another improvement in this RoI extraction layer is proposed as Generic RoI Extractor (Rossi et al., 2021), which boosts 1.1% AP in detecting the bounding box and 1.7% AP on instance segmentation. A complete-IoU (CIoU) loss and Cluster-NMS (Zheng et al., 2022) have been proposed to enhance geometric factors like bounding-box regression with faster R-CNN to achieve more performance gains.

Using a faster R-CNN object detector, a model named MaskLab (L.-C. Chen et al., 2018) that performs foreground/background segmentation has been proposed to solve the object detection and semantic segmentation problems simultaneously. It is based on the ResNet101 backbone and evaluated by the MS COCO dataset, showing promising mAP matrix results. A two-layer object detection model - D2Det (J. Cao et al., 2020), is proposed to address precise localization with a dense local regression and accurate classification. To reduce the background regional influence, they have used a binary overlap prediction on top of dense local regression. A discriminative RoI pulling is used to classify targeted objects more accurately. MS COCO and UAVDT (Du et al., 2018) datasets are used for their experiment with the ResNet101 backbone. Faster R-CNN is also seen to be used in medical image segmentation such as liver (Tang et al., 2018, 2020) (FR-CNN + DeepLab), brain tumor (Kaldera et al., 2019), heart (Z. Xu et al., 2018) (FR-CNN+Unet), multi-organ (Lei et al., 2020) (FR-CNN + Unet), optic disc and cup (Guo et al., 2021) (FR-CNN + VGG16).

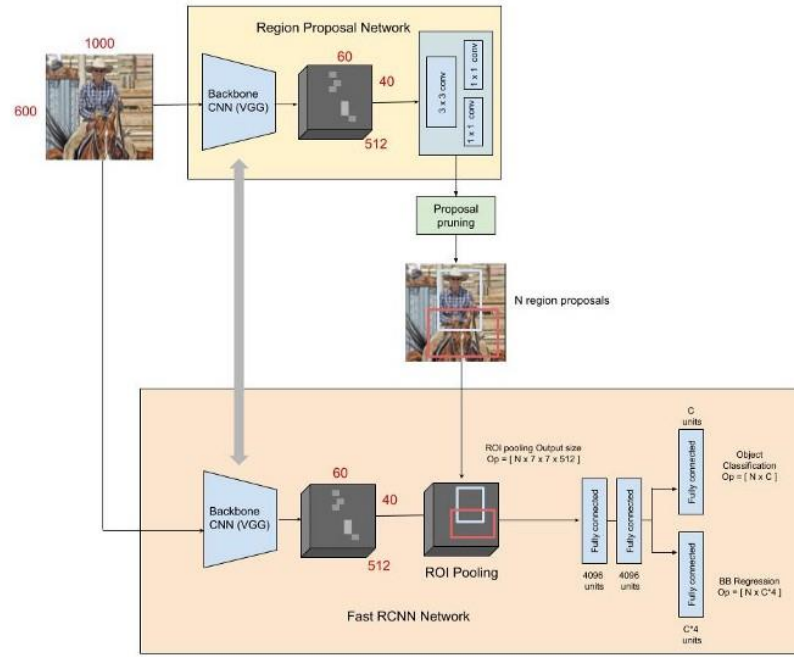


Figure 2.4 Faster R-CNN detection pipeline (RPN + Fast R-CNN). Image taken from (Ananth, 2021).

#### 2.2.3.4 Mask R-CNN

Mask R-CNN is another upgrade from Faster RCNN proposed by the Facebook AI Research group, which is conceptually more flexible, simple, and general, for instance segmentation. In addition to the existing branch for bounding box identification, it expands Faster R-CNN by adding a branch for predicting an object mask. MRCNN uses the Faster R-CNN's two-stage technique, with an RPN in the first stage, a binary mask for each RoI in the second stage, and the class and box outputs. It uses the Fast R-CNN's bounding box classification and regression in tandem, dramatically simplifying the original R-CNN's multi-stage pipeline. As claimed by the authors, this model outperformed all performance evaluated for the 2016 COCO challenge. The simple model concept is shown in Figure 2.5.

As the almost highest improved version of the R-CNN model, MRCNN has been used in many fields, especially in medical images. It has been used to automatically segment the nucleus with ResNet (Johnson, 2020) and U-Net (Vuola et al., 2019) under various

conditions. To diagnose early skin cancer, a transfer learning based on a sequence of chain processes initiated with MRCNN to segment the lesion; the resultant images are later passed to the DenseNet model for feature extraction and forwarded these extracted feature vectors to an entropy-controlled least square SVM (LS-SVM) model (Khan et al., 2021). For gastric cancer diagnosis, MRCNN has been used as an auxiliary method with a ResNet-101 backbone (G. Cao et al., 2019) and mask attention block (Deng et al., 2021) sequentially improved way. This model has also been used in medical image processing, such as Lung Nodule segmentation (FPN + ResNet101 as a backbone) (Liu et al., 2018), liver segmentation with holistically nested edge detection MRCNN (HED-Mask R-CNN) (Mulay Supriti and Deepika, 2020), 3D visualization diagnosis for Pulmonary Nodule (RPN + ResNet50) (L. Cai et al., 2020), tooth detection and segmentation (G. Zhu et al., 2020), Pancreas segmentation in CT image (MRCNN + 3D U-net) (Dogan et al., 2021), Femoral Cartilage segmentation from Knee Ultrasound Images (MRCNN + ImageNet) (Kompella et al., 2019), automated blood cell (Red & White) counting and classification (Dhieb et al., 2019), breast tumor detection (Chiao et al., 2019).

Along with medical images, MRCNN has been used in various agricultural fields. To detect apple leaf disease (Rehman et al., 2021), a real-time parallel framework of MRCNN and ResNet-50 has been used for segmentation and classification. Initially, they used a hybrid contrast enhancement method to improve the resolution of low-quality leaf images. A similar concept has been used to detect the ripeness of tomatoes by using fuzzy MRCNN (Huang et al., 2020) and to segment apple growth (D. Wang & He, 2022) using a fusion of attention modules into the backbone network. Another researcher used MRCNN with Resnet + FPN + RPN to detect the strawberry as rip or unrip in a uni structural environment (Yu et al., 2019). They successfully analyzed the shape and picking point for their strawberry harvesting robot. MRCNN is also used in Livestock (Qiao et al., 2019; B. Xu et al., 2020), counting from quadcopter images.

Although many researchers widely use Mask RCNN in various fields, it still has some updated versions. A Boundary-preserving Mask R-CNN (BMask R-CNN) (Cheng Tianheng and Wang, 2020) has been proposed to overcome issues like object boundaries and shape, leading to coarse and indistinct mask prediction results and imprecise localization. This model outperforms MRCNN on the COCO and Cityscapes datasets. Mask R-CNN ignores the disparity in spatial information between receptive fields of different sizes. A large-scale receptive field is more concerned with specific information, whereas a small-scale receptive field is more concerned with semantic data. As a result, the network cannot consider the relationship between the pixels at the object border, causing these pixels to be misclassified. The Mask-Refined R-CNN (MR R-CNN) (Y. Zhang et al., 2020) is a proposed method for adjusting the stride of ROI Align (region of interest align).

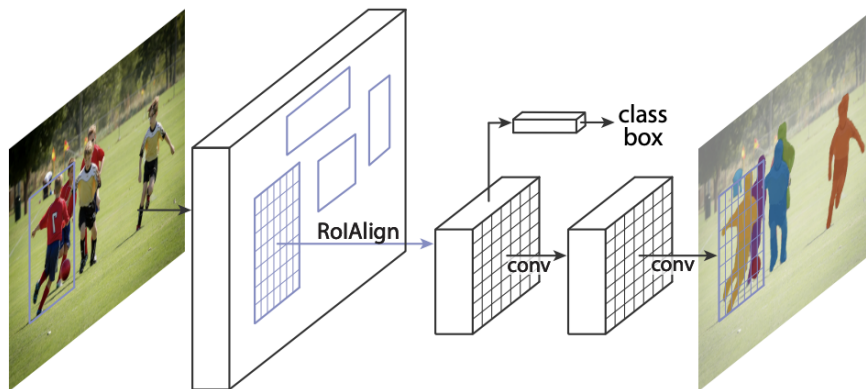


Figure 2.5 Simplified principal architecture of Mask R-CNN. Image taken from (K. He et al., 2017).

#### 2.2.4 Models with Transformer

Transformers have emerged as a groundbreaking paradigm in the field of computer vision, revolutionizing instance segmentation. Unlike traditional CNNs, transformers rely on self-attention mechanisms to capture global context and spatial relationships, making them well-suited for tasks that require fine-grained object delineation, such as instance segmentation. This shift in architecture has led to remarkable advances in the accuracy

and efficiency of instance segmentation algorithms, enabling the precise identification of individual objects within an image. By allowing models to learn complex, long-range dependencies and semantic relationships, transformers have propelled the state of the art in instance segmentation, opening the door to a wide range of applications in areas like object recognition, autonomous vehicles, and medical imaging. Many researchers are now using transformers as underlying architecture to propose their model in the field of instance segmentation as discussed below.

In deep learning models, usually end-to-end paradigms improve the detection accuracy, but for instance segmentation this was not possible due to higher output dimensions. To solve this, Jie Hu et al. proposed an instance segmentation transformer (ISTR) (Hu et al., 2021) which is the first end-to-end model of its kind. This framework achieved 38.6 mask AP using ResNet50-FPN backbone on MSCOCO dataset. Similar concept has been used for video instance segmentation (VIS) using transformer named VisTR (Y. Wang et al., 2020). This model uses similarity learning from same perspective, for instance segmentation and tracking. An inter-frame communication transformer (IFC) (Hwang et al., 2021) was used to further improve instance segmentation task on videos with fast runtime by efficiently encoding the context within an input clip, reducing information-passing overhead. They achieved an AP of 44.6 on YouTube-VIS 2019 dataset. Another model named SeqFormer (Wu, Jiang, et al., 2022) achieved 47.4 AP with ResNet-50, and 49.0 with ResNet-101 by locating an instance in each frame and aggregates temporal information to learn a video-level instance representation. Tianheng Cheng et al. proposes Sparse Instance Segmentation or SparseInst (Cheng et al., 2022), a novel framework for real-time instance segmentation that uses instance activation maps (IAM) to highlight informative regions for each object. SparseInst adopts single-level prediction, fully convolutional design, and simple post-processing without NMS or sorting. This model achieves 37.9 mask AP on COCO test-dev with 40.0 FPS, outperforming most state-of-

the-art methods for real-time instance segmentation. Shusheng Yang et al. proposed a model named Temporally Efficient Vision Transformer (TeViT) (S. Yang et al., 2022), a temporally efficient vision transformer for VIS. TeViT uses a messenger shift mechanism to fuse frame-level features and a spatiotemporal query interaction mechanism to build correspondence between video instances and queries. TeViT achieves state-of-the-art results on three VIS benchmarks with high inference speed. For offline VIS, a novel model named VITA (Heo et al., 2022) has been proposed that uses object tokens to associate frame-level objects. VITA uses an image object detector to distill object-specific contexts and a transformer-based model to build relationships between objects. It achieves state-of-the-art results on same three video instance segmentation benchmarks as TeViT with practical advantages such as handling long and high-resolution videos, freezing a frame-level detector, and fast convergence. A query-based framework named FastInst (J. He et al., 2023) was proposed for real-time instance segmentation which uses three key techniques: instance activation-guided queries, dual-path transformer decoder, and ground truth mask-guided learning. This model dynamically picks the pixel embeddings with high semantics from the underlying feature map as the initial queries for the transformer decoder. These queries contain rich information about potential objects and reduce the iteration update burden of the Transformer decoder. FastInst achieves state-of-the-art results on COCO and Cityscapes benchmarks with high inference speed and practical advantages. To achieve high-quality masks, such as segmentation, a new method known as PatchDCT (Wen et al., 2023) has been introduced, which addresses the difficulty of refining global Discrete Cosine Transform (DCT) vectors used to compress high-resolution images. They also highlight the limitation of the binary grid mask, which results in high training complexity and performance degradation. This model divides the decoded mask from DCT into multiple independent patches and refines each path using three class classifiers and a regressor. The classifier identifies and



corrects foreground and background patches, while the regressor predicts mixed patch DCT vectors containing boundary information. The paper claims that patching allows the model to focus on the refinement of local regions, fully exploiting the advantages of multi-stage refinement and high-resolution information compression.

Ke et al. (Ke et al., 2022) proposed Mask Transfiner, a novel method for high-quality and efficient instance segmentation that uses a quadtree and a transformer to improve mask quality and efficiency. It splits and encodes image regions as a tree of nodes, saving computation and memory. It also corrects errors in the nodes using self-attention and multi-scale feature fusion, beating other methods on three benchmarks: COCO, BDD100K, and Cityscapes. Along with offline models, an online framework for video instance segmentation based on contrastive learning and optimal transport is proposed termed IDOL (In Defense of OnLine) (Wu, Liu, et al., 2022). The framework aims to improve the quality of instance embeddings and association across frames and shows that the proposed method outperforms existing online and offline methods on three benchmarks, especially on complex videos with occlusion and deformation though it does ablation studies on different components of the contrastive head and the inference strategy.

Transformer models have been shown to be effective in instance segmentation tasks. They can refine the segmentation of sparse error-prone regions in the spatio-temporal volume, which is useful for high-quality and temporally consistent masks for moving objects. However, the computational and memory efficiency of the video transformer is also a concern (R. Sharma et al., 2022). Also, transformers still suffer from poor small object detection and unsatisfactory edge detail segmentation. In summary, transformer models are a promising approach for instance segmentation, but they still have some limitations that need to be addressed (T. Lin et al., 2022). For this dataset, along with Mask and

BMask R-CNN, two transformer models named FastInst and PatchDCT has been trained to compare results.

### **2.2.5 Backbone Network**

In the field of deep learning, both backbone models and pretrained models play crucial roles in the development and deployment of various machine learning tasks, particularly in computer vision. Backbone models refer to the architectural framework or structure of a neural network, typically comprising multiple layers such as convolutional and pooling layers, designed to extract hierarchical features from input data. These backbone models serve as the foundation upon which specific neural network architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), are built for specific tasks like image classification, object detection, or semantic segmentation. On the other hand, pretrained models are neural network models that have been trained on large-scale datasets, such as ImageNet, using a specific backbone architecture. These pretrained models learn generalizable features from the dataset during the training process, capturing complex patterns and representations that can be transferred or fine-tuned to new tasks or domains with minimal training data. While backbone models define the structural framework of a neural network, pretrained models leverage the learned representations from large-scale datasets to expedite training and improve performance on downstream tasks, making them indispensable tools in the field of deep learning and computer vision. The following subsections discuss the two most popular backbone networks that are used mainly in case of instance segmentation in which ResNet has been used as a backbone network in this thesis work.

#### **2.2.5.1 VGG-net**

The weighted 16-layer VGG-16 architecture (shown in Figure 2.6), based on CNN and has about 138 million parameters, won the ILSVR competition in 2014. It follows a straightforward construction of the max pool layer of 2x2 stride 2 and convolution layers

of 3x3 filter with stride 1 and similar padding. It has two fully connected layers at the end, followed by a SoftMax for output. This model has been used as a backbone network on Fast R-CNN (Girshick, 2015), Fast R-CNN + (ASDN + ASTN) (X. Wang et al., 2017), Faster R-CNN (X. Chen & Gupta, 2017) and CAFR-CNN (Guo et al., 2021). One of its major drawbacks is that the VGG16 network is extensive and requires more time to train its parameters. The depth and quantity of completely connected layers in the VGG16 model make it larger than 533MB. This lengthens the process of putting a VGG network into practice.

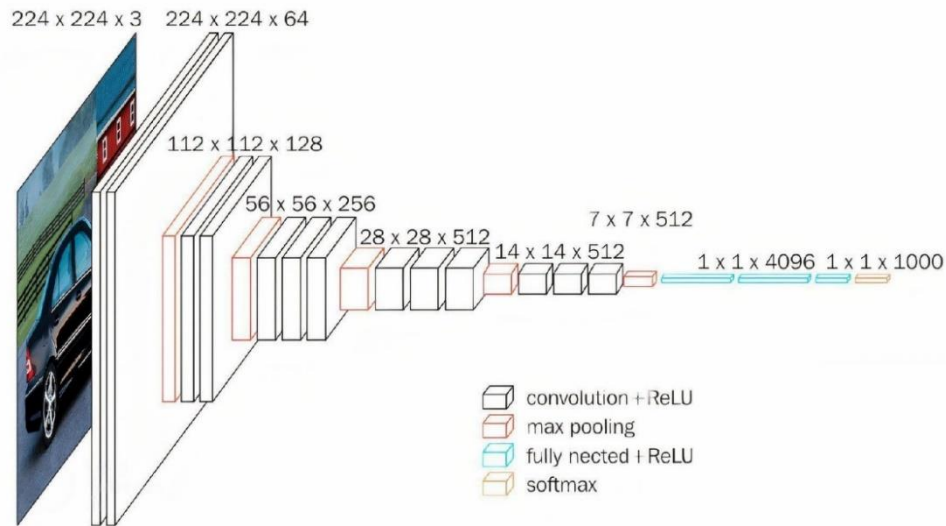


Figure 2.6 VGG-16 network architecture. Image taken from (Hassan, 2021).

### 2.2.5.2 ResNet

Since AlexNet, the cutting-edge CNN architecture has become increasingly complex due to added layers to its successor models. The VGG network and GoogleNet contained 19 and 22 convolutional layers, in contrast to AlexNet's five layers. Because of the well-known vanishing gradient problem, it is challenging to train deep networks because, as the gradient is backpropagated to older layers, repetitive multiplication may make the gradient infinitesimally small. As a result, the network's performance becomes saturated or declines quickly as it penetrates further. ResNet's central concept is to introduce a link

that bypasses one or more layers, called an "identity shortcut connection." Highway Network (Srivastava et al., 2015), which offered gated shortcut connections, used shortcut connections before ResNet. These parameterized gates control the amount of information that can pass through the shortcut. A parameterized forget gate in the Long-Term Short Memory (LSTM) cell, which regulates how much information flows to the following time step, uses a similar concept. ResNet can, therefore, be viewed as a particular instance of Highway Network. The general architecture of ResNet is illustrated in Figure 2.7.

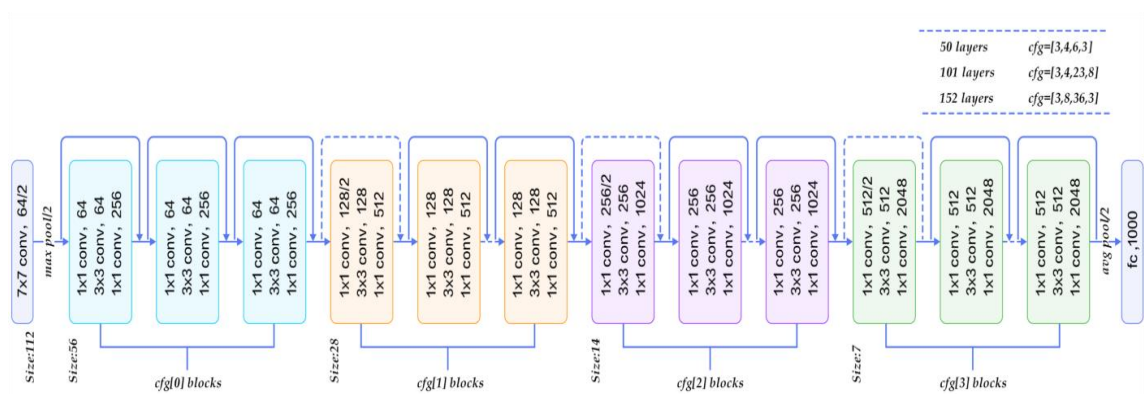


Figure 2.7 ResNet architecture. Image taken from (Rastogi, 2022).

## 2.3 Object Tracing

After the object segmentation, the focus is on how to trace them throughout their movement around the combustion area within a specific timeframe. Liquid droplet tracing has been hardly seen in the current literature study. Correspondence-based droplet tracing has been done by calculating displacement between two laser pulses by keeping one object as a reference point (Zhan et al., 2021). This procedure may work on constant movement-based tracing but will not work if objects are moving independently without maintaining the reference pointer location. A similar concept has been used in another droplet tracing case in agricultural spraying systems (Acharya et al., 2022). They used a custom model architecture to train and detect the droplets from the crop-spray system,

and for tracing, they used a reference droplet point with the same issue as the previously mentioned work.

A YOLO + DeepSORT (Durve et al., 2022) method was proposed for droplet tracing across sequential images by assigning them with a unique ID and predicting the future position of a particular object to avoid identity switches. They compared different YOLO (Durve et al., 2023) models to find out the best detection algorithm, as DeepSORT totally depends on the output of YOLO's detected objects. This model can trace well depending on how dense the objects are and how well the object detection model works. Using YOLO along with the Hungarian algorithm, a method was proposed to track objects in walking droplet and granular intruder experiments (Kara et al., 2023). But they also used similar plain, simple images, resulting in good object detection with YOLO and one directional smooth movement view. As no satisfactory relevant dense object tracing model or algorithm has been found, this thesis will focus on implementing a self-developed algorithm to trace objects.

## **2.4 Summary**

From the above sub-sections about droplet detection and tracing, it is notable that the importance of droplet detection is high for achieving different goals like improving atomizer, spray nuzzle, medical image understanding, etc. Deep learning models like YOLO, region-based CNN, and customized CNN with popular backbone like ResNet networks were primarily used for detection. DeepSORT is the latest model for droplet tracing found in the literature, which also depends on an object detection model and cannot trace dynamic objects in case of class transition. A summary of mentioned references is shown in Table A.1. which represents a comprehensive overview of object segmentation models, highlighting their diversity in approaches and performance across different evaluation metrics. Various models including R-CNN, Cascade R-CNN, and Mask R-CNN are listed, showcasing a spectrum of methodologies employed in object

segmentation tasks. Performance metrics such as mAP and accuracy are provided, offering insights into the effectiveness of each model. The evaluation is conducted on diverse datasets such as PASCAL VOC, Cityscapes and MS COCO, illustrating the applicability of these models to different image datasets. Furthermore, the applications of these models range from core object detection to specific tasks like lung nodule segmentation and gastric cancer detection, indicating their versatility in addressing various challenges in computer vision. Additionally, the table details the backbone networks utilized in these models, with architectures like ResNet and VGG supporting their functionalities. Overall, the table serves as a comprehensive summary of object segmentation models, emphasizing their performance, versatility, and potential applications in the field of computer vision.

### CHAPTER 3: METHODOLOGY

In liquid atomization process, quantitative analysis of object count is important to know because from this analysis, the engine performance or the combustion process can be understood. Multiple ways are available to directly count the generated objects through an atomizer, for example, PDA, PSD and PDPA as sensor-based methods. One major problem with sensor-based methods is that one must build the atomizer and test it, also it is prone to missing the overlapped objects. Instead of building the physical system to test, another popular method is to simulate the atomization mechanism and then detect the objects using computer vision techniques. It is easy to generate the simulation and can make changes based on outcome very easily compared to physical systems. That's why this thesis used simulated dataset and computer vision for object detection and object's path tracing.

The first step of detecting objects using computer vision is to prepare a suitable dataset that represents the target objects. This involves acquiring images or videos that contain objects of interest and annotating them with relevant information. For example, in medical image analysis, cells or anatomical structures might be labeled by experts in the field. If the objects are densely populated and comparatively large in number, it might be difficult to label them manually. So, a complete labelling may not be available in some cases. To make the situation worse, it is also possible to not have enough images to train a deep learning model. In these complex cases, image partial labelling and augmentation could help to overcome the lower image number and labelling issue. Once dataset related problems are resolved, next step would be to choose a process to detect objects. Various possible ways include object detection models, object segmentation models, color differentiation, displacement calculation and so on. Object detection process mostly depends on the expected outcome and required object's characteristics. If the dataset is

related to real life traffic scenario and goal is to detect vehicles in real time, the important thing would be to consider detection speed and vehicle location. It won't be necessary to detect pixel level accuracy which would take more time in inference. Based on the priority, in droplet detection procedure, it is necessary to detect object location in pixel level accuracy to know their location, breakdown process, tracing and shape leading to object segmentation. The movement of objects is necessary to trace to know the displacement in a given time. For example, an autonomous vehicle needs to continuously trace surrounding vehicles to avoid accident. In atomization process, object tracing is necessary to know the droplet breakdown density in a given time and location. The core principle of object tracing is to calculate the displacement of an object in corresponding next image frame. The challenge would be to figure out if the object is same in the next frame considering any overlapping happens in the process. Some deep learning models are being introduced recently like DeepSort that can trace a marked object throughout frames, but this is limited to single object tracing and in case of object breakdown, model fails to keep track of the separation.

Instance segmentation has been chosen for detecting objects as usual object detection only gives us coordinates of bounding boxes and not the object's shape. From the literature, it has been observed that Mask R-CNN is to date one of the most popular and widely used architecture for object detection and segmentation. So as a baseline model, it has been utilized for data training. As we are dealing with relatively small and dense objects, it is important to preserve the boundaries of each object so that overlapping issues can be addressed automatically. For doing so, another modified and updated version of Mask R-CNN has been used, named BMask RCNN. It has also been observed that, in recent years, transformer-based models are gaining popularity, and a wide range of applications are utilized by many researchers. So, to compare the results, two transformer-based models are also trained; named FastInst and PatchDCT. After object detection, it is also necessary



to know which path does object followed throughout its lifetime as well as how many objects changed its assigned classes and converted into lower classes. A custom method has been developed using graph algorithm to trace objects path. So that we can understand more about the atomization process. Fluid flow can cause object shape to change over time, so it is also possible for one object to be classified as different class once it changes its shape. This class transition has also been calculated, as fuel atomization focuses on how fast liquid fuel can convert to tiny droplets.

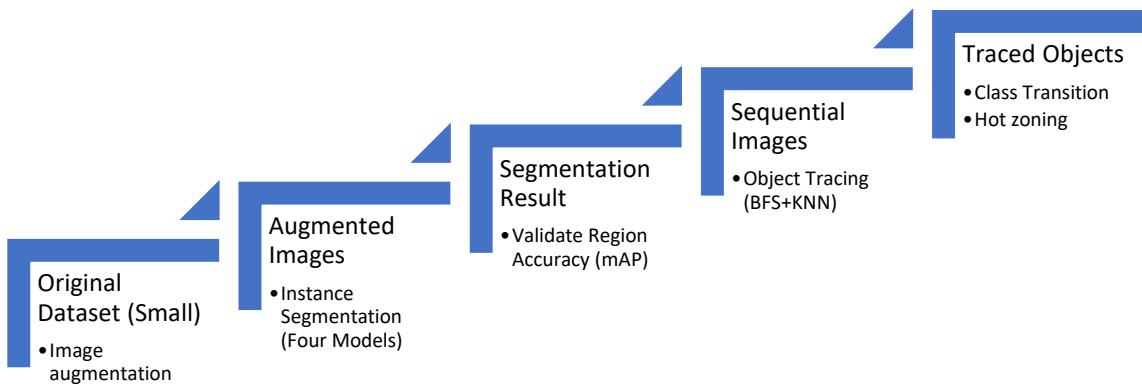
The methodology of the research involves a series of sequential steps aimed at achieving accurate object tracing and class transition analysis in a given dataset. Initially, this research utilizes an original dataset, a bit small in scale, as the foundation for subsequent analysis. To enhance the diversity and robustness of the dataset, image augmentation techniques are applied, resulting in an augmented dataset with a broader range of variations. Another purpose of dataset augmentation is to achieve a more general and nonexclusive image, so that trained model or similar procedure can later be used for another similar use case serving our first research objective (RO1). This augmented dataset serves as the input for the instance segmentation process, employing a Mask R-CNN, BMask RCNN and two Transformer models to precisely delineate object boundaries and generate segmentation results. The accuracy of these segmented regions is then validated using mean Average Precision (mAP) metrics, ensuring the reliability of subsequent analyses, focusing on our second objective (RO2). Later sequentially captured images are then used for object tracing, utilizing Breadth-First Search (BFS) and customized K-nearest neighbors (KNN) algorithms to trace the movement trajectories of objects across frames. The traced objects are further analyzed for class transition, examining objects transition between different classes over time, serving our final research objective (RO3). Additionally, a hot zoning analysis is conducted to identify areas of significant activity or change within the traced objects, providing insights into

object movements and breakdown. Through this comprehensive methodology, the research aims to achieve reasonable object tracing and class transition analysis, leveraging image augmentation, instance segmentation, and sequential image analysis techniques to extract valuable insights from the dataset. The steps of the working procedure are illustrated in Figure 3.1, (a) and a flowchart explaining the unified methodology targeting these steps are shown in Figure 3.1 (b).

Designing the methodology faces some technical challenges, starting with selected Mask R-CNN model, which is a comparatively old model using previous python, TensorFlow and CUDA versions. It was challenging to update the code to support latest python, TensorFlow and other libraries to run on CUDA 11.2 having RTX2070. This model was chosen for its well know reputation but being an old model architecture, it requires relatively large amount of time and memory to train the dataset making it computationally expensive model than others that are used though it worth the risk taken. After training it was upsetting that model was not able to detect expected number of objects. So, a new challenge came to address the issue where a divide and conquer technique was used to reduce object density by cropping the image into multiple small part. By comparing different models, it was noticed that slow doesn't mean bad result and fast doesn't mean better results are available. In this project opposite scenarios were observed in object detection rate. Technically it was challenging to choose the models based on their reported benchmarks. Also, it was difficult to know the suitable number of iterations that one model could have as different architectures were giving different results on same iterations. After getting object detection result, initial challenge was to somehow store the detected object properties for object tracing part, as inferencing on 1500+ images was taking relatively large amount of time, it was not possible to inference and trace at the same time. Another challenge is faced when calculating the corresponding objects of

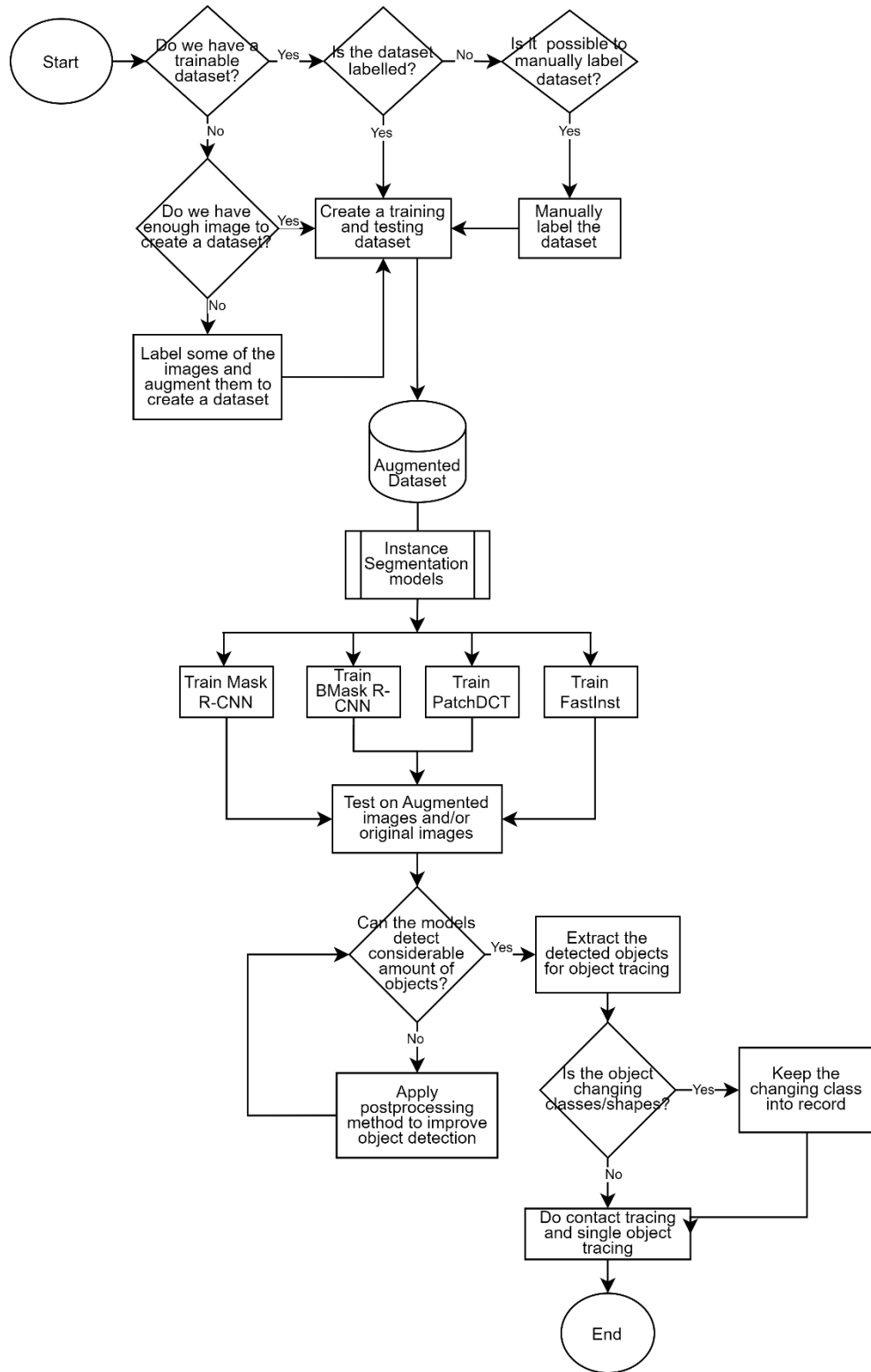
every other frame and store those objects' current state. Finally, graph algorithms were customized to search into the corresponding objects pool and trace the movement.

This chapter discusses research methodology, starting with the dataset in section 3.1. Initially, the dataset size was relatively small, and data augmentation was necessary, as discussed in section 3.2. Mask R-CNN, BMask R-CNN and transformer based model trainings on this dataset have been explained in sections 3.3, 3.4 and 3.5. A proposed image post-processing technique has been mentioned in section 3.6. Object tracing is described in section 3.7, model evaluation matrices are mentioned in section 3.8, and finally, this chapter is summarized in section 3.9.



(a) Steps of working procedure.

Figure 3.1 (a) Steps of working procedure and, (b) flowchart explaining the unified methodology focusing on these steps.



(b) Flowchart of unified methodology

Figure 3.1, continued; (a) Steps of working procedure and, (b) flowchart explaining the unified methodology focusing on these steps.

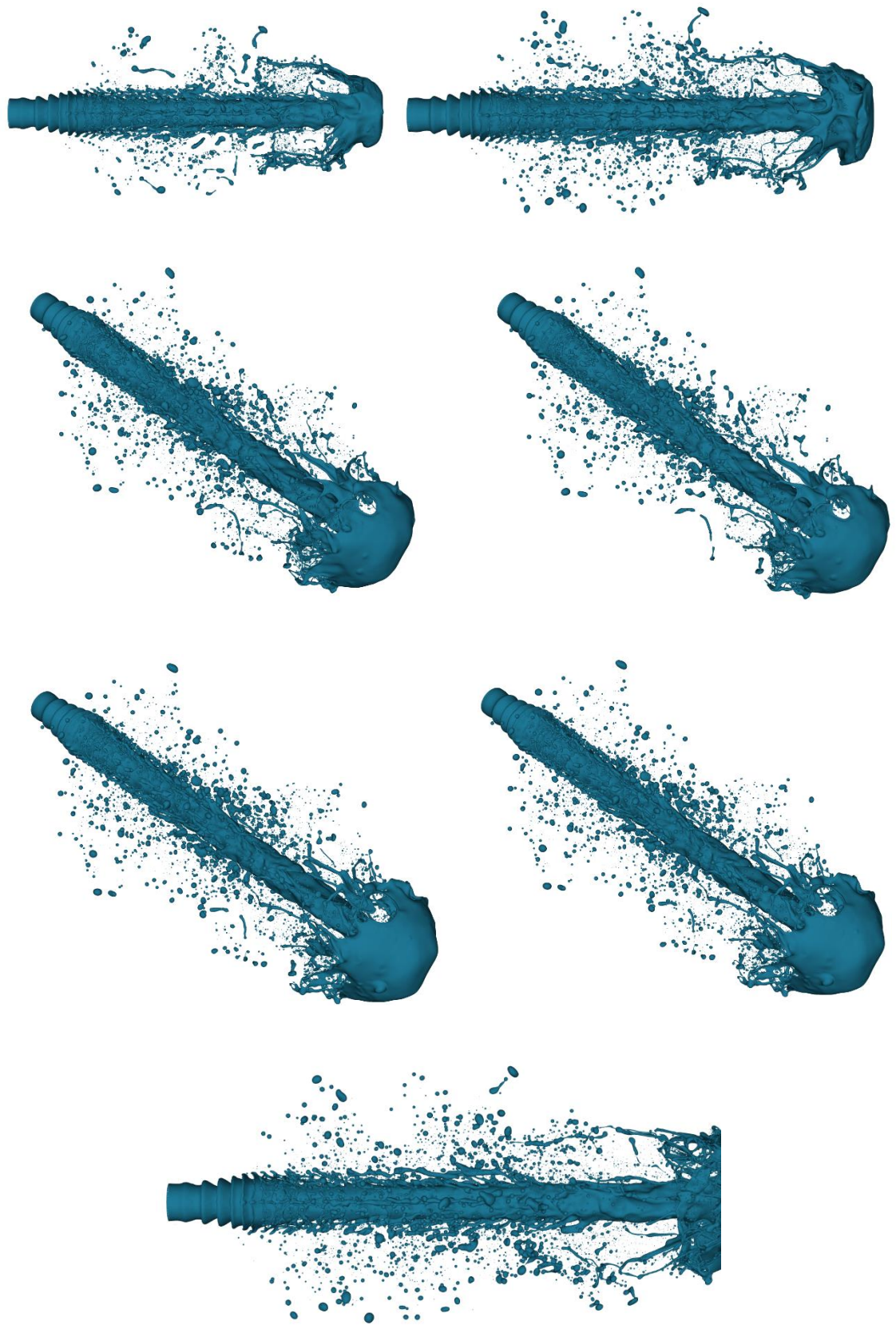
### 3.1 Dataset

Knowledge on how liquid jets break up and form sprays is essential for developing air-breathing propulsion systems. This applies to conventional engines like gas turbines, compression ignition engines and more advanced concepts like rotating detonation engines or scramjets for hypersonic propulsion. The spray formation process starts with injecting a liquid fuel column into a turbulent environment, where the presence of large density, temperature, and composition differences leads to the growth of non-linear surface instabilities. These instabilities eventually cause the liquid column to break, which is known as the primary breakup (Gorokhovski & Herrmann, 2008). The challenges hinder the progress in understanding primary breakup in spray formation in visualizing the optically dense spray region. Existing measurement methods are limited due to the reflection of visible light by the thick liquid core and dense cloud of droplets. Experimental techniques like ballistic imaging (Linne, 2013) and x-ray-based (Kastengren & Powell, 2014) methods have shown promise. Still, they cannot provide comprehensive four-dimensional information for detailed analysis (Bothell et al., 2020). As a result, developing a comprehensive theory for primary breakup remains a challenge, leading to limited predictive accuracy in design approaches. The lack of understanding of primary breakup hinders modeling air-breathing combustion systems. However, with advancements in computer algorithms and access to big data, researchers can now use computer simulations and machine learning techniques to investigate primary breakups.

This experiment works with an unlabeled small dataset that has been generated from high-fidelity direct numerical simulations (DNS). This DNS formulation has already been verified and used to study different types of multiphase flow issues like primary (Notaro et al., 2019) and secondary (Redding & Khare, 2022) breakdown causing droplet collision (X. Chen et al., 2011; Ganti, Khare, et al., 2020). Those results were also used to train

machine learning models to predict multiphase spatiotemporal flow fields (Ganti, Kamin, et al., 2020).

That simulation primarily generated only seven unevenly shaped, unlabeled images focusing on four classes: droplets, lobe, attached ligament, and detached ligament. As the dataset is unlabeled, the first priority was to label the objects as much as possible. A third-party tool named VIA (Dutta & Zisserman, 2019) has been used to label the objects as polygon points to get approximately 30 objects per image of four classes and those points were extracted into a JSON file for further analysis. All seven original images are shown in Figure 3.2 (a) and one sample image from the initial seven images is shown in Figure 3.2 (b), pointing to the classes. On the fluid body, the backward portion where the flow looks like waves is called the lobe; long objects connected to the main body called attached ligaments, detached from the main body called detached ligaments and tiny circular objects are called droplets. As the flow continues, droplets are generated from all over the body, including ligament breakdown. The simulated images are in RGB color spectrum, and two kinds of shapes are seen: the horizontally aligned spray image shape is 1200x600 pixels, and diagonally aligned images are 1600x1200 pixels. Initially, seven images were generated to test the model, and later, 1573 images were generated targeting the object tracing process. The image augmentation has been done on the original seven images and detailed process is discussed in next sub-section.



(a) Original seven images

Figure 3.2 (a) Showing all seven images and (b) Classes are pointed out in one image.

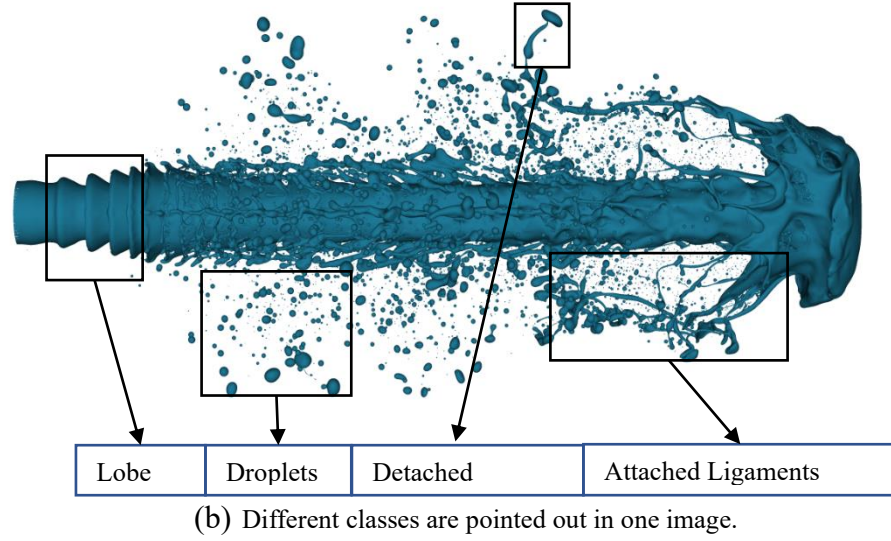


Figure 3.2, continued; (a) Showing all seven images and (b) Classes are pointed out in one image.

### 3.2 Data Augmentation

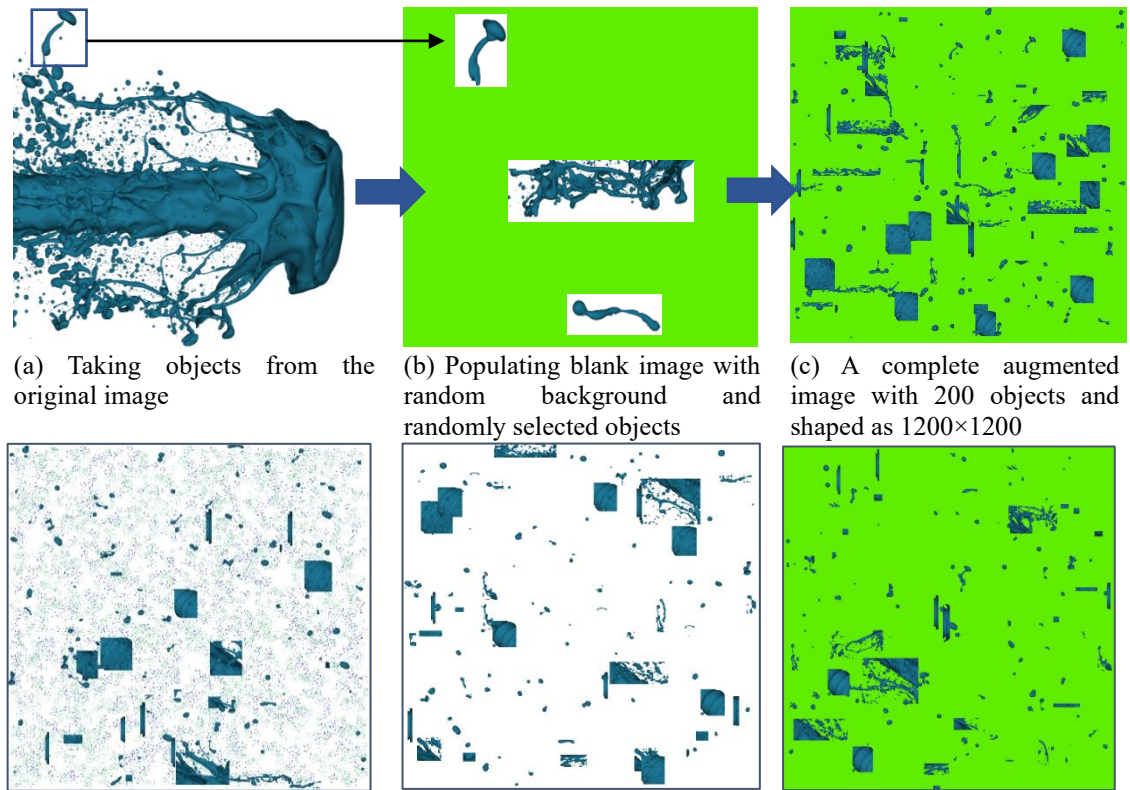
As four deep learning models have been chosen to train the dataset, having only seven images were too small to consider the training. So, image augmentation was almost mandatory. Instead of just rotating the image and creating typical changes in the visual aspect, a custom-intensive augmentation has been done to generalize the dataset and complicate the training process so that models are not overfitted, and the differences between classes are understood well. A new image set was created for augmentation based on cropped objects from the original seven images. Five background images were used, which have a plain white color, three different shades of light green, and one noisy white color. Another set of five types of noisy image backgrounds with different shapes are used for noise.

To begin with image augmentation, a blank 1200x1200 pixel image was taken with a randomly selected background and noise. An object pool from the original image set has been created with all the positional (x, y) parameters of the labeled objects and properties like class. From this object pool, randomly selected objects were put in random places in the prepared augmented image background, with 100 to 200 objects per image. Following

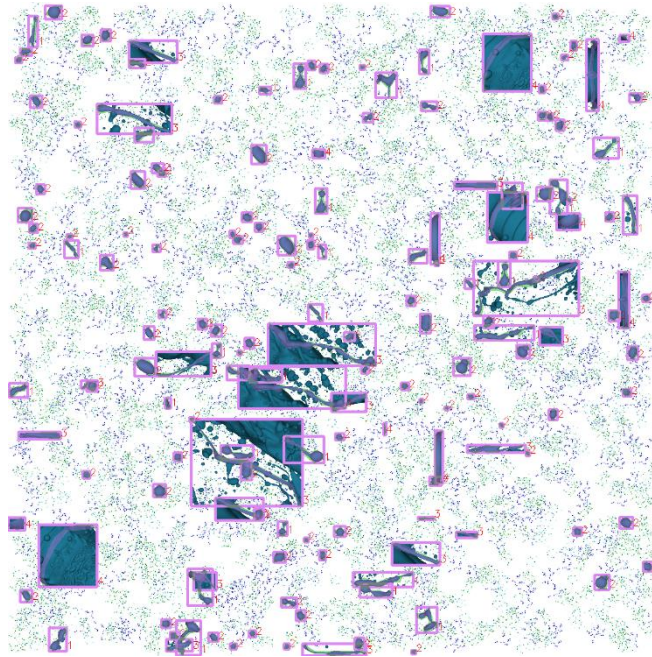


this procedure, a total of 1200 images were generated, of which 1000 images were used for training and 200 for testing. The process has been illustrated in Figure 3.3.

This augmentation also solves the auto-labeling issue, as now 1200 images have fully labeled objects, which completes the second-mentioned objective. All the information regarding the dataset and object labels was stored in a JSON file for subsequent processing in the model training. Finally, a visual check was done to ensure that the object position, label, and class were correctly stored. This dataset generation completed the first mentioned objective. After the dataset is generated, Mask R-CNN has been trained on it, later BMask R-CNN was trained for possible improvement opportunities, and finally two transformer models were trained for further experiment.



(d) Sample augmented images from test data with three different backgrounds.



(e) Verifying if object label, position and class are stored correctly.

Figure 3.3 Image augmentation procedure in (a), (b), (c), some sample images are shown in (d) from augmented image dataset, and finally, visual verification has been done in (e).

### 3.3 Training with Mask R-CNN

Mask R-CNN is a special kind of deep neural network that aims explicitly to do instance segmentation in computer vision tasks. From a general point of view, for one input image, this model can extract each object's bounding box coordinates, high-quality masks, and classes separately. Its architecture has two stages; the first stage extracts the proposals about the object's region using a region proposal network (RPN), such as which areas have the highest probability of having an object in it. The second stage predicts the class of that object and refines its positional coordinates or bounding boxes. Finally, it generates a mask of pixels based on the first stage result. Each stage gets its features from a backbone network that extracts them using a feature pyramid network (FPN). In the first stage, RPN scans those FPN outputs to get the proposal areas using region of interest (ROI) or another word called anchors, by which RPN can understand the object's size and coordinates using some intersection over union (IoU) values. In the second stage, previously selected regions are pooled using another neural network and assigned to a fixed-size feature map area from which RoIAlignment is done to choose relevant feature positions. From a side branch, object masks are generated at a pixel level, and object class and bounding boxes are generated from a fully connected layer. The simplified model architecture is shown in Figure 3.4. A similar technique like ref. (Abdulla, 2018) was followed to prepare the dataset for training with an input shape of  $1200 \times 1200$  pixels. The classes were labeled as 0-4 (0: background, 1: detached ligaments, 2: droplets, 3: attached ligaments, 4: lobe), including the background. The ground truth masks of objects were generated from the polygon labels. The default Mask R-CNN model input shape is  $1024 \times 1024$ , and it automatically resizes the input image to this shape, but the output maintains the input shape. Two ResNet (Resnet-50 and ResNet-101) backbone networks were used for training and an initial 1000 epochs with 100 steps per epoch and a 90% confidence has been maintained. After comparing the results between two ResNet

backbones, it was decided to continue with ResNet-50 as this showed faster training with relatively better results. A total of 5791 epochs was done with ResNet-50, and after training, three model weights at 1000th, 2791st, and 5791st were selected randomly to compare the result and progress with tracing. The original Mask R-CNN is incompatible with TensorFlow 2.x and CUDA 11, so coding adjustment was done for the training model and finally used Python 3.8, TensorFlow 2.6, CUDA 11.2 with RTX2070 with 16GB RAM on the Ubuntu system. Instead of running from the beginning, a pre-trained weight of Mask R-CNN on the MS COCO dataset has been used.

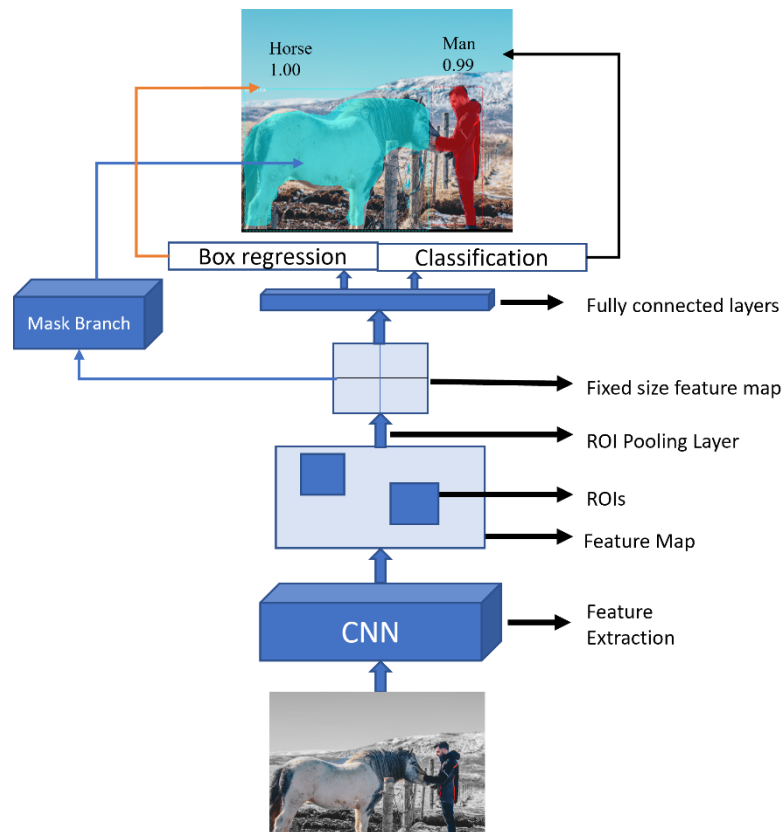


Figure 3.4 Simplified architecture of Mask R-CNN

### **3.4 Training with BMask R-CNN**

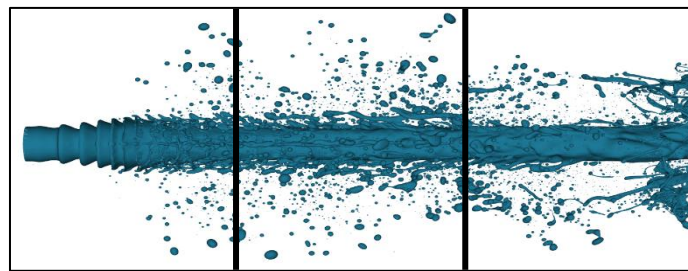
Boundary preserving Mask R-CNN or BMask R-CNN is an improved version of Mask R-CNN where an object's mask and bounding box are learned together through a feature fusion network. This way, the object's mask values are better adjusted with the bounding box. Facebook's detectron2 image processing library has been used along with BMask R-CNN configuration for training the data with pre-trained weight on MS COCO. Three different configurations (Base-RCNN-FPN, BMask RCNN-101-FPN-3x, Cascade BMask R-CNN R-101-FPN-1x) were used to train with this model with 1000 and 3000 epochs each for comparison. Base-RCNN-FPN consists of the BoundaryROIHeads layer in ROI\_HEADS and the BoundaryPreservingHead layer in ROI\_MASK\_HEAD. For BOUNDARY\_MASK\_HEAD, 28 pooler resolution with two convolutional layers has been used. For BMask R-CNN-101-FPN-3x config, predefined ImageNet weight has been used with ResNet-101. Cascade BMask R-CNN uses the CascadeBoundaryROIHeads layer in ROI\_HEADS along with ImageNet and ResNet-101. It also keeps the class agnostic bounding box regression with 2000 post-NMS top trains. On training, one image has been used per batch on a single Google Colab T4 GPU with a 0.02 base learning rate.

### **3.5 Training Transformers**

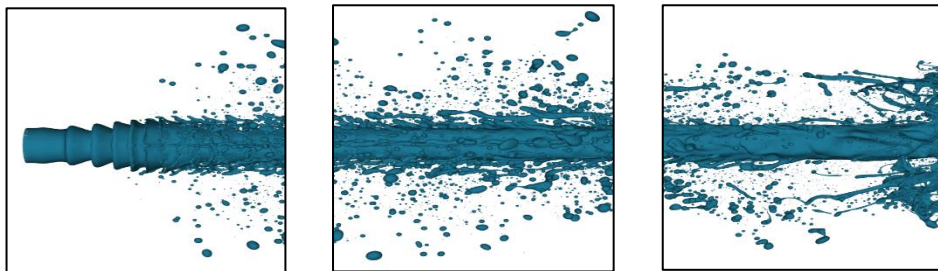
Similar configurations were maintained for both FastInst and PatchDCT and trained using detectron2 library with MS COCO's pretrained weight. ResNet-50 has been used as a backbone network with 0.02 base learning rate, 2000 pre-NMS and 1000 post-NMS on training, 1000 pre and post NMS for testing. A total of 20k epochs were trained with 100 checkpoint period. Unlike Mask or BMask R-CNN, transformer models were trained on another paid GPU machine having RTX 3090, where 20k iterations took approximately 8 hours.

### 3.6 Image Post Processing

As the dataset used in this thesis is about liquid spray, the density of objects or droplets is higher than the usual training dataset for MS COCO. By default, the detection model was not working well. The trained models were generating a high-quality mask of low number of objects, where the goal was to detect a high number of objects with a high-quality mask. A divide and conquer technique have been utilized by dividing images with different cropping window as the object detection rate is low. In this process, one image has been cropped in nine pre-defined window sizes as (width, height/3), (width/3, height), (width/3, height/3), (width, height/4), (width/4, height), (width/4, height/4), (width, height/5), (width/5, height) and (width/5, height/5). Dividing by different numbers refers to dividing the height or width in equal number of given portions (i.e. width/3 refers to dividing width into three equal segments, if no number is mentioned it means keeping the original size). After cropping the image, object density reduces per portion, and it is tested with previously trained models, showing a significant increase in object count. Later, cropped images were merged together to conquer the result. A sample cropping process is shown in Figure 3.5.

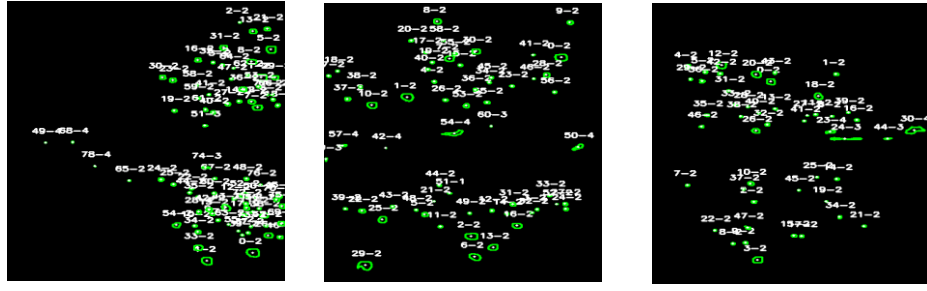


(a) Cropping image by (height, width/3) portion



(b) Three separate portions of the original image ready for individual inference.

Figure 3.5 Image cropping and testing procedure. Initiated with (a) cropping with given window size, (b) separating based on window size and (c) testing procedures of each individual section.



(c) Testing each portion (1<sup>st</sup> positional number is the object number, 2<sup>nd</sup> positional number is the class identification number)

Figure 3.5, continued: Image cropping and testing procedure. Initiated with (a) cropping with given window size, (b) separating based on window size and (c) testing procedures of each individual section.

After the image post-processing, all the objects' properties are extracted to a JSON file. To do so, some necessary steps are followed, as mentioned in section 3.6.1 and section 3.6.2 below.

### 3.6.1 Extract object mask using contour

Mask R-CNN model return masks result in a “pred\_masks” variable, a multilayer binary array of true/false pixel values (an object containing pixel is true, empty is false) of input image shape. Later, that 2D location\_coordinates (shown in Equation 3.1) array has been processed to have a polygon shape to put in a grayscale image (one mask per image), and the contour of that polygon is extracted using Python's OpenCV library. Along with the contour area, the center coordinates of that contour have also been calculated. This process is explained mathematically below. In Equation 3.1, location\_coordinates is a tuple containing row and column indices of pred\_masks's true values (r refers to row, c refers to column). Later, x and y are separated in Equations 3.2 and 3.3, and finally, the object's mask polygons are generated from Equation 3.4.

$$location\_coordinates = \{ (r, c) \mid pred\_masks[r, c] = True \} \quad 3.1$$

$$x = \{ r \mid (r, c) \in location\_coordinates \} \quad 3.2$$

$$y = \{ c \mid (r, c) \in location\_coordinates \} \quad 3.3$$

$$object\_polygon = \{ (x[i], y[i]) \mid i = 1, 2, \dots, N \} \quad 3.4$$

### 3.6.2 Realign objects coordinates

When images were cropped, the objects lost their original location information. After object detection, masks are being generated considering the image shape, and the coordinates of those masks are also based on the input image shape. As the input shape changed due to the cropping window index, it is necessary to realign the object's location based on the cropped window when merged after the final stage. A basic formula in Equation 3.5 has been followed for changing range, and later, using this formula, new x and y of the object's mask contour has been calculated in Equation 3.6 and 3.7. As the contour center point was extracted before, it also needs the range shift done by Equations 3.8 and 3.9, where m is the total contour count.

$$new\_value = newMin + (value - oldMin) * (newMax - newMin) / (oldMax - oldMin) \quad 3.5$$

$$x = newMin\_a + (data[i] - oldMin\_a) * (newMax\_a - newMin\_a) / (oldMax\_a - oldMin\_a) \quad 3.6$$

$$y = newMin\_b + (data[j] - oldMin\_b) * (newMax\_b - newMin\_b) / (oldMax\_b - oldMin\_b) \quad 3.7$$

$$cx = (\sum x) / m \quad 3.8$$

$$cy = (\sum y) / m \quad 3.9$$



After this procedure, all the relevant data is being saved in the following format:

```
Contour_list_array = {
  "image": name,
  "classes": classes,
  "ctr": [contour_list, center_x, center_y, class_value],
  "ctr-modified": [shifted_contour_list, shifted_center_x, shifted_center_y, class_value],
}
```

### 3.7 Object Tracing

After having all the relevant data for 1573 images about the object's location, mask, and class per image, it is time to explore object tracing options. The first step was calculating the object correspondence between two image frames and creating a list of probable connections within a range  $r$  (tested  $r = 20$  pixels). There are four classes to consider in object tracing. Still, as objects can break down from larger sizes to smaller particles, it is necessary to follow a rule for correspondence tracing. So, rules were applied to maintain a maximum distance limit ( $r$ ) between two object's centers as coordinates. The rule set is shown in Table 3.1, and the Euclidean distance between two points in a 2D Cartesian coordinate system formula has been shown in Equation 3.10.

$$\sqrt{(x1 - x2)^2 + (y1 - y2)^2} \quad 3.10$$

Table 3.1 Correspondence calculation rule.

Base Frame – a	Corresponding Frame – (a+1)
lobe	lobe, attach ligament, detach ligament, droplet
attach ligament	attach ligament, detach ligament, droplet
detach ligament	detach ligament, droplet
droplet	droplet

Having all the corresponding object data among all the images, a dictionary has been made connecting each frame with all the corresponding objects. With this dictionary,

tracing can be done for different images and objects in it. A BFS-based algorithm is written to search for a connection among the corresponding lists shown in Algorithm 1.

### **Algorithm 1:**

**Title:** Finding Tracing Indices

**Algorithm Description:** This algorithm finds an object's tracing indices based on a set of input data structures.

#### **Inputs:**

1. **object\_correspondence\_dictionary**: A data structure containing information about the nodes and their connections.
2. **index**: An integer representing the starting image index.
3. **total\_correspondence\_length**: An integer representing the maximum image length to trace the objects for.

#### **Outputs:**

1. **object\_tracing\_indices**: A list of core indices based on the input data.

#### **Algorithm Steps:**

1. Initialize an empty dictionary **all\_result** to store results.
2. Create a new queue **queue** for processing nodes.
3. Set the **index** to the specified initial image index value.
4. Check if there are elements in **object\_correspondence\_dictionary** at the given **index**:
  - 4.1. If there are elements, proceed with the following steps; otherwise, return an empty **object\_tracing\_indices** list.
5. Initialize an empty list **object\_tracing\_indices** to store traced indices.
6. For each element **current** in **object\_correspondence\_dictionary[index][0]**, do the following:
  - 6.1. Clear the queue **queue**.
  - 6.2. Create a new dictionary **individual\_result** to store results for this node.
  - 6.3. Set the **parent** to 0.
  - 6.4. Get the list of **next\_index\_list** from **object\_correspondence\_dictionary[index][0][current]**.
  - 6.5. For each **next\_item** in **next\_index\_list**, do the following:
    - 6.5.1. Add **[parent, index + 1, next\_item]** to the queue **queue**.
    - 6.5.2. Append **(index, current)** and **(index + 1, next\_item)** to **individual\_result**.
    - 6.5.3. Process items in the queue as follows:
      - 6.5.3.1. Dequeue an element as **[parent, next\_index, next\_item]**.
      - 6.5.3.2. Check if **(next\_index, next\_item)** is in **individual\_result**. If it is, skip this iteration.

- 6.5.3.3. Check if `next_index` is greater than `total_correspondence_length`. If it is, exit the loop.
- 6.5.3.4. Check if `next_item` is not in `object_correspondence_dictionary[next_index][0]`. If it is not, skip this iteration.
- 6.5.3.5. Check if there is more than one child in the next node. If there are, increment `parent`.
- 6.5.3.6. For each `next_i` in `object_correspondence_dictionary[next_index][0][next_item]`, do the following:
  - 6.5.3.6.1. Add `[parent, next_index + 1, next_i]` to the queue `queue`.
  - 6.5.3.6.2. Append `(next_index, next_item)` and `(next_index + 1, next_i)` to `individual_result`.
- 6.6. Append `individual_result` to `object_tracing_indices`.
7. Return `object_tracing_indices`.

As the object tracing is done, the “object\_tracing\_indices” variable has all the indices of the traced object started from a given image. Tracing is incomplete without visualization, as traced values are now in a list of dictionary format; a tree representation is needed to extract the raw values to human-understandable visuals. Another customized tree algorithm was written to do so and is shown in Algorithm 2. This representation targets one single object in the previously given image (“firstNpairs” refers to that target object index), from which the tracing tree generation will begin.

#### **Algorithm 2:**

**Title:** Building and Visualizing the Tracing Tree

**Algorithm Description:** This algorithm constructs a tree data structure from a given dictionary and visualizes the tree using the **graphviz** library.

#### **Inputs:**

1. **data:** A dictionary representing the tree structure.

#### **Outputs:**

1. A visualization of the tree.

### Algorithm Steps:

1. Initialize an empty dictionary **node\_dict** to store Node objects.
2. For each key in the **data** dictionary, do the following:
  - 2.1. Extract **value** and **idx** from the key.
  - 2.2. Create a unique **name** for the node based on **value** and **idx**.
  - 2.3. Create a Node object and store it in **node\_dict** with the key as its identifier.
3. For each key in the **data** dictionary, do the following:
  - 3.1. If the key exists in **node\_dict**, assign the corresponding Node object to **parent\_node**.
  - 3.2. For each **child** in the list of children associated with the key, do the following:
    - 3.2.1. If the **child** exists in **node\_dict**, assign the corresponding Node object to **child\_node**.
    - 3.2.2. If **child\_node** is not already a child of **parent\_node**, add it as a child of **parent\_node**.
4. Find the root node by selecting the first key from the **data** dictionary.
5. Return the root Node object.
6. Create an empty directed graph **dot** for visualization using **graphviz**.
7. Create an empty set **edges** to track processed edges.
8. Define a recursive function **add\_nodes(dot, node, edges)** to add nodes and edges to the graph:
  - 8.1. If **node.parent** is **None**, add the node as a single node to the graph.
  - 8.2. Otherwise, create an edge from the parent node to the current node and add both nodes to the graph.
  - 8.3. Recursively call the **add\_nodes** function for each child of the current node.
9. Build the tree by calling the **build\_tree** function with **firstNpairs** as the input and assign the root node to **root**.
10. Initialize an empty set **edges**.
11. Create an empty **graphviz** Digraph named **dot** for visualization.
12. Call the **add\_nodes** function with **dot**, **root**, and **edges** as arguments to add nodes and edges to the graph.
13. The visualization of the tree is now ready.

Finally, the visualization tree is ready to view the connected nodes (objects) along with the other leaf nodes with the main body. The result of this tracing is converted into a video to better understand the droplets' movement path.

### 3.8 Evaluation Matrix Used

Some popular evaluation matrices and formulas like Mean Average Precision (mAP), Mean Average Recall (mAR), and F1 score are used for segmentation. In instance

segmentation, average precision is calculated using a precision-recall curve where the x-axis represents recall, and the y-axis represents precision. This curve is calculated based on a confidence threshold like intersection over union (IoU) to measure the correctness of the predicted mask based on ground truth. The precision-recall graph calculates the average precision from the area under the curve. Representing the precision( $r$ ) at a specific recall value  $r$ , the AP formula is mentioned in Equation 3.11.

$$AP = \int_0^1 \text{precision}(r) dr \quad 3.11$$

Mean average precision means the whole performance summarization if multiple classes exist in a dataset. It is calculated by averaging each class's AP. If there are  $N$  classes, the mAP calculation formula is shown in Equation 3.12.

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad 3.12$$

Recall refers to the total number of correctly detected ground truths by the model. It is a fraction of the correctly detected ground truth of one target class, as shown in Equation 3.13.

$$Recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \quad 3.13$$

Average recall has been calculated based on recall on each image on one target class. If there are  $N$  images in one class, the average recall is calculated by Equation 3.14.

$$AR_i = \frac{1}{N} \sum_{j=1}^N Recall_{i,j} \quad 3.14$$

Finally, mAR is calculated across all the classes in a dataset. If there are  $C$  classes in a dataset, the mAR is calculated from the Equation 3.15.

$$mAR = \frac{C}{1} \sum_{i=1}^C AR_i \quad 3.15$$

F1 score usually means the balance between precision and recall, in this case mAP and mAR. The formula to calculate F1 score is pretty simple and shown in Equation 3.16.

$$F1\ score = \frac{2 \times (mAP \times mAR)}{mAP + mAR} \quad 3.16$$

### 3.9 Summary

This chapter explains the dataset, experimental procedure, used models with their parameters, and customized algorithms for object tracing. Four image segmentation models named Mask R-CNN, BMask R-CNN, FastInst and PatchDCT have been used with different epochs and backbone networks. After training, all four models and their different weights were tested on the generated augmented test dataset, the original seven-image set, and the final image set that is consists of 1573 images. The next chapter will discuss the results of this thesis.

## CHAPTER 4: RESULTS AND DISCUSSION

This chapter presents the findings obtained through the rigorous application of the research methodology outlined in the preceding chapters. This section aims to provide a comprehensive overview of the outcomes derived from the experimental procedures and analytical techniques employed. Through image augmentation, pre and post processing, model training and analysis, the results offer valuable insights into liquid fuel atomization process. This chapter serves as a platform for presenting quantitative outcomes, shedding light on the efficacy of the models employed and the implications of the findings. Moreover, it highlights any noteworthy entity, value, or anomalies observed within the model's test result, fostering a deeper understanding of the research objectives and contributing to the advancement of knowledge in the field of atomization and computer vision.

This chapter discusses the experimental results, which are divided into three significant sections. Section 4.1 shows all the detection results related to Mask R-CNN. The model is trained on 1000 augmented images and tested on 200 test images that has been generated using custom data augmentation method (discussed in section 3.2). Later, it is tested on original seven images for object segmentation and object count. But the object detection count is lower than expected for which a divide and conquer technique has been applied to reduce the image size in both height and width using different crop window. It is expected that, the model can detect better on less dense image frames which will lead to better object detection count. Section 4.2 is about BMask R-CNN results, which is expected to be a better version of Mask R-CNN. Similar result presentation structure is followed, consists of augmented image testing, later original seven image inference. Also, similar divide and conquer technique is applied expecting higher object detection count. Section 4.3 is about two transformer models, maintaining consistent result pattern, starts

with augmented image testing, continues to seven original image testing and finally applied the divide and conquer technique. Transformer models are expected to be fast and accurate and comparable with the Mask R-CNN and BMask R-CNN. These results will help to achieve the second objective of this research, which is to get the comparatively better object detection model. After getting results of each mentioned models and comparing the results with different set of images, section 4.4 discusses the result achieved from final dataset that has 1573 images in it in a sequential way. Based on the object count results, suitable models will be selected for object tracing task as it requires consistent object detection throughout the whole image sequence. Later, section 4.5 discusses the object tracing outcome using the selected models. One extra model has been used named DeepSort for tracing the object movement, but it has its own limitation over our goal. Finally, using the selected models, object tracing and hot zoning has been done. It is expected to contact trace object movement and figure out object trajectories. These tracing results is expected to achieve the mentioned third and final objective of this research. An in-depth discussion about the results of different models' detection rate and comparison is mentioned in section 4.6.

#### **4.1 Results of Mask R-CNN**

Mask R-CNN allowed extra parameter tuning during the test through a config file where max instance detection, pre-non-maximum suppression (NMS), post-NMS, ROIs train limit was changed to 2000, ROIs positive ratio as 0.50 and detection minimum confidence to 0.5. These mentioned parameters looked suitable based on the gained detection and segmentation accuracy comparing with other tested parameters. With this configuration, initially model has been tested on augmented test images and later has been applied to the original image set to extract objects and their mask properties. As three randomly selected weights were considered (1000<sup>th</sup>, 2791<sup>st</sup> and 5791<sup>st</sup>) for the testing, all the results would be based on those.



#### 4.1.1 Object detection on the augmented and preliminary dataset

There are 1000 train and 200 test images, generated in the data augmentation process. After training those 1000 images, testing has been done on multiple batches. With a 75% IoU value, mean average precision (mAP), mean average recall (mAR), and F1 score were calculated for the 1000th, 2791st, and 5791st weights. 75.69% mAP and 67.09% mAR were achieved from the 1000<sup>th</sup> weight, 73.96% mAP and 55.3% mAR were acquired from the 2791<sup>st</sup> weight, and 74.93% mAP and 57.18% mAR were achieved from 5791<sup>st</sup> weight files. From mAP and mAR, the F1 score was 71.12% for 1000, 63.28% for 2791, and 64.86% for the 5791<sup>st</sup> weight file. This comparison has been illustrated in Figure 4.1, along with some sample test images from the mentioned three model weights in Figure 4.2.

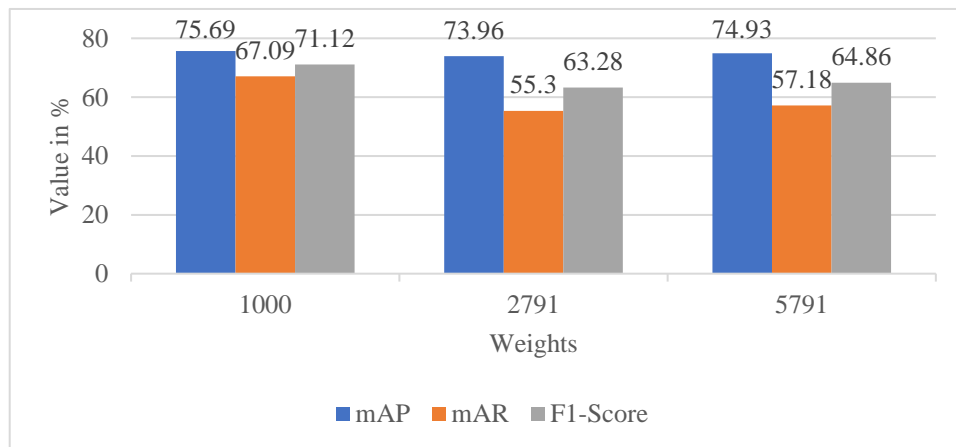
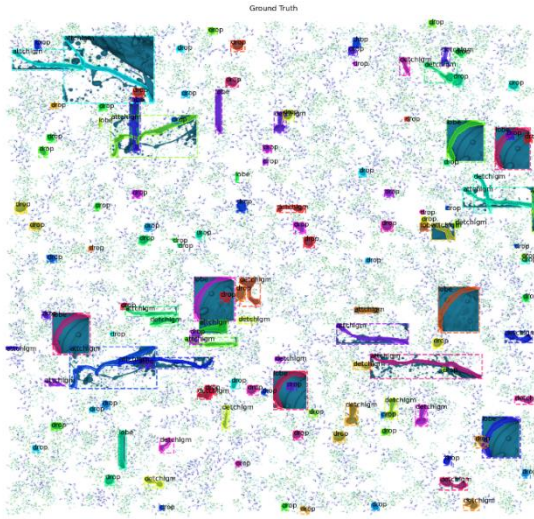
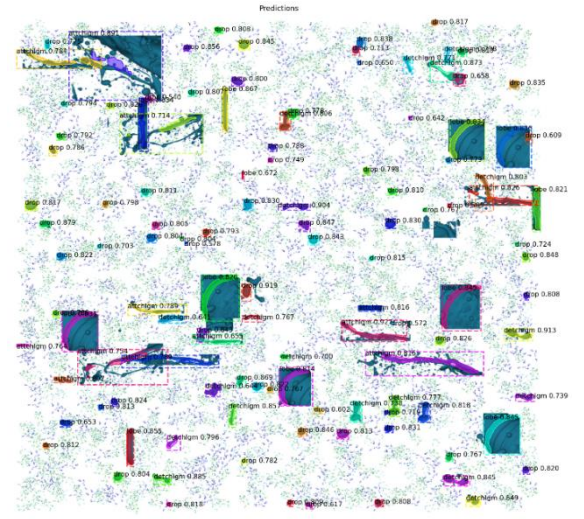


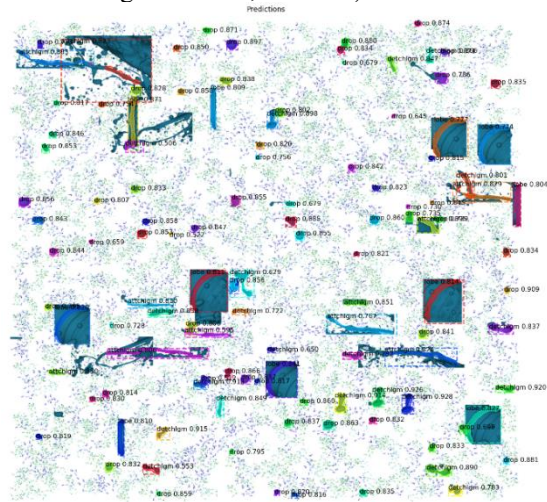
Figure 4.1 Comparison between mAR, mAP, F1 score among different weight testing.



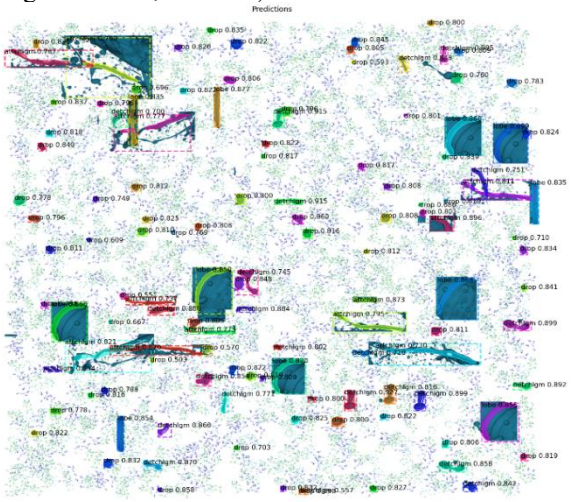
(a) Test image with ground truth label, having objects (droplets:82, detached ligaments: 23, attached ligaments: 15, lobes:15)



(b) 1000<sup>th</sup> weight's detected result, having objects (droplets:72, detached ligaments: 20, attached ligaments: 13, lobes:12)



(c) 2791<sup>st</sup> weight's detected result, having objects (droplets:72, detached ligaments: 21, attached ligaments: 11, lobes:11)



(d) 5791<sup>st</sup> weight's detected result, having objects (droplets:74, detached ligaments: 23, attached ligaments: 13, lobes:11)

Figure 4.2 Detection result for multiple weight files on augmented test images. Original test image is shown in (a), 1000<sup>th</sup> weight test is shown in (b), 2791<sup>st</sup> weight test is shown in (c) and 5791<sup>st</sup> weight test is shown in (d).

The randomly selected test image shown in Figure 4.2 (a) has 82 droplets, 23 detached ligaments, 15 attached ligaments, and 15 lobes (a total of 135 objects). Comparing this with the number of objects detected by trained models, 72 droplets, 20 detached ligaments, 13 attached ligaments, and 12 lobes are detected by the 1000<sup>th</sup> weight file; 72 droplets, 21 detached ligaments, 11 attached ligaments, and 11 lobes are detected by 2791<sup>st</sup> weight file. A small increment of droplet detection is shown in the 5791<sup>st</sup> weight file as 74. Also, ligaments of 23 in detached and 13 in attached are shown with previously detected 11 lobes. The illustration of object detection comparison is shown in Figure 4.3.

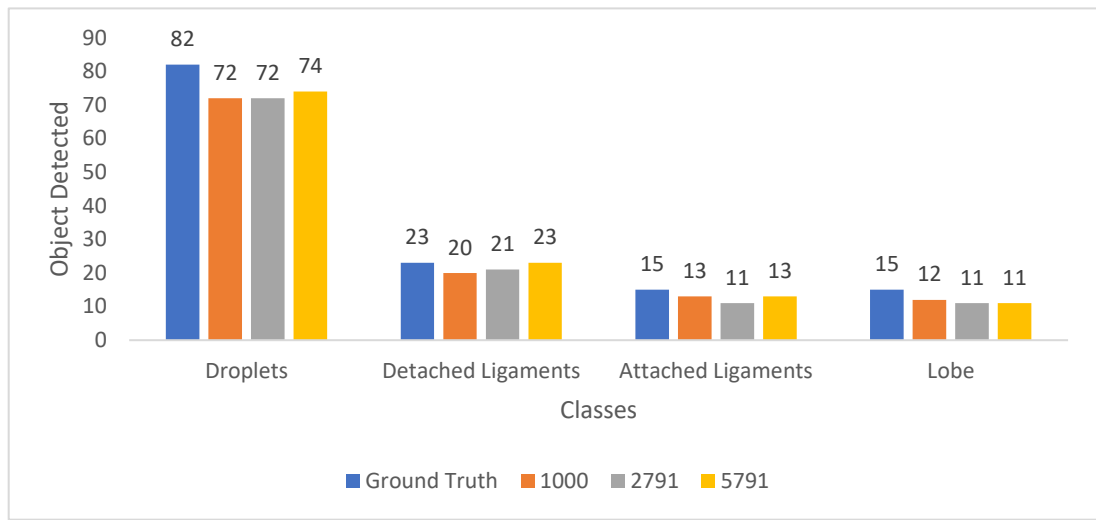


Figure 4.3 Comparison of detected objects on different weights in one image.

From the tested augmented images, it has been observed that the model is successfully segmenting the object shapes and the object detection count has achieved 89.6% accuracy for 5791<sup>st</sup> weight (121 objects detected out of 135) which seems good. This test result validates the learning capabilities of Mask R-CNN with respect to the given dataset, fulfilling the aim of model training and now ready to be applied on the original seven image dataset.

After augmented test images are validated, the original initial seven images are tested and compared with different weights. Technically, these images were never known to the

model in the training process; also, all the objects were not labeled due to the suspicion of overfitting. At first glance, the objects detected in these images were low, but the detected objects' masks were precise. Also, higher epoch weights were giving better detection results than lower weights. In the case of droplets that were too small, all three tested weights failed to detect them. Preliminary results also show a low detection rate of lobes in dynamic positions. The highest number of droplets detected on the 5<sup>th</sup> image was 84 for 1000<sup>th</sup>, 66 for 2791<sup>st</sup> and 100 for 5791<sup>st</sup> weight. The highest number of detections for detached ligaments has been shown in 16 for 1000 and 2791, and 19 for 5791. A lower number of attached ligaments and lobes were detected for all three weights. The total summary of detected objects on all seven images for three weights is shown in Figure 4.4. Among those seven images, one image result is shown for three weights in Figure 4.5, additionally, some large area of missing objects that were visible to the model but undetected have been identified and marked in red boxes.

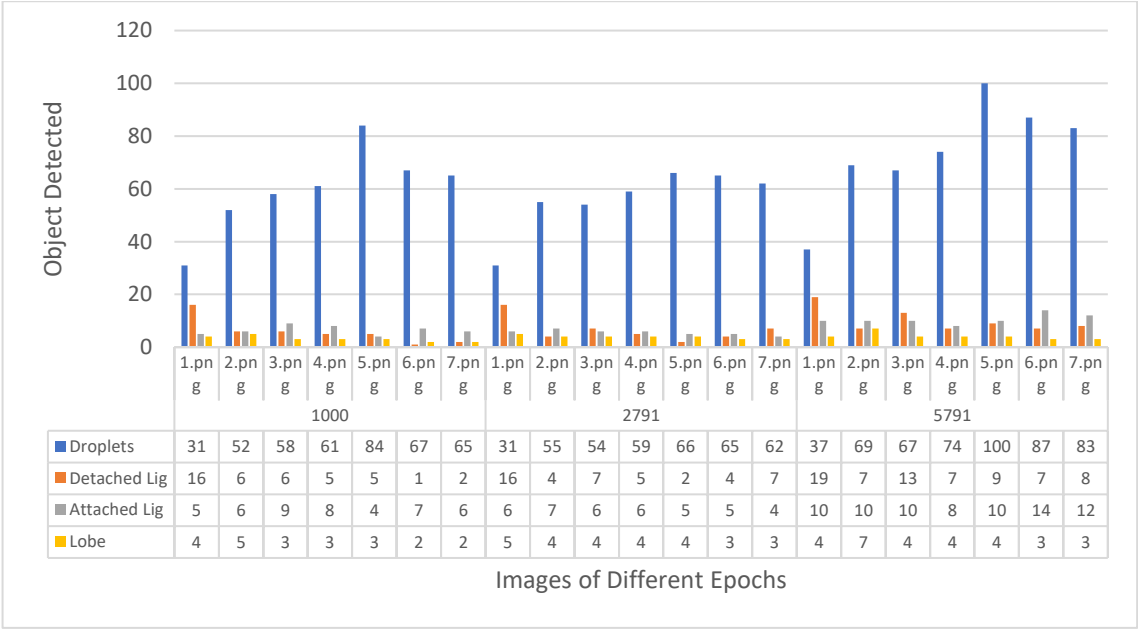


Figure 4.4 Summary of detected objects on original images using all three weights.



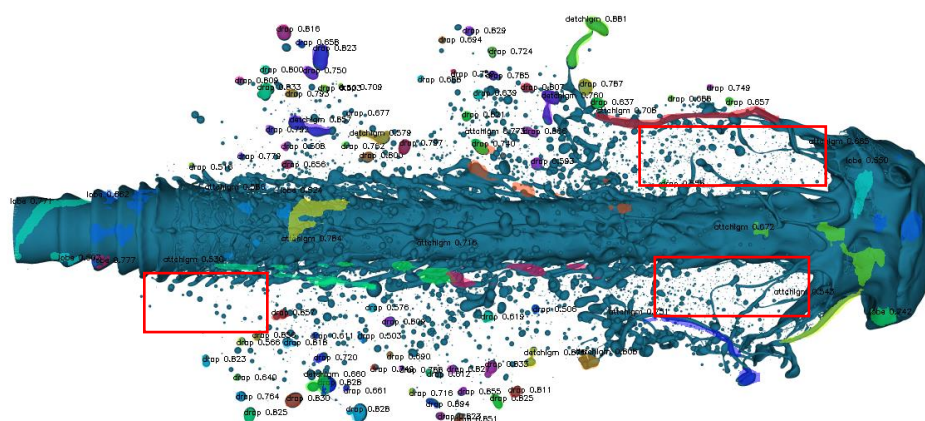
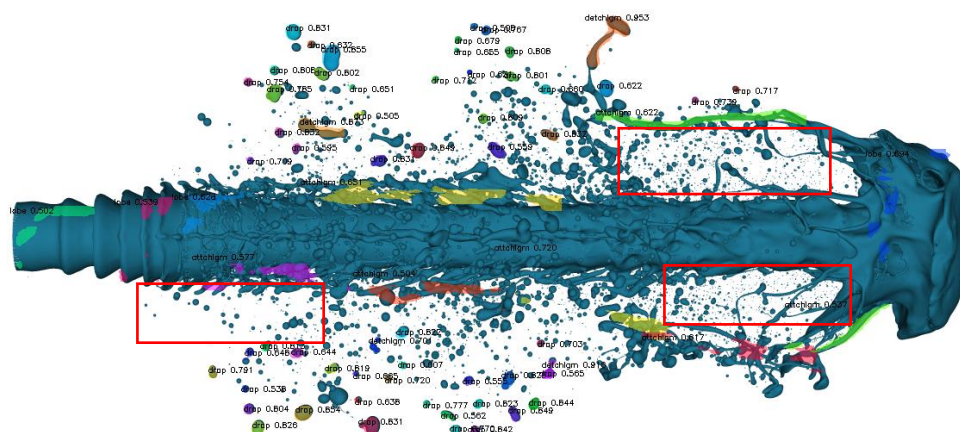
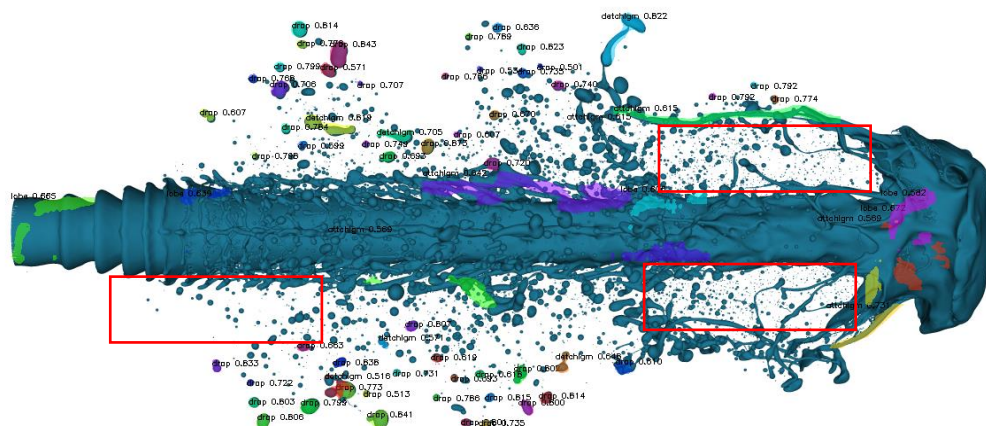


Figure 4.5 One original image test result shown for (a) 1000<sup>th</sup>, (b) 2791<sup>st</sup> and (c) 5791<sup>st</sup> weight (missed object detection areas are marked in red boxes)

Raw weight without any post-processing works well on augmented test datasets with high object counts and precise object masks, but the target is to apply it to the main dataset. But when applied to the real seven images, a lot of objects are seen to remain undetected at the right middle portion of the body, as well as other parts of the body when objects are relatively looking small. One image is estimated to have around 1500 objects, but initial detection is only having around 100 objects, which is too low. Some major areas that miss the object detection is marked in red boxes in Figure 4.5. As the detection rate is not up to expectations, the divide and conquer method was applied to reduce the object density and hoping to increase the object count near estimation. Object detection after cropping results is shown in the following subsection.

#### **4.1.2 Object detection after divide and conquer**

As the trained model is not working well on the original dataset, one suspected reason is that higher object density in the image may cause the model to fail in the region proposal network. To reduce this object density, a different cropping window is applied to crop a section of the image and test it to see the differences. It was unknown which cropping window would give the best result, so a total of nine windows has been applied shaping as (width, height/3), (width/3, height), (width/3, height/3), (width, height/4), (width/4, height), (width/4, height/4), (width, height/5), (width/5, height), (width/5, height/5). For the 1000<sup>th</sup> weight, a total of 32 ligaments, 565 droplets, 52 attached ligaments, and 44 lobes are detected from the (height, width/3) crop window on seven images. A total of 693 objects are detected from this crop window. Though some significantly improved object detection is seen on crop window (height/3, width/3), (height/4, width/5), and (height/5, width/5), based on the mask precision on them (height/3, width/3) seems better, also has highest droplet detection of 2830 among all windows. Though detected object is high, comparing with visual inspection of mask prediction, (height, width/3) shows better

values. A comparison of all crop windows and summation of detected objects for 1000<sup>th</sup> weight is shown in Table 4.1.

For the 2791<sup>st</sup> weight, a similar process has been applied, and significant improvement in object detection on (height/3, width/3) crop window has been seen. The total object count on this window is 3275, having a maximum droplet count of 2717 (compared with other windows). As there is no complete ground truth value, human visuals were applied to test the best mask outcome, and this window shows better results than others. Though (height/5, width/5) offers overall high detection, the number of lobes is misclassified, reducing the droplet count. For this model, stable and good mask prediction values are seen on (height, width/3) window. A similar comparison of these object detections is shown in Table 4.2.

Finally, the 5791<sup>st</sup> weight is tested with a similar crop window and original images. Like the previous weight, this also results in better detection and mask prediction on the (height/3, width/3) window, as the total object detection count is 3497 with the highest 2887 droplets. This time (height/4, width/4) window shows the highest object detection rate, but lobe detection is still insufficient due to the misclassification of the droplet as lobe. Also comparing with mask prediction quality, the (height, width/3) shows a consistency which is necessary for object tracing. Summary comparison for this weight is shown in Table 4.3. One sample image after processing with (height, width/3) window for 5791<sup>st</sup> weight is shown in Figure 4.6, focusing on the areas that were previously marked red, due to undetected objects (marked in yellow boxes).

Table 4.1 Comparison of the total detected object for 1000<sup>th</sup> weight's test result on different crop windows for original seven images.

Crop Shape	Window	Total Detached Ligaments	Total Droplets	Total Attached Ligaments	Total Lobes	Sum
height, width		41	418	45	22	526
height, width/3		32	565	52	44	693
height/3, width		43	461	53	31	588
height/3, width/3		58	2830	354	247	3489
height, width/4		52	1305	88	98	1543
height/4, width		47	454	57	22	580
height/4, width/4		27	2660	600	713	4000
height, width/5		32	750	42	48	872
height/5, width		55	403	63	20	541
height/5, width/5		17	1628	810	1502	3957

Table 4.2 Comparison of the total detected object for the 2791<sup>st</sup> weight's test result on different crop windows for original seven images.

Crop Shape	Window	Total Detached Ligaments	Total Droplets	Total Attached Ligaments	Total Lobes	Sum
height, width		45	392	39	27	503
height, width/3		103	1138	67	49	1357
height/3, width		59	433	32	35	559
height/3, width/3		150	2717	215	193	3275
height, width/4		103	1104	63	47	1317
height/4, width		56	426	33	13	528
height/4, width/4		73	2289	428	597	3387
height, width/5		96	1137	65	43	1341
height/5, width		61	382	15	12	470
height/5, width/5		52	1281	737	1419	3489

Table 4.3 Comparison of the total detected object for 5791<sup>st</sup> weight's test result on different crop windows for original seven images.

Crop Shape	Window	Total Detached Ligaments	Total Droplets	Total Attached Ligaments	Total Lobes	Sum
height, width		70	517	74	29	690
height, width/3		70	1231	100	56	1457
height/3, width		74	504	61	31	670
height/3, width/3		90	2887	275	245	3497
height, width/4		89	1408	90	66	1653
height/4, width		69	473	62	13	617
height/4, width/4		35	2439	419	812	3705
height, width/5		99	1420	122	55	1696
height/5, width		64	454	73	17	608
height/5, width/5		15	1195	535	1486	3231



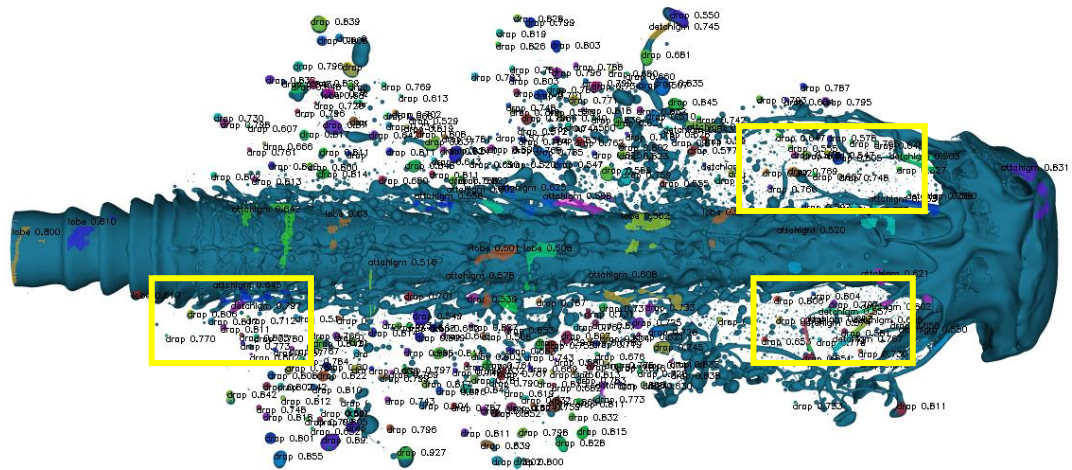


Figure 4.6 One sample image using (height, width/3) crop window from 5791st weight, showing increment in the object detection. Notable object detection increased on smaller droplets; this image is comparable with Figure 4.5 (c). Yellow boxes are to point out region that previous had undetected objects, but now it has a lot of object detected after using divide and conquer method.

The summation of object count of the original seven images are shown in above three tables. Additionally, the detailed object detection result for individual images for all three weights and all cropping windows are shown in the Appendix Table A.2 on Page 122. From all these results, the (height, width/3) window is selected to analyze the final 1573 images-based dataset, because it has detected a total of 1457 objects, which is close to the expected 1500 number. Also, this window has a lower lobe detection and higher droplet detection. Finally, a visual inspection was done among different windows' object detection results to support the selected window. From there, detected object properties are saved in a JSON file for object tracing tasks. It seems apparent from the literature that BMask R-CNN was specifically modified to identify object boundaries more accurately than Mask R-CNN. which is a crucial area for identifying objects that are densely populated. In order to determine whether better results are achievable, BMask R-CNN has been trained and used as a performance comparison as well as an improvement opportunity.

## 4.2 Results of BMask R-CNN

BMask R-CNN is faster in training compared with Mask R-CNN. A total of 3000 epochs were done on this model for previously mentioned configurations; among them, 1000<sup>th</sup> and 3000<sup>th</sup> weights are selected for testing, and mAP has been calculated using MS COCO evaluation format with the detectron2 library. A test threshold of 50% is used for evaluation, and detailed results are shown in the section below.

### 4.2.1 Object detection on the augmented and preliminary dataset

The model is initially tested on augmented test images, getting a mean average precision of 51.49% for 50% IoU and 17.76% for 75% IoU on FPN-based BMask R-CNN with 1000 epochs. With 3000 epochs and the same configuration, mAP of 64.99% for 50% IoU and 44.680% for 75% IoU is achieved. Using Base BMask R-CNN configuration, mAP achieved 44.743% for 50% IoU and 14.78% for 75% IoU with 1000 epochs and 63.88% for 50% IoU and 42.07% for 75% IoU has been achieved for 3000 epochs. Using Cascade BMask R-CNN mAP is calculated as 54.85% for 50% IoU and 27.89% for 75% IoU with 1000 epochs, 66.95% for 50% IoU, and 47.45% for 75% IoU with 3000 epochs. Individual APs are also calculated for four classes among three configurations and two weights; details are shown in Table 4.4, and the detailed mAP comparison is in Table 4.5.

Table 4.4 Comparison of mAP per class on different configurations with selected weights.

Config	Weight	Detached Ligament	Droplet	Attached Ligament	Lobe
FPN	1000	30.821	25.353	23.734	16.383
	3000	47.668	30.566	40.060	41.930
Base	1000	10.203	23.726	20.903	23.975
	3000	38.637	38.614	35.704	37.944
Cascade	1000	36.735	21.064	25.630	32.100
	3000	47.699	35.750	41.249	42.126

Table 4.5 Comparison of mAP among different configurations with selected weights.

Config	Weight	AP	AP50	AP75
FPN	1000	24.073	51.492	17.776
	3000	40.056	64.999	44.680
Base	1000	19.702	44.743	14.788
	3000	37.725	63.886	42.076
Cascade	1000	28.882	54.852	27.898
	3000	41.706	66.959	47.449

One sample object detection image has been shown on the same sample augmented test image as previously mentioned in Figure 4.3 With different configurations and weights, BMask R-CNN achieved a maximum of 60 droplet counts for 3000 epochs on base configuration. With this weight, 15 detached ligaments, 10 attached ligaments, and 12 lobes are detected. Compared with the Mask R-CNN detection result, BMask R-CNN shows less detection on augmented images along with low mAP. A detailed comparison of detected objects is shown in Figure 4.7, and the tested object detection image is shown in Figure 4.8 for each weight and configuration.

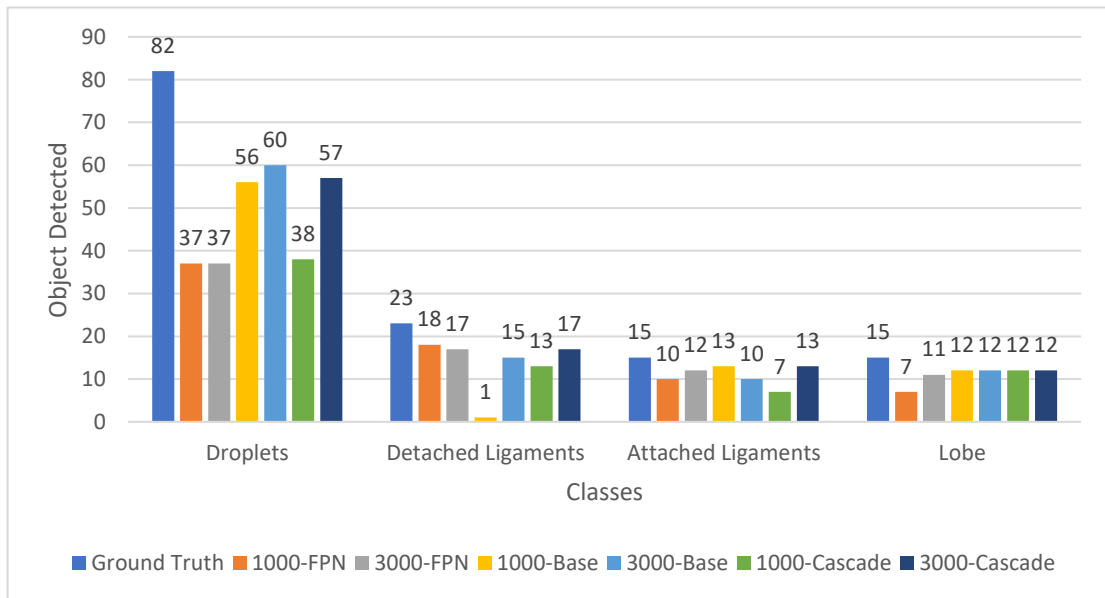
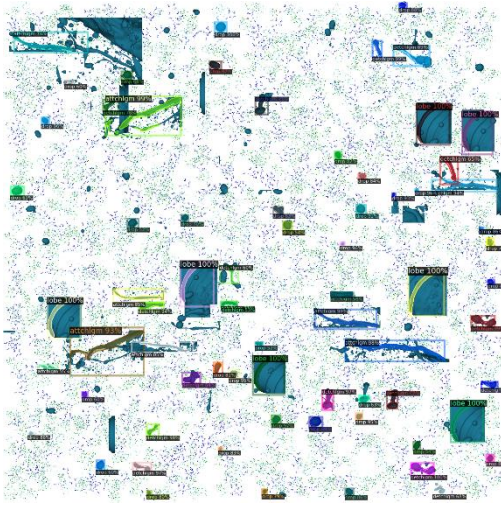
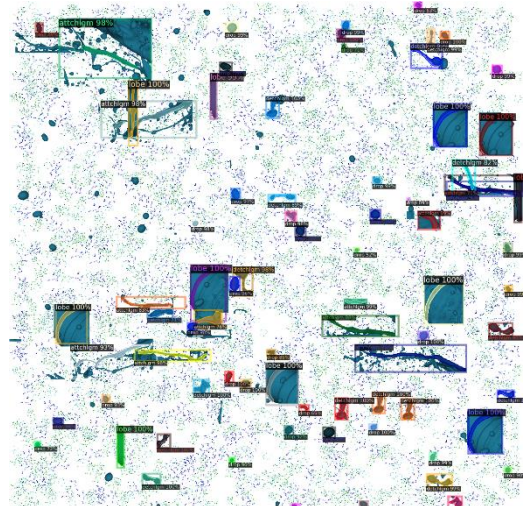


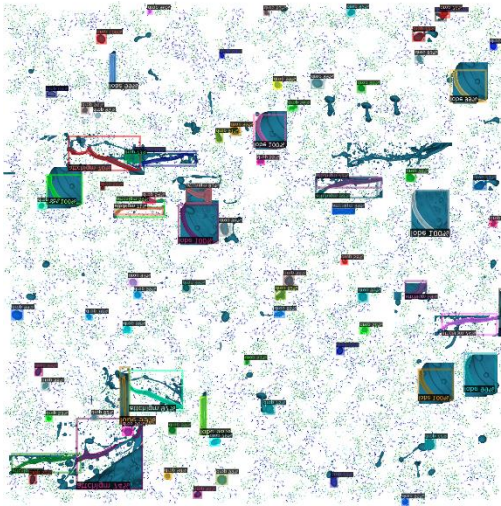
Figure 4.7 Comparison of object count between ground truth and selected weights for all configurations and selected weights.



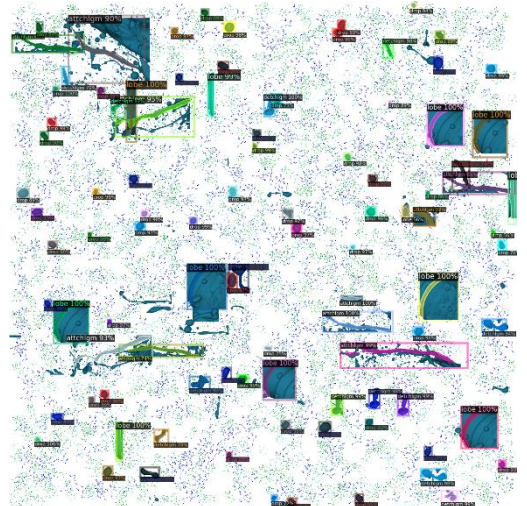
(a) 1000<sup>th</sup> FPN detection result, having objects (droplets:37, detached ligaments: 18, attached ligaments: 10, lobes:7)



(b) 3000<sup>th</sup> FPN detection result, having objects (droplets:37, detached ligaments: 17, attached ligaments: 12, lobes:11)



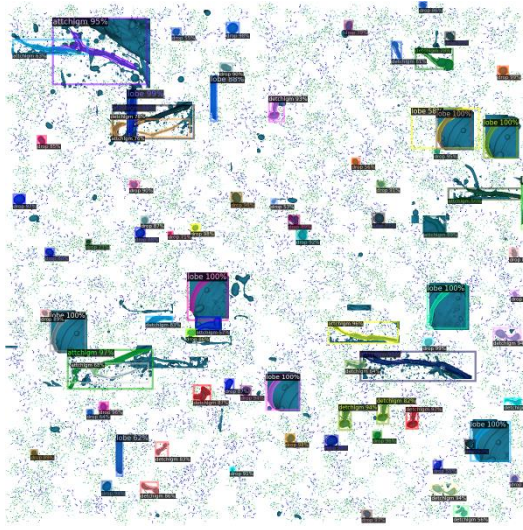
(c) 1000<sup>th</sup> Base detection result, having objects (droplets:56, detached ligaments: 1, attached ligaments: 13, lobes:12)



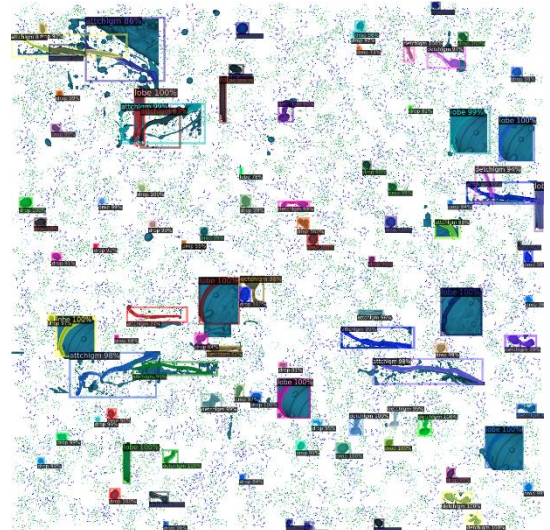
(d) 3000<sup>th</sup> Base detection result, having objects (droplets:60, detached ligaments: 15, attached ligaments: 10, lobes:12)

Figure 4.8 Detection result for multiple configurations with different weights on one selected augmented test image. FPN based result with 1000 epoch is shown in (a) and 3000 epoch result is shown in (b). Base model result with 1000 epoch is shown in (c) and 3000 epoch result is shown in (d). Finally, Cascade model result with 1000 epoch is shown in (e) and 3000 epoch result is shown in (f).





(e) 1000<sup>th</sup> Cascade detection result, having objects (droplets:38, detached ligaments: 13, attached ligaments: 7, lobes:12)



(f) 3000<sup>th</sup> Cascade detection result, having objects (droplets:57, detached ligaments: 17, attached ligaments: 13, lobes:12)

Figure 4.8, continued: Detection result for multiple configurations with different weights on one selected augmented test image. FPN based result with 1000 epoch is shown in (a) and 3000 epoch result is shown in (b). Base model result with 1000 epoch is shown in (c) and 3000 epoch result is shown in (d). Finally, Cascade model result with 1000 epoch is shown in (e) and 3000 epoch result is shown in (f).

Though BMask R-CNN was expected to have better detection capabilities due to the object boundary preservation method, it showed slightly lower performance on object count as 73.33% accuracy (99 object detected out of 135) and mask precision than Mask R-CNN which achieved 89.62% object detection accuracy. One possible reason could be the modified model layers which causes the BMask R-CNN to speed up the training time but lost features in between causing performance loss.

After testing the model on augmented test images, it's time to apply the models to the initial original seven images. As the initial test mAP was low for test images, this time, testing thresholding has been reduced to 40%, which dramatically increases the object detection count, outperforming the number for Mask R-CNN. The maximum object detection is caused by the Cascade model with 3000 epochs, with the highest droplet count of 193 on the 5<sup>th</sup> image. Also, the average droplet count is 119 among seven images. Comparing the three model configurations, the lowest average object count is seen from the Base model, which detected 72 droplets, 4 detached ligaments, 1 attached ligament

and 1 lobe in all seven images. Improvement is seen in updated 3000<sup>th</sup> weights, as average droplet detection increases from 72 to 112, detached ligaments from 4 to 5, attached ligaments from 1 to 5, and lobes from 1 to 4. Similar improvement is seen in the other two configurations; in FPN-based BMask R-CNN, the 1000<sup>th</sup> weight was detected on average 73 droplets, whereas the 3000<sup>th</sup> weight average is 82. Also, in cascade, the 1000<sup>th</sup> weight initial detection was 113, but the 3000<sup>th</sup> weight increases this to 119. The summary representation for seven images tested on 3000<sup>th</sup> weight with three configuration models is shown in Figure 4.9. Also, the similar image that has been mentioned in Mask R-CNN original test image detection in Figure 4.5 is shown here in this section tested with BMask R-CNN models for comparison in Figure 4.10.

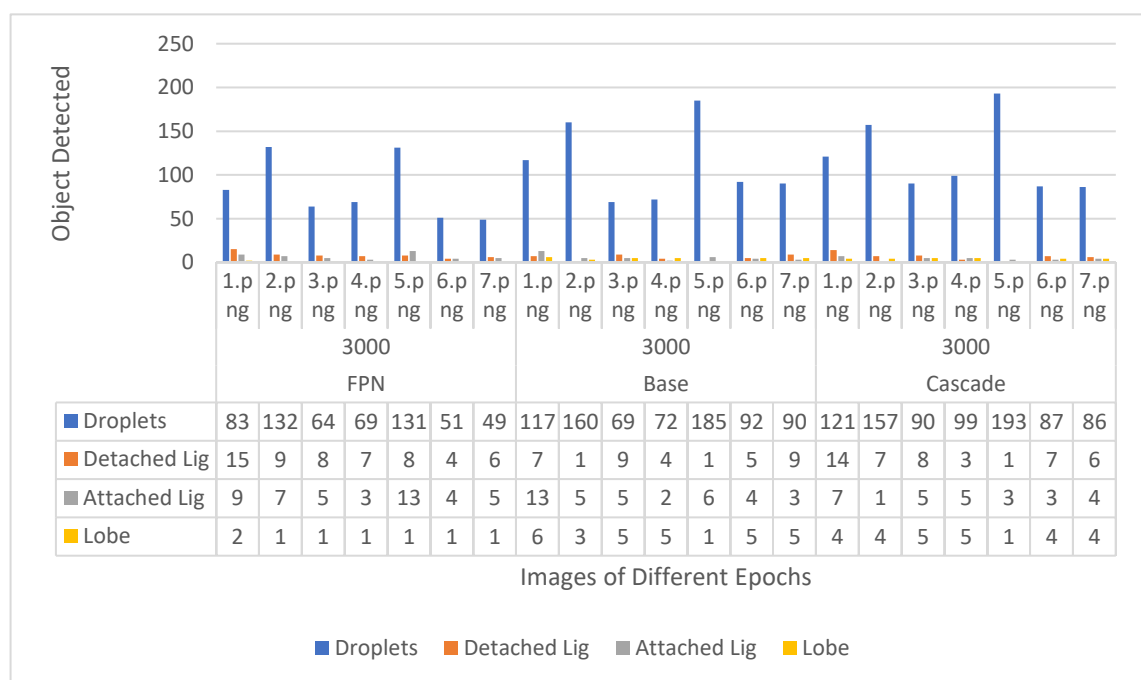
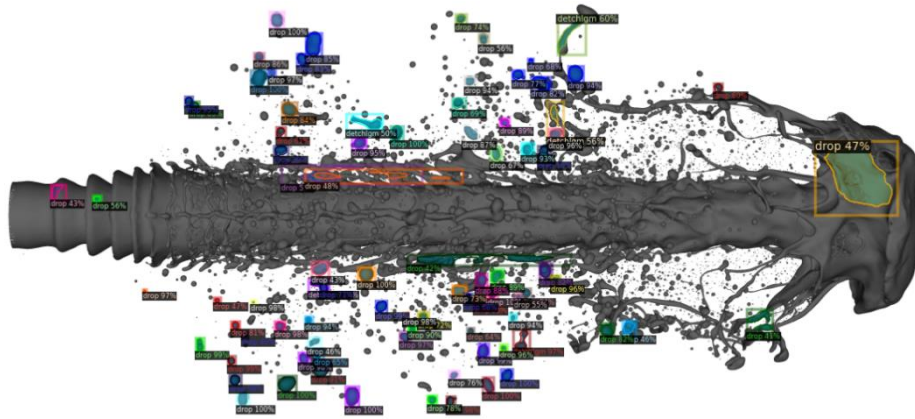
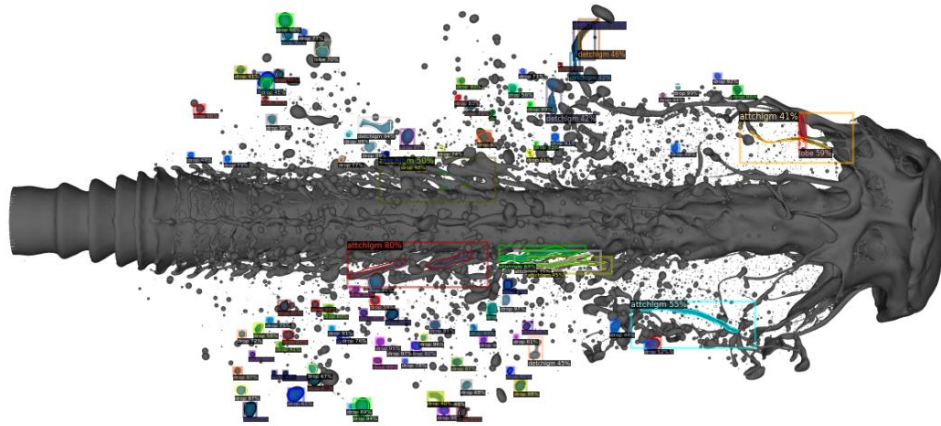


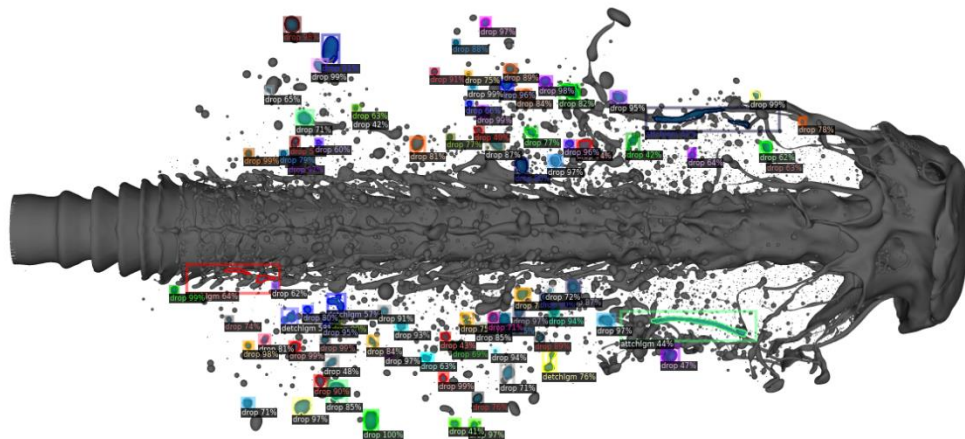
Figure 4.9 Summary of detected objects on original seven images with three configuration models on 3000<sup>th</sup> weight.



(a) FCN with 1000<sup>th</sup> weight



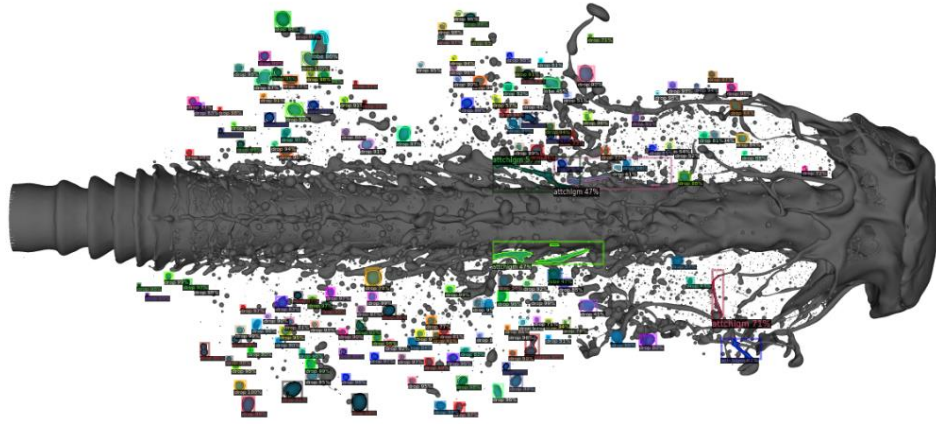
(b) FCN with 3000<sup>th</sup> weight



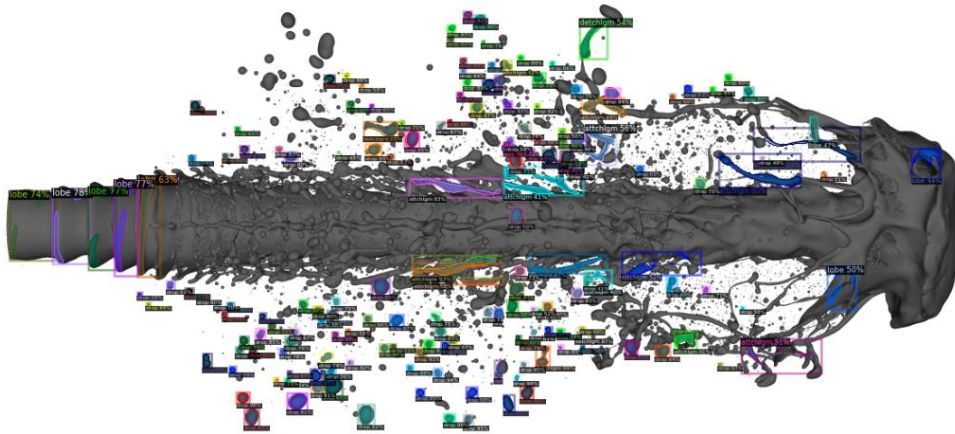
(c) Base with 1000<sup>th</sup> weight

Figure 4.10 Original image testing with different model configurations and epochs. FCN based result for 1000<sup>th</sup> weight is shown in (a), 3000<sup>th</sup> weight is shown in (b). Base model result with 100<sup>th</sup> weight is shown in (c) and 3000<sup>th</sup> weight is shown in (d). Finally, Cascade model result with 1000<sup>th</sup> weight is shown in (e) and 3000<sup>th</sup> weight is shown in (f).

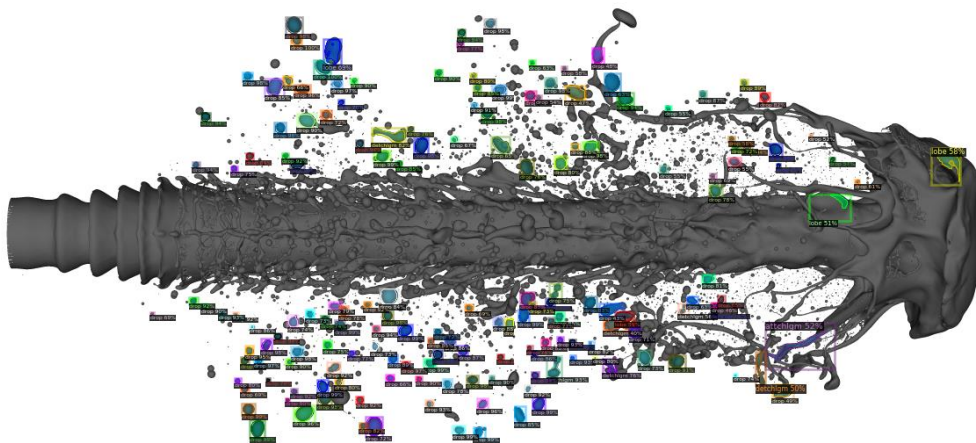




(d) Base with 3000<sup>th</sup> weight



(e) Cascade with 1000<sup>th</sup> weight



(f) Cascade with 3000<sup>th</sup> weight

Figure 4.10, continued: Original image testing with different model configurations and epochs. FCN based result for 1000<sup>th</sup> weight is shown in (a), 3000<sup>th</sup> weight is shown in (b). Base model result with 100<sup>th</sup> weight is shown in (c) and 3000<sup>th</sup> weight is shown in (d). Finally, Cascade model result with 1000<sup>th</sup> weight is shown in (e) and 3000<sup>th</sup> weight is shown in (f).



The average droplet detection for cascade 3000<sup>th</sup> weight is about 119, and the average object detected on 5791<sup>st</sup> weight of Mask R-CNN is 73. This indicates a higher object detection rate though the average droplet detection on (height, width/3) cropped window with Mask R-CNN with 5791<sup>st</sup> weight is 175. So, the raw BMask R-CNN improvement is still low. For this reason, a similar cropping window has been applied to the models on all weight files, and the next section discusses these results.

#### 4.2.2 Object detection after divide and conquer

Though the number of object detections has increased, it is still lower than expected object count, which is approximately 1500. So, a similar nine-cropping window and processing have been used with different configurations and weights for BMask R-CNN. The goal is to detect high droplet count, but a problem seen with cropping is that droplets are detected as lobes in some cases. Due to this issue, human visual verification was used to check cropping window performance. Another filter is to select a window with a high droplet count, with a general low lobe count of less than 200. For FPN based model with 1000<sup>th</sup> weight, a total of 1701 objects were detected by (height, width/3) window where droplet count is 1540 and lobe is 47. Also, with (height, width/4), 1793 droplets and 113 lobes were detected, and with (height, width/5) 2050 droplets and 112 lobes were detected. A comparison of the total detected object for this model and weight is shown in Table 4.6.

Table 4.6 Comparison between crop windows for FPN-based BMask R-CNN with 1000<sup>th</sup> weight.

Crop Window Shape	Total Detached Ligaments	Total Droplets	Total Attached Ligaments	Total Lobes	Sum
height, width	54	511	9	5	579
height, width/3	68	1540	46	47	1701
height/3, width	103	1394	53	20	1570
height/3, width/3	55	2907	148	772	3882
height, width/4	72	1793	46	113	2024
height/4, width	132	1627	48	30	1837
height/4, width/4	30	3119	245	1803	5197
height, width/5	83	2050	56	122	2311
height/5, width	136	1801	78	39	2054
height/5, width/5	36	3245	531	3085	6897

With the 3000<sup>th</sup> weight of the same FPN model, similar scenarios are seen with droplet and lobe detection. High droplet count and low lobes count are also from the same (height, width/3) having 1329 droplets, 100 lobes; (height, width/4) having 1600 droplets, 128 lobes; (height, width/5) having 1684 droplets and 168 lobes. This model's comparison is shown in Table 4.7, and additionally the complete raw data comparison between 1000 and 3000 weight is shown in Appendix Table A.3 on Page 124. One sample image from FPN based BMask R-CNN using (height, width/3) with 3000<sup>th</sup> weight is shown in Figure 4.11.

Table 4.7 Comparison between crop windows for FPN-based BMask R-CNN with 3000<sup>th</sup> weight.

Crop Shape	Window	Total Detached Ligaments	Total Droplets	Total Attached Ligaments	Total Lobes	Sum
height, width		57	579	46	8	690
height, width/3		38	1329	70	100	1537
height/3, width		44	1692	118	165	2019
height/3, width/3		13	2339	218	1253	3823
height, width/4		47	1600	88	128	1863
height/4, width		56	2113	183	195	2547
height/4, width/4		17	2338	450	2523	5328
height, width/5		50	1684	110	168	2012
height/5, width		51	2384	215	263	2913
height/5, width/5		7	1927	868	4089	6891

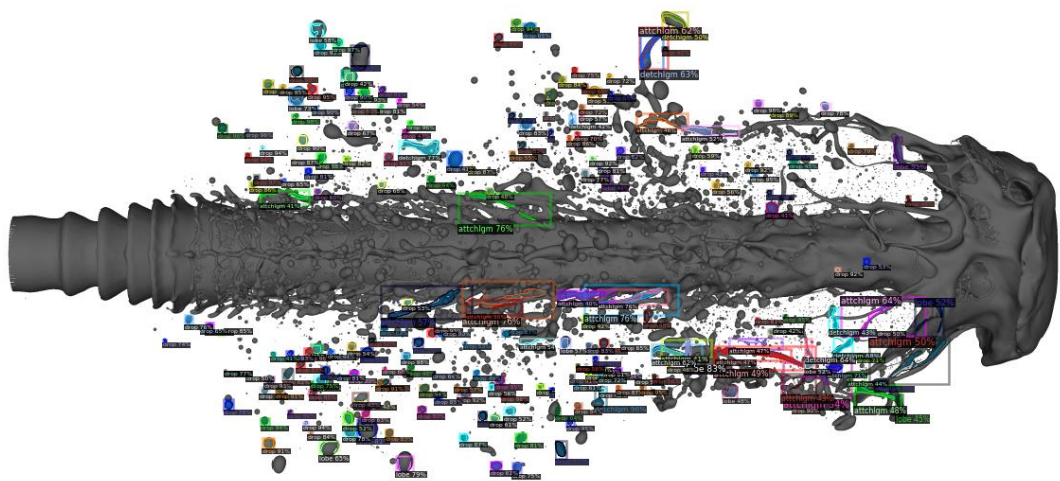


Figure 4.11 One sample image using (height, width/3) window with FPN based BMask R-CNN and 3000<sup>th</sup> weight, showing the increment in the object detection. This image is comparable with Figure 4.10 (b).

After FPN based model, Base BMask R-CNN's result is shown here for the 1000<sup>th</sup> weight. This mode has better detection on lobes as the number of lobe detection is low, and droplets are relatively high for previously focused weights. A total droplet of 1437 and 57 lobes is seen for (height, width/3) window; 1572 droplets, 93 lobes for (height, width/4); 1787 droplets, 99 lobes for (height, width/5), which is a little lower than FPN-1000<sup>th</sup> result. A summary of this model's result is shown in Table 4.8.

Table 4.8 Comparison between crop windows for Base BMask R-CNN with 1000<sup>th</sup> weight.

Crop Window Shape	Total Detached Ligaments	Total Droplets	Total Attached Ligaments	Total Lobes	Sum
height, width	30	510	13	5	558
height, width/3	23	1437	49	57	1566
height/3, width	46	1112	36	5	1199
height/3, width/3	30	2352	244	590	3216
height, width/4	22	1572	79	93	1766
height/4, width	57	1270	45	6	1378
height/4, width/4	34	1829	395	1756	4014
height, width/5	27	1784	108	99	2018
height/5, width	64	1333	70	5	1472
height/5, width/5	13	1481	634	3052	5180

Base BMask R-CNN with 3000<sup>th</sup> weight showed a slight improvement in droplet detection on previously focused three windows compared with 1000<sup>th</sup> weight. This configuration detects 1488 droplets and 242 lobes with (height, width/3) window; 1593 droplets and 268 lobes with (height, width/4) window; 1662 droplets and 312 lobes with (height, width/5) window. Noticeably, (height/3, width) also detects a high number of droplets and a low number of lobes. The summary result is shown in Table 4.9, and the detailed comparison of raw result between 1000<sup>th</sup> and 3000<sup>th</sup> weight is shown in Appendix Table A.4 on Page 125. One sample image from Base BMask R-CNN using (height, width/3) with 3000<sup>th</sup> weight is shown in Figure 4.12.

Table 4.9 Comparison between crop windows for Base BMask R-CNN with 3000<sup>th</sup> weight.

Crop Window Shape	Total Detached Ligaments	Total Droplets	Total Attached Ligaments	Total Lobes	Sum
height, width	36	785	38	30	889
height, width/3	17	1488	62	242	1809
height/3, width	16	1690	91	276	2073
height/3, width/3	4	1615	155	1962	3736
height, width/4	17	1593	58	268	1936
height/4, width	18	1915	151	336	2420
height/4, width/4	2	1003	204	3495	4704
height, width/5	20	1662	102	312	2096
height/5, width	27	2057	216	408	2708
height/5, width/5	0	491	348	4976	5815

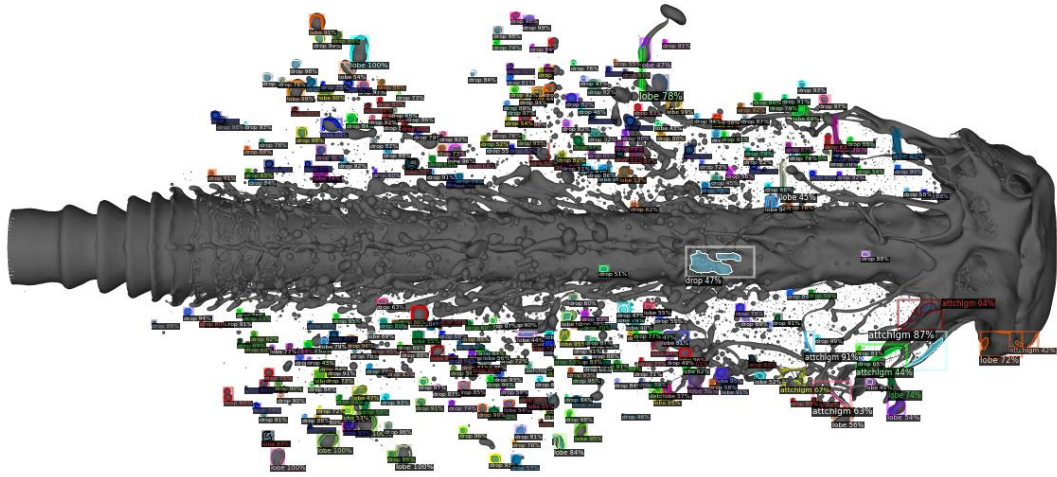


Figure 4.12 One sample image using (height, width/3) window with Base BMask R-CNN and 3000<sup>th</sup> weight, showing the increment in the object detection. This image is comparable with Figure 4.10 (d)

Cascade BMask R-CNN on 1000<sup>th</sup> weight shows better mAP than other 1000<sup>th</sup> models and weights. The results are also better than previous models in the focused crop windows. For (height, width/3), 1697 droplets, 160 lobes; (height, width/4) detected 1959 droplets, 208 lobes, and (height, width/5) detected 2169 droplets, 246 lobes. An increment of droplet detection is seen here, compared to the 1000<sup>th</sup> weight of models: Cascade detected more droplets than FPN based model as 157 on (height, width/3), 166 on (height, width/4), and 119 on (height, width/5) windows, Similar increment is seen with other

objects as on average 193 attached ligament and 116 lobe detection but reduces the detached ligament detected by 19 on average. A summary of the Cascade model with 1000<sup>th</sup> weight is shown in Table 4.10.

Table 4.10 Comparison between crop windows for Cascade BMask R-CNN with 1000<sup>th</sup> weight.

Crop Window Shape	Total Detached Ligaments	Total Droplets	Total Attached Ligaments	Total Lobes	Sum
height, width	59	791	63	44	957
height, width/3	47	1697	213	160	2117
height/3, width	32	1476	166	112	1786
height/3, width/3	19	1852	452	900	3223
height, width/4	68	1959	249	208	2484
height/4, width	47	1721	218	175	2161
height/4, width/4	20	1364	723	1896	4003
height, width/5	84	2169	309	246	2808
height/5, width	35	1756	197	164	2152
height/5, width/5	13	989	1276	2836	5114

The Cascade model with 3000<sup>th</sup> weight shows the highest mAP50 of 66.959 and mAP75 of 47.449 compared to FPN and Base models of all weights. With 3000<sup>th</sup> weight, it detects 1578 droplets, 98 lobes on (height, width/3), 1630 droplets, 145 lobes on (height, width/4), 1762 droplets, 188 lobes on (height, width/5) window. The droplet detection increased compared with the other two models with this weight as 249 with (height, width/3), 30 with (height, width/4), 78 with (height, width/5) from FPN; 90 with (height, width/3), 37 with (height, width/3) and 100 with (height, width/3) from Base BMask R-CNN. A summary is shown in Table 4.11 for cascade with 3000<sup>th</sup> weight, and details comparison of both weight's raw result is shown in Appendix Table A.5 on Page 127. One sample image from Base BMask R-CNN using (height, width/3) with 3000<sup>th</sup> weight is shown in Figure 4.13.

Table 4.11 Comparison between crop windows for Cascade BMask R-CNN with 3000<sup>th</sup> weight.

Crop Window Shape	Total Detached Ligaments	Total Droplets	Total Attached Ligaments	Total Lobes	Sum
height, width	46	833	28	27	934
height, width/3	19	1578	17	98	1712
height/3, width	34	1846	37	163	2080
height/3, width/3	6	1696	75	1346	3123
height, width/4	27	1630	29	145	1831
height/4, width	46	2083	54	203	2386
height/4, width/4	1	1257	104	2677	4039
height, width/5	24	1762	38	188	2012
height/5, width	50	2283	66	224	2623
height/5, width/5	0	575	123	3951	4649

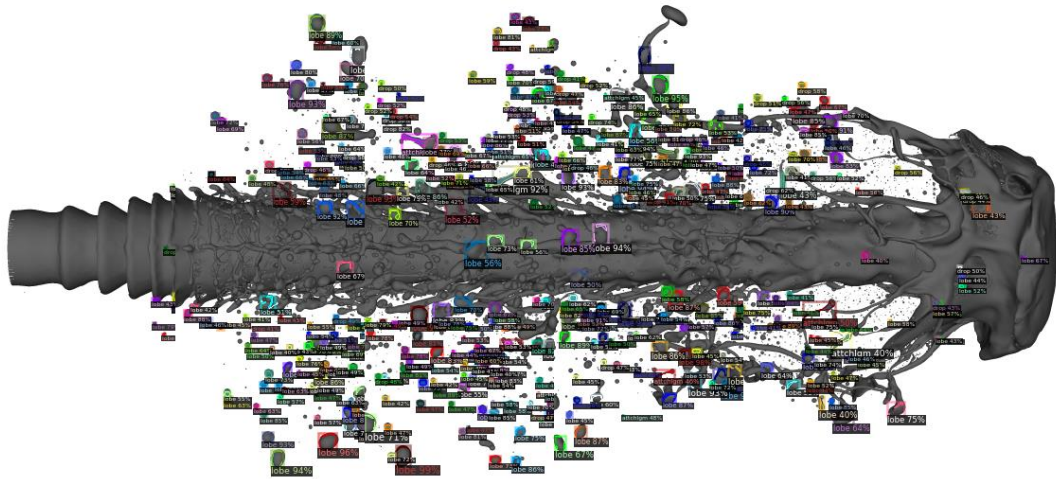


Figure 4.13 One sample image using (height, width/3) window with Cascade BMask R-CNN and 3000<sup>th</sup> weight, showing the increment in the object detection. This image is comparable with Figure 4.10 (f)

Based on the result analysis of BMask R-CNN with different weights and model configurations, it can be noticed that, dividing width is giving more suitable detection result than dividing height and width both. FPN based BMask R-CNN has shown a balanced object detection of high droplet count and low lobe count with 3000<sup>th</sup> weight, though the cascade model's detection with 3000 epoch is highest on droplet count for

seven original images compared with other models based on its detection rate, mAP, and human visual verification on width window cropping values.

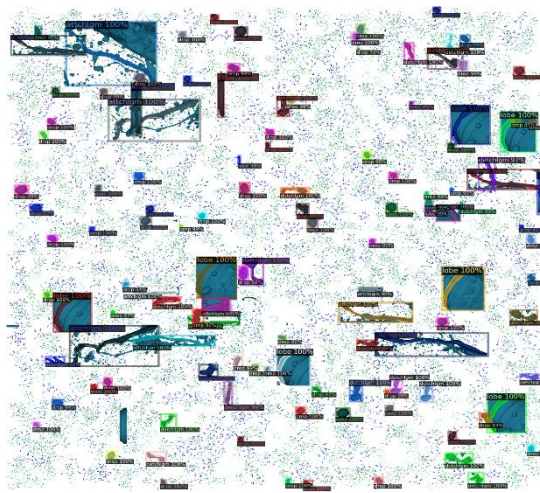
### **4.3 Result of Transformer Models**

As transformer being the latest and one of the most popular architectures, two models are trained for instance segmentation of the targeted image set to compare the results with previously evaluated Mask and BMask R-CNN model results. A total of 20k epochs were trained for FastInst and PatchDCT models with RTX 3090 considering previously mentioned configurations. Among all the epochs, 5k, 15k and 20k are considered for result comparison. For mAP calculations, 75% IoU has been maintained just like previous model comparisons. Detailed results are shown in the following sub-sections.

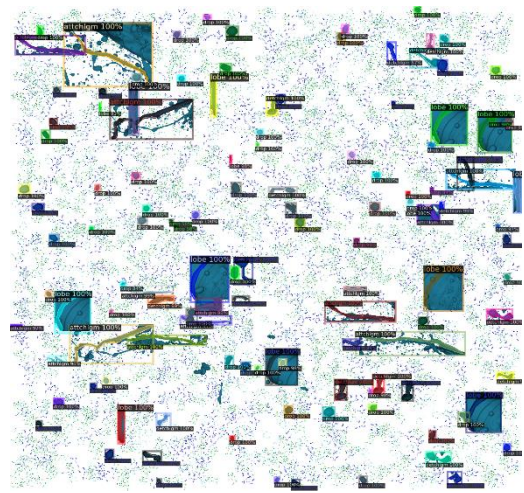
#### **4.3.1 Object detection on the augmented and preliminary dataset**

Maintaining the previous result flow, both transformer models were initially tested on augmented test dataset, later original seven image datasets. Four checkpoints were selected to compare the result on every five thousand epochs. PatchDCT achieved an average precision of 78.92% with 75% IoU on 5000 epochs, which later increased to 80.99% in 10k epochs, 80.08% on 15k and finally 82.69% in 20k. Applying the checkpoint weights to the same previously used test augmented image, a total of 123 to 125 objects are detected. Segmented objects result on test image is shown in Figure 4.14 and result comparison is in Figure 4.15.

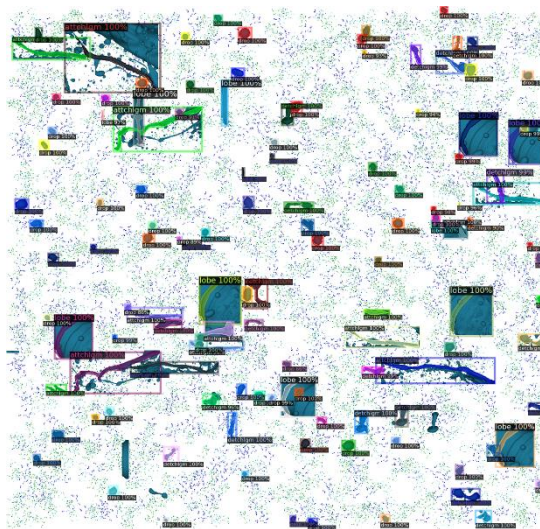




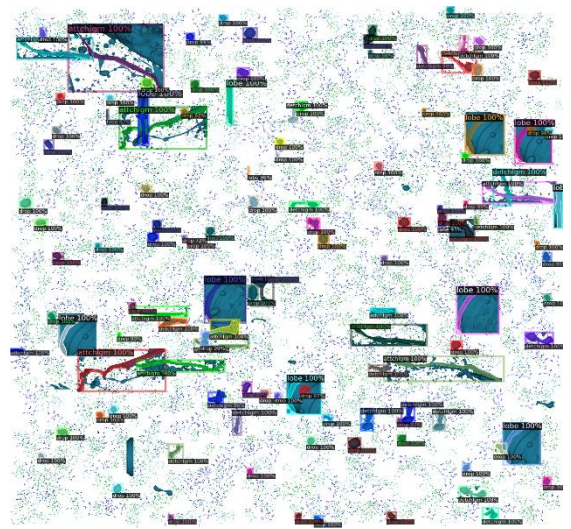
(a) Result of 5k epochs



(b) Result of 10k epochs



(c) Result of 15k epochs



(d) Result of 20k epochs

Figure 4.14 Result of object segmentation on different epochs for PatchDCT. Result for 5k epochs is shown in (a), 10k epochs is shown in (b), 15k epoch result is shown in (c) and 20k epoch result is shown in (d).

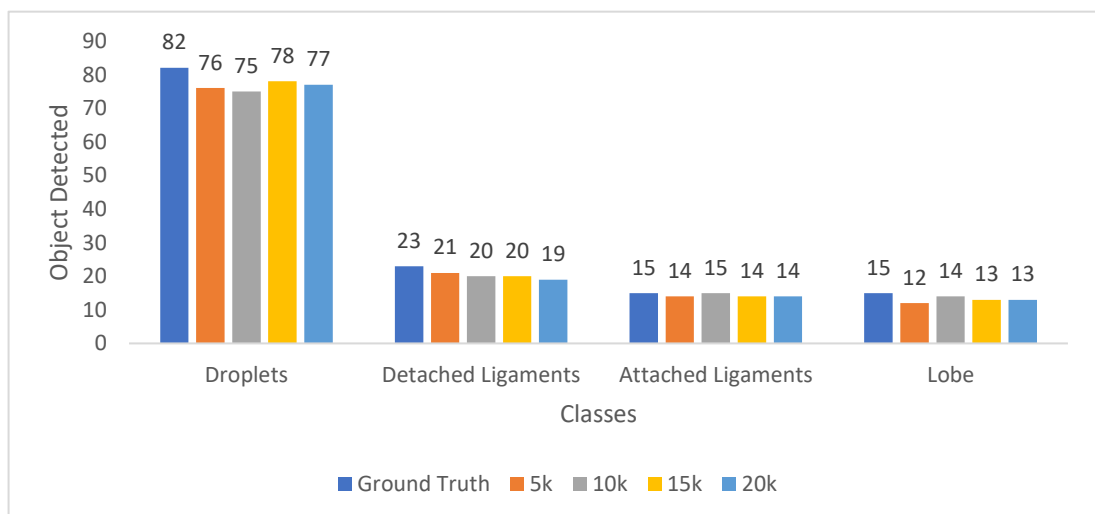


Figure 4.15 Comparison of detected objects on different weights in one image for PatchDCT.



The initial training of PatchDCT on augmented dataset is showing better result comparing with Mask R-CNN on the object count. The maximum droplet detected with Mask R-CNN (5791 epochs) was 74, and with BMask R-CNN (3000 epochs) was 60, and with PatchDCT (15k epochs), it is 78 with 15k epochs. It gives a very promising result and opens the door to explore further on the other two datasets aiming to achieve higher object detection.

After the testing on augmented images, again extracted checkpoints are being tested on original 7 images. Similar kind of detection increment has been seen based on incremental training iterations. On first 5k epochs, the maximum droplets detected in 99 on 5<sup>th</sup> image, which increased to 119 in 10k epochs. Later improvements have been seen again on next 15k and 20k epochs having 105 and 126 droplets being detected maintaining the highest object detection comparing with other images. A total summary has been shown in Figure 4.16 and one sample test image with object segmentation result has been shown in Figure 4.17.

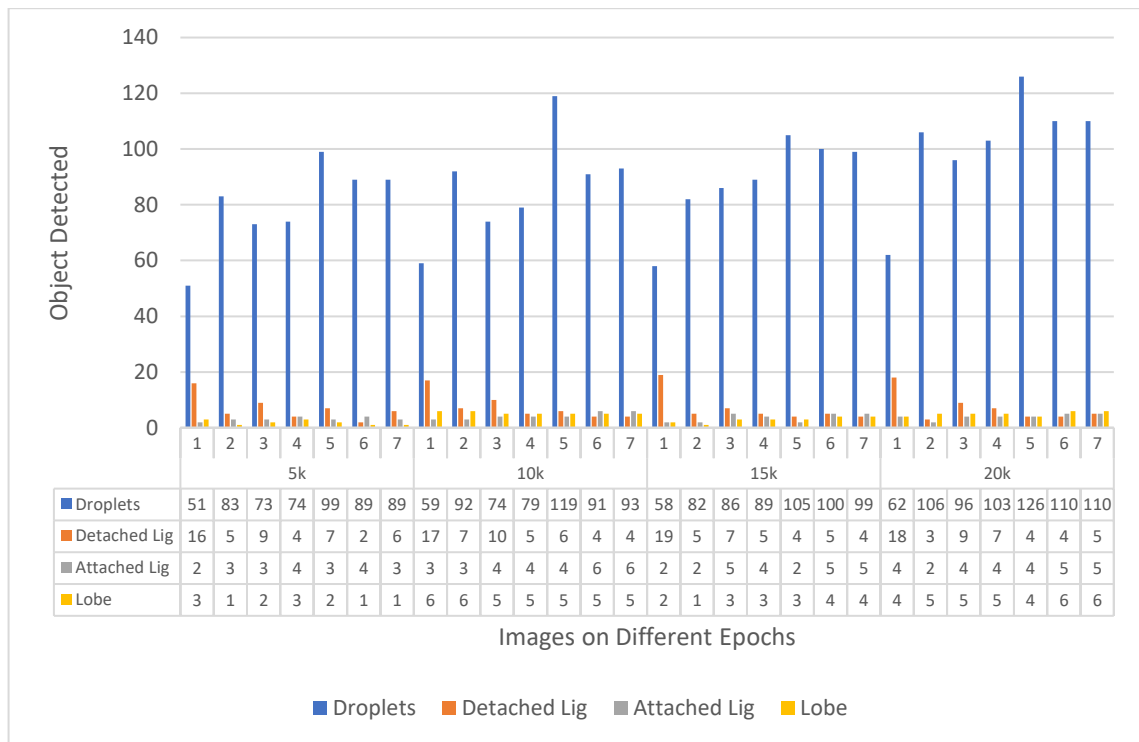
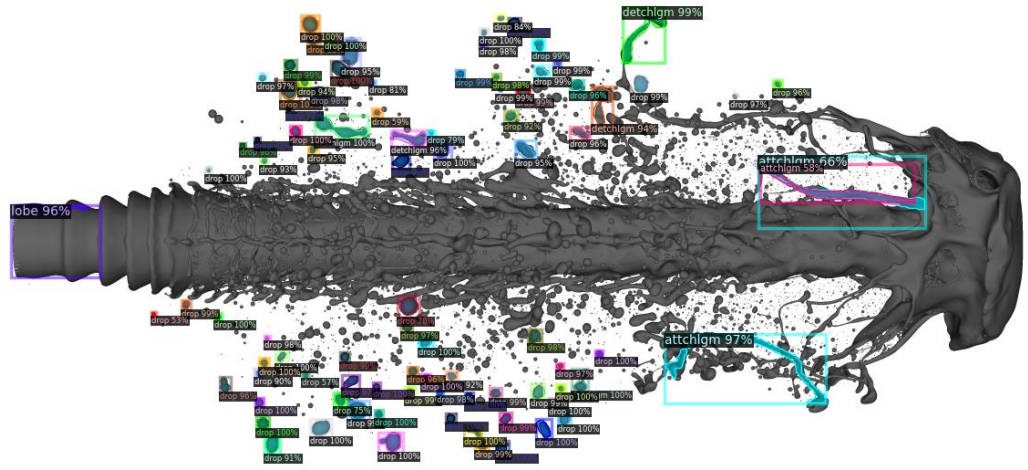
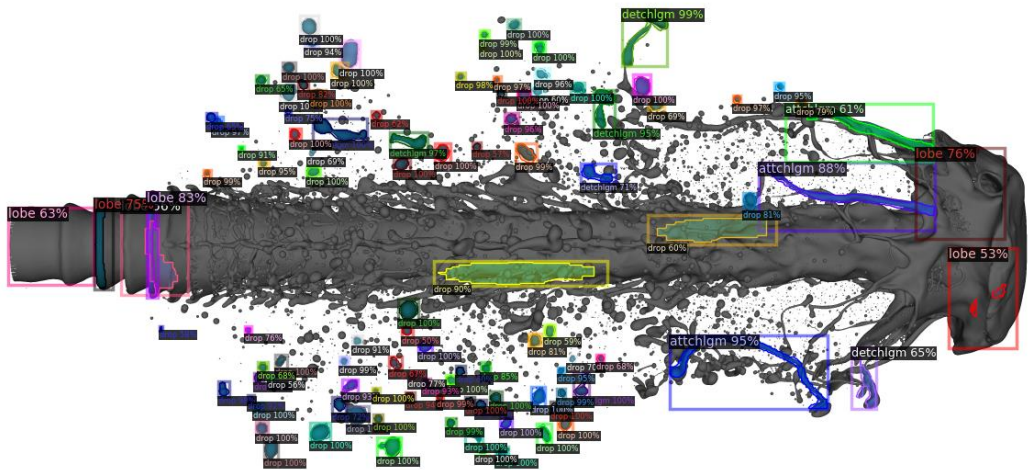


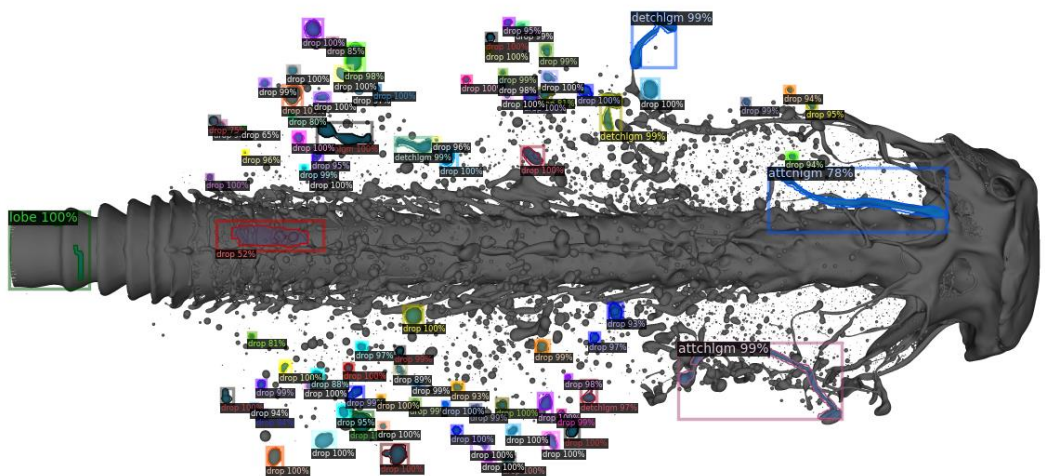
Figure 4.16 Summary of detected objects on original seven images with different weights for PatchDCT.



(a) Original image test using 5k epochs with PatchDCT.

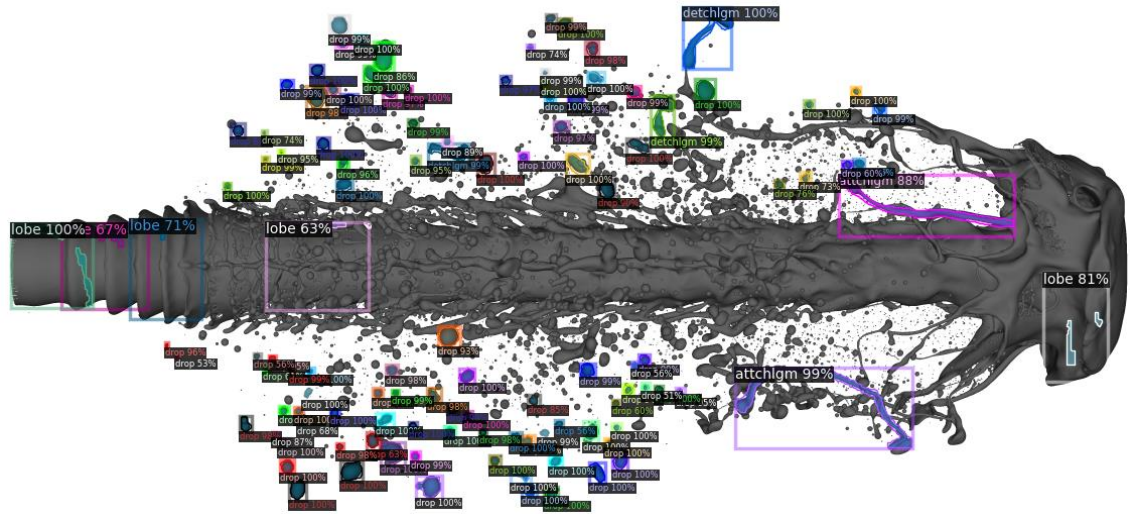


(b) Original image test using 10k epochs with PatchDCT.



(c) Original image test using 15k epochs with PatchDCT.

Figure 4.17 One original image test result shown for (a) 5k epochs, (b) 10k epochs, (c) 15k epochs, (d) 20k epochs using PatchDCT model.



(d) Original image test using 20k epochs with PatchDCT.

Figure 4.17, continued: One original image test result shown for (a) 5k epochs, (b) 10k epochs, (c) 15k epochs, (d) 20k epochs using PatchDCT model.

Similar experiments were conducted for FastInst model, one core problem with this model's default GitHub codebase is that it will show total object in inference regardless of assigned confidence threshold. So, a customized function was written to override default behavior and have the expected filtered result on 50% minimum confidence threshold. Four weight checkpoints were taken for testing and comparing with PatchDCT. With 5k epochs, FastInst shows a 41.49% average precision with 75% IoU, 46.93% AP75 for 10k epochs, 46.89% AP75 for 15k epochs and 46.04% for 20k epochs. One selected image (same image for all model testing) from augmented dataset has been tested for object count, refers to a total of 95 object detection with 5k epochs where droplet count is 56, total of 91 objects for 10k epochs with 54 droplets, 92 total objects for 15k epochs with 54 droplets and finally 90 objects for 20k epochs having 55 droplets. Inferred image from each epoch for FastInst is shown in Figure 4.18 and comparison graph is shown in Figure 4.19. Comparing with PatchDCT, lower average precision and object detection is seen.



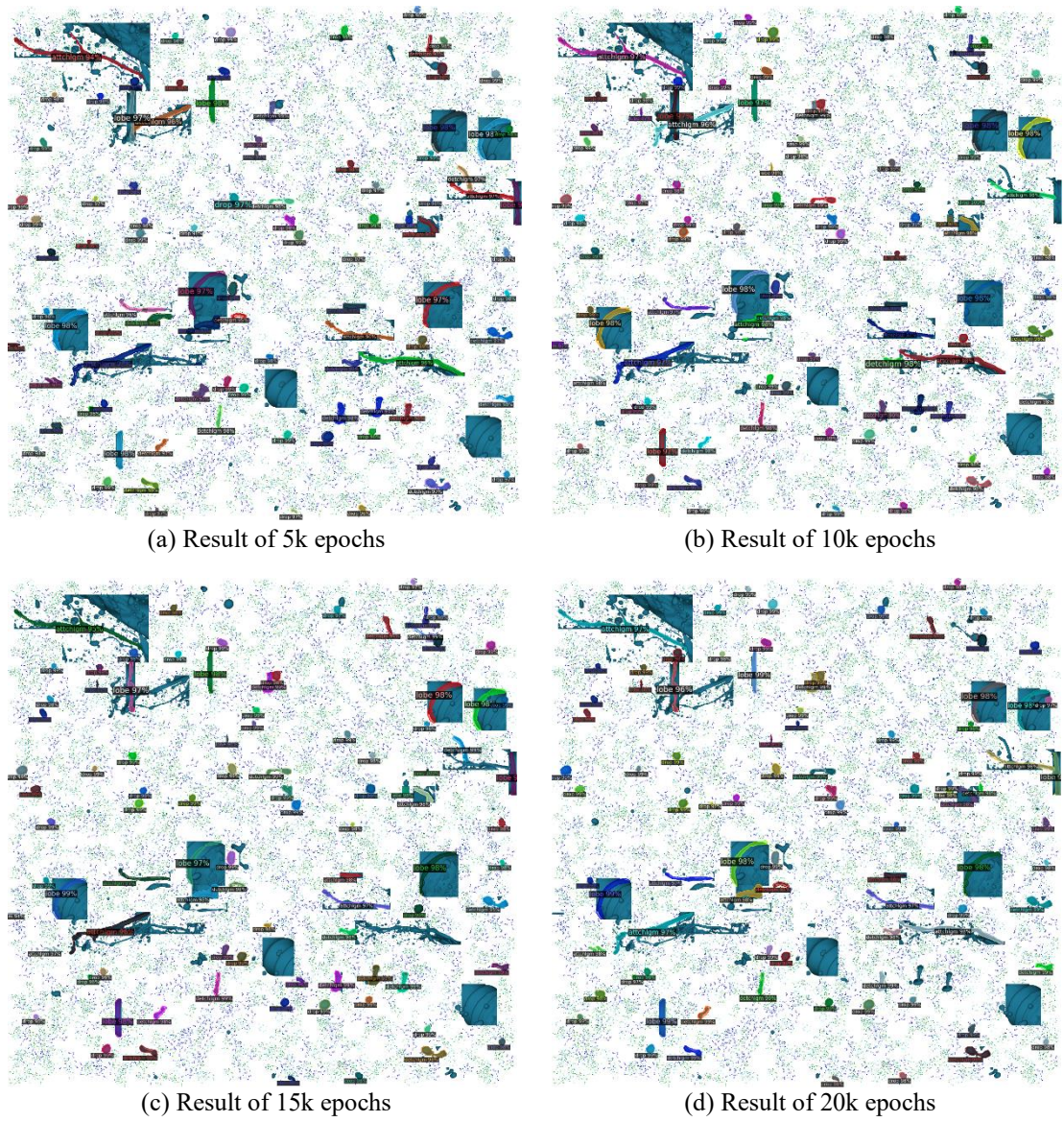


Figure 4.18 Result of object segmentation on (a) 5k epochs, (b) 10 epochs, (c) 15k epochs and (d) 20k epochs for FastInst.

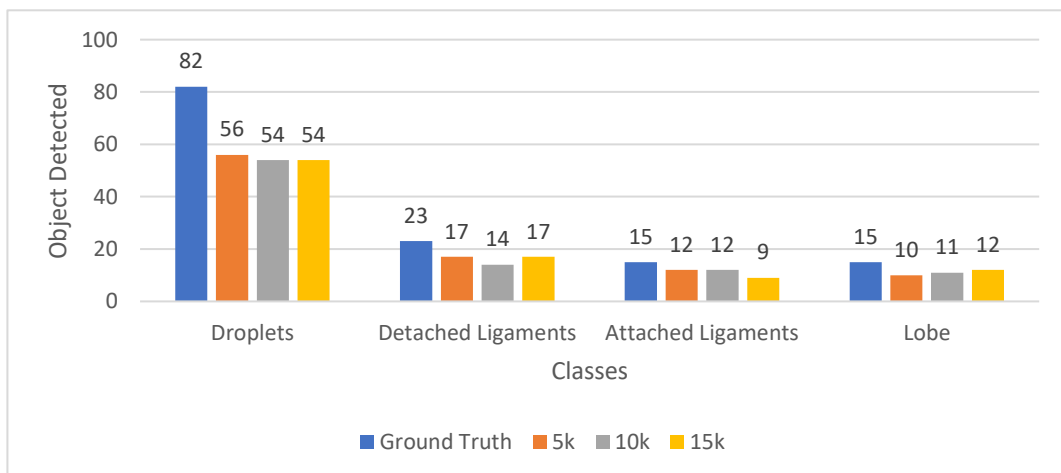


Figure 4.19 Comparison of detected objects on different weights in one image for FastInst.

Though it was expected that, FastInst would perform at least nearly similar to PatchDCT but for the architectural differences, it is losing object count comparing with other models. But from BMask R-CNN it has already been noticed that it is possible for model to perform better on original images even though augmented test object detection rate is low. From this supporting point, further experiments continue with two other image datasets consisting of seven and 1573 images.

After the testing with augmented test dataset, original seven images were inferenced with four model checkpoints. On 5k weight, a maximum droplet of 67 has been detected in 6<sup>th</sup> image, 62 droplets are detected on 10k weight on same image. With slight increment, a total of 63 droplets are detected with 15k weight and finally 66 droplets are detected with 20k weight. Compared with PatchDCT, decrement is seen on object detection, but increment is seen with attached ligament detection. A full comparison of object detection on different weights is shown in Figure 4.20 for FastInst model, a sample image from each inference is also shown in Figure 4.21, similar image is used for fare visual comparison with other models.

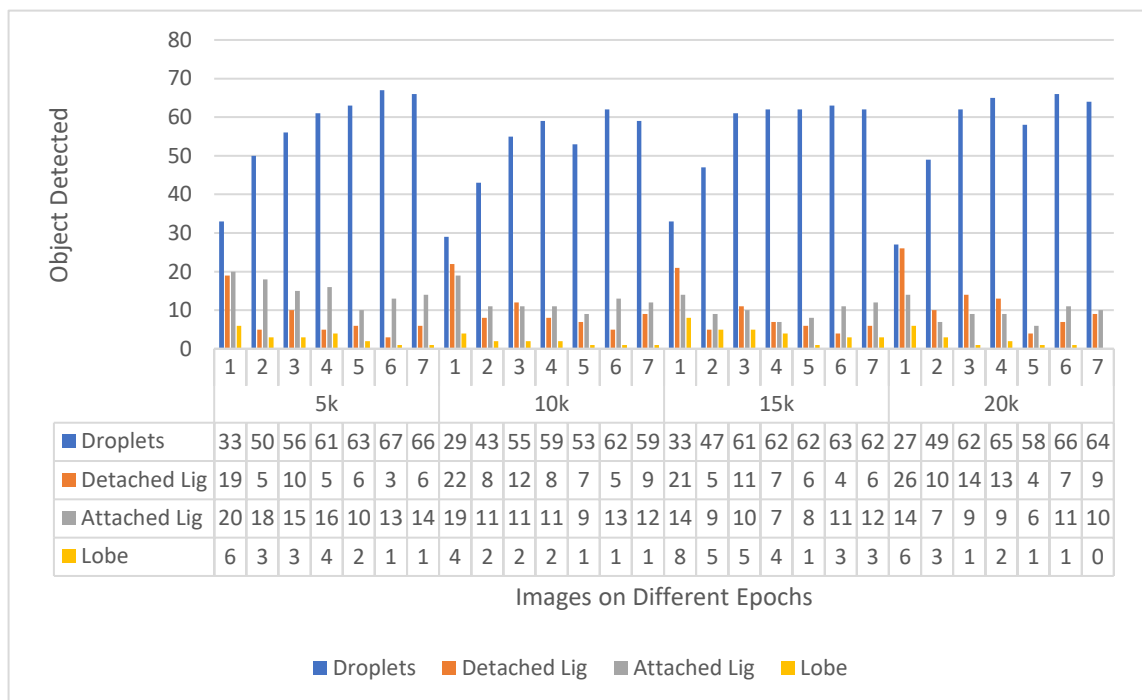
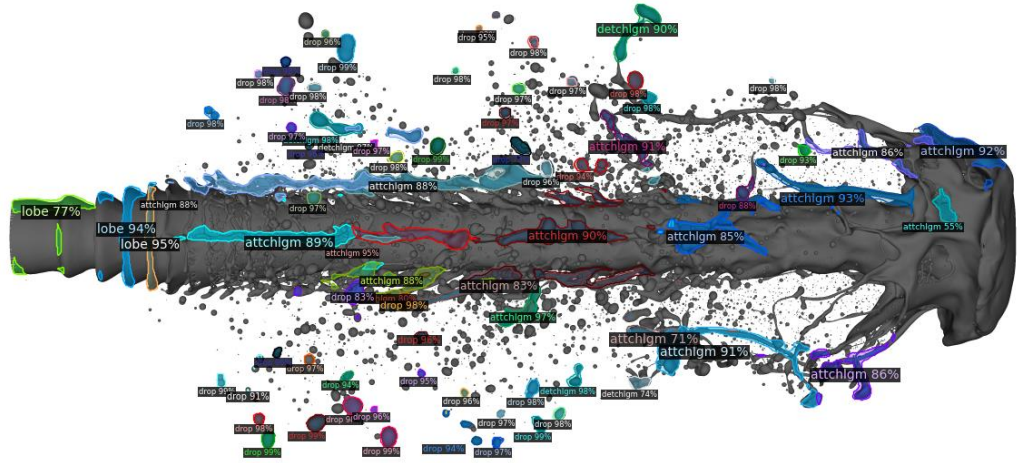
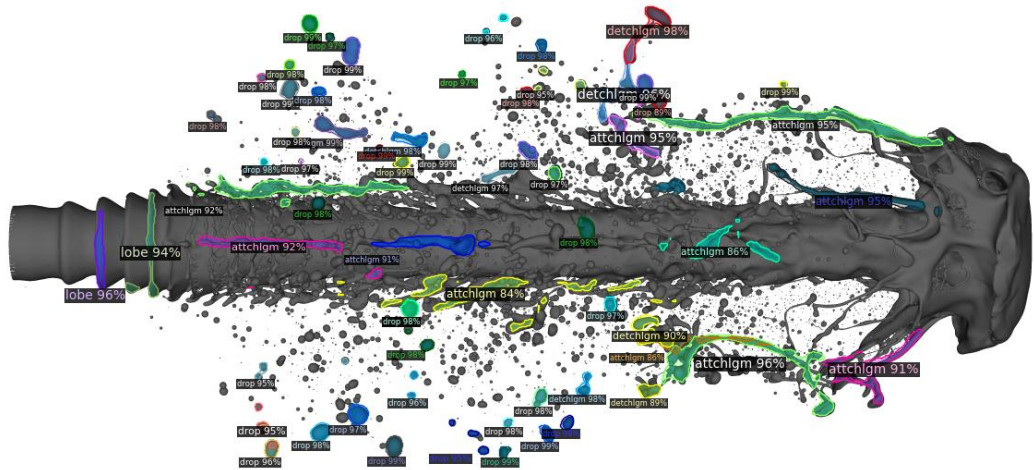


Figure 4.20 Summary of detected objects on original seven images with different weights for FastInst.

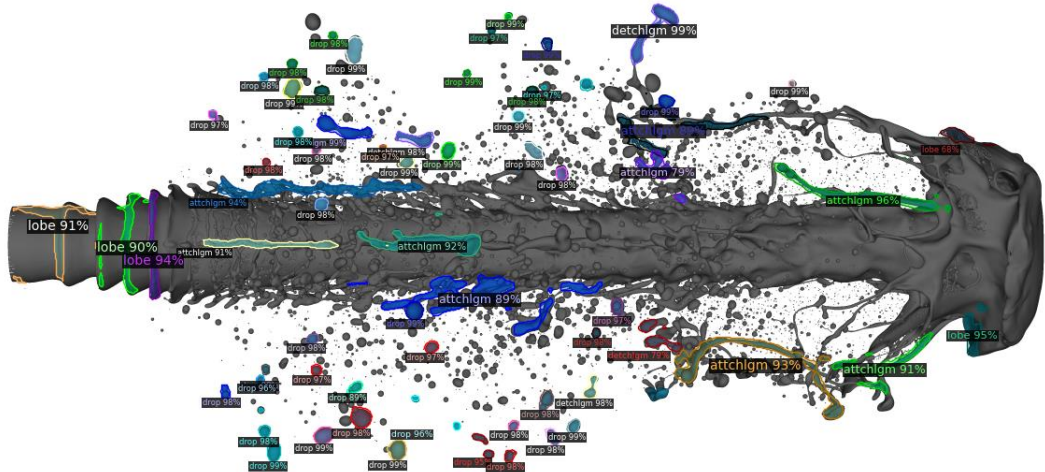




(a) Original image test using 5k epochs with FastInst.



(b) Original image test using 10k epochs with FastInst.



(c) Original image test using 15k epochs with FastInst.

Figure 4.21 One original image test result shown for (a) 5k epochs, (b) 10k epochs, (c) 15k epochs, (d) 20k epochs using FastInst model.

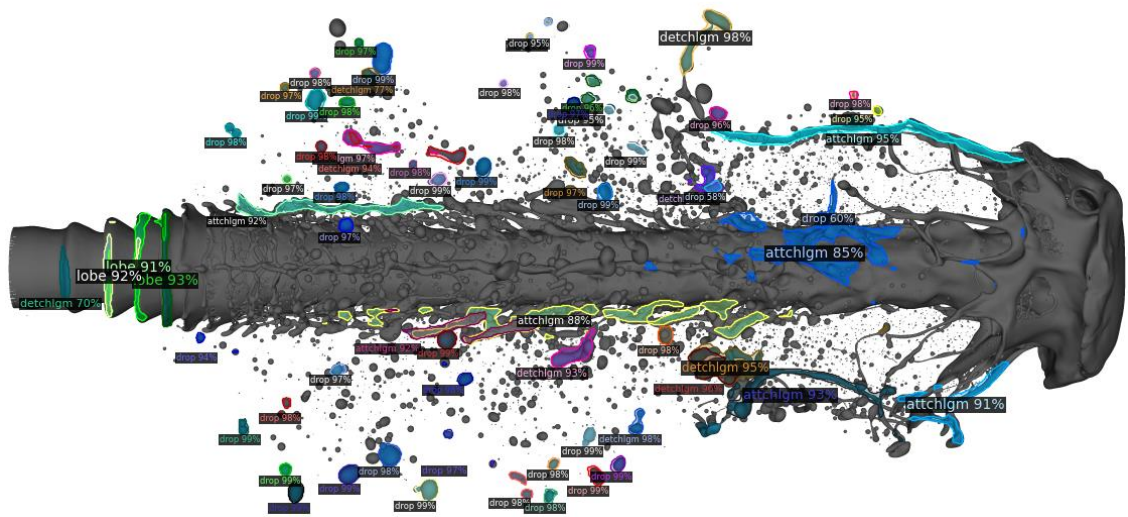


Figure 4.21, continued: One original image test result shown for (a) 5k epochs, (b) 10k epochs, (c) 15k epochs, (d) 20k epochs using FastInst model.

From the visual aspect, transformer models detected good masks on objects, but the number of object detection is low compared with other models after cropping. Between PatchDCT and FastInst, PatchDCT showed good object detection rate along with mean average precision. Further testing is done with cropping window technique and result is shown in next section.

### 4.3.2 Object detection after divide and conquer

Previously mentioned cropping windows are used with both transformer models to evaluate the result, starting with PatchDCT, it showed a good amount of object detection after applying cropping window along with visual verification as no ground truth is available to compare results. Mask R-CNN showed a good performance with (height/3, width/3) window and BMask R-CNN with (height, width/3), so focus will be on these two windows for transformers to see if the result actually improved. Using PatchDCT, without any crop window, applying to seven original images, a total of 642 object are detected with 5k epochs, which is significantly increased for (height, width/3) window to 1765. Also, for other checkpoint weights like 10k, object detection increased from 727 to

2586, for 15k it increased from 713 to 2232 and for 20k it increased from 826 to 1962. Verifying visually, the masks are also good, compared to Mask R-CNN and BMask R-CNN on mean average precision ground. As from previous models' results it is expected that, one test image would have approximately 1500 objects. But PatchDCT shows unexpectedly large object detection count for (height/3, width/3) as 4651, 5971, 5021, 3684 for 5k, 10k, 15k and 20k epochs respectively. When visual check is done on PatchDCT result, it is noticed that a huge number of detected objects are misclassification, and it has detected multiple objects as a small part of the big original object, resulting in overcounting. Hence, for further experiment, (height, width/3) window is chosen as a wise reasonable cropping window for PatchDCT. A detailed comparison is shown in Table 4.12, mentioned window result is marked as light blue color. One sample image from PatchDCT using (height, width/3) with 20k weight is shown in Figure 4.22.

Table 4.12 Comparison of the total detected object for 5k,10k,15k and 20k weight's test result on different crop windows for PatchDCT.

Crop window	Epochs	Detached Ligaments	Droplets	Attached Ligaments	Lobe	Sum
height, width	5k	49	558	22	13	642
	10k	53	607	30	37	727
	15k	49	619	25	20	713
	20k	50	713	28	35	826
height, width/3	5k	39	1657	26	43	1765
	10k	45	2409	46	86	2586
	15k	37	2058	53	84	2232
	20k	39	1819	51	53	1962
height/3, width	5k	51	767	31	21	870
	10k	63	865	43	45	1016
	15k	54	840	34	34	962
	20k	51	856	29	39	975
height/3, width/3	5k	18	3869	47	717	4651
	10k	25	4976	111	859	5971
	15k	7	4063	160	791	5021
	20k	6	3205	182	291	3684
height, width/4	5k	42	1836	17	53	1948
	10k	50	2788	40	99	2977
	15k	38	2154	52	99	2343
	20k	37	1936	37	63	2073
height/4, width	5k	58	980	27	23	1088
	10k	73	1148	37	36	1294
	15k	58	1130	37	25	1250
	20k	63	900	27	28	1018
height/4, width/4	5k	2	3236	81	2387	5706
	10k	4	4560	207	2916	7687
	15k	0	3149	192	2707	6048



Crop window	Epochs	Detached Ligaments	Droplets	Attached Ligaments	Lobe	Sum
	20k	1	2535	403	1169	4108
height, width/5	5k	46	1813	26	72	1957
	10k	57	2454	45	130	2686
	15k	43	2118	47	115	2323
	20k	41	1834	48	65	1988
height/5, width	5k	69	1105	47	33	1254
	10k	79	1425	59	54	1617
	15k	63	1181	45	23	1312
	20k	63	900	40	33	1036
height/5, width/5	5k	1	3536	126	4468	8131
	10k	3	4814	327	5129	10273
	15k	0	2262	281	5000	7543
	20k	2	1805	646	2159	4612

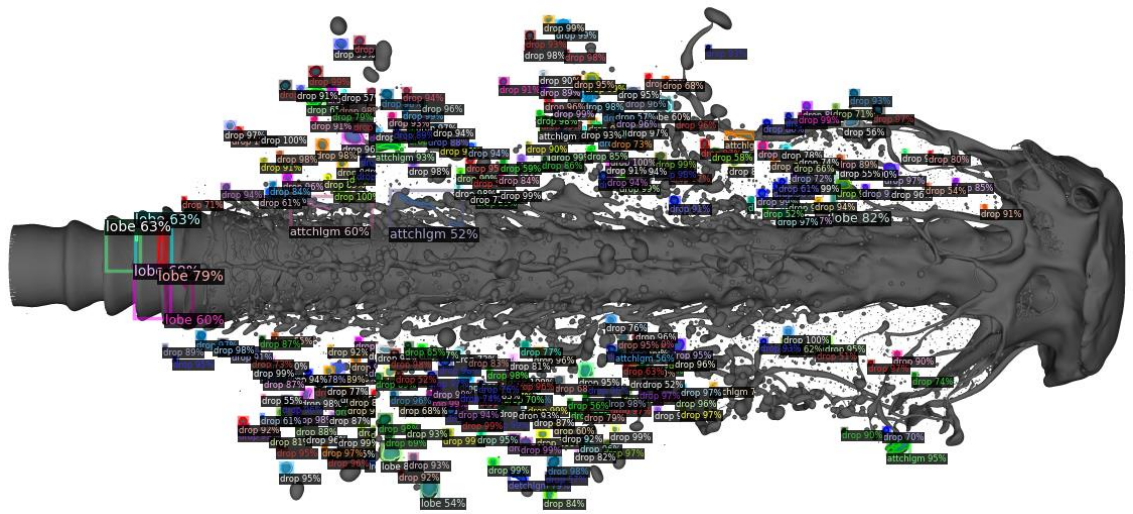


Figure 4.22 One sample image using (height, width/3) window with PatchDCT and 20k weight, showing the increment in the object detection. This image is comparable with Figure 4.17 (d).

For FastInst, focusing on similar crop window to compare the result with PatchDCT and other models along with visual verification. Almost same kind of multiple objects misclassification has been noticed for (height/3, width/3) and unexpected object detection is seen above 3000 number having more than 2000 droplets each image. For (height, width/3) a more reasonable object detection is seen and validated by visual inspection. For 5k epochs, a total of 1496 objects are identified, for 10k the number of droplet detection is reduced totaling to 1382 objects, for 15k epochs the total object count is 1444 where droplets are 1075 and finally for 20k epochs, a total of 1371 objects are detected.

Though (height, width/3) detected low number of objects compared with other crop window, but it shows good masks prediction and reasonable detection number. Hence, this window is selected for further inspection on other experiments. A detailed object detection count on crop windows for FastInst is shown in Table 4.13 (mentioned window result is marked as light blue color) and comparison of PatchDCT and FastInst raw value for 10k weight is shown in Table A.6. One sample image from FastInst using (height, width/3) with 20k weight is shown in Figure 4.23.

Table 4.13 Comparison of the total detected object for 5k,10k,15k and 20k weight's test result on different crop windows for FastInst.

Crop window	Epochs	Detached Ligaments	Droplets	Attached Ligaments	Lobe	Sum
height, width	5k	54	396	106	20	576
	10k	71	360	86	13	530
	15k	60	390	71	29	550
	20k	83	391	66	14	554
height, width/3	5k	89	1032	289	86	1496
	10k	113	940	252	77	1382
	15k	77	1075	209	83	1444
	20k	138	965	190	78	1371
height/3, width	5k	79	610	247	44	980
	10k	116	585	189	43	933
	15k	88	665	153	74	980
	20k	170	627	143	43	983
height/3, width/3	5k	170	2667	715	103	3655
	10k	221	2396	631	139	3387
	15k	102	2735	418	234	3489
	20k	357	2505	385	177	3424
height, width/4	5k	95	1110	345	153	1703
	10k	144	1017	296	138	1595
	15k	94	1136	236	179	1645
	20k	183	1079	219	133	1614
height/4, width	5k	83	706	305	67	1161
	10k	128	704	229	58	1119
	15k	89	737	182	94	1102
	20k	199	732	152	65	1148
height/4, width/4	5k	218	3933	986	254	5391
	10k	302	3497	868	318	4985
	15k	119	3703	687	469	4978
	20k	488	3429	640	325	4882
height, width/5	5k	114	1291	421	131	1957
	10k	167	1164	366	128	1825
	15k	106	1306	289	201	1902
	20k	245	1189	279	119	1832
height/5, width	5k	90	821	327	74	1312
	10k	140	794	234	72	1240
	15k	93	791	177	93	1154
	20k	215	789	169	66	1239
height/5, width/5	5k	396	5066	1512	477	7451
	10k	410	4262	1525	549	6746



473 droplets, 9 detached ligaments, 36 attached ligaments, and 30 lobes were detected. The last image is shown in Figure 4.24 and the video of (height, width/3) window detection result along with all the images is in Appendix B.1 on Page 131.

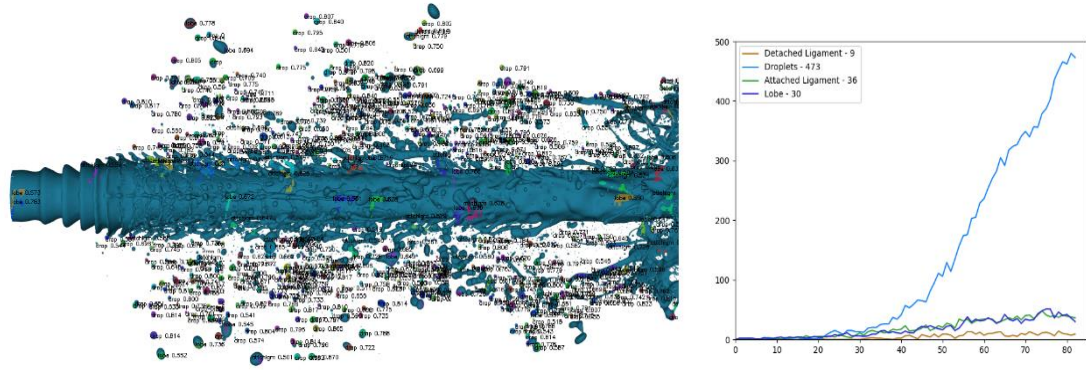


Figure 4.24 Object detection on 1558<sup>th</sup> image with 5791<sup>st</sup> weight of Mask R-CNN and (height, width/3) cropping window.

With BMask R-CNN, the cascade model worked better for seven images, but for the 1573 image set, it has not been able to detect many objects. In this case, FPN based model with 3000<sup>th</sup> weight was able to detect 212 droplets with (height, width/3), 236 droplets with (height, width/4), and 251 droplets with (height, width/5) window. On the other hand, the Base model detected 164, 144, and 149 droplets with (height, width/3), (height, width/4) and (height, width/5) respectively. Finally, the cascade-based model detected the lowest 79, 61, and 61 droplets. One reason could be that the image is continuous and cannot cope with image size and object locations. The last image at 1558 index, the resulting image with FPN based model with (height, width/3), is shown in Figure 4.25. Detailed images with videos are shown in Appendix B.2 on Page 131. As the FPN-based model detected more objects than others, it was chosen to extract the contour list with (height, width/3) cropping window for object tracing.

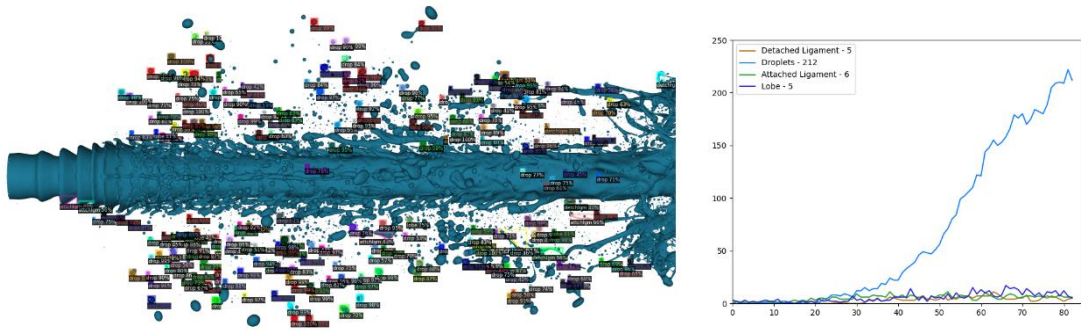


Figure 4.25 Object detection on the 1558<sup>th</sup> image with 3000<sup>th</sup> weight of FPN base BMask R-CNN and (height, width/3) cropping window.

For PatchDCT with (height, width/3) cropping window, selected 83 image set has been evaluated to test the model compatibility and effectiveness. From mAP and seven image tests, it looked like that model would work good on large similar dataset, but unfortunately when tested the model, it shows multiple wrong detection shown as spike in the graph. Though in the last portion of the graph, it shows smooth increment in droplet detection. But for continuous detection rate, it is not suitable for object tracing experiments. The last image from the detection with 10k weight and (height, width/3) crop window is shown in Figure 4.26.

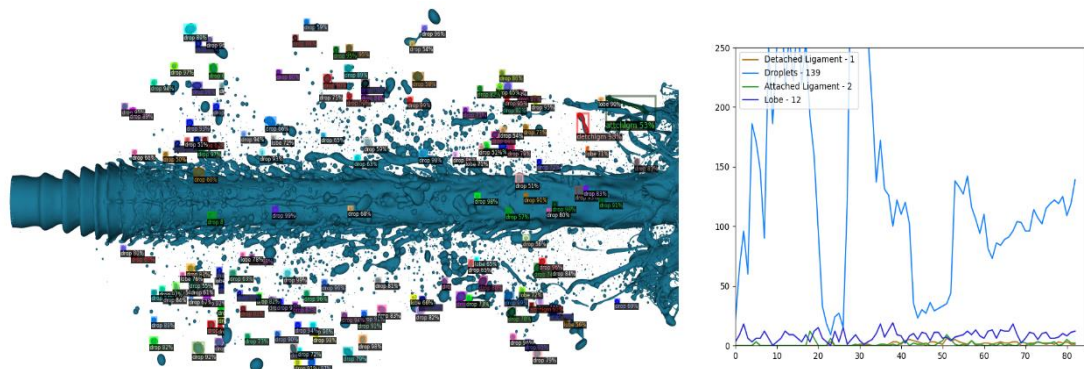


Figure 4.26 Object detection on the 1558<sup>th</sup> image with 10k weight of PatchDCT and (height, width/3) cropping window.

With FastInst, a similar issue is seen in detection rate with (height, width/3) crop window. The total number of object detection is still low, and awkward spike is seen throughout the detection graph. Though at the last portion of the detection, a smooth upwards trend



has been seen in droplet detection having a maximum count of 171, but at the starting images, model confuses between droplet and lobes. Hence, it also makes it unqualified for object tracing experiment. The last image of the detection is shown in Figure 4.27. Full details of both model's object detection and video is shown in B.3.

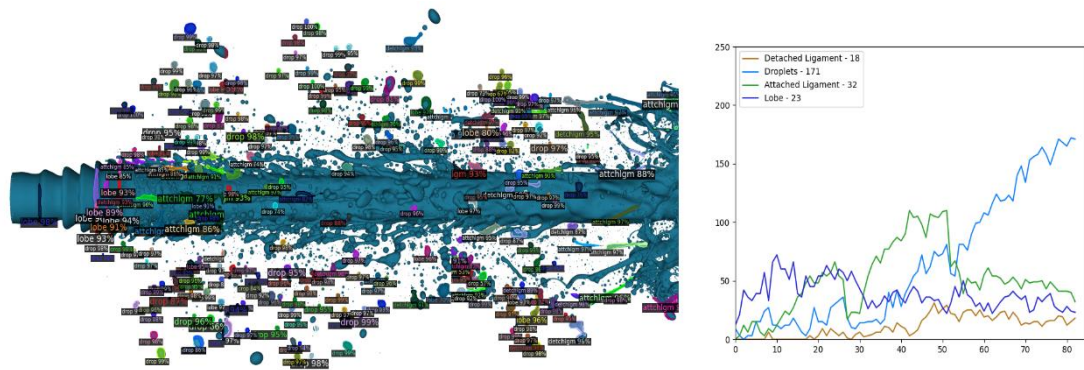


Figure 4.27 Object detection on the 1558th image with 10k weight of FastInst and (height, width/3) cropping window.

As Mask R-CNN and BMask R-CNN have shown consistent object detection throughout the minimized image set, another inferencing has been done for both models on a full-scale dataset consisting of 1573 images. During this inference, the object's properties like mask, class, index, contour, and center of contour were extracted and saved in a JSON file for further processing in object tracing. Due to the inconsistency in object detection between frames, both transformer models are noted as incapable of object tracing tasks.

#### 4.5 Results of Object Tracing

After detecting the objects from the full-length dataset of 1573 images, object tracing has been done for both Mask R-CNN and BMask R-CNN. As one object is not guaranteed to be detected on each frame, it is essential to handle skipping objects. Multiple skipping windows have been used to get the smooth tracing result. Selecting one certain point in a frame and an object within that frame, tracing began to find near objects (based on radius threshold  $r$ ) in the given distance. When an object is in the range, the selected initial object will mark it as a contacted object and keep tracing it throughout the rest of the frame. The

tracing process also works when some object comes in range from outside. This process is named Hot Zone Tracing, which is based on the idea of COVID-19 contact tracing. From this tracing result, the goal is to gain the object location information in time to analyze the object density in a certain place of the combustion chamber. To begin the tracing, a correspondence list is calculated, and the 1573<sup>rd</sup> image with the 5791<sup>st</sup> weight of Mask R-CNN (which is the last image consisting of a complete correspondence list) is shown in Figure 4.28 (with the process). The video consisting of complete correspondence list tracing is shown in Appendix B.4 on Page 131. For FPN-based BMask R-CNN, the correspondence tree is shown in Appendix B.5 on Page 131 using 3000<sup>th</sup> weight and (heigh, width/3) crop window. Object hot zone tracing result is discussed in the following subsection.

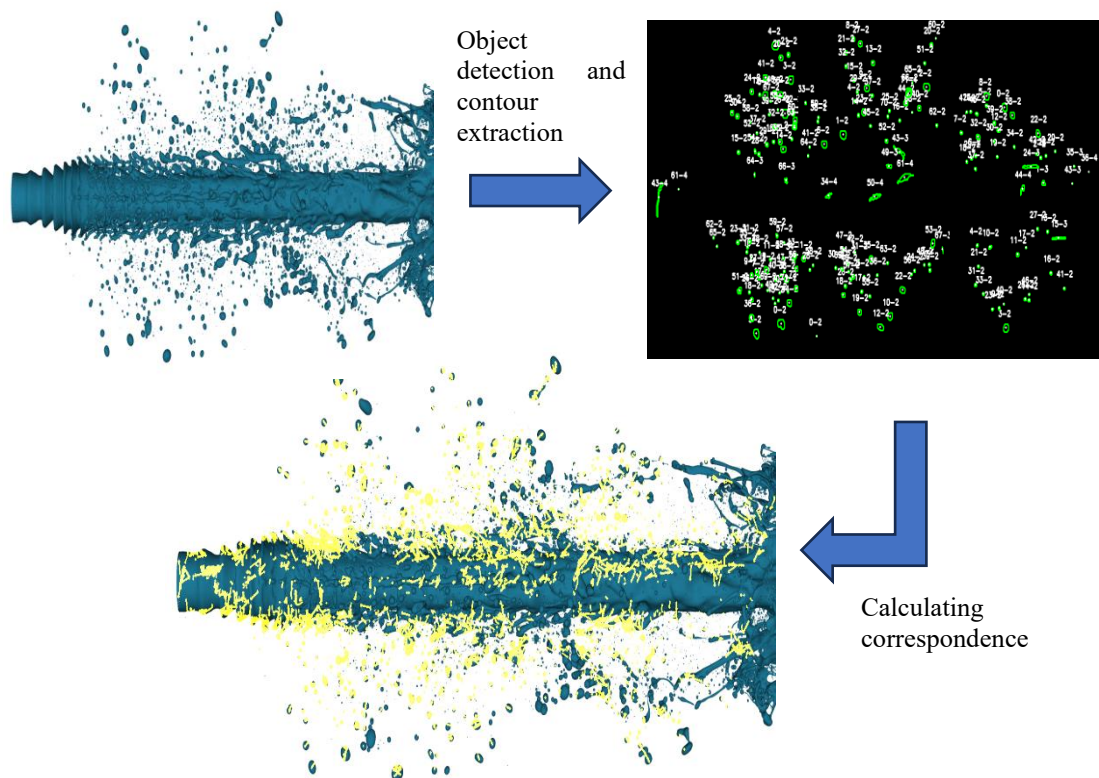


Figure 4.28 Correspondence detection of 1573<sup>rd</sup> image with 5791<sup>st</sup> weight of Mask R-CNN.

#### **4.5.1 DeepSort Tracing**

DeepSort generally depends on an object detection model for object bounding box inputs. Later, it processes those box inputs and assigns a number to each box based on its location; subsequent box locations are assigned the same number considering as one object. Applying DeepSort with Mask R-CNN based object detection model with 5791<sup>st</sup> weight and (height, width/3) window, a video is generated showing complete object detection and tracing throughout the total 1573 images in Appendix B.6 on Page 131. DeepSort can do the tracing on one single object. It cannot identify if an object shifts its class or breaks into another object class or cannot identify collisions. One single object tracing video with a similar configuration is shown in Appendix B.7 on Page 131.

#### **4.5.2 Object Hot Zone Tracing – with Mask R-CNN**

There are a total of 1573 images available in the primary dataset. Initially, whole correspondence values are extracted from this dataset using the (height, width/3) crop window on 5791<sup>st</sup> weight with the method referred to in subsection 3.7. From which 840<sup>th</sup> image was randomly selected where there are 49 object correspondences are available. The 4<sup>th</sup> indexed object is chosen from these objects as it has the highest tracing path consisting of 12340 objects, with an average distance between each object of 3.82 pixels. As the pixel difference is low between frames, a test has been done by skipping some consistent frames in between to speed up the tracing process. Skipping one object in between, a total correspondence list is taken with 786 images. This time, to focus on the same object, the 420<sup>th</sup> image and the 4<sup>th</sup> object are taken. The total tracing length now consists of 4766 objects with an average distance of 4.83 pixels. Two more skipping are done with 2 and 3 frames in between to analyze the tracing further with inconsistent object detection. Two frames skipping results in selecting the 280<sup>th</sup> image and the same 4<sup>th</sup> object. This configuration results in 2978 object tracing with an average distance of 4.91 pixels. 210<sup>th</sup> Image is chosen for three frames skipping analysis on the 4<sup>th</sup> object, resulting



in 2306 object inclusion having an average distance of 5.25 pixels. Considering the original tracing result of 12340 objects, the single-frame skipping result gains 77.24% of the original length, two-frame skipping gains 72.39%, and three-frame skipping gains 74.74%.

Calculating gain on all objects detected on the initial image is pretty high, like 90.59% for the 420<sup>th</sup> frame, 92.55% for the 280<sup>th</sup> frame, and 92.42% for the 210<sup>th</sup> frame. Skipping frames are considered fast and effective if the original number of images is high and real-time processing is needed. Extracting the object tracing tree as (image index, object index), a miniature sample is shown in Figure 4.29, and a full tree image is added as a supplementary document as the image size is pretty high. This tree starts with an image at index 840, and the selected object index is 4. This 4<sup>th</sup> object of the 840<sup>th</sup> image is connected with the 14<sup>th</sup> object of the 841<sup>st</sup> image. For multiple connected nodes, two possible cases can happen: either the 5<sup>th</sup> object of the 891<sup>st</sup> image breaks into the 11<sup>th</sup> and 12<sup>th</sup> objects of the 892<sup>nd</sup> image, or there are two objects in 891<sup>st</sup> image that came from outside the object's searchable region. As they came near, those are now marked as connected and will continue to trace their tracing paths. Object tracing of the original tree and videos are shown in Appendix B.8 on Page 131.

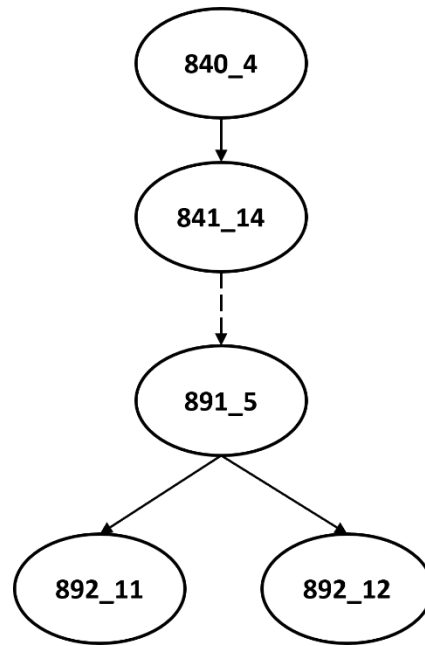


Figure 4.29 Sample object tracing tree.

From the correspondence data, objects' transition rates are also calculated using the object correspondence rule mentioned in Table 3.1. The dataset has been divided into three sections: the first consists of images from 1-523, the second has 524-799, and the third has 800-1573. As objects at the initial stage are low and high at the end, this division would help to understand the transition rate at a specific part of the fluid flow. In the first section, 39.95% of detached ligaments are seen to convert from detached ligaments to droplets, which decreased in middle section to 33.79% but increased to 46.03% in the last image set. The detailed percentage of object transition is illustrated in Figure 4.30.

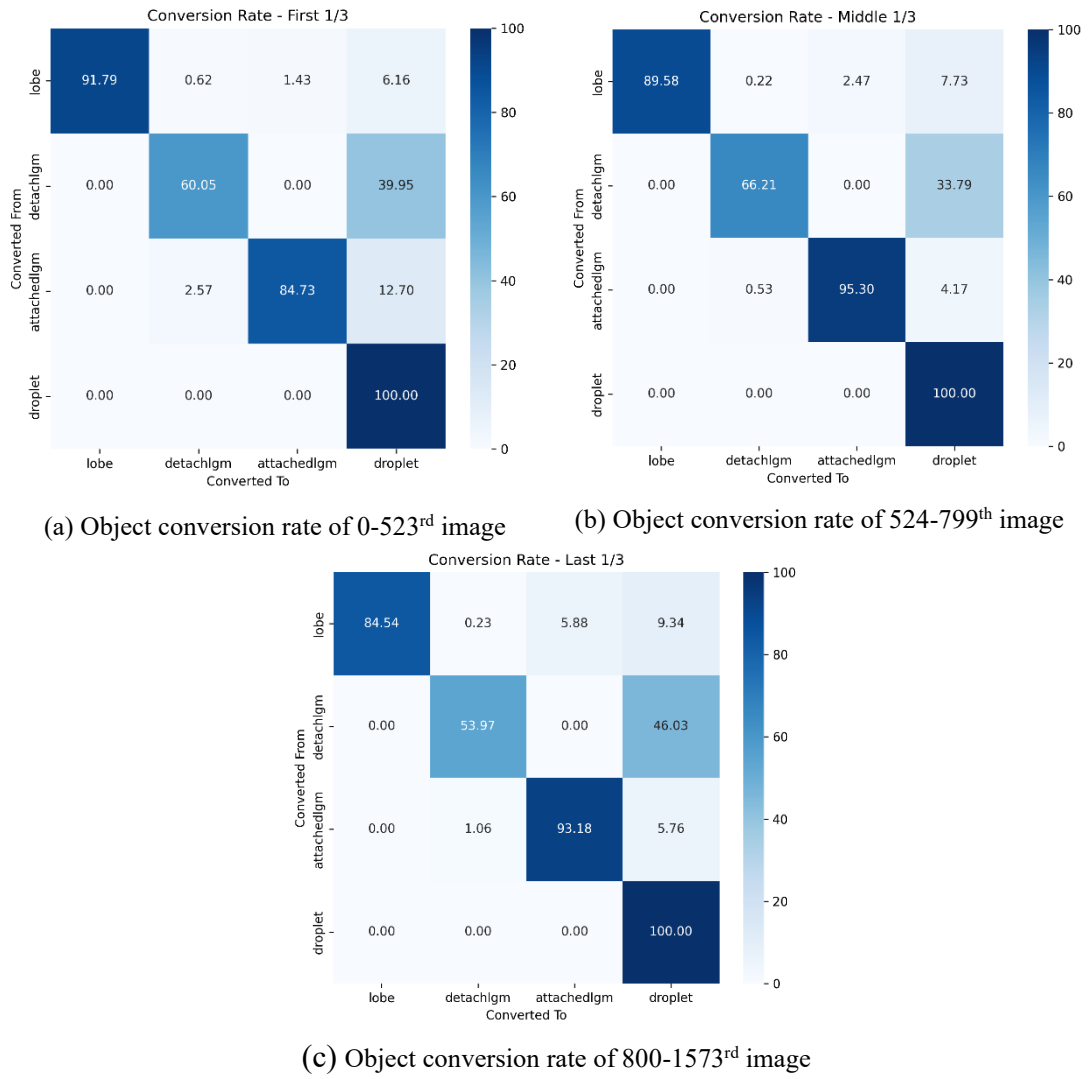


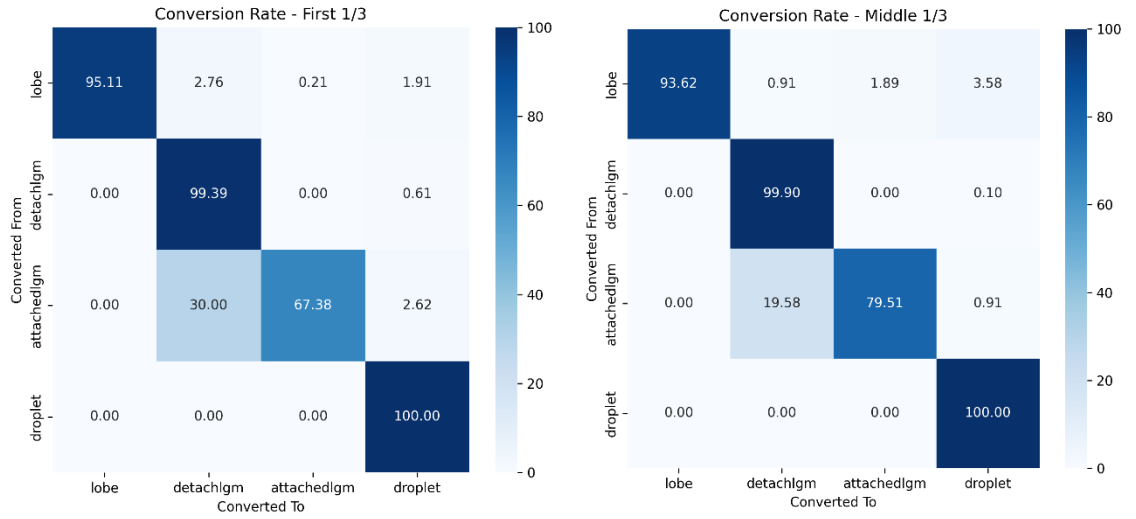
Figure 4.30 Object transition rate from Mask R-CNN on three sections as (a) 0-523<sup>rd</sup> image range, (b) 524-799<sup>th</sup> image range and (c) 800-1573<sup>rd</sup> image range.

### 4.5.3 Object Hot Zone Tracing – with BMask R-CNN

Using (height, width/3) window with FPN-based BMask R-CNN, a complete corresponding object list has been extracted for all 1573 images. The 840th image is selected as a test image with the same skipping method to compare with Mask R-CNN. A total of 63 objects are detected on the 840th frame, having an average distance of 5.69 pixels; the maximum tracing object count is seen for the 25<sup>th</sup> object, with 4190 objects traced, which is around 1/3<sup>rd</sup> of what Mask R-CNN based tracing has detected. For one frame skipping data, the 25<sup>th</sup> object traced a total of 1348 objects, which is a 93.43% gain

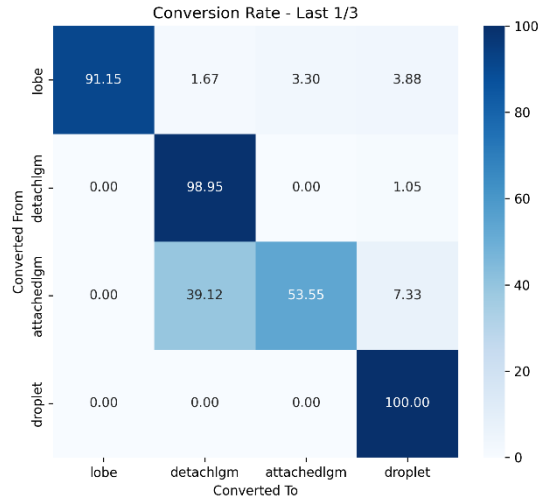
compared to no skipping. The same 25<sup>th</sup> object of two frames skipping data traced only 230 objects, gaining only 16.46%, and three frames skipping traced 220 objects, gaining 16.22%. The BMask R-CNN based tracing is not up to expectations. On total traced objects, one frame skipping achieved a 93.43% gain, two frames skipping achieved a 69.05% gain, and two frames skipping achieved 82.52% gain, which is relatively low and keeping the single object tracing in mind, this model might not be usable in case of real-time object tracing with current configurations. The 25<sup>th</sup> object tracing video with no skipping frame is shown in Appendix B.9 on Page 131. Compared with Mask R-CNN, the 25<sup>th</sup> object that has been detected with BMask R-CNN is the same object with index 16 for Mask R-CNN. A comparison has been drawn between these two object tracing paths for Mask R-CNN; the total traced object count is 11822 with no skipping, which is higher than BMask R-CNN. A comparison video has been generated for better understanding, as shown in Appendix B.10 on Page 131.

A similar object transition rate has been calculated, showing a detached ligament breakdown rate of 0.61% and an attached ligament breakdown rate of 2.62% to droplets in the first image section. In the middle image set, 0.10 of detached ligaments and 0.91% of attached ligaments break down into droplets. In the last section, the 1.05% of detached ligaments and 7.33% of attached ligaments are broken down into droplets, which can be considered an improvement. The detailed percentage of object transition is illustrated in Figure 4.31.



(a) Object conversion rate of 0-523<sup>rd</sup> image

(b) Object conversion rate of 524-799<sup>th</sup> image



(c) Object conversion rate of 800-1573<sup>rd</sup> image

Figure 4.31 Object transition rate for BMask R-CNN on three sections as (a) 0-523<sup>rd</sup> image range, (b) 524-799<sup>th</sup> image range and (c) 800-1573<sup>rd</sup> image range.

## 4.6 Discussion

Fluid object detection has always been a complex task. Even the most popular segmentation models like Mask R-CNN also suffers to detect these objects. Among all the epochs, three weights are selected to compare the results in between to analyze the improvement. After the analysis, it was seen that model detection showed promising results but was not up to expectations. This leads to a problem of object density as the number of objects counted in a small portion of the image is high. Multiple cropping has been applied to reduce this density. Among them, Mask R-CNN and BMask R-CNN acted

differently on different models. Mask R-CNN with 5791<sup>st</sup> weight has an average precision of 74.93 with 75% IoU, whereas FPN-based BMask R-CNN has only 44.68%. On augmented images, testing on the same image, Mask R-CNN detected 74 droplets, 23 detached ligaments, 13 attached ligaments, and 11 lobes, whereas FPN detected only 37 droplets, 17 detached ligaments, 12 attached ligaments, and 11 lobes. Surprisingly, FPN-BMask detected a total of 579 droplets on the initial seven original images, where Mask RCNN detected 517, though another object detection is low as 57 detached ligaments, 46 attached ligaments, and 8 lobes; where Mask RCNN has 70 detached ligaments, 74 attached ligaments, and 29 lobes. Among focused cropping windows of (height/3, width/3), 5791<sup>st</sup> weight of Mask R-CNN detected a total of 2887 droplets, whereas FPN-based BMask R-CNN detected 2339 droplets on this window. Also, the focused window of (height, width/3) for the FPN model detected 1329 droplets, whereas Mask R-CNN detected 1231 droplets on this window; detailed differences are shown in Table 4.14, two mentioned windows has shown better result than others and highlighted as light blue color. Object correspondence is extracted from original 1573 images for object tracing based on the differences.

Table 4.14 Differences between Mask R-CNN with 5791<sup>st</sup> weight and FCN-based BMask R-CNN with 3000<sup>th</sup> weight on different crop windows.

	Total Detached Ligaments		Total Droplets		Total Attached Ligaments		Total Lobes		Sum	
	Mask RCNN	BMask RCNN	Mask RCNN	BMask RCNN	Mask RCNN	BMask RCNN	Mask RCNN	BMask RCNN	Mask RCNN	BMask RCNN
height, width	70	57	517	579	74	46	29	8	690	690
height, width/3	70	38	1231	1329	100	70	56	100	1457	1537
height/3, width	74	44	504	1692	61	118	31	165	670	2019
height/3, width/3	90	13	2887	2339	275	218	245	1253	3497	3823
height, width/4	89	47	1408	1600	90	88	66	128	1653	1863
height/4, width	69	56	473	2113	62	183	13	195	617	2547
height/4, width/4	35	17	2439	2338	419	450	812	2523	3705	5328

Comparing transformer models, PatchDCT shows good results as mAP75 of 81% is achieved with 10k epochs. With this model, a total of 727 objects are detected with original seven image testing and after applying cropping window it increased to a stable number of 2586 total object count for (height, width/3) window. Though other cropping windows show higher or similar object count, after visual inspection, misclassification has been seen with overlapping mask issues. Almost the same cases are seen from FastInst model testing. It also has a stable reliable object count of 1382 on (height, width/3) window with 10k epochs. Other cropping windows messes up with its object detection capabilities due to similar kinds of overlapping issues. Also compared with Mask R-CNN, BMask R-CNN and PatchDCT, FastInst has the lowest mAP values. For both transformer models, inferencing on original 1573 image set, they failed to produce a smooth incremental object count graph, which is crucial to object tracing, hence these models are excluded from object tracing experiments. A comparison table between transformer models are shown in Table 4.15, highlighting the best window result as light blue color.

Table 4.15 Differences between PatchDCT and FastInst with 10k-th weight on different crop windows.

	Total Detached Ligaments		Total Droplets		Total Attached Ligaments		Total Lobes		Sum	
	Patch DCT	Fast Inst	Patch DCT	Fast Inst	Patch DCT	Fast Inst	Patch DCT	Fast Inst	Patch DCT	Fast Inst
height, width	53	71	607	360	30	86	37	13	727	530
height, width/3	45	113	2409	940	46	252	86	77	2586	1382
height/3, width	63	116	865	585	43	189	45	43	1016	933
height/3, width/3	25	221	4976	2396	111	631	859	139	5971	3387
height, width/4	50	144	2788	1017	40	296	99	138	2977	1595
height/4, width	73	128	1148	704	37	229	36	58	1294	1119
height/4, width/4	4	302	4560	3497	207	868	2916	318	7687	4985
height, width/5	57	167	2454	1164	45	366	130	128	2686	1825
height/5, width	79	140	1425	794	59	234	54	72	1617	1240
height/5, width/5	3	410	4814	4262	327	1525	5129	549	10273	6746

Best outcome is seen for Mask R-CNN on 5791 weight, BMask R-CNN on FPN-3000 weight, PatchDCT on 20k weight and FastInst on 10k weight, based on mAP and detected object count. Comparison of images before and after using divide and conquer has already been shown in Figure 4.6, Figure 4.11, Figure 4.13, Figure 4.22 and Figure 4.23. A summary comparison of previously mentioned best models is shown in Table 4.16, mentioning detected number of droplets before and after using divide and conquer technique.

Table 4.16 Comparison of droplet detection before and after divide and conquer technique on four best model weights.

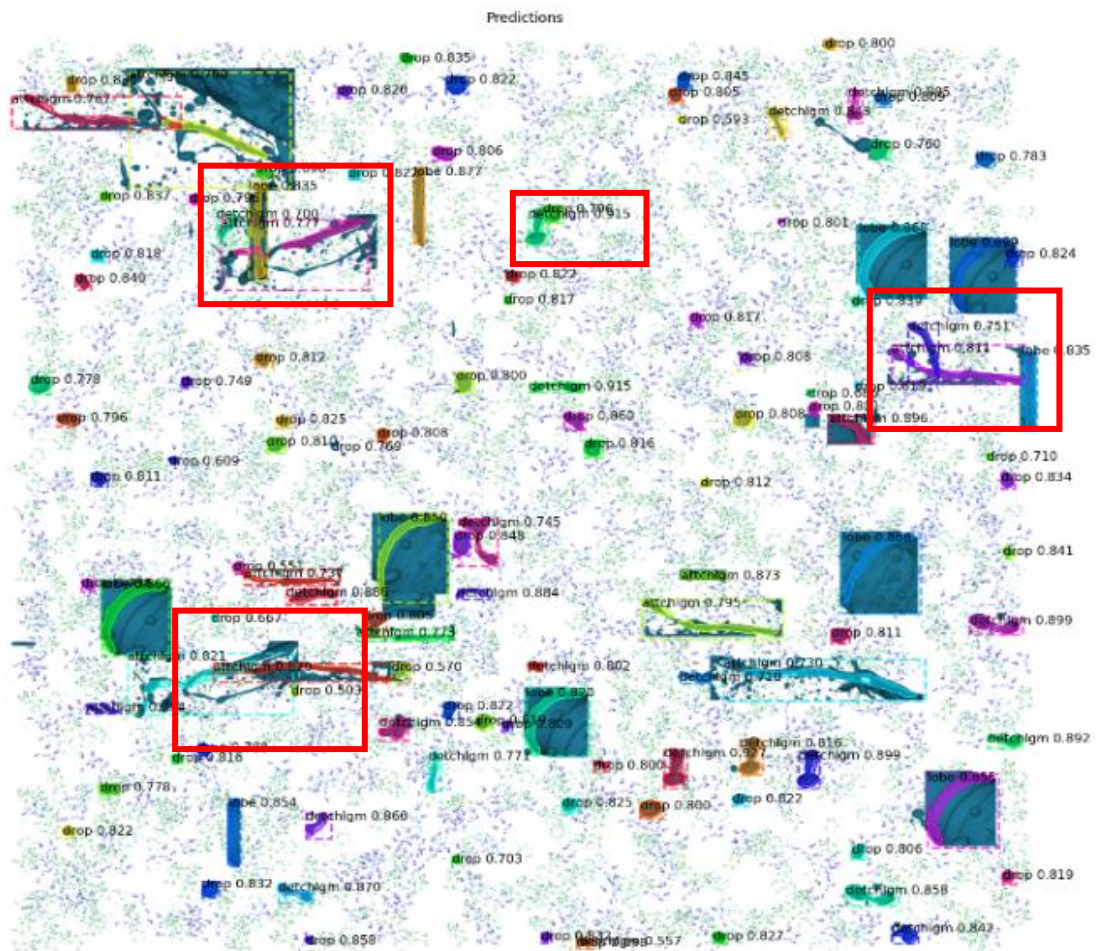
Image No.	Before Divide and Conquer				After Divide and Conquer			
	Mask R-CNN	BMask R-CNN	FastInst	PatchDCT	Mask R-CNN	BMask R-CNN	FastInst	PatchDCT
1	37	83	29	62	168	173	120	173
2	69	132	43	106	272	263	169	295
3	67	64	55	96	91	134	102	233
4	74	69	59	103	99	148	108	245
5	100	131	53	126	357	295	178	350
6	87	51	62	110	124	158	132	262

The post processing method is not dependent on fluid flow direction. But in some cases, it depends on image orientation. If the image is based on horizontal flow, it will work as usual. As vertical cropping is tested, any vertical flow would also work. Just the best cropping window may be different. Only if the image is diagonal, it needs to be adjusted accordingly. Comparing all the results, Mask R-CNN has shown overall better performance than other models based on full-length dataset testing results that was mentioned in section 4.4. With this result in place, objective two (RO2) has been achieved. Though it has some limitations, like the model training epochs couldn't go higher due to the limited computational resources. Also considering the fact that, images are highly dense, and objects are comparatively smaller in sizes, it is difficult for models to detect them. The available results might be improvable further, but current result would



be a supporting point for further improvement considering a jump start to the computer vision application on these kinds of dense liquid spray images.

Regarding the object overlapping and occlusion scenarios, it has been noticed that in augmented images, overlapping objects are well detected based on general object shapes. As object density in this augmented image dataset is lower than original images, the detection accuracy is generally better with mask prediction. At the time of generating the augmented dataset, the position of object placement was chosen randomly so that it can have some overlaps with other objects and the model could train on those kinds of scenarios. One sample image is shown in that was inferenced on MRCNN, pointing out the overlapping object detection areas (marked in red boxes). Object overlapping has also been seen in original images in both before and after divide and conquer technique, but due to unavailability of ground truth, quantitative analysis was not possible. So, a visual inspection was done to observe multi-color overlapping occurrence and considered them as a detection of overlapped objects. As the object size and shapes are not regular and the models are not yet able to detect 100% objects, the primary focus remained on improving object detection procedure as a future work direction.



After this object detection result on seven images, a full-length dataset of 1573 images were tested, and contour was extracted for object tracing. During this test, on the selected 1558<sup>th</sup> image, Mask R-CNN detected a total of 473 droplets, 9 detached ligaments, 36 attached ligaments, and 30 lobes. In contrast, FPN-based BMask R-CNN detected only 212 droplets, 5 detached ligaments, 6 attached ligaments, and 5 lobes, which causes the object tracing tree to be smaller than the Mask R-CNN based tree.

For object tracing, it was expected to have good competition between these two selected models. But Mask R-CNN created a good tracing tree with less pixel distance between objects where FPN-based BMask R-CNN couldn't follow. With the initially selected 840<sup>th</sup> image and 4<sup>th</sup> object, Mask R-CNN based tracing created a tracing tree of length 12340;

skipping one frame causes the tree to be small but still traced 4766 objects with a distance of 4.83pixel that gains a 90.59% of the original length. Skipping two frames resulted in 2978 objects in the tracing tree, having a gain of 92.55% on the original tree, and finally, three frames skipping has a gain of 92.42% with a traced object length of 1306. On the other hand, the FPN model shows a poor result in this case, as the object detection on the original length shows a lower count than Mask R-CNN. With the object tracing tree, final research objective (RO3) has been achieved. One limitation of this object tracing is that it has been calculated with manual algorithms, which is accurate enough depending on the object detection consistency. But it might not be suitable for real time applications. Still, future improvements would be to focus on object tracing using neural networks architecture merging with object detection model so that both can happen parallelly. Still, this result is reliable based on visual observations and highly customizable with respect to expected outcomes. Finally, from all the results and comparisons, Mask R-CNN with 5791 epochs ultimately led to better detection and tracing results.

The key research design idea was based on three major points as: generalization and robustness of the dataset, instance segmentation solution and object movement path tracing. The research methodology was mostly based on the fluid spray dataset, that was not labelled, and dataset is not unique to its characteristics. For example, a car is a car no matter its surroundings. On the other hand, a droplet can be in many forms, locations, shapes having different surroundings. Training on one kind of spray dataset will be very less useful in case of spray characteristic changes. So, the dataset augmentation was based on the intuition of robustness so that it can handle diverse scenarios. As now the model is trained on objects, not on spray itself, regardless of simulation or atomization parameters, model is supposed to recognize similar kind of objects in a wide range of fuel and atomizer variations. To select an instance segmentation model, many things were kept in mind like availability, computational time and resource requirements, usefulness,

backbone architecture and benchmarking. Considering these keys, four models were trained from relatively old Mask R-CNN to latest transformer-based models. Intuition was to see better performance on the latest transformer-based models which was proven to be incorrect as Mask R-CNN showed overall better performance. After initial result of object detection, it was detecting much less objects than expected. An idea was formed suggesting that the detection performance may be lacking due to the high object density in most of the spray body. To solve this problem, a divide and conquer method was proposed which is proven to show better detections. For object tracing, the key idea was to trace object movement by calculating the differences between pixels in between two frames and continue this process till it breaks down into two objects or converts into another class. One major challenge in tracing was to determine if the object is same as previous frame as it is gradually moving and changing position. To solve this, a radius search was done as discussed in section 4.5. Maintaining this process a total object tracing tree was created and later visualized in the form of a video.

The methodology used in this research demonstrates the generalization capabilities of image processing in the atomization post-processing process. By employing advanced instance segmentation models like Mask R-CNN, BMask R-CNN, and transformer-based models (PatchDCT and FastInst), the study effectively detects and segments droplets from synthetic liquid spray images. The approach includes training these models on augmented datasets to enhance their robustness and adaptability to various spray conditions. Additionally, the use of customized algorithms for object tracing and class transition analysis ensures that the methodology can be applied to different spray scenarios with minor adjustments as per requirements, providing valuable insights into the atomization process and enabling improvements in fuel injector design and performance.

## CHAPTER 5: CONCLUSION

The atomization of liquid spray droplets in a combustion engine plays a crucial role in fuel burning and performance efficiency. By understanding and optimizing the factors that influence atomization, it is possible to improve the fuel delivery and combustion process, leading to improved engine performance and reduced emissions. To understand these parameters, objects related to liquid spray need to be detected and segmented to analyze the fuel injector's insights. This thesis analyzed CFD simulation images of fluid spray, focusing on four classes: droplets, detached ligaments, attached ligaments, and lobes. For instance, a state-of-the-art model named Mask R-CNN and one of its variations named BMask R-CNN, along with two transformer models named PatchDCT and FastInst have been used for the segmentation of those images.

Initially, there were only seven images, for which image augmentation has been applied to increase the number of images with a custom method. This augmented dataset can be considered as generalized dataset as it is not depending on spray direction, pattern or fuel type, full filling the first objective. Later, model inferencing is done to check the object detection rate while keeping an eye on the mAP value. With Mask R-CNN a maximum number of 74 droplets has been detected, where BMask R-CNN detected 60, PatchDCT detected 78 and FastInst detected 56 on a test augmented image that has 82 droplets. Later testing with original seven images, a maximum total of 690 objects has been detected with both Mask R-CNN with 5791<sup>st</sup> weight and FPN based BMask R-CNN with 3000<sup>th</sup> weight, PatchDCT detected 826 with 20k epochs and FastInst detected 576 objects with 5k epochs. Default testing of the trained models shows a very small portion of objects that existed in the dataset.

To improve object count, a cropping window method with multiple window sizes is introduced to the system. Four models with different configurations and epochs are tested. Finally, the (height, width/3) window is selected for Mask R-CNN, FPN-based BMask R-CNN and transformer models as compatible window size. With this window size, the object detection has significantly improved to 1457 for Mask R-CNN with 5791<sup>st</sup> weight, 1537 for FPN-based BMask R-CNN with 3k epochs, 1962 for PatchDCT with 20k epochs and 1496 for FastInst with 5k epochs. After comparing these results, (height, width/3) window is used to inference a subset of 1573 images to observe the object detection consistency among sequential frames. From the observations, it is noticed that Mask R-CNN and BMask R-CNN shows a consistent detection where transformer models failed. Finally Mask R-CNN and BMask R-CNN models are selected to inference a full-length dataset of 1573 images to extract the detected object's mask as a contour, full filling the second objective of model selection and evaluation. Those contours are analyzed, and the correspondence object list is calculated using a customized algorithm to maintain a distance threshold. Finally, object tracing and hot zoning are done on those correspondence object lists, addressing the third and final objective of this thesis. The main contribution of this thesis is the procedure that extracted the expected object classes and traced their path. These detection and tracing results are crucial for atomizer designers who want to know the quantifiable value of atomizer performance to improve their performance.

## **5.1 Limitations of Study**

This study has some limitations as image labels were never available, and the Mask R-CNN and BMask R-CNN model training system is done on an in-house basic computer; it was impossible to continue the training for a higher epoch. So, results for higher epochs are not available to compare initially. Subsequent transformer models were trained using the most recent RTX3090, but a comparison of the results reveals that Mask R-CNN

outperforms these models with significantly larger weight counts. When 1573 images were inferenced, it was noticed that the same object that moved some pixels around remained undetected throughout some consecutive frames. This leads to inconsistent tracing of object breakdown or transition from higher class to lower class.

## **5.2 Future Direction**

One may extend this current research by increasing the model training epoch for Mask R-CNN to see if it improves further or is just stuck or overfitted. Also, it needs some more experiments and adjustments to work with transformer models as they showed some good mask predictions. As the images are densely populated, it would be nice to experiment with graph networks techniques, considering objects as nodes, and movement direction after a time value as vertices. Another improvement can be made if some other image post-processing method can be applied to detect model results to improve the object count. Then, object breakdown tracing can be enhanced by considering the consistency of object detection throughout the frames.

## PUBLICATIONS

### Published Papers:

- a. Pathan, R. K., Lim, W. L., Lau, S. L., Ho, C. C., Khare, P., & Koneru, R. B. (2022). Experimental Analysis of U-Net and Mask R-CNN for Segmentation of Synthetic Liquid Spray. *2022 IEEE International Conference on Computing (ICOCO)*, 237–242. <https://doi.org/10.1109/ICOCO56118.2022.10031951>
- b. Pathan, R. K., Lim, W. L., Lau, S. L., Ho, C. C., Bravo, L., Koneru, R. B., & Khare, P. (2023, October). Application of Computer Vision Techniques to Identify Spray Primary Breakup Structures in High-Speed Flow. In *2023 IEEE International Conference on Computing (ICOCO)* (pp. 468-474). IEEE. <https://doi.org/10.1109/ICOCO59262.2023.10398036>
- c. Lim, W. L., Teow, M. Y., Wong, R. T., Pathan, R. K., Lau, S. L., Ho, C. C., ... & Khare, P. (2023, October). Performance Assessment of U-Net for Semantic Segmentation of Liquid Spray Images with Gaussian Blurring. In *2023 IEEE International Conference on Computing (ICOCO)* (pp. 462-467). IEEE. <https://doi.org/10.1109/ICOCO59262.2023.10397704>
- d. Lim, W. L., Teow, M. Y., Wong, R. T., Pathan, R. K., Ho, C. C., Koneru, R. B., ... & Lau, S. L. (2024). Semantic Liquid Spray Understanding with Computer-Generated Images. *IEEE Access*, vol. 12, pp. 39968-39977, 2024, doi: 10.1109/ACCESS.2024.3373459.



## APPENDIX A

### DETAILED EXPERIMENTAL RESULTS

Table A.1 Summary of segmentation models.

Ref.	Model	Evaluation	Dataset	Application	Backbone
(Girshick et al., 2014)	R-CNN	53.3% - mAP	PASCAL VOC-2012	Core model	
(Van de Sande et al., 2011)	Selective search with SVM	96.7% - Accuracy	PASCAL VOC 2007-2010	Core concept	
(Z. Cai & Vasconcelos, 2019)	Cascade R-CNN	42.8% - AP (resnet101) 50.9% - AP(resnext152)	MS COCO, VOC, KITTI, CityPerson, WiderFac	Core model	ResNet-101, ResNeXt-152
(Dong et al., 2019)	Cascade R-CNN	44.4% - mAP	YouTube-Video Instance Segmentation Challenge	video instance segmentation	ResNext-101
(Y. Zhu et al., 2019)	Rotated cascade RCNN	49.2% - mAP on Task 1	ICPR 2018 Contest on Object Detection in Aerial Images, CTW1500)	Core model	ResNet-50 (DOTA), (ResNeXt-101) ICPR contest on object detection in aerial images,
(Khoreva et al., 2016)	DeepLabBO X	51.4% - ABO on VOC12+COCO 49.1% - ABO on VOC12	PASCAL VOC 2012, MS COCO	Core model	DeepLabBO X
(Hariharan et al., 2015)	Hypercolumns	60% - AP	PASCAL VOC 2012	Core model	
(X. Wang et al., 2017)	Fast R-CNN + (ASDN + ASTN)	69.0% - mAP – VOC, 25.7% - AP – COCO matrix	PASCAL VOC 2012, MS COCO	Core model	VGG16
(Li et al., 2017)	SAF R-CNN	65.01% - AP in KITTI	Caltech, INRIA and ETH, KITTI	Core model	VGG16
(X. Chen & Gupta, 2017)	Faster R-CNN	28.3% - AP	PASCAL VOC 2007, MS COCO 2014	Faster RCNN framework from Caffe to TensorFlow	VGG16
(Rossi et al., 2021)	GROIE	35.8% - AP (with mask - rcnn+GROIE) 37.2% - AP (with GC-net+GROIT)	MS COCO	Core model	ResNet-50+FPN

Ref.	Model	Evaluation	Dataset	Application	Backbone
(L.-C. Chen et al., 2018)	MaskLab+	37.3% - mAP	MS COCO	Refining Object Detection with Semantic and Direction Features	ResNet-101
(J. Cao et al., 2020)	D2Det	50.1% - AP	MS COCO	Core Model	ResNet-101
(T.-T. Zhang et al., 2021)	EHTC	94.3% AP	Fish4knowledge	Detecting fish	ResNeXt-101
(Gong et al., 2021)	MHTCUN	26.87% PSNR	BioSR	Core Model	U-Net
(K. He et al., 2017)	Mask RCNN	37.1 – AP on ResNeXt-101-FPN (instance segmentation)	MS COCO	Core Model	ResNet-101-C4 ResNet-101-FPN ResNeXt-101-FPN
(Khan et al., 2021)	MASK RCNN	accuracy 93.6% - ISBI2016 92.7% - ISBI2017	ISBI2016 and ISIC2017	skin lesion detection and recognition	DenseNet
(G. Cao et al., 2019)	Mask R-CNN	AP value of 61.2	Custom 1400 pathological images (2048 x 2048 pixels) included 1120 positive samples (with cancer) and 280 negative samples (no cancer).	gastric cancer	ResNet-50, ResNet-101
(Liu et al., 2018)	Mask R-CNN	0.7965% - mAP	LIDC-IDRI	Lung Nodule segmentation	ResNet-101+ FPN
(Mulay Supriti and Deepika, 2020)	HED-Mask R-CNN	0.9 - Dice coefficient	CHAOS	liver segmentation	Resnet-101+FPN
(L. Cai et al., 2020)	Mask R-CNN	88.2% - AP@50 score of using Mask R-CNN with weighted loss	LUNA16	3D visualization diagnosis for Pulmonary Nodule	ResNet-50 + RPN
(G. Zhu et al., 2020)	Mask RCNN	97.4% - Pixel accuracy	collect 100 images from the hospital	automatic tooth detection and segmentation	ResNet101 + FPN
(Dogan et al., 2021)	MRCNN + 3D U-net	The average values of DSC, JI, REC and ACC are computed as 86.15%, 75.93%, 86.27%, 86.27% and 99.95%	NIH-82 dataset	Segmentation of Pancreas in CT Imaging	ResNet-101

Ref.	Model	Evaluation	Dataset	Application	Backbone
(Kompella et al., 2019)	MRCNN + ImageNet	0.88 maximum Dice Similarity Coefficient (DSC) of and an average DSC of 0.80 is achieved	US images of the knee cartilage	Segmentation of Femoral Cartilage from Knee Ultrasound Images	ImageNet
(Chiao et al., 2019)	MRCNN	75% mAP for the detection and segmentation. The overall accuracy of benign/malignant classification was 85%.	A total of 80 cases were recruited and the image datasets were composed of 307 images of ultrasound images obtained during echo guide biopsy.	breast lesions with ultrasound images	-
(Rehman et al., 2021)	MRCNN	99.1% - Accuracy	The Plant Village dataset	apple leaf diseases	ResNet-50+FPN ResNet-101+FPN
(Huang et al., 2020)	fuzzy Mask R-CNN	98% - accuracy	Tomato images with dimensions of $1108 \times 1478$ pixels per image were collected from a greenhouse cooperative farm located in Tainan	ripening tomatoes	ResNet-50 ResNet-101
(Yu et al., 2019)	Mask-RCNN	89.85% -mean intersection over union (MIoU)	Experimental images were acquired from several strawberry gardens in Shunyi District, Beijing, China during April to May 2018	Fruit detection for strawberry harvesting robot	Resnet-50
(B. Xu et al., 2020)	Mask R-CNN	accurately classify the livestock with an accuracy of 96% estimate the number of cattle and sheep to within 92% of the visual ground truth	FriesianCattle	Livestock classification and counting	ResNet-101
(Qiao et al., 2019)	Mask R-CNN	92% - Mean Pixel Accuracy (MPA)	Beef cattle in Australian Country Choice's Brisbane	Cattle segmentation and contour extraction	ResNet-101 + FPN

Ref.	Model	Evaluation	Dataset	Application	Backbone
			Valley feedlot was research target		
(Cheng Tianheng and Wang, 2020)	BMask R-CNN	36.6% AP: R50+FPN 38.0% -AP: R101	Cityscapes MS COCO	Core model	ResNet-50+FPN
(Y. Zhang et al., 2020)	Mask-refined R-CNN	37.6% - AP	Cityscapes MS COCO	Core model	ResNet-101+FPN

Table A.2 Experimental results for Mask R-CNN with 1000<sup>th</sup>, 2791<sup>st</sup> and 5791<sup>st</sup> weight on different crop window for seven original images, notable results are highlighted with light blue color.

Crop Window Size	Image	Detached Ligaments			Droplets			Attached Ligaments			Lobe		
		1000	2791	5791	1000	2791	5791	1000	2791	5791	1000	2791	5791
height, width	1	16	16	19	31	31	37	5	6	10	4	5	4
	2	6	4	7	52	55	69	6	7	10	5	4	7
	3	6	7	13	58	54	67	9	6	10	3	4	4
	4	5	5	7	61	59	74	8	6	8	3	4	4
	5	5	2	9	84	66	100	4	5	10	3	4	4
	6	1	4	7	67	65	87	7	5	14	2	3	3
	7	2	7	8	65	62	83	6	4	12	2	3	3
height, width/3	1	0	24	17	5	165	168	5	16	11	6	9	11
	2	0	27	13	17	242	272	4	17	17	3	6	9
	3	9	8	7	120	104	91	11	7	14	7	8	5
	4	7	5	4	125	113	99	11	5	16	7	8	5
	5	0	21	19	16	281	357	2	13	19	5	4	12
	6	7	8	5	142	117	124	9	4	12	8	7	7
	7	9	10	5	140	116	120	10	5	11	8	7	7
height/3, width	1	14	16	16	37	34	30	5	6	6	4	6	8
	2	3	8	10	59	63	68	4	6	7	5	5	3
	3	6	9	11	55	56	67	10	4	11	5	5	7
	4	6	7	11	60	62	72	8	3	11	5	5	7
	5	6	8	5	90	80	99	9	4	7	6	4	4
	6	2	5	10	83	72	85	8	4	10	3	5	1
	7	6	6	11	77	66	83	9	5	9	3	5	1
height/3, width/3	1	10	20	18	228	219	216	68	32	34	33	38	29
	2	12	23	15	372	337	352	58	41	33	52	30	47
	3	10	20	14	387	378	401	35	22	36	25	21	33
	4	6	17	16	431	415	446	37	20	36	25	23	31
	5	8	19	6	466	441	447	60	50	38	50	33	43
	6	5	26	10	482	474	522	47	25	45	31	23	31
	7	7	25	11	464	453	503	49	25	53	31	25	31
height, width	1	8	26	23	163	159	191	13	13	14	16	10	21
	2	7	24	15	260	235	300	16	16	10	11	8	11

Crop Wind ow Size	Ima -ge	Detached Ligaments			Droplets			Attached Ligaments			Lobe		
		1000	2791	5791	1000	2791	5791	1000	2791	5791	1000	2791	5791
	3	11	9	10	129	100	109	10	5	12	13	5	5
	4	7	7	11	138	105	124	11	6	12	12	5	5
	5	8	25	16	330	275	388	19	12	12	18	7	14
	6	5	5	7	143	116	150	10	5	15	14	6	5
	7	6	7	7	142	114	146	9	6	15	14	6	5
height/4, width	1	15	17	18	32	27	31	6	3	5	4	2	3
	2	6	9	6	61	61	58	9	7	9	2	1	2
	3	6	7	13	56	53	60	8	4	9	3	3	2
	4	5	5	10	62	59	67	7	3	8	3	3	2
	5	7	6	10	84	83	98	8	5	14	0	0	0
	6	4	4	5	80	72	80	9	5	8	5	2	2
	7	4	8	7	79	71	79	10	6	9	5	2	2
height/4, width/4	1	4	13	5	148	120	113	90	56	54	99	107	124
	2	4	13	7	226	157	173	107	67	56	118	107	178
	3	4	6	5	436	323	412	67	47	55	85	43	75
	4	5	12	5	498	461	476	63	45	53	85	54	75
	5	4	11	2	281	224	234	103	104	68	149	158	196
	6	3	10	5	552	518	532	84	53	67	88	64	82
	7	3	8	6	519	486	499	86	56	66	89	64	82
height, width/5	1	0	27	25	35	168	188	5	18	14	4	9	15
	2	0	25	20	68	235	299	4	9	19	2	7	10
	3	9	6	10	125	105	119	9	7	16	9	6	4
	4	6	4	12	136	108	126	6	4	17	9	6	4
	5	2	18	13	95	276	376	5	13	18	6	3	16
	6	7	8	8	146	123	157	7	7	19	9	6	3
	7	8	8	11	145	122	155	6	7	19	9	6	3
height/5, width	1	16	16	15	23	24	30	5	3	5	4	1	3
	2	4	8	6	54	47	60	10	4	9	3	1	2
	3	9	8	10	50	49	60	8	1	12	3	3	4
	4	10	8	7	54	56	65	11	1	11	3	3	4
	5	9	10	10	74	66	82	6	3	10	1	0	2
	6	3	5	8	74	71	78	12	2	14	3	2	1
	7	4	6	8	74	69	79	11	1	12	3	2	1
height/5, width/5	1	3	5	1	51	37	12	129	76	59	184	164	158
	2	1	2	2	66	64	23	167	153	97	270	245	245
	3	3	10	2	318	261	252	82	87	71	159	153	169
	4	2	9	3	371	297	283	84	87	70	170	167	182
	5	2	6	0	88	55	23	180	163	102	373	320	330
	6	3	9	3	379	299	312	83	86	68	171	186	205
	7	3	11	4	355	268	290	85	85	68	175	184	197

Table A.3 Experimental results for FPN based BMask R-CNN with 1000<sup>th</sup> and 3000<sup>th</sup> weight on different crop window for seven original images, notable results are highlighted with light blue color.

Crop Window Size	Image name	Detached Ligaments		Droplets		Attached Ligaments		Lobe	
		1000	3000	1000	3000	1000	3000	1000	3000
height, width	1.png	15	15	65	83	1	9	1	2
	2.png	5	9	81	132	0	7	0	1
	3.png	9	8	62	64	4	5	1	1
	4.png	6	7	70	69	1	3	1	1
	5.png	6	8	91	131	3	13	0	1
	6.png	6	4	70	51	0	4	1	1
	7.png	7	6	72	49	0	5	1	1
height, width/3	1.png	14	12	204	173	8	23	15	32
	2.png	10	6	235	263	6	17	10	30
	3.png	13	8	185	134	7	4	3	1
	4.png	8	4	195	148	6	3	4	2
	5.png	9	2	272	295	3	16	8	31
	6.png	6	3	224	158	7	3	3	2
	7.png	8	3	225	158	9	4	4	2
height/3, width	1.png	25	14	137	221	5	33	1	55
	2.png	13	3	177	317	6	23	1	44
	3.png	14	9	208	191	7	6	3	3
	4.png	12	4	207	206	7	4	2	3
	5.png	20	5	208	368	6	35	1	52
	6.png	9	4	237	192	12	9	6	4
	7.png	10	5	220	197	10	8	6	4
height/3, width/3	1.png	7	2	361	198	40	74	140	265
	2.png	4	0	434	286	45	50	194	356
	3.png	13	4	384	363	10	12	53	52
	4.png	7	3	392	393	8	10	49	47
	5.png	5	2	463	322	27	54	248	443
	6.png	10	1	439	392	9	9	46	44
	7.png	9	1	434	385	9	9	42	46
height, width/4	1.png	12	11	247	181	6	23	33	29
	2.png	6	3	255	272	11	16	32	36
	3.png	16	10	244	222	5	8	5	11
	4.png	12	6	246	236	5	6	5	8
	5.png	7	3	280	314	7	20	32	35
	6.png	10	6	259	188	5	8	3	5
	7.png	9	8	262	187	7	7	3	4
height/4, width	1.png	28	15	163	289	5	43	4	44
	2.png	19	6	222	397	4	44	0	65
	3.png	18	11	216	226	11	12	6	4
	4.png	12	7	222	242	7	12	6	4
	5.png	17	6	252	488	10	53	1	68
	6.png	19	4	280	238	5	10	6	5

Crop Window Size	Image name	Detached Ligaments		Droplets		Attached Ligaments		Lobe	
		1000	3000	1000	3000	1000	3000	1000	3000
	7.png	19	7	272	233	6	9	7	5
height/4, width/4	1.png	2	1	308	154	64	96	283	486
	2.png	2	1	247	189	51	121	436	644
	3.png	6	5	563	400	14	35	139	150
	4.png	4	3	588	452	14	31	139	150
	5.png	2	1	257	229	73	130	508	778
	6.png	7	3	588	471	14	18	149	157
	7.png	7	3	568	443	15	19	149	158
height, width/5	1.png	22	12	253	199	12	25	38	42
	2.png	7	5	357	309	10	20	32	49
	3.png	14	13	256	183	6	13	4	4
	4.png	12	4	261	197	4	12	4	4
	5.png	10	5	332	359	12	25	33	51
	6.png	8	5	296	218	5	6	6	9
	7.png	10	6	295	219	7	9	5	9
height/5, width	1.png	26	12	195	343	7	58	2	69
	2.png	22	4	244	482	6	45	3	73
	3.png	19	11	283	234	11	14	7	9
	4.png	12	5	263	243	10	13	7	9
	5.png	16	6	286	547	14	61	2	83
	6.png	19	5	270	266	15	14	9	10
	7.png	22	8	260	269	15	10	9	10
height/5, width/5	1.png	0	1	378	65	118	187	441	595
	2.png	1	1	274	121	128	263	632	908
	3.png	10	1	531	360	34	35	280	336
	4.png	11	1	561	401	28	31	296	355
	5.png	1	0	260	94	146	297	776	1068
	6.png	6	1	627	452	38	28	330	413
	7.png	7	2	614	434	39	27	330	414

Table A.4 Experimental results for Base BMask R-CNN with 1000<sup>th</sup> and 3000<sup>th</sup> weight on different crop window for seven original images, notable results are highlighted with light blue color.

Crop Window Size	Image name	Detached Ligaments		Droplets		Attached Ligaments		Lobe	
		1000	3000	1000	3000	1000	3000	1000	3000
height, width	1.png	12	7	70	117	4	13	1	6
	2.png	3	1	82	160	3	5	0	3
	3.png	5	9	64	69	0	5	1	5
	4.png	3	4	65	72	0	2	1	5
	5.png	2	1	88	185	4	6	0	1
	6.png	2	5	72	92	1	4	1	5
	7.png	3	9	69	90	1	3	1	5

Crop Window Size	Image name	Detached Ligaments		Droplets		Attached Ligaments		Lobe	
		1000	3000	1000	3000	1000	3000	1000	3000
height, width/3	1.png	4	6	165	191	18	23	16	59
	2.png	2	3	233	268	16	7	14	75
	3.png	7	3	193	164	2	8	3	6
	4.png	6	1	188	178	2	5	2	5
	5.png	3	0	253	300	9	11	12	84
	6.png	0	2	203	194	1	4	5	6
	7.png	1	2	202	193	1	4	5	7
height/3, width	1.png	14	5	128	203	8	28	1	70
	2.png	5	1	182	283	9	18	0	79
	3.png	9	2	135	208	4	6	1	8
	4.png	3	1	140	217	3	3	1	4
	5.png	7	2	191	334	12	18	0	98
	6.png	3	2	166	225	0	8	1	8
	7.png	5	3	170	220	0	10	1	9
height/3, width/3	1.png	5	0	231	44	88	44	114	374
	2.png	4	1	303	60	60	35	142	536
	3.png	4	2	333	329	14	14	43	105
	4.png	4	1	358	352	10	12	38	103
	5.png	3	0	344	59	58	37	178	649
	6.png	5	0	398	388	6	6	36	97
	7.png	5	0	385	383	8	7	39	98
height, width/4	1.png	4	4	192	199	25	17	19	63
	2.png	2	1	237	289	14	5	29	75
	3.png	8	5	214	182	7	10	4	14
	4.png	6	2	219	191	5	7	3	12
	5.png	0	0	272	306	20	7	29	87
	6.png	1	3	220	215	4	5	4	8
	7.png	1	2	218	211	4	7	5	9
height/4, width	1.png	20	3	131	208	8	38	1	74
	2.png	6	3	183	322	6	32	1	107
	3.png	8	2	154	226	10	11	1	8
	4.png	4	1	167	250	6	9	1	9
	5.png	8	0	214	389	7	34	0	122
	6.png	5	3	213	261	3	12	1	8
	7.png	6	6	208	259	5	15	1	8
height/4, width/4	1.png	8	0	64	25	98	49	292	509
	2.png	4	0	88	17	90	36	397	726
	3.png	5	1	359	228	32	20	121	312
	4.png	4	1	381	255	29	14	133	327
	5.png	0	0	94	17	93	44	484	865
	6.png	6	0	429	237	24	19	164	386
	7.png	7	0	414	224	29	22	165	370
height, width/5	1.png	5	4	213	191	34	26	26	68
	2.png	3	3	276	296	26	16	30	91
	3.png	6	4	219	189	12	13	4	7



Crop Window Size	Image name	Detached Ligaments		Droplets		Attached Ligaments		Lobe	
		1000	3000	1000	3000	1000	3000	1000	3000
	4.png	7	3	238	206	9	10	3	6
	5.png	0	0	326	333	21	17	29	108
	6.png	3	4	261	225	3	10	3	16
	7.png	3	2	251	222	3	10	4	16
height/5, width	1.png	18	6	136	227	13	44	0	75
	2.png	7	3	178	337	8	40	1	123
	3.png	9	4	175	244	10	18	1	13
	4.png	8	1	183	254	6	18	1	15
	5.png	10	3	233	426	15	56	0	145
	6.png	6	4	216	285	9	19	1	18
	7.png	6	6	212	284	9	21	1	19
height/5, width/5	1.png	2	0	37	30	142	81	446	611
	2.png	2	0	41	35	143	68	654	886
	3.png	5	0	323	89	52	31	251	531
	4.png	4	0	339	98	48	30	275	582
	5.png	0	0	27	61	141	92	797	1030
	6.png	0	0	366	88	54	24	317	681
	7.png	0	0	348	90	54	22	312	655

Table A.5 Experimental results for Cascade BMask R-CNN with 1000<sup>th</sup> weight on different crop window for seven original images, notable results are highlighted with light blue color.

Crop Window Size	Image name	Detached Ligaments		Droplets		Attached Ligaments		Lobe	
		1000	3000	1000	3000	1000	3000	1000	3000
height, width	1.png	19	14	101	121	17	7	10	4
	2.png	5	7	147	157	14	1	8	4
	3.png	11	8	90	90	8	5	5	5
	4.png	7	3	97	99	4	5	6	5
	5.png	7	1	179	193	10	3	4	1
	6.png	4	7	91	87	5	3	5	4
	7.png	6	6	86	86	5	4	6	4
height, width/3	1.png	9	6	187	188	58	7	35	24
	2.png	6	2	259	268	50	4	41	28
	3.png	9	4	218	185	16	3	10	6
	4.png	4	2	228	206	10	0	10	5
	5.png	6	1	320	335	54	1	44	29
	6.png	6	3	244	198	12	0	10	3
	7.png	7	1	241	198	13	2	10	3
height/3, width	1.png	9	5	168	208	56	8	20	38
	2.png	5	3	269	305	41	9	37	43
	3.png	7	10	175	239	12	2	6	11
	4.png	5	4	187	245	12	1	5	7

Crop Window Size	Image name	Detached Ligaments		Droplets		Attached Ligaments		Lobe	
		1000	3000	1000	3000	1000	3000	1000	3000
	5.png	2	2	293	345	33	12	32	51
	6.png	2	5	191	252	7	2	6	6
	7.png	2	5	193	252	5	3	6	7
height/3, width/3	1.png	1	1	90	46	135	23	148	260
	2.png	2	0	143	55	123	12	226	392
	3.png	5	1	334	344	29	4	54	62
	4.png	2	0	362	384	30	1	55	60
	5.png	5	0	136	67	96	31	294	452
	6.png	2	2	398	404	20	2	62	61
	7.png	2	2	389	396	19	2	61	59
height, width/4	1.png	15	5	204	154	70	6	40	34
	2.png	10	1	330	237	57	6	51	44
	3.png	12	8	237	233	19	5	17	9
	4.png	10	5	252	251	17	4	17	6
	5.png	5	1	381	289	63	3	57	41
	6.png	8	5	279	233	12	1	13	5
	7.png	8	2	276	233	11	4	13	6
height/4, width	1.png	7	7	168	220	56	14	40	52
	2.png	7	7	263	332	52	11	45	50
	3.png	11	10	263	259	15	3	10	9
	4.png	6	6	277	277	17	2	8	9
	5.png	5	3	297	415	48	15	42	65
	6.png	6	7	227	291	13	5	15	9
	7.png	5	6	226	289	17	4	15	9
height/4, width/4	1.png	2	0	38	14	166	16	294	457
	2.png	5	0	38	16	185	23	443	640
	3.png	4	1	288	266	63	15	129	190
	4.png	3	0	317	312	52	10	137	198
	5.png	0	0	51	14	170	28	566	738
	6.png	3	0	321	327	43	5	166	230
	7.png	3	0	311	308	44	7	161	224
height, width/5	1.png	10	5	229	185	72	10	51	43
	2.png	8	1	350	308	77	6	62	50
	3.png	17	8	263	206	21	5	21	8
	4.png	10	3	290	234	23	5	18	8
	5.png	13	1	445	365	79	4	58	59
	6.png	13	3	296	232	19	4	18	10
	7.png	13	3	296	232	18	4	18	10
height/5, width	1.png	8	7	173	248	55	14	35	50
	2.png	6	4	314	387	46	12	38	60
	3.png	8	12	207	271	12	3	7	10
	4.png	2	10	229	302	16	2	6	8
	5.png	4	5	359	463	35	21	44	76
	6.png	2	5	239	310	16	8	17	10
	7.png	5	7	235	302	17	6	17	10

Crop Window Size	Image name	Detached Ligaments		Droplets		Attached Ligaments		Lobe	
		1000	3000	1000	3000	1000	3000	1000	3000
height/5, width/5	1.png	1	0	63	13	294	19	366	528
	2.png	0	0	94	40	338	13	569	715
	3.png	3	0	170	124	60	12	282	395
	4.png	2	0	184	140	57	16	305	448
	5.png	1	0	57	11	388	19	669	873
	6.png	3	0	216	127	71	22	324	501
	7.png	3	0	205	120	68	22	321	491

Table A.6 Experimental results for PatchDCT and FastInst with 10k weight on different crop window for seven original images, notable results are highlighted with light blue color.

Crop Window Size	Image name	Detached Ligaments		Droplets		Attached Ligaments		Lobe	
		Patch DCT	FastInst	Patch DCT	FastInst	Patch DCT	FastInst	Patch DCT	FastInst
height, width	1.png	17	22	59	29	3	19	6	4
	2.png	7	8	92	43	3	11	6	2
	3.png	10	12	74	55	4	11	5	2
	4.png	5	8	79	59	4	11	5	2
	5.png	6	7	119	53	4	9	5	1
	6.png	4	5	91	62	6	13	5	1
	7.png	4	9	93	59	6	12	5	1
height, width/3	1.png	8	32	219	120	11	63	16	16
	2.png	0	16	319	169	6	49	18	10
	3.png	9	14	416	102	6	30	9	11
	4.png	9	12	429	108	5	30	9	11
	5.png	4	18	379	178	5	38	12	5
	6.png	7	10	326	132	7	21	11	12
	7.png	8	11	321	131	6	21	11	12
height/3, width	1.png	21	29	77	81	6	27	9	6
	2.png	7	16	131	72	8	35	7	10
	3.png	11	16	140	78	7	25	5	6
	4.png	7	12	120	84	6	24	4	6
	5.png	8	17	140	85	8	25	9	4
	6.png	3	12	129	91	4	25	6	5
	7.png	6	14	128	94	4	28	5	6
height/3, width/3	1.png	7	49	346	294	28	137	149	34
	2.png	1	39	294	362	27	146	262	24
	3.png	8	27	931	288	8	61	54	14
	4.png	7	24	968	308	6	59	54	16
	5.png	0	42	402	453	25	100	219	9
	6.png	1	20	1029	346	8	64	60	21
	7.png	1	20	1006	345	9	64	61	21
	1.png	9	39	462	130	6	66	22	23

Crop Window Size	Image name	Detached Ligaments		Droplets		Attached Ligaments		Lobe	
		Patch DCT	FastInst	Patch DCT	FastInst	Patch DCT	FastInst	Patch DCT	FastInst
height, width/4	2.png	1	21	349	187	7	57	23	27
	3.png	12	16	293	115	5	33	9	18
	4.png	8	12	302	116	4	35	8	18
	5.png	7	30	432	203	6	51	17	17
	6.png	7	12	477	134	6	27	10	17
	7.png	6	14	473	132	6	27	10	18
height/4, width	1.png	22	33	83	103	8	23	5	8
	2.png	8	19	147	90	6	38	3	6
	3.png	11	19	175	94	3	32	4	7
	4.png	10	16	186	95	1	32	4	6
	5.png	7	21	170	124	9	32	1	9
	6.png	7	8	192	100	5	35	10	11
	7.png	8	12	195	98	5	37	9	11
height/4, width/4	1.png	1	56	649	385	36	174	436	71
	2.png	0	76	181	537	43	230	644	60
	3.png	2	32	741	439	20	91	257	34
	4.png	1	26	800	461	15	90	267	23
	5.png	0	61	182	624	41	138	703	53
	6.png	0	24	1001	532	26	72	304	39
	7.png	0	27	1006	519	26	73	305	38
height, width/5	1.png	9	35	576	157	7	74	24	28
	2.png	1	30	451	197	5	63	43	25
	3.png	13	21	214	135	7	32	6	14
	4.png	11	19	223	138	7	37	6	14
	5.png	5	33	449	213	5	69	22	14
	6.png	8	14	269	164	8	44	15	16
	7.png	10	15	272	160	6	47	14	17
height/5, width	1.png	21	33	169	88	13	31	9	8
	2.png	11	19	263	120	8	33	9	12
	3.png	16	15	204	100	5	30	5	7
	4.png	12	15	163	110	5	36	5	8
	5.png	8	27	245	154	13	31	10	5
	6.png	5	14	190	113	8	37	8	16
	7.png	6	17	191	109	7	36	8	16
height/5, width/5	1.png	0	75	1133	475	57	285	574	103
	2.png	0	105	591	633	71	379	870	126
	3.png	0	41	662	532	32	155	610	59
	4.png	0	38	632	580	32	133	670	51
	5.png	0	96	378	735	65	288	936	92
	6.png	1	26	715	659	36	139	750	59
	7.png	2	29	703	648	34	146	719	59

## APPENDIX B

### RESULT IMAGE AND VIDEOS

- B.1 [Mask R-CNN object detection images with video with 5791<sup>st</sup> weight and \(height, width/3\) window.](#)
- B.2 [BMask R-CNN object detection images with video for \(height, width/3\), \(height, width/4\) and \(height, width/5\) for all model configurations.](#)
- B.3 [Transformer object detection images for all crop window and epochs and videos for \(height, width/3\) window.](#)
- B.4 [Mask R-CNN object correspondence tracing video.](#)
- B.5 [BMask R-CNN object correspondence tracing video.](#)
- B.6 [DeepSort tracing with Mask R-CNN \(5791 weight\) on \(height,width/3\) window using 1573 images.](#)
- B.7 [DeepSort tracing of one object with Mask R-CNN \(5791 weight\) on \(height, width/3\) window.](#)
- B.8 [Mask R-CNN object tracing with skipping video and tree.](#)
- B.9 [BMask R-CNN object tracing with no skipping video and tree.](#)
- B.10 [Mask R-CNN object tracing comparison video with BMask R-CNN considering same object location.](#)

## REFERENCES

- Abdulla, W. (2018). *Splash of Color: Instance Segmentation with Mask R-CNN and TensorFlow*. <https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46>
- Acharya, P., Burgers, T., & Nguyen, K.-D. (2022). AI-enabled droplet detection and tracking for agricultural spraying systems. *Computers and Electronics in Agriculture*, 202, 107325. <https://doi.org/10.1016/j.compag.2022.107325>
- Ainsalo, A., Sallinen, R., Kaario, O., & Larmi, M. (2019). OPTICAL INVESTIGATION OF SPRAY CHARACTERISTICS FOR LIGHT FUEL OIL, KEROSENE, HEXANE, METHANOL, AND PROPANE. *Atomization and Sprays*, 29(6), 521–544. <https://doi.org/10.1615/AtomizSpr.2019029626>
- Ananth, S. (2021). *Faster R-CNN for object detection - Towards Data Science*. <https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46>
- Bhargavi, K., & Jyothi, S. (2014). A survey on threshold based segmentation technique in image processing. *International Journal of Innovative Research and Development*, 3(12), 234–239.
- Bothell, J. K., Machicoane, N., Li, D., Morgan, T. B., Aliseda, A., Kastengren, A. L., & Heindel, T. J. (2020). Comparison of X-ray and optical measurements in the near-field of an optically dense coaxial air-assisted atomizer. *International Journal of Multiphase Flow*, 125, 103219. <https://doi.org/10.1016/j.ijmultiphaseflow.2020.103219>
- Bradley, D., & Roth, G. (2007). Adaptive thresholding using the integral image. *Journal of Graphics Tools*, 12(2), 13–21.
- Cai, L., Long, T., Dai, Y., & Huang, Y. (2020). Mask R-CNN-Based Detection and Segmentation for Pulmonary Nodule 3D Visualization Diagnosis. *IEEE Access*, 8, 44400–44409. <https://doi.org/10.1109/ACCESS.2020.2976432>
- Cai, Z., & Vasconcelos, N. (2019). Cascade R-CNN: high quality object detection and instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5), 1483–1498.
- Cao, G., Song, W., & Zhao, Z. (2019). Gastric Cancer Diagnosis with Mask R-CNN. *2019 11th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, 60–63. <https://doi.org/10.1109/IHMSC.2019.00022>
- Cao, J., Cholakkal, H., Anwer, R. M., Khan, F. S., Pang, Y., & Shao, L. (2020). D2Det: Towards High Quality Object Detection and Instance Segmentation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11482–11491. <https://doi.org/10.1109/CVPR42600.2020.01150>
- Cerruto, E., Manetto, G., Privitera, S., Papa, R., & Longo, D. (2022). Effect of Image Segmentation Thresholding on Droplet Size Measurement. *Agronomy*, 12(7), 1677. <https://doi.org/10.3390/agronomy12071677>

- Chaussonnet, G., Lieber, C., Yikang, Y., Gu, W., Bartschat, A., Reischl, M., Koch, R., Mikut, R., & Bauer, H.-J. (2019). *Towards DeepSpray: Using Convolutional Neural Network to post-process Shadowgraphy Images of Liquid Atomization*. <https://doi.org/10.5445/IR/1000097897/v3>
- Chen, C., Wang, B., Lu, C. X., Trigoni, N., & Markham, A. (2020). *A Survey on Deep Learning for Localization and Mapping: Towards the Age of Spatial Machine Intelligence*.
- Chen, L.-C., Hermans, A., Papandreou, G., Schroff, F., Wang, P., & Adam, H. (2018). MaskLab: Instance Segmentation by Refining Object Detection with Semantic and Direction Features. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4013–4022. <https://doi.org/10.1109/CVPR.2018.00422>
- Chen, X., & Gupta, A. (2017). *An Implementation of Faster RCNN with Study for Region Sampling*.
- Chen, X., Ma, D., Khare, P., & Yang, V. (2011, January 4). Energy and Mass Transfer during Binary Droplet Collision. *49th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. <https://doi.org/10.2514/6.2011-771>
- Cheng, T., Wang, X., Chen, S., Zhang, W., Zhang, Q., Huang, C., Zhang, Z., & Liu, W. (2022). Sparse Instance Activation for Real-Time Instance Segmentation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4423–4432. <https://doi.org/10.1109/CVPR52688.2022.00439>
- Cheng Tianheng and Wang, X. and H. L. and L. W. (2020). Boundary-Preserving Mask R-CNN. In H. and B. T. and F. J.-M. Vedaldi Andrea and Bischof (Ed.), *Computer Vision – ECCV 2020* (pp. 660–676). Springer International Publishing.
- Chiao, J.-Y., Chen, K.-Y., Liao, K. Y.-K., Hsieh, P.-H., Zhang, G., & Huang, T.-C. (2019). Detection and classification the breast tumors using mask R-CNN on sonograms. *Medicine*, 98(19), e15200. <https://doi.org/10.1097/MD.00000000000015200>
- Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1251–1258.
- Chourasiya, S. K., Si, A., Singh, G., Gautam, G., & Singh, D. (2019). A novel technique for automatic quantification of porosities in spray formed warm rolled Al-Si-graphite composite. *Materials Research Express*, 6(11), 116565. <https://doi.org/10.1088/2053-1591/ab4a58>
- Cristea, A., Van Houtte, J., & Doulgeris, A. P. (2020). Integrating incidence angle dependencies into the clustering-based segmentation of SAR images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13, 2925–2939.
- Deng, Y., Lan, J., Huang, Y., Gao, Q., Zhang, H., Tong, T., & Chen, G. (2021). Predicting Differentiation Degree of Gastric Cancer Pathology Images Based on Mask Attention R-CNN. *2021 2nd International Conference on Computer Engineering and Intelligent Control (ICCEIC)*, 98–102. <https://doi.org/10.1109/ICCEIC54227.2021.00027>
- Dhieb, N., Ghazzai, H., Besbes, H., & Massoud, Y. (2019). An Automated Blood Cells Counting and Classification Framework using Mask R-CNN Deep Learning Model.

2019 31st International Conference on Microelectronics (ICM), 300–303.  
<https://doi.org/10.1109/ICM48031.2019.9021862>

- Dogan, R. O., Dogan, H., Bayrak, C., & Kayikcioglu, T. (2021). A Two-Phase Approach using Mask R-CNN and 3D U-Net for High-Accuracy Automatic Segmentation of Pancreas in CT Imaging. *Computer Methods and Programs in Biomedicine*, 207, 106141. <https://doi.org/10.1016/j.cmpb.2021.106141>
- Dong, M., Wang, J., Huang, Y., Yu, D., Su, K., Zhou, K., Shao, J., Wen, S., & Wang, C. (2019). Temporal feature augmented network for video instance segmentation. *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 0.
- Du, D., Qi, Y., Yu, H., Yang, Y., Duan, K., Li, G., Zhang, W., Huang, Q., & Tian, Q. (2018). The Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer Vision – ECCV 2018* (pp. 375–391). Springer International Publishing.
- Durve, M., Orsini, S., Tiribocchi, A., Montessori, A., Tucny, J.-M., Lauricella, M., Camposeo, A., Pisignano, D., & Succi, S. (2023). *Benchmarking YOLOv5 and YOLOv7 models with DeepSORT for droplet tracking applications*.
- Durve, M., Tiribocchi, A., Bonaccorso, F., Montessori, A., Lauricella, M., Bogdan, M., Guzowski, J., & Succi, S. (2022). DropTrack—Automatic droplet tracking with YOLOv5 and DeepSORT for microfluidic applications. *Physics of Fluids*, 34(8), 082003. <https://doi.org/10.1063/5.0097597>
- Dutta, A., & Zisserman, A. (2019). The VIA Annotation Software for Images, Audio and Video. *Proceedings of the 27th ACM International Conference on Multimedia*, 2276–2279. <https://doi.org/10.1145/3343031.3350535>
- Emerson, N. I. (2023). *Thresholding*. <https://www.ni.com/docs/en-US/bundle/ni-vision-concepts-help/page/thresholding.html>
- Fan, S., Chen, S., Wu, Z., Wu, S., Chen, Y., Liu, D., Yao, Y., & Huang, J. (2022). Analysis of droplet size distribution and selection of spray parameters based on the fractal theory. *Journal of Cleaner Production*, 371, 133315. <https://doi.org/10.1016/j.jclepro.2022.133315>
- Frazão, J., Palma, S. I. C. J., Costa, H. M. A., Alves, C., Roque, A. C. A., & Silveira, M. (2021). Optical Gas Sensing with Liquid Crystal Droplets and Convolutional Neural Networks. *Sensors*, 21(8), 2854. <https://doi.org/10.3390/s21082854>
- Fredericks, S. A., Magidow, L. C., He, Z., Hogan, C. J., & Hong, J. (2020). Quantification of drifting droplets in sprays experiencing crosswind shear via digital image analysis techniques. *2020 ASABE Annual International Virtual Meeting, July 13-15, 2020*. <https://doi.org/10.13031/aim.202000634>
- Ganesan, R., & Merline, A. (2017). Fuzzy-C-means clustering based segmentation and CNN-classification for accurate segmentation of lung nodules. *Asian Pacific Journal of Cancer Prevention: APJCP*, 18(7), 1869.
- Ganti, H., Kamin, M., & Khare, P. (2020). Design Space Exploration of Turbulent Multiphase Flows Using Machine Learning-Based Surrogate Model. *Energies*, 13(17), 4565. <https://doi.org/10.3390/en13174565>



- Ganti, H., Khare, P., & Bravo, L. (2020). Binary collision of CMAS droplets—Part I: Equal-sized droplets. *Journal of Materials Research*, 35(17), 2260–2274. <https://doi.org/10.1557/jmr.2020.138>
- Gao, W., Zhang, X., Yang, L., & Liu, H. (2010). An improved Sobel edge detection. *2010 3rd International Conference on Computer Science and Information Technology*, 5, 67–71.
- Ghiani, L., Sassu, A., Piccirilli, D., Marcialis, G. L., & Gambella, F. (2020). *Development of a Matlab Code for the Evaluation of Spray Distribution with Water-Sensitive Paper* (pp. 845–853). [https://doi.org/10.1007/978-3-030-39299-4\\_91](https://doi.org/10.1007/978-3-030-39299-4_91)
- Girshick, R. (2015). Fast r-cnn. *Proceedings of the IEEE International Conference on Computer Vision*, 1440–1448.
- Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 580–587.
- Gong, D., Ma, T., Evans, J., & He, S. (2021). Deep Neural Networks for Image Super-Resolution in Optical Microscopy by Using Modified Hybrid Task Cascade U-Net. In *Progress In Electromagnetics Research* (Vol. 171).
- Gorokhovski, M., & Herrmann, M. (2008). Modeling Primary Atomization. *Annual Review of Fluid Mechanics*, 40(1), 343–366. <https://doi.org/10.1146/annurev.fluid.40.111406.102200>
- Gould, S., Gao, T., & Koller, D. (2009). Region-based Segmentation and Object Detection. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, & A. Culotta (Eds.), *Advances in Neural Information Processing Systems* (Vol. 22). Curran Associates, Inc. <https://proceedings.neurips.cc/paper/2009/file/a7aeed74714116f3b292a982238f83d2-Paper.pdf>
- Guo, Y., Peng, Y., & Zhang, B. (2021). CAFR-CNN: coarse-to-fine adaptive faster R-CNN for cross-domain joint optic disc and cup segmentation. *Applied Intelligence*, 51(8), 5701–5725. <https://doi.org/10.1007/s10489-020-02145-w>
- Hariharan, B., Arbelaez, P., Girshick, R., & Malik, J. (2015). Hypercolumns for object segmentation and fine-grained localization. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 447–456. <https://doi.org/10.1109/CVPR.2015.7298642>
- Hassan, M. U. (2021). *VGG16 – Convolutional Network for Classification and Detection*. <https://neurohive.io/en/popular-networks/vgg16/>
- He, J., Li, P., Geng, Y., & Xie, X. (2023). *FastInst: A Simple Query-Based Model for Real-Time Instance Segmentation*.
- He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. *Proceedings of the IEEE International Conference on Computer Vision*, 2961–2969.
- Heo, M., Hwang, S., Oh, S. W., Lee, J.-Y., & Kim, S. J. (2022). *VITA: Video Instance Segmentation via Object Token Association*.
- Hu, J., Cao, L., Lu, Y., Zhang, S., Wang, Y., Li, K., Huang, F., Shao, L., & Ji, R. (2021). *ISTR: End-to-End Instance Segmentation with Transformers*.

- Huang, Y.-P., Wang, T.-H., & Basanta, H. (2020). Using Fuzzy Mask R-CNN Model to Automatically Identify Tomato Ripeness. *IEEE Access*, 8, 207672–207682. <https://doi.org/10.1109/ACCESS.2020.3038184>
- Huzjan, F., Jurić, F., Lončarić, S., & Vujanović, M. (2023). Deep Learning-based Image Analysis Method for Estimation of Macroscopic Spray Parameters. *Neural Computing and Applications*, 35(13), 9535–9548. <https://doi.org/10.1007/s00521-022-08184-3>
- Hwang, S., Heo, M., Oh, S. W., & Kim, S. J. (2021). Video instance segmentation using inter-frame communication transformers. *Advances in Neural Information Processing Systems*, 34, 13352–13363.
- Iannizzotto, G., & Vita, L. (2000). Fast and accurate edge-based segmentation with no contour smoothing in 2-D real images. *IEEE Transactions on Image Processing*, 9(7), 1232–1237.
- Jensen, J. B., Beaton, S. P., Stith, J. L., Schwenz, K., Colón-Robles, M., Rauber, R. M., & Gras, J. (2020). The Giant Nucleus Impactor (GNI)—A System for the Impaction and Automated Optical Sizing of Giant Aerosol Particles with Emphasis on Sea Salt. Part I: Basic Instrument and Algorithms. *Journal of Atmospheric and Oceanic Technology*, 37(9), 1551–1569. <https://doi.org/10.1175/JTECH-D-19-0109.1>
- Johnson, J. W. (2020). Automatic Nucleus Segmentation with Mask-RCNN. In K. Arai & S. Kapoor (Eds.), *Advances in Computer Vision* (pp. 399–407). Springer International Publishing.
- Jüngst, N., Ersoy, V., Smallwood, G. J., & Kaiser, S. A. (2024). Neural networks for classification and segmentation of thermally-induced droplet breakup in spray-flame synthesis. *Journal of Aerosol Science*, 176, 106314. <https://doi.org/10.1016/j.jaerosci.2023.106314>
- Jüngst, N., Smallwood, G. J., & Kaiser, S. A. (2022). Visualization and image analysis of droplet puffing and micro-explosion in spray-flame synthesis of iron oxide nanoparticles. *Experiments in Fluids*, 63(3), 60. <https://doi.org/10.1007/s00348-022-03411-y>
- Kaganami, H. G., & Beiji, Z. (2009). Region-based segmentation versus edge detection. *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 1217–1221.
- Kaldera, H. N. T. K., Gunasekara, S. R., & Dissanayake, M. B. (2019). Brain tumor Classification and Segmentation using Faster R-CNN. *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, 1–6. <https://doi.org/10.1109/ICASET.2019.8714263>
- Kara, E., Zhang, G., Williams, J. J., Ferrandez-Quinto, G., Rhoden, L. J., Kim, M., Kutz, J. N., & Rahman, A. (2023). *Deep Learning Based Object Tracking in Walking Droplet and Granular Intruder Experiments*.
- Kastengren, A., & Powell, C. F. (2014). Synchrotron X-ray techniques for fluid dynamics. *Experiments in Fluids*, 55(3), 1686. <https://doi.org/10.1007/s00348-014-1686-8>
- Ke, L., Danelljan, M., Li, X., Tai, Y.-W., Tang, C.-K., & Yu, F. (2022). Mask Transfiner for High-Quality Instance Segmentation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4402–4411. <https://doi.org/10.1109/CVPR52688.2022.00437>

- Khan, M. A., Akram, T., Zhang, Y.-D., & Sharif, M. (2021). Attributes based skin lesion detection and recognition: A mask RCNN and transfer learning-based deep learning framework. *Pattern Recognition Letters*, 143, 58–66. <https://doi.org/10.1016/j.patrec.2020.12.015>
- Khoreva, A., Benenson, R., Hosang, J., Hein, M., & Schiele, B. (2016). *Simple Does It: Weakly Supervised Instance and Semantic Segmentation*.
- Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2019). Panoptic segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9404–9413.
- Kirsch, R. A. (1971). Computer determination of the constituent structure of biological images. *Computers and Biomedical Research*, 4(3), 315–328.
- Kompella, G., Antico, M., Sasazawa, F., Jeevakala, S., Ram, K., Fontanarosa, D., Pandey, A. K., & Sivaprakasam, M. (2019). Segmentation of Femoral Cartilage from Knee Ultrasound Images Using Mask R-CNN. *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 966–969. <https://doi.org/10.1109/EMBC.2019.8857645>
- Koračin, N., Zupančič, M., Vrečer, F., Hudovornik, G., & Golobič, I. (2022). Characterization of the spray droplets and spray pattern by means of innovative optical microscopy measurement method with the high-speed camera. *International Journal of Pharmaceutics*, 629, 122412. <https://doi.org/10.1016/j.ijpharm.2022.122412>
- Kumar, S. S., Li, C., Christen, C. E., Hogan, C. J., Fredericks, S. A., & Hong, J. (2019). Automated droplet size distribution measurements using digital inline holography. *Journal of Aerosol Science*, 137, 105442. <https://doi.org/10.1016/j.jaerosci.2019.105442>
- Lei, Y., Zhou, J., Dong, X., Wang, T., Mao, H., McDonald, M., Curran, W. J., Liu, T., & Yang, X. (2020). Multi-organ segmentation in head and neck MRI using U-Faster-RCNN. In B. A. Landman & I. Išgum (Eds.), *Medical Imaging 2020: Image Processing* (p. 117). SPIE. <https://doi.org/10.1117/12.2549596>
- Li, J., Liang, X., Shen, S., Xu, T., Feng, J., & Yan, S. (2017). Scale-aware Fast R-CNN for Pedestrian Detection. *IEEE Transactions on Multimedia*, 1–1. <https://doi.org/10.1109/TMM.2017.2759508>
- Lin, T., Wang, Y., Liu, X., & Qiu, X. (2022). A survey of transformers. *AI Open*, 3, 111–132. <https://doi.org/10.1016/j.aiopen.2022.10.001>
- Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2117–2125.
- Linne, M. (2013). Imaging in the optically dense regions of a spray: A review of developing techniques. *Progress in Energy and Combustion Science*, 39(5), 403–440. <https://doi.org/10.1016/j.pecs.2013.06.001>
- Liu, M., Dong, J., Dong, X., Yu, H., & Qi, L. (2018). Segmentation of Lung Nodule in CT Images Based on Mask R-CNN. *2018 9th International Conference on Awareness Science and Technology (ICAST)*, 1–6. <https://doi.org/10.1109/ICAwST.2018.8517248>

- Liu, M., Duan, Y., Zhang, T., & Xu, Y. (2011). Evaluation of unsteadiness in effervescent sprays by analysis of droplet arrival statistics – The influence of fluids properties and atomizer internal design. *Experimental Thermal and Fluid Science*, 35(1), 190–198. <https://doi.org/10.1016/j.expthermflusci.2010.09.001>
- Luo, K., Shao, C., Chai, M., & Fan, J. (2019). Level set method for atomization and evaporation simulations. *Progress in Energy and Combustion Science*, 73, 65–94. <https://doi.org/10.1016/j.pecs.2019.03.001>
- Manuputty, M. Y., Lindberg, C. S., Botero, M. L., Akroyd, J., & Kraft, M. (2019). Detailed characterisation of TiO<sub>2</sub> nano-aggregate morphology using TEM image analysis. *Journal of Aerosol Science*, 133, 96–112. <https://doi.org/10.1016/j.jaerosci.2019.04.012>
- Minov, S., Cointault, F., Vangeyte, J., Pieters, J., & Nuytens, D. (2016). Spray Droplet Characterization from a Single Nozzle by High Speed Image Analysis Using an In-Focus Droplet Criterion. *Sensors*, 16(2), 218. <https://doi.org/10.3390/s16020218>
- Mlkvik, M., Stähle, P., Schuchmann, H. P., Gaukel, V., Jedelsky, J., & Jicha, M. (2015). Twin-fluid atomization of viscous liquids: The effect of atomizer construction on breakup process, spray stability and droplet size. *International Journal of Multiphase Flow*, 77, 19–31. <https://doi.org/10.1016/j.ijmultiphaseflow.2015.06.010>
- Mueller, M., Segl, K., & Kaufmann, H. (2004). Edge-and region-based segmentation technique for the extraction of large, man-made objects in high-resolution satellite imagery. *Pattern Recognition*, 37(8), 1619–1628.
- Mukherjee, S., & Acton, S. T. (2014). Region based segmentation in presence of intensity inhomogeneity using legendre polynomials. *IEEE Signal Processing Letters*, 22(3), 298–302.
- Mulay Supriti and Deepika, G. and J. S. and R. K. and S. M. (2020). Liver Segmentation from Multimodal Images Using HED-Mask R-CNN. In R. and D. B. and L. X. Li Quanzheng and Leahy (Ed.), *Multiscale Multimodal Medical Imaging* (pp. 68–75). Springer International Publishing.
- Notaro, V., Khare, P., & Lee, J. G. (2019). Mixing Characteristics of Non-Newtonian Impinging Jets at Elevated Pressures. *Flow, Turbulence and Combustion*, 102(2), 355–372. <https://doi.org/10.1007/s10494-018-9955-x>
- Pathan, R. K., Lim, W. L., Lau, S. L., Ho, C. C., Khare, P., & Koneru, R. B. (2022). Experimental Analysis of U-Net and Mask R-CNN for Segmentation of Synthetic Liquid Spray. *2022 IEEE International Conference on Computing (ICOCO)*, 237–242. <https://doi.org/10.1109/ICOCO56118.2022.10031951>
- Poozesh, S., Akafuah, N. K., Campbell, H. R., Bashiri, F., & Saito, K. (2020). Experimental and Mathematical Tools to Predict Droplet Size and Velocity Distribution for a Two-Fluid Nozzle. *Fluids*, 5(4), 231. <https://doi.org/10.3390/fluids5040231>
- Prewitt, J. M. S. (1970). Object enhancement and extraction. *Picture Processing and Psychopictorics*, 10(1), 15–19.
- Purkait, P., Zhao, C., & Zach, C. (2017). SPP-Net: Deep absolute pose regression with synthetic views. *ArXiv Preprint ArXiv:1712.03452*.

- Qiao, Y., Truman, M., & Sukkarieh, S. (2019). Cattle segmentation and contour extraction based on Mask R-CNN for precision livestock farming. *Computers and Electronics in Agriculture*, 165, 104958. <https://doi.org/10.1016/j.compag.2019.104958>
- Rajab, M. I., Woolfson, M. S., & Morgan, S. P. (2004). Application of region-based segmentation and neural network edge detection to skin lesions. *Computerized Medical Imaging and Graphics*, 28(1–2), 61–68.
- Rastogi, A. (2022). *ResNet50 - Dev Genius*. <https://blog.devgenius.io/resnet50-6b42934db431>
- Ray, R., Henshaw, P., & Biswas, N. (2019). Characteristics of Spray Atomization for Liquid Droplets Formed Using a Rotary Bell Atomizer. *Journal of Fluids Engineering*, 141(8). <https://doi.org/10.1115/1.4042562>
- Redding, J. P., & Khare, P. (2022). A computational study on shock induced deformation, fragmentation and vaporization of volatile liquid fuel droplets. *International Journal of Heat and Mass Transfer*, 184, 122345. <https://doi.org/10.1016/j.ijheatmasstransfer.2021.122345>
- Rehman, Z. ur, Khan, M. A., Ahmed, F., Damaševičius, R., Naqvi, S. R., Nisar, W., & Javed, K. (2021). Recognizing apple leaf diseases using a novel parallel real-time processing framework based on MASK RCNN and transfer learning: An application for smart agriculture. *IET Image Processing*, 15(10), 2157–2168. <https://doi.org/10.1049/ipr2.12183>
- Reitz, R. D. (2013). Directions in internal combustion engine research. *Combustion and Flame*, 160(1), 1–8. <https://doi.org/10.1016/j.combustflame.2012.11.002>
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28.
- ROGOWSKA, J. (1999). Overview and Fundamentals of Medical Image Segmentation. In *Handbook of Medical Image Processing and Analysis* (pp. 73–90). Elsevier. <https://doi.org/10.1016/B978-012373904-9.50013-1>
- Rong, W., Li, Z., Zhang, W., & Sun, L. (2014). An improved CANNY edge detection algorithm. *2014 IEEE International Conference on Mechatronics and Automation*, 577–582.
- Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 234–241.
- Rossi, L., Karimi, A., & Prati, A. (2021). A Novel Region of Interest Extraction Layer for Instance Segmentation. *2020 25th International Conference on Pattern Recognition (ICPR)*, 2203–2209. <https://doi.org/10.1109/ICPR48806.2021.9412258>
- Sharma, N., Bachalo, W. D., & Agarwal, A. K. (2020). Spray droplet size distribution and droplet velocity measurements in a firing optical engine. *Physics of Fluids*, 32(2), 023304. <https://doi.org/10.1063/1.5126498>
- Sharma, R., Saqib, M., Lin, C. T., & Blumenstein, M. (2022). A Survey on Object Instance Segmentation. *SN Computer Science*, 3(6), 499. <https://doi.org/10.1007/s42979-022-01407-3>

- Sikka, R., Halstensen, M., & Lundberg, J. (2022). Spray characterization in air-assist atomizers using flow-induced acoustic vibrations and multivariate analysis. *Flow Measurement and Instrumentation*, 86, 102209. <https://doi.org/10.1016/j.flowmeasinst.2022.102209>
- Siva Raja, P. M., & rani, A. V. (2020). Brain tumor classification using a hybrid deep autoencoder with Bayesian fuzzy clustering-based segmentation approach. *Biocybernetics and Biomedical Engineering*, 40(1), 440–453. <https://doi.org/10.1016/j.bbe.2020.01.006>
- Snyder, W., Bilbro, G., Logenthiran, A., & Rajala, S. (1990). Optimal thresholding—a new approach. *Pattern Recognition Letters*, 11(12), 803–809.
- Sonawane, U., & Agarwal, A. K. (2022). *Spray Breakup Modelling for Internal Combustion Engines* (pp. 57–85). [https://doi.org/10.1007/978-981-16-8618-4\\_4](https://doi.org/10.1007/978-981-16-8618-4_4)
- Song, L., Huang, J., Liang, X., Yang, S. X., Hu, W., & Tang, D. (2020). An Intelligent Multi-Sensor Variable Spray System with Chaotic Optimization and Adaptive Fuzzy Control. *Sensors*, 20(10), 2954. <https://doi.org/10.3390/s20102954>
- Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Training Very Deep Networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 28). Curran Associates, Inc. <https://proceedings.neurips.cc/paper/2015/file/215a71a12769b056c3c32e7299f1c5ed-Paper.pdf>
- Tang, W., Zou, D., Yang, S., & Shi, J. (2018). DSL: Automatic Liver Segmentation with Faster R-CNN and DeepLab. In V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, & I. Maglogiannis (Eds.), *Artificial Neural Networks and Machine Learning – ICANN 2018* (pp. 137–147). Springer International Publishing.
- Tang, W., Zou, D., Yang, S., Shi, J., Dan, J., & Song, G. (2020). A two-stage approach for automatic liver segmentation with Faster R-CNN and DeepLab. *Neural Computing and Applications*, 32(11), 6769–6778. <https://doi.org/10.1007/s00521-019-04700-0>
- Tippett, J. T., Borkowitz, D. A., Clapp, L. C., Koester, C. J., & Vanderburgh Jr, A. (1965). *Optical and electro-optical information processing*. Massachusetts Inst of Tech Cambridge.
- Van de Sande, K. E. A., Uijlings, J. R. R., Gevers, T., & Smeulders, A. W. M. (2011). Segmentation as selective search for object recognition. *2011 International Conference on Computer Vision*, 1879–1886.
- Vuola, A. O., Akram, S. U., & Kannala, J. (2019). Mask-RCNN and U-Net Ensembled for Nuclei Segmentation. *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, 208–212. <https://doi.org/10.1109/ISBI.2019.8759574>
- Wallington, T. J., Kaiser, E. W., & Farrell, J. T. (2006). Automotive fuels and internal combustion engines: a chemical perspective. *Chemical Society Reviews*, 35(4), 335. <https://doi.org/10.1039/b410469m>
- Wang, D., & He, D. (2022). Fusion of Mask RCNN and attention mechanism for instance segmentation of apples under complex background. *Computers and Electronics in Agriculture*, 196, 106864. <https://doi.org/10.1016/j.compag.2022.106864>

- Wang, X., Shrivastava, A., & Gupta, A. (2017). A-Fast-RCNN: Hard Positive Generation via Adversary for Object Detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3039–3048. <https://doi.org/10.1109/CVPR.2017.324>
- Wang, Y., Xu, Z., Wang, X., Shen, C., Cheng, B., Shen, H., & Xia, H. (2020). *End-to-End Video Instance Segmentation with Transformers*.
- Wei, M., Chen, S., Sun, M., Liang, J., Liu, C., & Wang, M. (2020). Atomization simulation and preparation of 24CrNiMoY alloy steel powder using VIGA technology at high gas pressure. *Powder Technology*, 367, 724–739. <https://doi.org/10.1016/j.powtec.2020.04.030>
- Wen, Q., Yang, J., Yang, X., & Liang, K. (2023). *PatchDCT: Patch Refinement for High Quality Instance Segmentation*.
- Wu, J., Jiang, Y., Bai, S., Zhang, W., & Bai, X. (2022). *SeqFormer: Sequential Transformer for Video Instance Segmentation* (pp. 553–569). [https://doi.org/10.1007/978-3-031-19815-1\\_32](https://doi.org/10.1007/978-3-031-19815-1_32)
- Wu, J., Liu, Q., Jiang, Y., Bai, S., Yuille, A., & Bai, X. (2022). *In Defense of Online Models for Video Instance Segmentation* (pp. 588–605). [https://doi.org/10.1007/978-3-031-19815-1\\_34](https://doi.org/10.1007/978-3-031-19815-1_34)
- Xin, M., Wang, Y., & Suo, X. (2021). Based on Fast-RCNN Multi Target Detection of Crop Diseases and Pests in Natural Light. In J. Abawajy, Z. Xu, M. Atiquzzaman, & X. Zhang (Eds.), *2021 International Conference on Applications and Techniques in Cyber Intelligence* (pp. 132–139). Springer International Publishing.
- Xu, B., Wang, W., Falzon, G., Kwan, P., Guo, L., Sun, Z., & Li, C. (2020). Livestock classification and counting in quadcopter aerial images using Mask R-CNN. *International Journal of Remote Sensing*, 41(21), 8121–8142. <https://doi.org/10.1080/01431161.2020.1734245>
- Xu, Z., Wu, Z., & Feng, J. (2018). *CFUN: Combining Faster R-CNN and U-net Network for Efficient Whole Heart Segmentation*. <https://arxiv.org/abs/1812.04914>
- Yang, S., Wang, X., Li, Y., Fang, Y., Fang, J., Liu, W., Zhao, X., & Shan, Y. (2022). *Temporally Efficient Vision Transformer for Video Instance Segmentation*.
- Yang, Z., Chung, F.-L., & Shitong, W. (2009). Robust fuzzy clustering-based image segmentation. *Applied Soft Computing*, 9(1), 80–84.
- Yu, Y., Zhang, K., Yang, L., & Zhang, D. (2019). Fruit detection for strawberry harvesting robot in non-structural environment based on Mask-RCNN. *Computers and Electronics in Agriculture*, 163, 104846. <https://doi.org/10.1016/j.compag.2019.06.001>
- Zhan, C., Luo, H., Chang, F., Nishida, K., Ogata, Y., Tang, C., Feng, Z., & Huang, Z. (2021). Experimental study on the droplet characteristics in the spray tip region: Comparison between the free and impinging spray. *Experimental Thermal and Fluid Science*, 121, 110288. <https://doi.org/10.1016/j.expthermflusci.2020.110288>
- Zhang, T.-T., Chow, C.-Y., & Zhang, J.-D. (2021). Fish Image Instance Segmentation: An Enhanced Hybrid Task Cascade Approach. *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, 306–313. <https://doi.org/10.1109/ICSC50631.2021.00058>

- Zhang, Y., Chu, J., Leng, L., & Miao, J. (2020). Mask-Refined R-CNN: A Network for Refining Object Details in Instance Segmentation. *Sensors*, 20(4), 1010. <https://doi.org/10.3390/s20041010>
- Zheng, Z., Wang, P., Ren, D., Liu, W., Ye, R., Hu, Q., & Zuo, W. (2022). Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation. *IEEE Transactions on Cybernetics*, 52(8), 8574–8586. <https://doi.org/10.1109/TCYB.2021.3095305>
- Zhu, G., Piao, Z., & Kim, S. C. (2020). Tooth Detection and Segmentation with Mask R-CNN. *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 070–072. <https://doi.org/10.1109/ICAIIC48513.2020.9065216>
- Zhu, Y., Ma, C., & Du, J. (2019). Rotated cascade R-CNN: A shape robust detector with coordinate regression. *Pattern Recognition*, 96, 106964.