

# Configuration Manual

MSc Research Project  
Data Analytics

**Abhishek Sunil Padalkar**

Student ID: x19221576

School of Computing  
National College of Ireland

Supervisor: Dr. Paul Stynes

**National College of Ireland  
Project Submission Sheet  
School of Computing**



<b>Student Name:</b>	Abhishek Sunil Padalkar
<b>Student ID:</b>	x19221576
<b>Programme:</b>	Data Analytics
<b>Year:</b>	2021
<b>Module:</b>	MSc Research Project
<b>Supervisor:</b>	Dr. Paul Stynes
<b>Submission Due Date:</b>	20/09/2021
<b>Project Title:</b>	An Object Detection and Scaling Model for Plastic Waste Sorting
<b>Word Count:</b>	2226
<b>Page Count:</b>	11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

<b>Signature:</b>	
<b>Date:</b>	20/09/2021

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission</b> , to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project</b> , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Abhishek Sunil Padalkar  
x19221576

## 1 Introduction

This configuration manual provides a detailed specification of software and hardware used to implement the research project and a step-by-step guide implemented in the research project. This manual provides information to replicate the research project fully.

## 2 Hardware and Software Specification

### 2.1 Hardware Setup

Table 1 demonstrates all hardware setup utilized in implementing this research project. A local machine(Macbook Pro) with a graphics processing unit(GPU) is primarily required to carry out this project. For high-end object detection model training, free cloud service IDE, Google Colaboratory(Colab) Pro, which provides free 16GB memory P100/V100 GPU, is used.

Table 1: Hardware Specifications

Host Machine	Macbook Pro 16" 2019
Operating System	Mac OS Catalina Version 10.15.6 (19G2021)
Processor	2.3 GHz 8-Core Intel Core i9
Memory	16 GB 2667 MHz DDR4
Graphics / Graphics Card	Intel UHD Graphics 630 1536 MB / AMD Radeon Pro 5500M
Hard Disk	APPLE SSD AP1024N (1TB)
System Type	64-bit Operating System
Cloud IDE GPU (Google Colab Pro)	Tesla P100 / Tesla V100, 16gb graphics card with 32GB memory

### 2.2 Software Setup

Table 2 provides all the software and library used to build and train object detection models, web applications(App) used for plastic waste object detection dataset, transforming the dataset according to the required model input type, and for diagram creation for the report. To find further information on each software, click on the software name.

**Note:** No installation is required for setting up google colab pro. However, a monthly subscription as a pro member is required to obtain high GPU performance.

This configuration manual provides a detailed specification of software and hardware used to implement the research project and a step-by-step guide implemented in the research project. This manual provides information to replicate the research project fully.

Table 2: Software Specifications, Web Application tools and Important Libraries

Programming Language	Python
Cloud IDE	Google Colab Pro
Local IDE	Jupyter Notebook
Annotating Bounding Box to dataset	Labelbox (Web App)
Dataset transformation	Roboflow (Web App)
Model building	Tensorflow, Pytorch, Keras library
Model Evaluation & Results	Seaborn, Matplotlib library
Methodology & Architecture design	draw.io (Web App)

### 3 Data Preparation

Steps taken to prepare an ideal “WaDaBa” plastic waste **object detection** dataset is presented below:

#### Step 1: Data Downloading

The “WaDaBa” dataset consisting of 4000 plastic object images was downloaded from the official website of the dataset published by Bobulski and Piatkowski (2018).<sup>1</sup> A license agreement was signed before using this dataset for research purposes only.

#### Step 2: Bounding Box Annotation

The “WaDaBa” dataset published by Bobulski and Piatkowski (2018) is an image classification dataset. For this dataset to be trained on object detection models, bounding box annotations need to be created around each plastic object in all the images. The dataset was uploaded on Labelbox Web App, which helps the user to create a custom object detection dataset to perform bounding box annotations.<sup>2</sup> Using the Labelbox app, bounding box around each plastic object image was manually annotated. Fig. 1 displays a screenshot of the Labelbox annotation interface while creating the HDPE plastic-type object annotation.

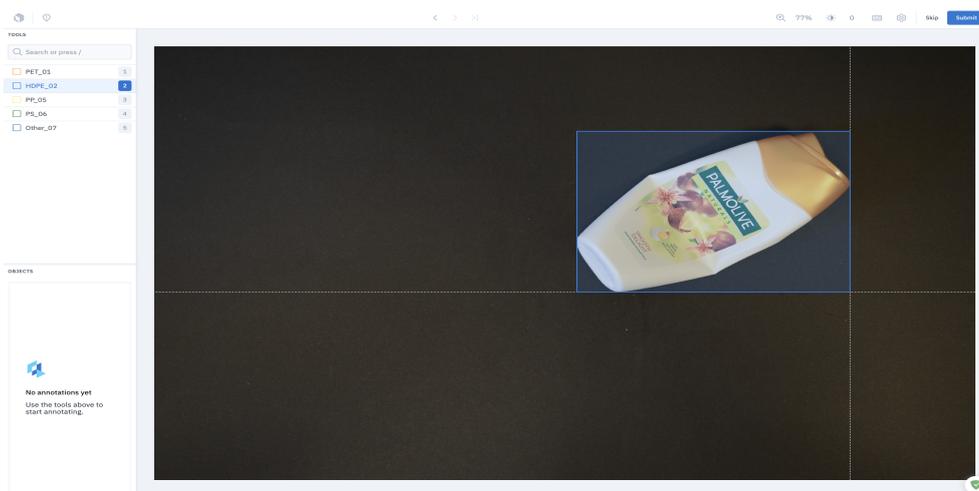


Figure 1: Labelbox bounding box annotation

<sup>1</sup>“WaDaBa” Dataset: <http://wadaba.pcz.pl/index.html>

<sup>2</sup>Labelbox Website Link: <https://app.labelbox.com/>

After annotating all the plastic object images, the Labelbox dataset annotations are exported as a .json file which includes bounding box annotations with classes for all the images. Fig. 2 displays the screenshot of the export option for all the annotated images after performing annotations.

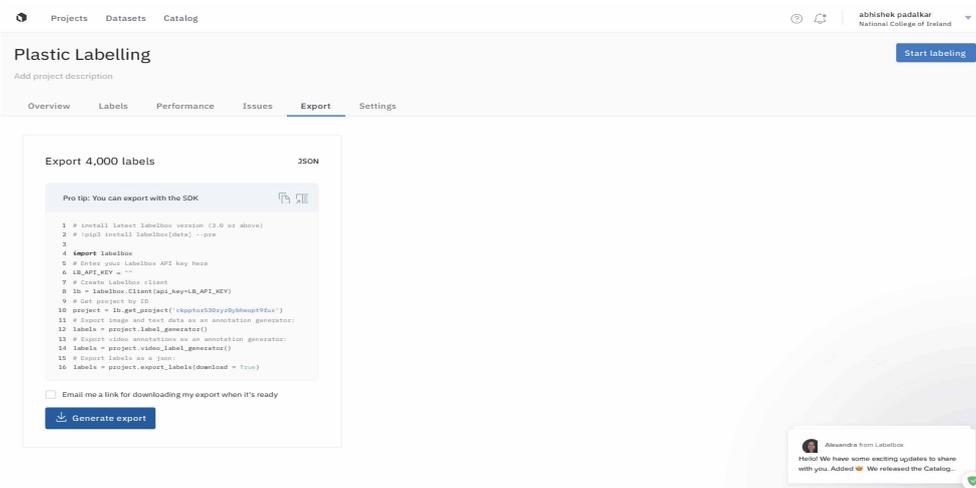


Figure 2: Labelbox Json export

### Step 3: Data Pre-processing and Data transformation

The annotated dataset created is in the Labelbox format, and for the Object detection and Scaling models to read the input, it is necessary to transform it to the required model format. First, we upload the Labelbox format dataset to Roboflow Web app.<sup>3</sup> Roboflow Web app is built for computer vision tasks of data pre-processing, different types of data augmentation, and transforming the data to required object detection models format. Fig. 3 shows the Labelbox annotated JSON uploading to the Roboflow web app.

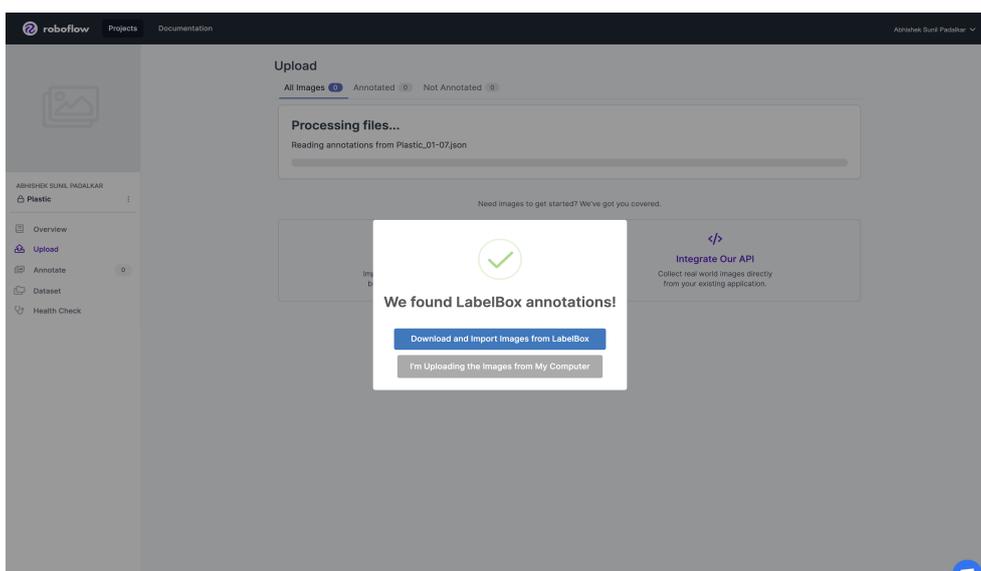


Figure 3: Labelbox Json Upload to Roboflow

First, the data is split into 70:20:10 ratio as train, validation, and test set after uploading the annotated dataset to Roboflow. Then, two versions of the dataset are created. On both versions,

<sup>3</sup>Roboflow Computer Vision Web App: <https://app.roboflow.com/>

the auto-orientation pre-processing step is applied. After applying auto-orientation, one dataset version formed is of original size images. For the other version of the dataset, further pre-processing is performed to resize the image dataset to 416x416 for faster model training and performance. Thus, the second resized version of the dataset is formed. Fig. 4a and fig. 4b shows performed data pre-processing for the original size version and resized version of the plastic dataset.

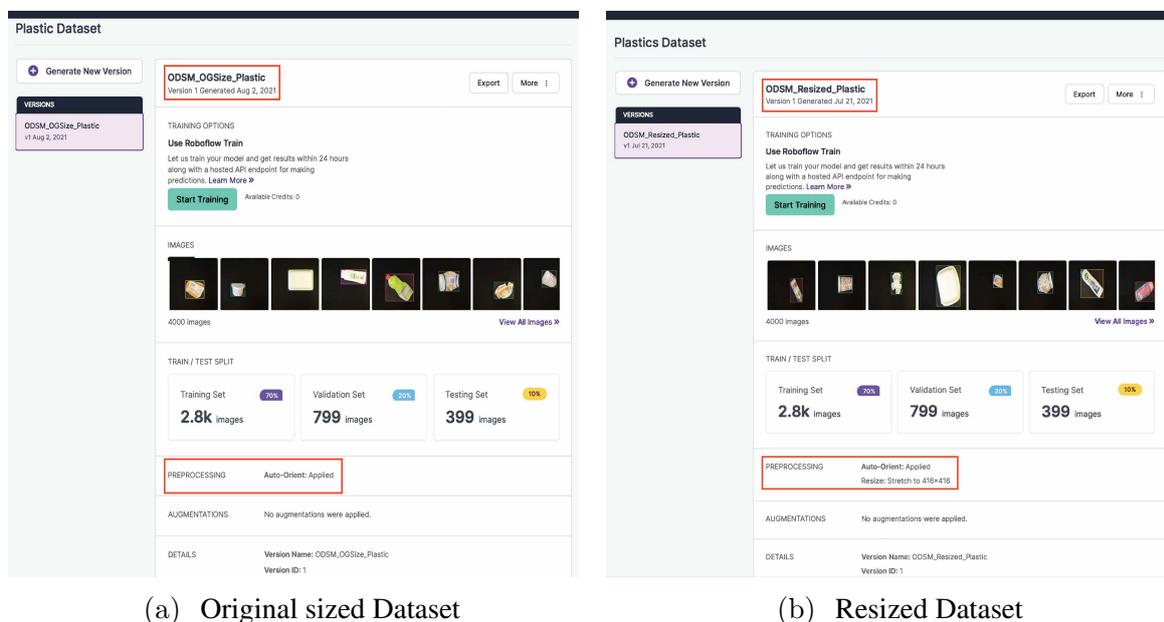


Figure 4: Data Pre-processing for two versions of dataset

After performing data pre-processing, the datasets are exported to the required object detection format of Scaled-Yolov4 and EfficientDet model. Fig. 5 shows the screenshot of exporting of the dataset in the two required formats.

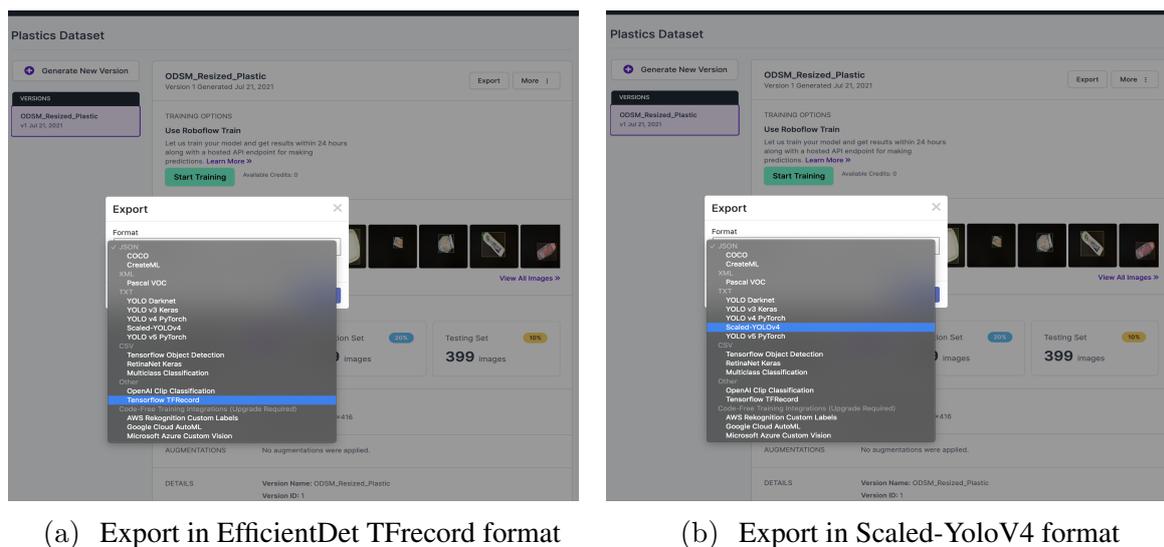


Figure 5: Exporting the data in required model format

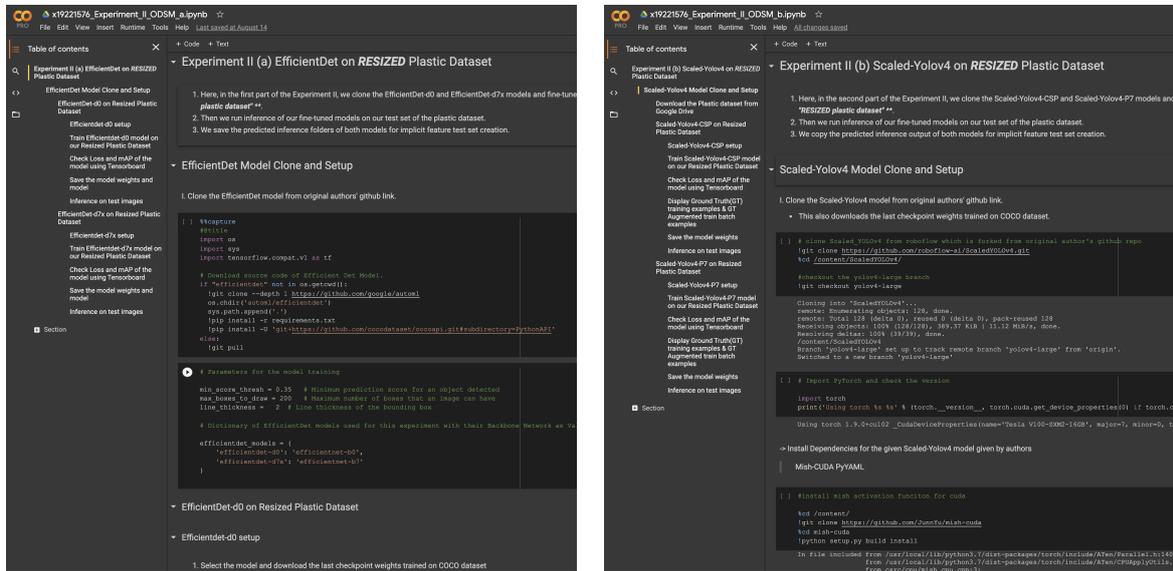
Scaled-Yolov4 format and EfficientDet TFRecord format is downloaded for both resized and original-sized transformed dataset.

# 4 Object Detection and Scaling Model(ODSM) Development

In the ODSM part of the research project, Scaled-Yolov4 and EfficientDet models are finetuned on these two transformed datasets (Wang et al.; 2021; Tan et al.; 2020). These object detection models are complex high-end models cloned from Github from their original author’s repository.<sup>4,5</sup> All the ODSMs’ model development, training and inference are performed on Google Colab Pro.

## 4.1 Model Setup

The models are first cloned from their respective author’s Github repository. After cloning the models, the required dependencies of the models are installed or imported. These requirements are followed by the instructions given in their respective repository. Fig. 6 display screenshots of Model Setup for EfficientDet and Scaled-Yolov4 for resized data.



(a) Original sized Dataset

(b) Resized Dataset

Figure 6: Data Pre-processing for two versions of dataset

For EfficientDet Setup, the model is cloned, and required variables for model training are initialized. The pre-trained weights of the COCO dataset, the last checkpoint, are downloaded and copied in a respective folder of the model. EfficientDet model is implemented using the TensorFlow library.

For Scaled-Yolov4, the model is cloned, and pre-trained weights of the COCO dataset are downloaded. Pytorch library is imported to run Scaled-Yolov4. Dependencies such as Mish-CUDA and PyYAML are installed.

<sup>4</sup>Scaled-Yolov4 Github link: <https://github.com/WongKinYiu/ScaledYOLOv4/tree/676800364a3446900b9e8407bc880ea2127b3415>

<sup>5</sup>EfficientDet Github link: <https://github.com/google/automl/tree/9c58b0b487995d5e6b95ba366cb56cff8f17cd26/efficientdet>

## 4.2 Model Training

The EfficientDet and Scaled-Yolov4 models were then trained with correct parameters to the required train function. Both models were trained for 100 epochs, and batch size differed for attaining convergence without consuming RAM above available RAM. Fig. 7 shows the screenshot of code snippet of model training for EfficientDet-d0 and Scaled-Yolo-CSP on resized plastic data.

(a) EfficientDet-d0 Training (b) Scaled-Yolov4-CSP Training

Figure 7: ODSMs Training

## 4.3 Model Testing and Inference

Following training, another important process of inference testing on the test set is performed with these trained models. Fig. 8 shows models EfficientDet-d0 and Scaled-Yolov4-CSP run inference on the test set. The inference run predicts and detects the plastic object type.

(a) EfficientDet-d0 Test run (b) Scaled-Yolov4-CSP Test run

Figure 8: ODSMs Inference Run on Test set of resized plastic object images

The inference test images are saved in the given folder for each model, which is displayed to check the results visually. Fig.9 shows the screenshots of the inference test images of both models.

Same code style, steps, and procedure are followed to train EfficientDet-d7x, Scaled-Yolov4-P7 on the resized data and to perform a similar experiment on original sized data.

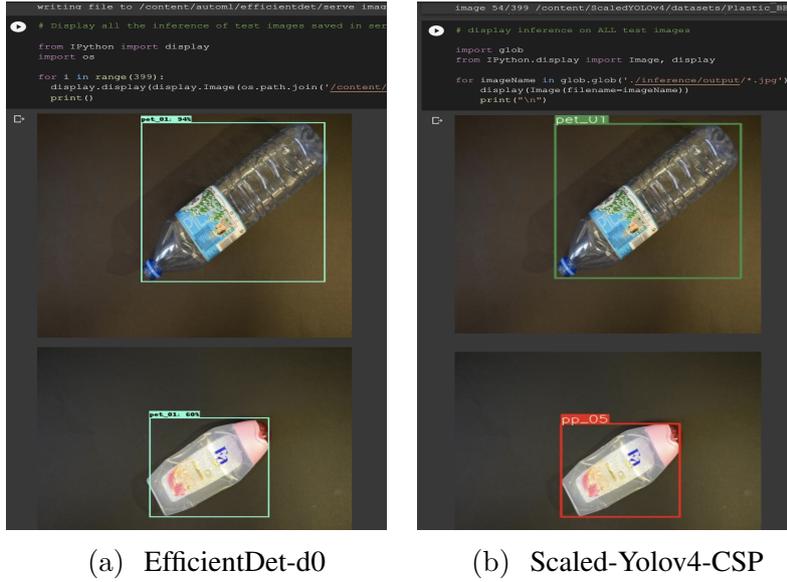


Figure 9: Inference on Test Images for both ODSMs

## 5 Object Detection and Scaling Model(ODSM) Evaluation

The prediction results from the Scaled-Yolov4 model are copied from the output of the inference test. Whereas the EfficientDet model only saved inference images directly, a small python code is developed to type in the predicted output in text format. Fig. 10 displays the code-snippet to get the predicted output from EfficientDet inference results.

```

EfficientDet-d0
In [1]: from IPython import display
import os

test_set = ['testdata/test/0018_a01b04c2d0e0f0g1h4_jpg.rf.965ab96a05715f96f3f161ede55f78f9.jpg', 'testdata/test/005
test_set = [i[14:] for i in test_set]
test_dir = '/Users/abhishekpadaalkar/Documents/IRELAND Admissions/NCI/Course Modules/Modules/Sem 3/Thesis/datasets/P
pred_dir = '/Users/abhishekpadaalkar/Downloads/Downloads/EfficientDet_training/EfficientDet-d0/serve_image_out/'
y_true = []
y_pred = []

for i, img in enumerate(test_set):
    print('Actual Image: ', img)
    display.display(display.Image(os.path.join(test_dir, img)))
    y_true.append(img[6:8])
    print('\nPredicted Image: {}.jpg'.format(i))
    display.display(display.Image(os.path.join(pred_dir, '{}.jpg'.format(i))))
    y_pred.append((str('0')+input('What is predicted?: ')))

```

Figure 10: Getting Predicted output in list format for evaluation

With the predicted class and true class of the test set, the models are evaluated with metrics like accuracy, f1-score, mean average precision(mAP). The train time and inference time is noted while performing training and inference tasks along with the size of the trained model. With these results and measures, a total of 8 models are critically evaluated and compared against each other.

## 5.1 Evaluation and Results

All the scores for the above metrics and measures were manually inputted in a dictionary for each model on each dataset type. Using pandas, seaborn, and matplotlib libraries, the final comparisons and visualization of all the models were performed. Fig. 11 shows the screenshot of a scatterplot with Accuracy vs. mAP achieved by each of the eight trained models.

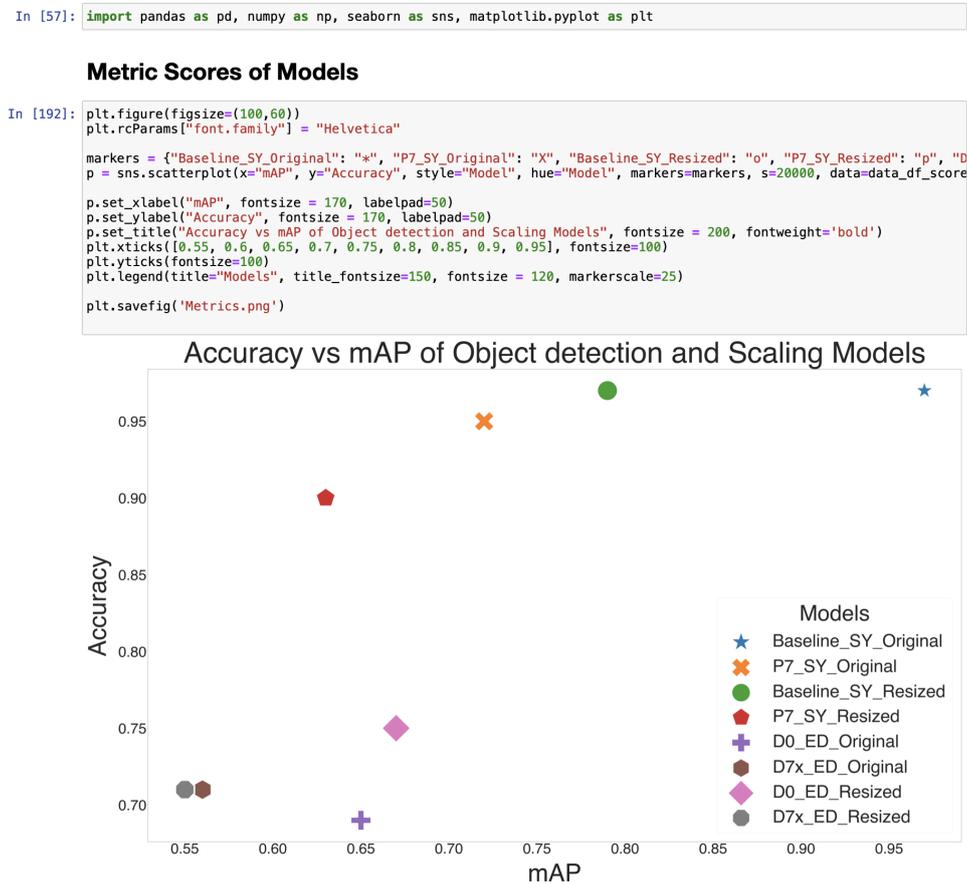


Figure 11: Code snippet of plotting scatterplot of Accuracy vs. mAP for ODSMs

## 6 Data Preparation for Artificial Neural Network (ANN) model

In the next part of the project, the implicit features dataset of the plastic objects is created to train the ANN model.

The “WaDaBa” dataset has an extra set of features for each plastic object image stored in the image file. Pandas library was used to create this new dataset of implicit features. Using python, filenames of the train and validation set of image dataset are extracted, and a train set CSV file is created with implicit features paired with all the plastic types present in the data. This data has 11 features and one true output classification input. The first five are binary features of the predicted class: PET(0/1), HDPE(0/1), PP(0/1), PS(0/1), and Other(0/1). The rest six are the implicit features from the file name: color, type of light, deformation level, dirt level, cap, ring. Each plastic object image is paired with all five predicted classes making sure

no bias in the experiment. Fig. 12 shows the screenshot of code-snippet of implicit features dataset creation.

```

def create_csv_datasets(train_dir, test_dir): # This function is main function to create implicit feature dataset
# Train and Validation image set are joined to create single train set for impl
# which will be used to train the ANW model.

# Test set is created from the inference output of the object detection models
# used as test set on ANW model.

data_train = []
data_test = []

train = '/Users/abhishekpadaalkar/Documents/IRELAND Admissions/NCI/Course Modules/Modules/Sem 3/Thesis/datasets/'
val = '/Users/abhishekpadaalkar/Documents/IRELAND Admissions/NCI/Course Modules/Modules/Sem 3/Thesis/datasets/Pl'
test = '/Users/abhishekpadaalkar/Documents/IRELAND Admissions/NCI/Course Modules/Modules/Sem 3/Thesis/datasets/P

for file_name in os.listdir(train):
    if '.jpg' in file_name:
        row_list = generate_df(file_name, 0)
        data_train += row_list

for file_name in os.listdir(val):
    if '.jpg' in file_name:
        row_list = generate_df(file_name, 0)
        data_train += row_list

for file_name in os.listdir(test):
    if '.jpg' in file_name:
        row_list = generate_df(file_name, 1)
# data_test.append(get_eff_det_pred(file_name) + row_list) # Use this for creating test set of efficient
data_test.append(get_yolo_pred(file_name) + row_list) # Use this for creating test set of scaled-yolov

col_names = ['PET_01', 'HDPE_02', 'PP_05', 'PS_06', 'Other_07', 'Color', 'Type of light', 'deformation level',
data_train_df = pd.DataFrame(data_train, columns = col_names)
data_train_df.to_csv(train_dir, index = False)
data_test_df = pd.DataFrame(data_test, columns = col_names)
data_test_df.to_csv(test_dir, index = False)
return data_train_df, data_test_df

if __name__ == '__main__':
train_set_dir = '/Users/abhishekpadaalkar/Documents/IRELAND Admissions/NCI/Course Modules/Modules/Sem 3/Thesis/d
test_set_dir = '/Users/abhishekpadaalkar/Documents/IRELAND Admissions/NCI/Course Modules/Modules/Sem 3/Thesis/da
train_df, test_df = create_csv_datasets(train_set_dir, test_set_dir)
# train and test set saved in the train_set_dir and test_set_dir, respectively.

```

Figure 12: Code-Snippet for implicit features dataset creation

For the test set of implicit features data creation, the predicted class for the test images for each model is extracted and paired with the test image object implicit features. The test set data file is different for all the models as each model had different predicted values. Fig. 13 demonstrates the implicit feature dataset with train and test for EfficientDet on the original sized dataset.

PET_01	HDPE_02	PP_05	PS_06	Other_07	Color	Type of light	deformation level	dirt level	Presence of cap	Presence of ring	True Label
1	0	0	0	0	00	1	0	0	0	0	05
0	1	0	0	0	00	1	0	0	0	0	05
0	0	1	0	0	00	1	0	0	0	0	05
0	0	0	1	0	00	1	0	0	0	0	05
0	0	0	0	1	00	1	0	0	0	0	05
1	0	0	0	0	00	1	3	1	1	0	01
0	1	0	0	0	00	1	3	1	1	0	01
0	0	1	0	0	00	1	3	1	1	0	01
0	0	0	1	0	00	1	3	1	1	0	01
0	0	0	0	1	00	1	3	1	1	0	01

(a) EfficientDet Train set

PET_01	HDPE_02	PP_05	PS_06	Other_07	Color	Type of light	deformation level	dirt level	Presence of cap	Presence of ring	True Label
1	0	0	0	0	03	2	1	0	1	1	01
0	1	0	0	0	01	2	3	0	0	0	06
0	0	1	0	0	00	2	2	0	0	0	05
0	1	0	0	0	01	2	3	0	0	1	01
0	1	0	0	0	01	1	2	0	1	0	07
1	0	0	0	0	00	1	2	0	0	0	05
0	0	1	0	0	01	2	2	0	0	0	06
1	0	0	0	0	01	1	2	0	1	0	07
0	0	1	0	0	01	1	0	1	0	0	02
0	1	0	0	0	01	1	2	0	0	0	06
0	0	1	0	0	01	1	3	0	0	0	06

(b) EfficientDet-d0 Test set

Figure 13: Implicit features dataset

## 7 Artificial Neural Network (ANN) Model Development

The ANN model is trained on implicit features created to investigate the changes in classification results of ODSMs. For each ODSM, the inference run created different predicted outputs, thus creating a test set of implicit features. Keras library is used to create a simple ANN model.

The ANN model created is made up of Dense layers. The input layer consists of 128 neurons with input shape as (11,) and “tanh” activation function is used for the classification task. Additional three hidden layers with 64, 32, and 16 nodes and “tanh” activation function are present. The final output layer has eight nodes to classify five classes with a “softmax” activation function. Fig. 14 displays the ANN model creation for EfficientDet-d0 resized dataset trained on the implicit features.

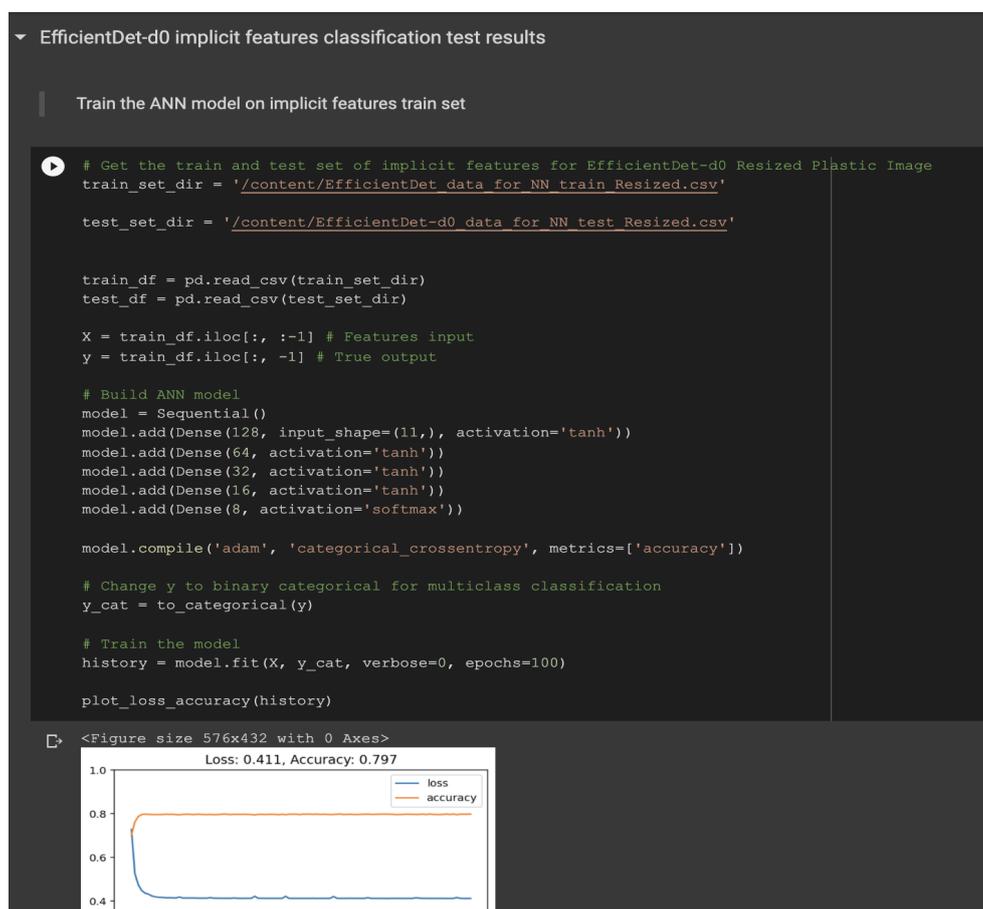


Figure 14: ANN model building and training on implicit features

Fig. 15 shows class prediction using this implicit feature-based trained ANN model on the EfficientDet-d0 predicted output on resized images. Further, to evaluate the model performance, a confusion matrix and classification report are used.

This process is applied for all the 8 ODSMs to investigate the impact of additional implicit feature knowledge in class prediction.

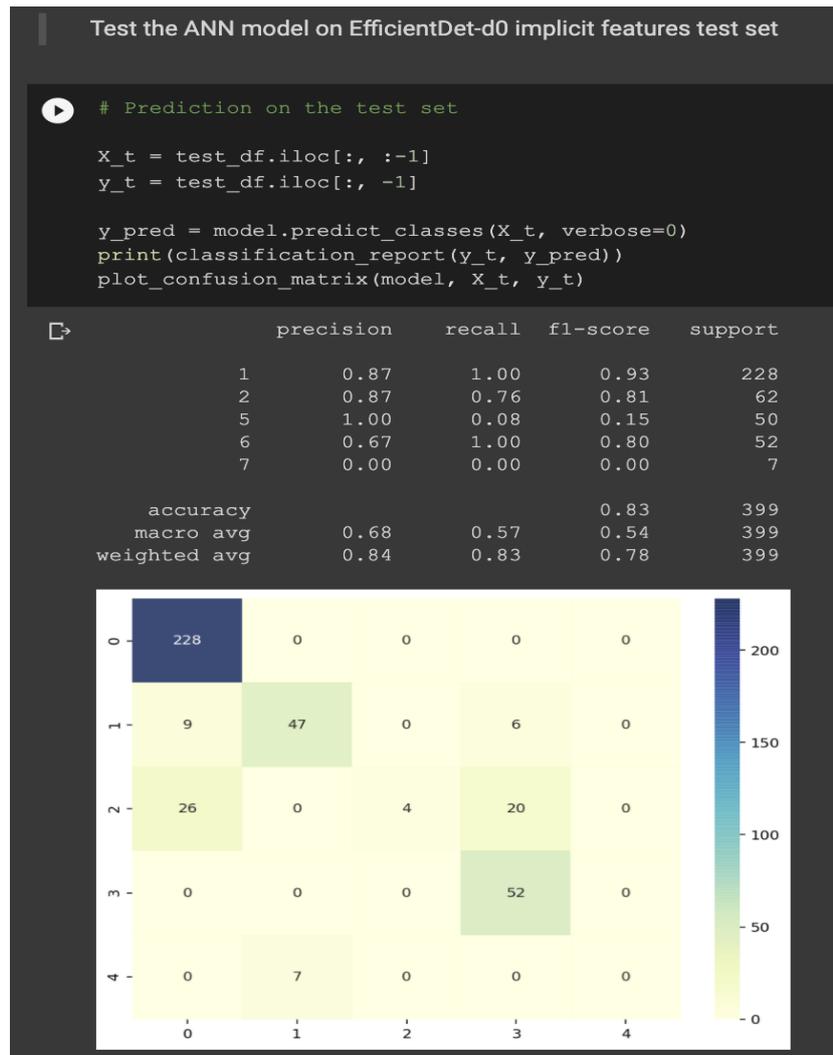


Figure 15: ANN model testing on EfficientDet-d0 prediction results

## References

- Bobulski, J. and Piatkowski, J. (2018). Pet waste classification method and plastic waste database - wadaba, pp. 57–64.
- Tan, M., Pang, R. and Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Wang, C.-Y., Bochkovskiy, A. and Liao, H.-Y. M. (2021). Scaled-yolov4: Scaling cross stage partial network, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13029–13038.