**DePondFi’23 Challenge Submission**

**NCVPRIPG2023**

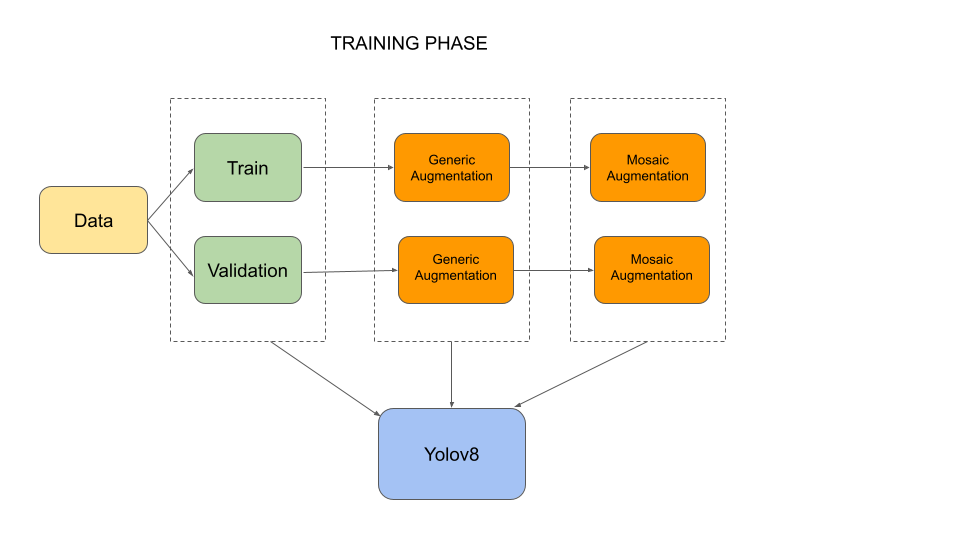
**Team Sahajeevis**

**Suman Paul Choudhury (Senior Data Scientist, SAHAJ AI)**

**Rohit Singh (Senior Data Scientist, SAHAJ AI)**

1. **Block Diagram and Algorithm Explanation**

**Training phase:**

****

The data that we received had around 5951 images along with their corresponding labels. We went with two types of augmentation, a generic augmentation and mosaic data augmentation to augment our data accordingly. The generic data augmentation created another 5951 images with labels and mosaic data augmentation generated around 1000 images with labels. All the images were put together and then we divide our dataset into train and validation using a 80:20 split.

We trained our model with the latest yolov8 object detection[1] module which is one of the most accurate object detection models available. It has been shown to achieve state-of-the-art results on a variety of benchmark datasets and is very fast too.

The model was trained on A100 gpu which took around 1.5 hours to train completely for 40 epochs.

The model was tested on validation dataset and got an mAP of around 0.82

Below are the metrics information

**Metrics**

COCO METRICS:

AP: 0.5331136321021929

AP50: 0.8584553766271087

AP75: 0.5629034392360094

APsmall: 0.2803512691599763

APmedium: 0.4663987909347095

APlarge: 0.6632216025117753

AR1: 0.06273687798227676

AR10: 0.3892978868438991

AR100: 0.5896898432174507

ARsmall: 0.3671380471380472

ARmedium: 0.5388311467030158

ARlarge: 0.7093780464028077

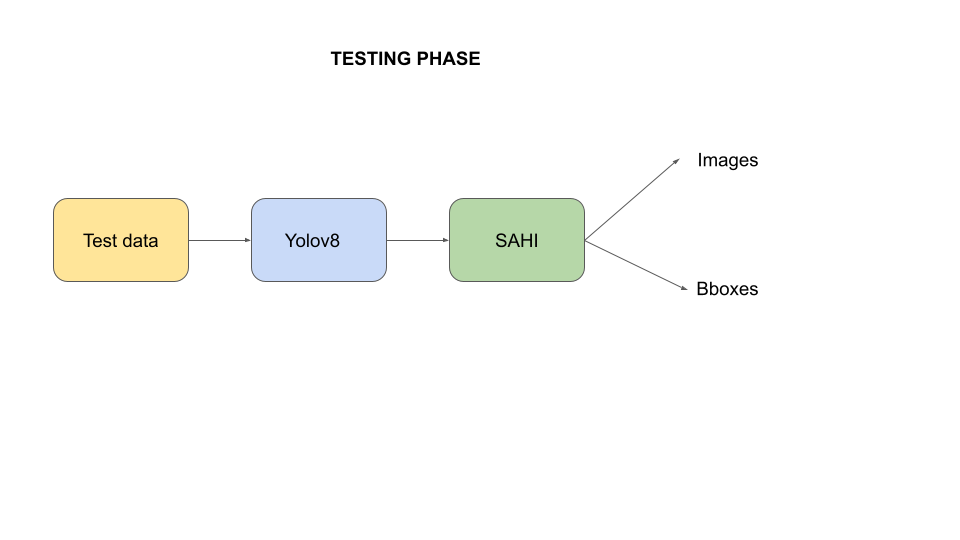
PASCAL METRIC (AP per class)

0: 0.8623294336593308

PASCAL METRIC (mAP)

mAP: 0.8623294336593308

**Testing phase:**

****

After our model was trained , we got out testing the phase-1 dataset and we passed the images to our model which detects the images and boxes. As a post processing step , we pass our images through SAHI [2] which aided in detecting more boxes.

1. **Summary of the approach and experimental result – Phase-I and Phase- II**

**(Detail Explanation)**

*Our Approach:*

1. Training Phase

We received a dataset consisting of 5951 images along with their corresponding labels. To enhance our dataset, we applied two types of augmentation techniques: generic augmentation and mosaic data augmentation.

**Data Augmentation**

**a. Image augmentation(generic)**

Given the domain understanding , we believe that augmentation like gaussian blur[3] , Image contrasting etc will actually replicate a real scenario of detecting fishes and help in our detection process.

We performed one of these types of augmentation in the images

* **HorizontalFlip**
* **VerticalFlip**
* **ShiftScaleRotate**
* **RandomBrightnessContrast**
* **RandomResizedCrop**
* **RGBShift**
* **GaussianBlur**

One example of generic augmentation is as follows:



**b. Mosaic data augmentation**

The idea behind Mosaic is very simple. Take 4 images and combine them into a single image. Mosaic does this by resizing each of the four images, stitching them together, and then taking a random cutout of the stitched images to get the final Mosaic image.

The Mosaic data augmentation[4] refers to combining 4 different images onto one but by changing the sizes of each and putting them under the overall image size which is 640\*640 under these cases.

One example of Mosaic generated image is shown below



We utilized the latest YOLOv8 object detection module to train our model. YOLOv8 is renowned for its exceptional accuracy in object detection and has consistently achieved state-of-the-art results on various benchmark datasets. Furthermore, it is known for its efficient processing speed, making it a fast and effective choice for our training process.

The model was trained on A100 gpu which took around 1.5 hours to train completely for 40 epochs.

The model was tested on validation dataset and got an mAP of around 0.82

1. Testing Phase 1

During the testing phase 1, around 1100 images were given to test out the performance of the Yolov8 model. There was no post processing involved in the model inference

1. Testing Phase 2

During the testing phase 2 , another 1100 images were given to test out the performance of the Yolov8 model. Here we use SAHI (Slicing Aided Hyper Inference) is an open-source framework for small object detection. It introduces a pipeline that leverages slicing aided inference and fine-tuning techniques to improve the detection accuracy of small objects in images.

SAHI works by dividing the input image into overlapping slices. Each slice is then passed through a pre-trained object detection model, and the results are combined to produce a final detection output. This approach allows SAHI to better detect small objects, which are often difficult to detect with traditional object detection methods.

**Tabulation**

| **S.No.** | **Parameters** | **Details** |
| --- | --- | --- |
| 1 | Train Count | 4761 |
| 2 | Validation Count | 1190 |
| 3 | Augmentation | 5951 |
| 4 | Loss Function Name | Classification loss,Localization loss ,Confidence loss |
| 5 | Ensemble Learning (Yes/No) | No |
| 6 | Experimental Setup Details |  |
| * TensorFlow/Pytorch | Pytorch |
| * Learning Rate | 0.001 |
| * Optimizer | Adam |
| * Others (if any) |  |
| 7 | Model Details with number of parameters | 25,843,813 |
| 8 | Details about Pre-Processing | Generic Augmentation,Mosaic Data Augmentation |
| 9 | Details about Post-Processing | SAHI |
| 10 | Others (if any specific) |  |

1. **Conclusion**

Our Yolov8 model when aided with SAHI helps in detecting more fishes in the pond. Though the inference time increases,but if the objective is to detect the fishes in the pond accurately , this method serves the purpose accordingly

1. **References**

[1] YOLOv8: A New State-of-the-Art Computer Vision Model : <https://yolov8.com/>

[2] Slicing Aided Hyper Inference (SAHI) for Small Object Detection | Explained: <https://encord.com/blog/slicing-aided-hyper-inference-explained/>

[3] Image Data Augmentation: A Survey: https://arxiv.org/abs/2204.08610

[4] Mosaic Data Augmentation for Object Detection: https://arxiv.org/abs/1903.12476