Computer Vision Enabled Dropwise Condensation Heat Transfer Measurement

ME-399 (Project Course)

Prof. Soumyadip Sett

Discipline of Mechanical Engineering, IIT Gandhinagar

Mayur Patil (21110126)

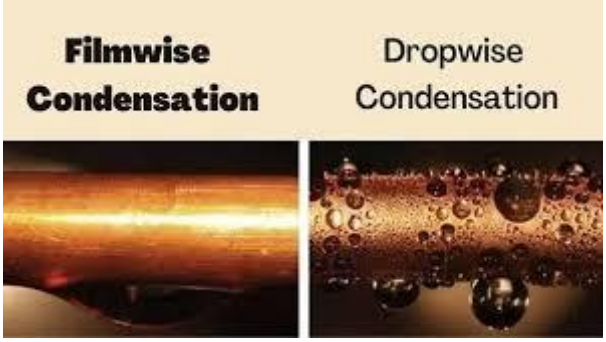
Aditya Gupte (21110012)

Table of Contents:

Introduction:

Condensation is a critical phase change process that occurs when vapor comes into contact with a surface that is at a lower temperature, transforming it into a liquid state. This process plays a pivotal role in various industrial applications such as power generation, desalination, and air conditioning. There are two primary methods of condensation: film wise and dropwise.

Film wise condensation occurs when a continuous film of liquid forms on a surface, which presents a higher resistance to heat transfer due to the insulating layer of liquid. In contrast, dropwise condensation is characterized by the formation of discrete droplets on a surface, which coalesce and grow until they slip away. This method of condensation is of particular interest due to its significantly higher heat transfer rate, which is approximately 5-7 times greater than that of film wise condensation for water.



Condensation Types, *source*

The enhanced heat transfer in dropwise condensation offers considerable benefits for improving the efficiency and performance of thermal systems. However, controlling and maintaining dropwise condensation can be challenging due to factors such as surface characteristics and droplet behaviour. This project aims estimate quantification of total heat transfer coefficient. By understanding the nuances of dropwise condensation, we can unlock the potential for greater energy efficiency and innovation in various thermal applications.

We conducted a comparative study to assess the efficacy of computer vision and deep learning methods in detecting and tracking droplets during dropwise condensation. Our focus was to calculate the overall heat flux value purely using computational methods. Computer vision techniques provided a foundational approach for observing droplet behaviour, while deep learning offered advanced capabilities for accurate droplet detection and tracking. By analysing these methodologies, we aimed to optimize the quantification of dropwise condensation and enhance the understanding of heat transfer in this context. The findings from this study provide valuable insights into improving thermal systems' efficiency and performance.

YOLO:

YOLO, short for "You Only Look Once," is a type of computer vision algorithm that can quickly identify and locate objects in images or videos. It was first introduced in 2015 by a team of researchers and marked a major advancement in the field of real-time object detection.

Before YOLO, other object detection methods like R-CNN were slower and less efficient. YOLO changed the game by using a single neural network that can look at an entire image at once and predict the locations and types of objects within it in one go. This approach made YOLO very fast and suitable for real-time applications.

Since its initial release, different versions of YOLO have been developed by various research teams, each bringing improvements in speed, accuracy, and performance. The latest version, YOLOv8, is the most advanced iteration yet.

YOLOv8:

YOLOv8 is the latest iteration of the YOLO (You Only Look Once) family of object detection algorithms. In the YOLOv8 architecture, there are three essential blocks: the Backbone, Neck, and Head. These blocks work together to process input images and generate predictions for object detection.

Backbone: The backbone is responsible for extracting meaningful features from input images. It consists of multiple convolutional layers that capture simple patterns like edges and textures in the initial layers and more complex patterns as the network progresses. These layers create a rich, hierarchical representation of the input image.

- Convolutions: The backbone uses convolutions with small filters (kernels) to analyse different aspects of the image.
- Feature Maps: The network generates feature maps at various scales, allowing it to understand different levels of abstraction in the image.

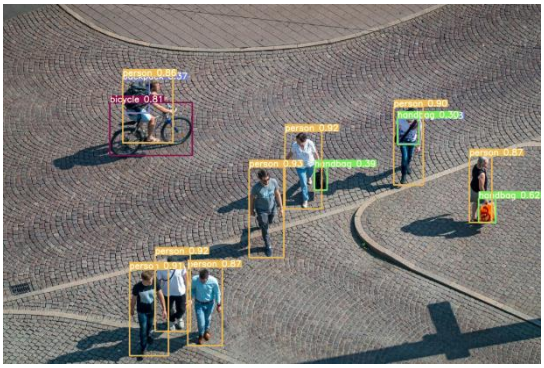
Neck: The neck serves as a bridge between the backbone and the head. It performs feature fusion and integrates contextual information to improve detection accuracy. It combines feature maps from different layers of the backbone to detect objects of different sizes.

- Feature Fusion: The neck fuses feature from multiple layers, allowing the network to consider broader context and improve detection.
- Dimensionality Reduction: It may also reduce the spatial resolution of feature maps, facilitating computation and increasing speed.

Head: The head is the final part of the network and generates the network's outputs, such as bounding boxes and confidence scores for object detection.

- Bounding Boxes: The head predicts bounding boxes for detected objects, indicating their location and size in the image.
- Confidence Scores: It assigns confidence scores to each bounding box, indicating the likelihood of an object being present.
- Class Probabilities: The head also predicts class probabilities, classifying objects within the bounding boxes.

These blocks work together to enable YOLOv8 to perform efficient and accurate object detection. By using a CNN-based architecture with convolutional layers, batch normalization, and other techniques, YOLOv8 effectively processes images and generates precise predictions.



Pre-trained YOLOv8 model, *source*

It comes in five different variants, each tailored for different use cases and levels of complexity. These variants are designed based on the number of parameters they contain, affecting their size and performance. Here is a breakdown of each variant:

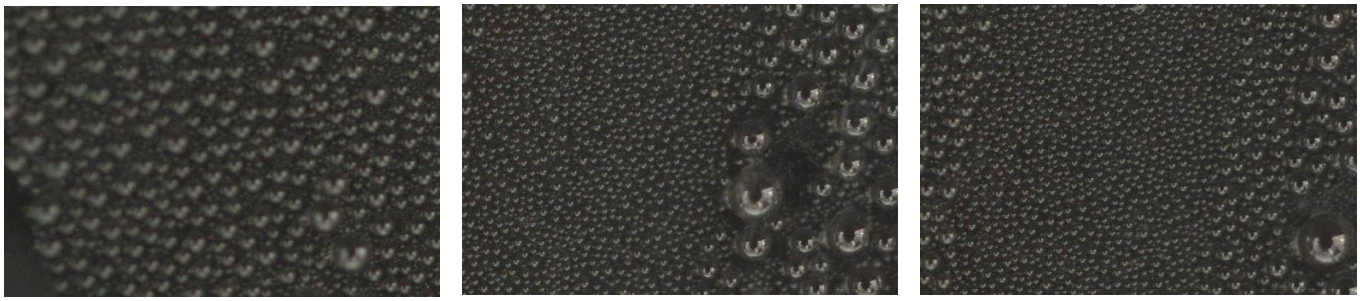
Nano (n), Small (s), Medium (m), Large (l), Extra Large (x)

All YOLOv8 variants can be used for different tasks such as classification (identifying the category of an object), object detection (finding and labelling objects in an image or video), and segmentation (identifying

the exact shape and boundaries of objects). Depending on your specific needs, you can choose the appropriate variant to optimize your model's performance for speed, accuracy, or a balance of both.

Raw Images and Data Preprocessing:

Below are the raw images of droplets taken using camera. The plate here was copper plate, black in color.



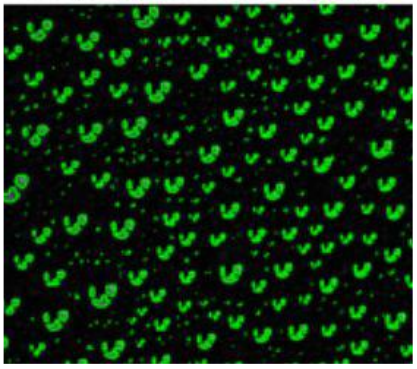
The white coloured circles are the droplets. As seen clearly, these images have been largely impacted by the light of camera. Half portion of droplets is invisible due to reflection of light from camera. Some of the images are found to be blurred. Also, images were of different sizes. Hence, every image is resized to 1920*1280 pixels. Other parameters were also changes like brightness, contrast, RGB values, but it resulted into more bad quality of images.

Training Dataset Preparation:

In the process of developing a model for detecting and tracking droplets, one of the crucial steps is preparing a training dataset. This dataset is essential for training the model how to recognize and identify droplets accurately. We explored two methods for creating this dataset:

- **Using contour area method:** The contour area method involves identifying the edges of droplets in images and tracing their contours to calculate their area. This approach uses computer vision techniques to recognize and extract the shapes of the droplets based on their boundaries. Here is how the process works:
    - **Edge Detection**: It begins by using edge detection algorithms to identify the contours of droplets in the images. Common methods include using the Canny edge detector.
    - **Contour Tracing**: Once the edges are detected, the contours are traced around the droplets. This creates a closed loop that encircles the droplet, as shown in below.
    - **Area Calculation**: After identifying contours, the area enclosed by each contour is calculated. This gives us the size and shape of the droplet. Here, areas, less than 0.05 units was neglected.

Contours of Edited Image



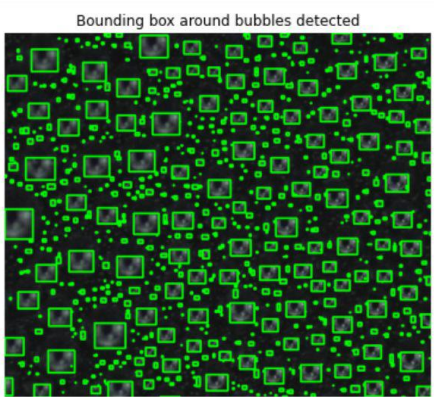
While this method is effective at identifying the size and shape of droplets, it presents some challenges. Since, light was reflected from droplets, each droplet appeared to be in the shape semicircle. Hence, the contours are also drawn in same way (i.e. in the shape of semi-circle) as shown in figure, which creates errors in determination of radius of drop, which is essential part of heat transfer calculation.

Also, this can be sensitive to noise and variations in image quality, which may lead to inaccuracies in contour detection.

- **Using the bounding box method:**

  The bounding box method involves creating rectangular boxes around droplets in images, providing a simpler and more straightforward approach to preparing the training dataset. Here is how the process works:

  - **Object Detection**: It uses object detection algorithms to identify the locations of droplets in the images. The algorithms scan the images and recognize patterns associated with droplets.
  - **Bounding Box Generation**: Once the droplets are detected, it generates rectangular boxes that encompass each droplet. These boxes are defined by their x-y coordinates of centre of box, width of box, height of box, and class number of object detected.
  - **Labelling and Annotation**: Each bounding box is then labelled with the class of the object it contains (in our case, droplets). This information is used to annotate the dataset.



The bounding box method offers several advantages over the contour area method. It is less sensitive to variations in image quality and noise, as it focuses on identifying regions of interest rather than precise contours. Here, the width of bounding box is considered as radius of droplet.

Below is the code used to make and draw annotations of the image, and then create .txt file of annotated values for corresponding image.

```
[ ]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import cv2
     import os
```

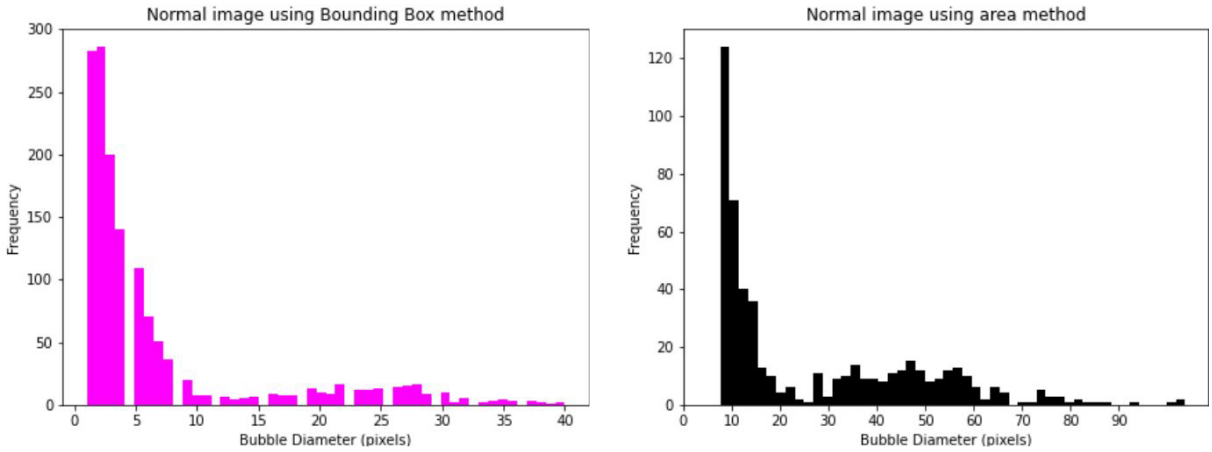
```
[ ]  def Draw_Annotations(image):
         gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
         _, binary = cv2.threshold(gray, 50, 255, cv2.THRESH_BINARY)
         contours, _ = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

         dia = []
         for contour in contours:
             x, y, w, h = cv2.boundingRect(contour)
             dia.append(w)
             cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
         plt.figure(figsize=(10,10))
         plt.imshow(image), plt.axis("off"), plt.title("Bounding box around bubbles detected"), plt.show()
         return dia
       #this function returns only the diameters
```

```
     def Make_Annotations(image):
         gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
         _, binary = cv2.threshold(gray, 50, 255, cv2.THRESH_BINARY)
         contours, _ = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

         arr = []
         i = 0
         for contour in contours:
             x, y, w, h = cv2.boundingRect(contour)
            # print(x)
             arr.append(f"{0} {x/1920} {y/1280} {w/1920} {h/1280}")
             i = i + 1
             cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
         return arr
       #this function returns YOLO parameters
```

```
[ ]  def Text_File_Creator(img, name):
       df = pd.Series(Make_Annotations(img))
       df.to_csv(f'/content/drive/MyDrive/Annotations/{name[:-4]}.txt', index=False, header=False)
```



As seen from above graphs, we have two main reasons for considering Bounding box method to be more effective:

- More number of total droplets are detected in total as compared to contour area method.
- Also, more drops of smaller radii are detected.

```
0 0.6729166666666667 0.9984375 0.0010416666666666667 0.0015625
0 0.5802083333333333 0.9984375 0.0010416666666666667 0.0015625
0 0.5635416666666667 0.9984375 0.0010416666666666667 0.0015625
0 0.36354166666666665 0.9984375 0.0041666666666666667 0.0015625
0 0.13072916666666667 0.99765625 0.0005208333333333333 0.00078125
0 0.09270833333333334 0.99765625 0.0020833333333333333 0.00234375
0 0.7614583333333333 0.996875 0.0020833333333333333 0.003125
0 0.6760416666666667 0.996875 0.0020833333333333333 0.003125
0 0.46770833333333334 0.996875 0.0005208333333333333 0.00078125
0 0.41510416666666666 0.996875 0.0026041666666666665 0.003125
0 0.20416666666666666 0.996875 0.004166666666666667 0.003125
0 0.7833333333333333 0.9953125 0.003125 0.0046875
0 0.7677083333333333 0.9953125 0.004166666666666667 0.0046875
0 0.42291666666666666 0.9953125 0.00625 0.0046875
0 0.4015625 0.9953125 0.0015625 0.003125
0 0.4 0.9953125 0.0005208333333333333 0.00078125
0 0.4036458333333333 0.99453125 0.0005208333333333333 0.00078125
0 0.8536458333333333 0.99375 0.0005208333333333333 0.00078125
0 0.7979166666666667 0.99375 0.0020833333333333333 0.003125
```

Annotated File created using above code

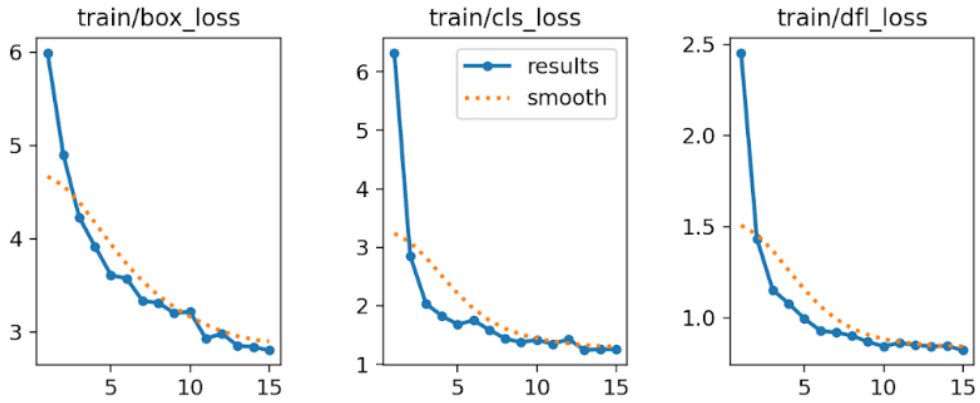
Training and Evaluation of YOLOv8.s:

After training various versions of YOLO, we found that YOLOv8.s fits well as compared to other models. Here are some results that we got through this model.

- *Model summary: 225 layers, 11135987 parameters, 11135971 gradients, 28.6 GFLOPs*
- *Optimizer: AdamW(learning_rate = 0.002, momentum = 0.9) with parameter groups 57 weight (decay=0.0), 64 weight (decay=0.0005), 63 bias (decay=0.0)*
- *Training was done over 15 epochs, which took around 0.783 hours.*

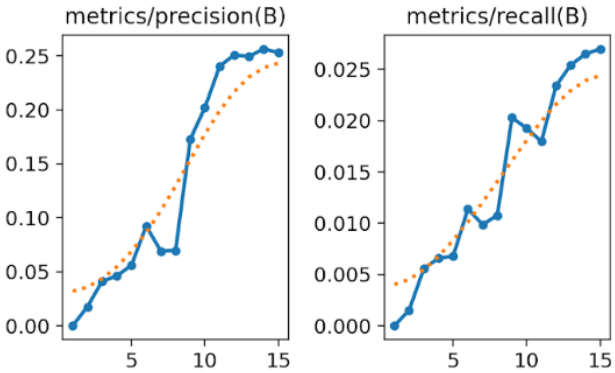
These training metrics are used to train model:

- **box_loss:** The loss value associated with the bounding box predictions during training. Lower values are preferred, indicating the model is making more accurate predictions.
- **cls_loss:** The loss value related to the classification of objects during training. Lower values suggest better model performance.
- **dfl_loss:** The loss value related to the depth for localization (DFL) during training. It is designed to improve localisation accuracy.



These are evaluation metrics used in model:

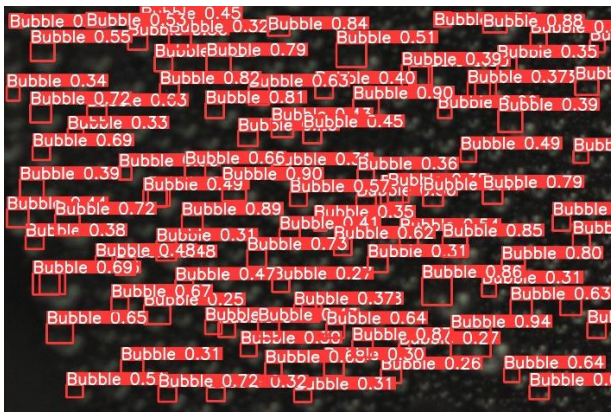
- **Precision:** Precision measures the accuracy of the model's positive predictions. It is the ratio of correctly predicted positive detections (true positives) to the total number of positive predictions (true positives + false positives). It is particularly important in YOLO because it indicates how well the model can detect objects without mistakenly labelling background areas as objects.
- **Recall:** Recall measures the model's ability to detect all relevant objects in an image. It is the ratio of correctly predicted positive detections (true positives) to the total number of actual positives (true positives + false negatives). It is essential because it shows how good the model is at not missing any objects in the images, especially small or partially occluded objects.

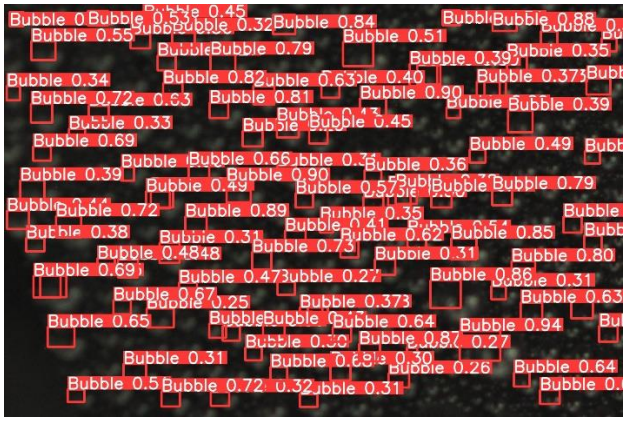


Results:

The maximum number of droplets predicted by model are nearly 100.

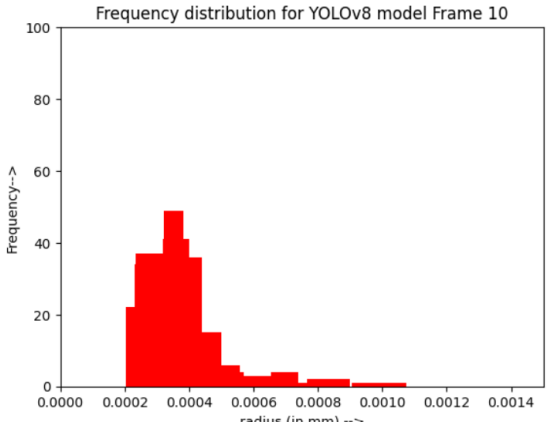
Below are the images of some of the best results we got out from the model while testing:

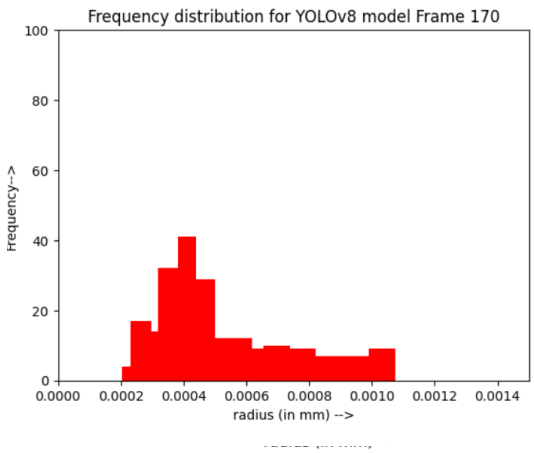


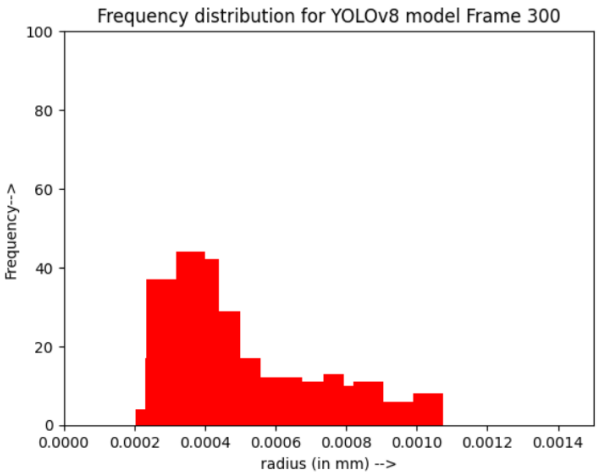


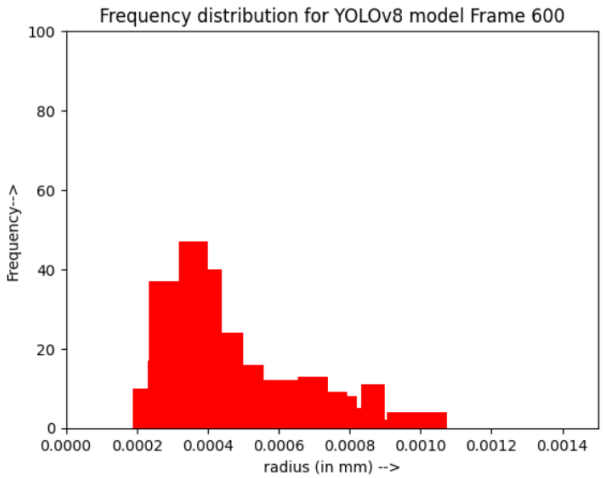
**Bubbles here means droplets*

Plots for Frequency vs. Radius:







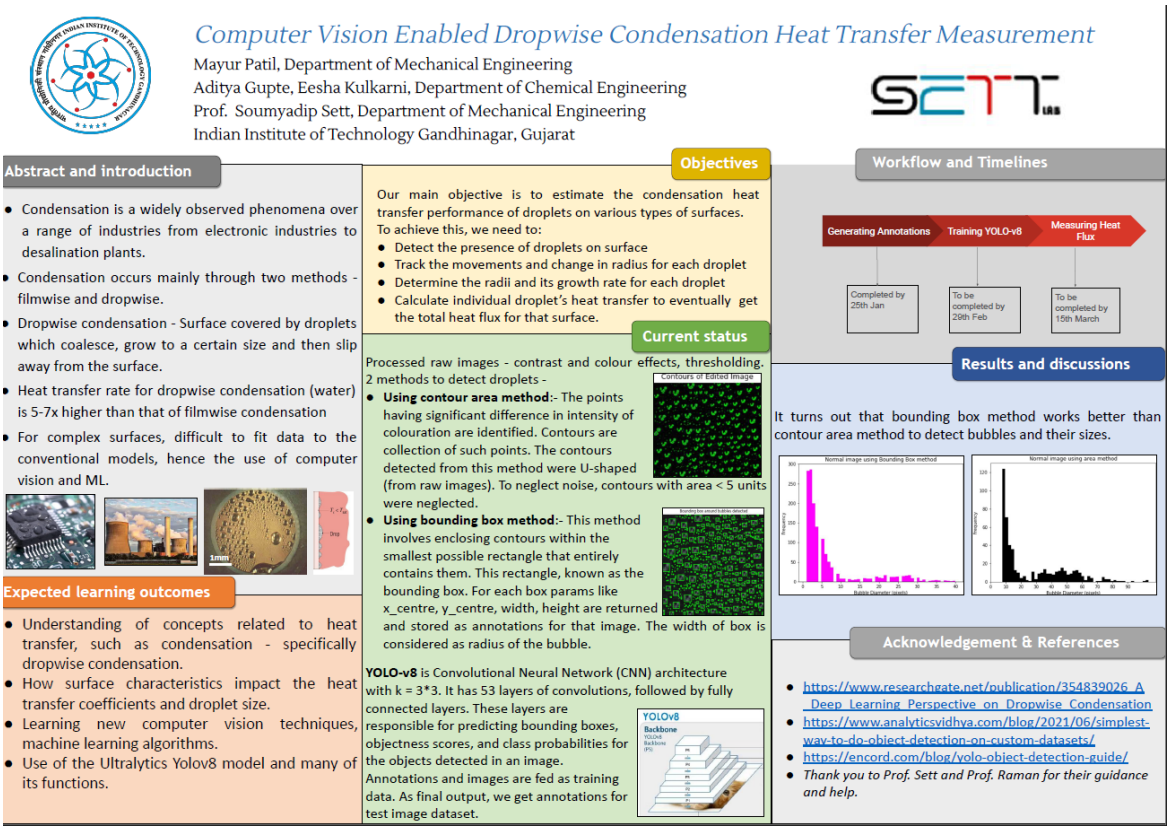


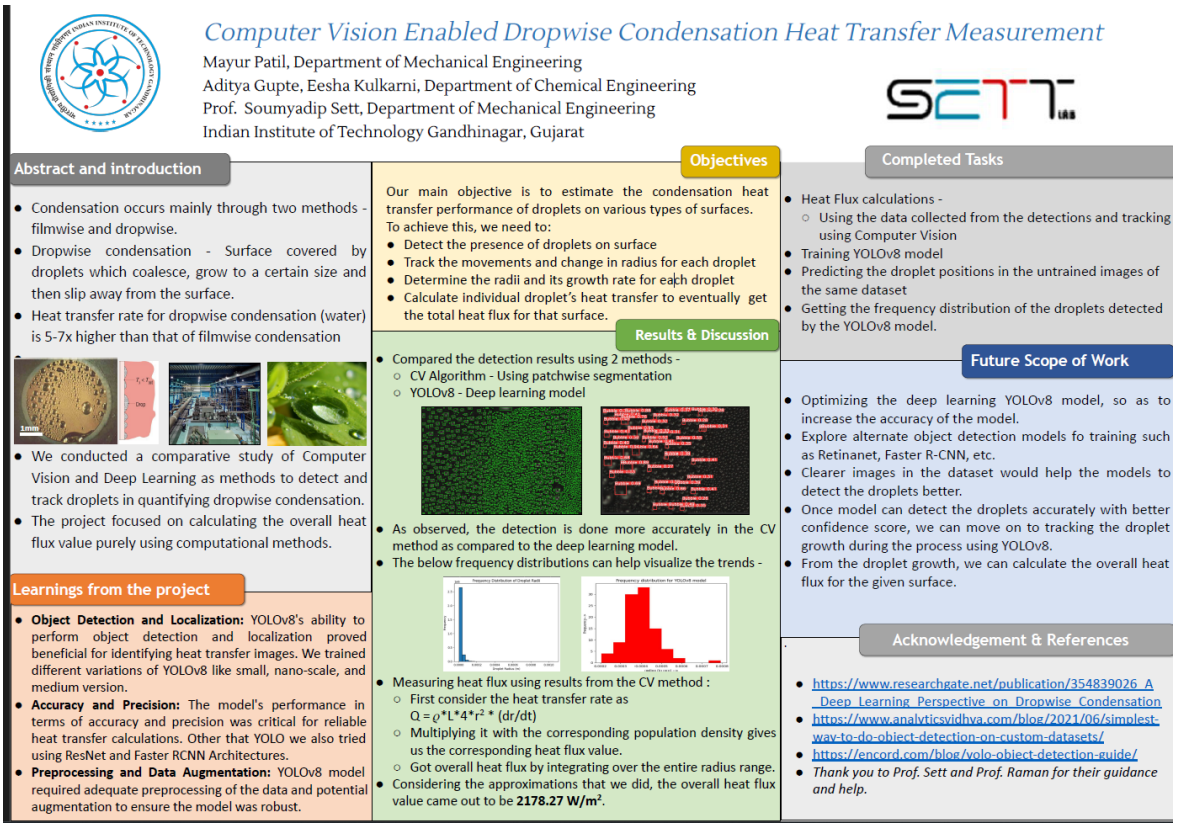
Please find here the link for the Jupyter Notebook for the implementation of the model.

Please find here the link for the final video of graphs in different time frames.

Please find here the link for other models trained for this.

Poster Presentation:





Future Works:

- We will try to optimise this model, so that number of droplets detected per image would increase.
- We are planning to implement Segmentation Anything computer vision model by Meta AI.

References:

- *A Deep Learning Perspective on Dropwise Condensation*. (2021, September). Research Gate. Retrieved April 21, 2024, from https://www.researchgate.net/publication/354839026_A_Deep_Learning_Perspective_on_Dropwise_Condensation
- Acharya, A. (2024, April 4). *YOLO Object Detection Explained: Evolution, Algorithm, and Applications*. https://encord.com/blog/yolo-object-detection-guide/
- Pedro, J. (2023, December 11). *Detailed Explanation of YOLOv8 Architecture — Part 1*. Medium. https://medium.com/@juanpedro.bc22/detailed-explanation-of-yolov8-architecture-part-1-6da9296b954e
- *OpenCV: Canny Edge Detection*. (n.d.). https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html
- Boesch, G. (2024, March 27). *A Guide to YOLOv8 in 2024*. viso.ai. https://viso.ai/deep-learning/yolov8-guide/

Acknowledgement:

As we conclude our work for this semester, we extend our heartfelt thanks to **Prof. Soumyadip Sett**, Mechanical Engineering at IIT Gandhinagar, for his exceptional guidance. We are genuinely appreciative of your support. We would also like to acknowledge **Eesha Kulkarni** for their consistent motivation and assistance. Furthermore, our gratitude goes to the **SETT Lab** for furnishing us with the necessary resources crucial to the successful culmination of the project course.