

Configuration Manual

MSc Research Project
MSc in Data Analytics

Madhusudan Narasimhan
Student ID: x20197888

School of Computing
National College of Ireland

Supervisor: Prof. Prashanth Nayak

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Madhusudan Narasimhan
Student ID: 20197888
Programme: MSc in Data Analytics **Year:** 2021-22
Module: Research Project
Lecturer: Prof. Prashant Nayak
Submission Due Date: 15-12-2022
Project Title: Printed Circuit Board Defect Detection using YOLOv7

Word Count: 1596 **Page Count:** 9

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 authors' written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Madhusudan N

Date: 15-12-2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

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

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

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Madhusudan Narasimhan

Student ID: x20197888

1. Introduction

This document provides step-by-step instructions for the project “PCB defect detection using YOLOv7” that is given in a technical report. The goal of this document is to guide the reader to recreate the steps to help them get the output and results discussed in the technical report. A large range of libraries, technologies and configurations are used to implement the project.

1.1 Project Overview

This project aims to evaluate the performance of YOLOv7 in detecting defects on a printed circuit board. YOLOv7, being the latest in the YOLO family is claimed to be the fastest yet algorithm when compared to other object detection algorithms.

2. Pre-requisites

System Requirements:

The following system configuration was used in constructing the model.

- **CPU:** Intel(R) Core (TM) i7-10750H CPU @ 2.60GHz 2.59 GHz
- **GPU:** NVIDIA GeForce RTX 2060, 6 GB
- **RAM:** 16 GB DDR4
- **Memory:** 1 TB SSD

Programming Language: Python

Development Tools: Visual Studio, cudnn

Other Platforms: app.roboflow.com

Python is the primary programming language used in this study. Python 3.7.9 was used in this study and can be sourced for free from anaconda distributions. Data annotation, data pre-processing and data augmentation is performed on a platform called Roboflow¹ which allows seamless conversion of data into the YOLOv7 format. This website can also be used with a free subscription plan. This project uses Visual Studio for code editing and modification. Visual Studio also has a free version and can be downloaded from its official website. CUDA GPU accelerators were also used in this project. To use the CUDA GPU accelerators a physical NVIDIA GPU is necessary. For this project cuda 11.7 version is used. The distribution for installing cudnn is available on the official Nvidia website for free.

¹ https://app.roboflow.com/thesispcbdefect/pcb_defect_detection-pegfm/5

```
(RP) D:\College_Material\Sem3\Research_Project\yolov7>nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
Built on Wed_Jun__8_16:59:34_Pacific_Daylight_Time_2022
Cuda compilation tools, release 11.7, V11.7.99
Build cuda_11.7.r11.7/compiler.31442593_0
```

Figure 1 CUDA version

3. Software Installation Guide

- a. Download & install Python v3.7.9 version from Anaconda.
- b. Download and install the v11.7 distribution of cudnn from Nvidia.
- c. Download and install the community edition of Visual Studio and install necessary dependencies provided in the 'requirements.txt' to run python code.

4. Project Implementation Guide

This section explains in detail on the implementation of the project. It provides a brief explanation of the necessary libraries installed, methods written and necessary parameters.

4.1 YOLOv7 code setup

The code² for YOLOv7 is available publicly on the authors GitHub repository. The code can be downloaded, and a local setup can be performed to run the code on the user's local machine. Figure 2 shows the folder structure of the source code.

² <https://github.com/WongKinYiu/yolov7>

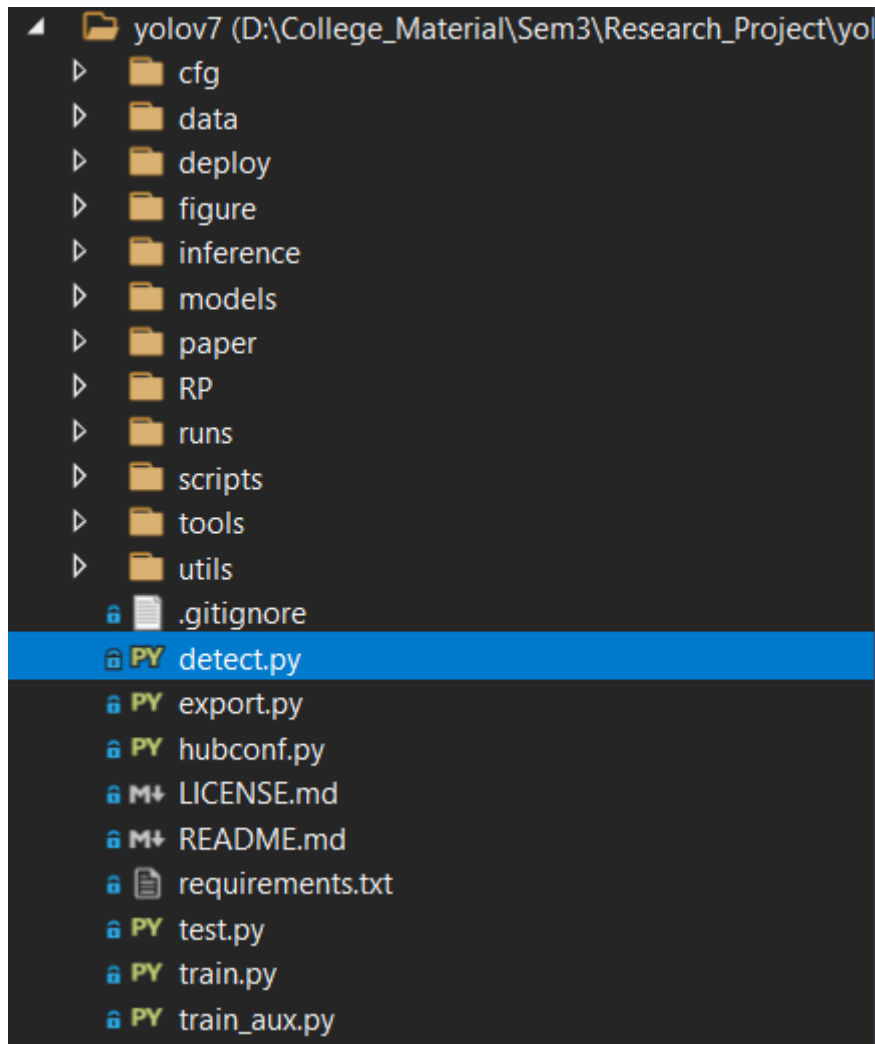


Figure 2 Code Folder Structure

Below are the steps for setting up the code on the reader's local machine.

- a. Install the necessary software mentioned in section 3.
- b. Clone the code into local machine or download a zipped version.
- c. **Recommended step:** Create a virtual environment to prevent any version mismatch errors for the required libraries. Figure 3 shows the command to create a virtual environment.

```
D:\College_Material\Sem3\Research_Project\yolov7>python -m venv RP
```

Figure 3 Virtual Environment Creation

- d. Activate the environment by navigating to the scripts folder and passing the command 'activate' in the command prompt as shown in Figure 4.

```
D:\College_Material\Sem3\Research_Project\yolov7>cd RP
D:\College_Material\Sem3\Research_Project\yolov7\RP>cd Scripts
D:\College_Material\Sem3\Research_Project\yolov7\RP\Scripts>activate
```

Figure 4 Virtual Environment Activation

- e. Once the virtual environment is activated, install the necessary libraries mentioned in the file 'requirements.txt' as shown in the Figure 5.

```
(RP) D:\College_Material\Sem3\Research_Project\yolov7>pip install -r requirements.txt
```

Figure 5 Installing necessary dependencies

These steps complete the code setup, and the code is ready to be run on a custom dataset.

4.2 Data Pre-processing & Augmentation

The data pre-processing and data augmentation are performed on a platform called Roboflow as shown in Figure 6. Roboflow is a platform which aids in running computer vision-based model building, code running and model evaluation. An account needs to be created before using the platform. Roboflow allows you to create our own workspace where we can work on our own custom dataset.

YOLOv7 accepts input for training only in a specific format, i.e., the dataset must contain a txt file with the dimensions of bounding boxes in the images. The dataset sourced had only a xml format of the annotations and needs to be converted into the specific YOLOv7 version.

Roboflow allows one to convert the data into any suitable format necessary provided the annotations are available. Below are the steps to upload a dataset into roboflow and perform data pre-processing and data augmentation.

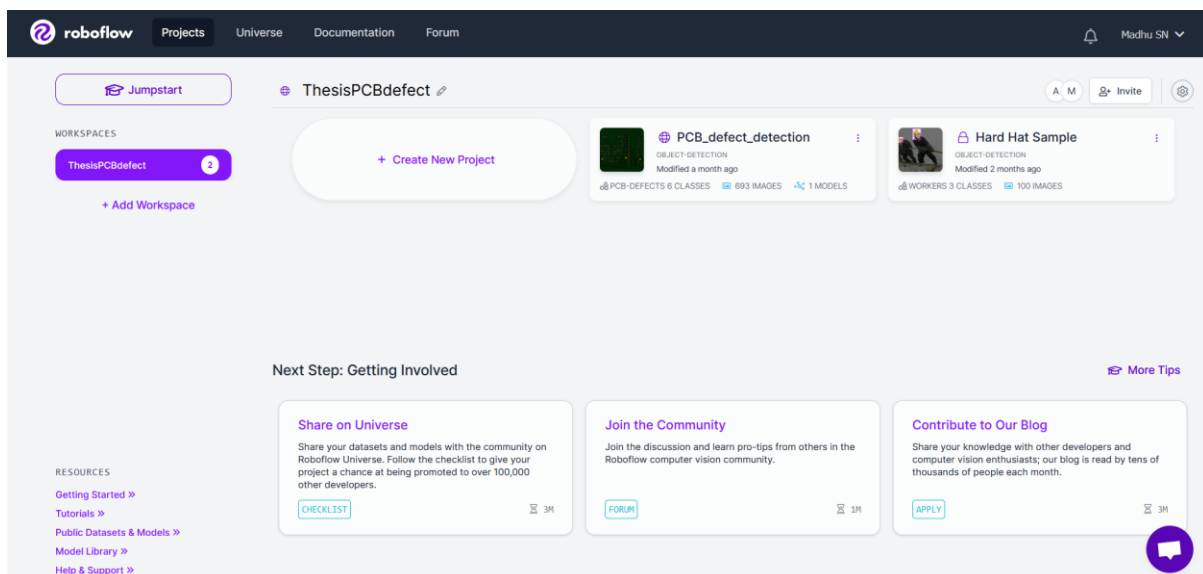


Figure 6 Roboflow Platform

- a. Create a project and provide the following details as shown in Figure 7.

Create Project

×

ThesisPCBdefect / [New Public Project](#)

Project Type What is This?

Object Detection (Bounding Box)
▾

What Are You Detecting? ?

E.g., `dogs` or `cars` or `words`

Project Name

E.g., `Dog Breeds` or `Car Models` or `Text Finder`.

License

CC BY 4.0
▾

Cancel

Create Public Project

Figure 7 Creating Workspace on Roboflow

- b. Navigate to upload tab and upload the custom dataset to be trained or augmented as shown in Figure 8.

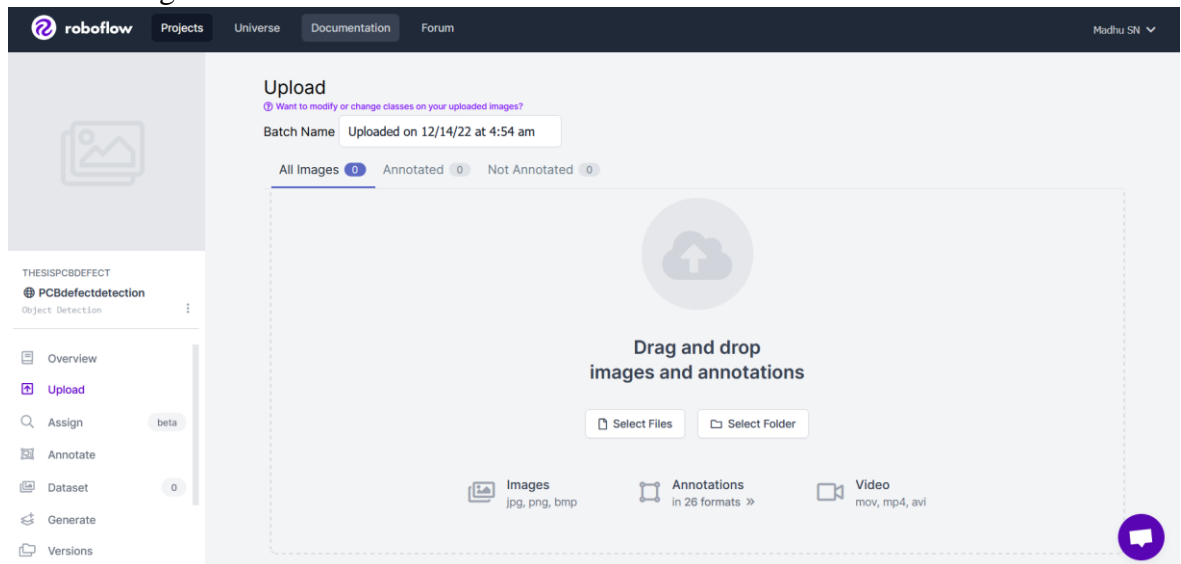


Figure 8 Uploading dataset

- c. The Images are required to be uploaded and folder wise and the corresponding annotations are to be input as well. Once the data is annotated, the user would be able to perform pre-processing and augmentation steps on the annotated dataset.

- d. For the purpose of this research, the images are augmented in three different ratios, namely, 1:1, 1:2 and 1:3. The results of these augmentations and the operations performed are as shown in the Figure 9.

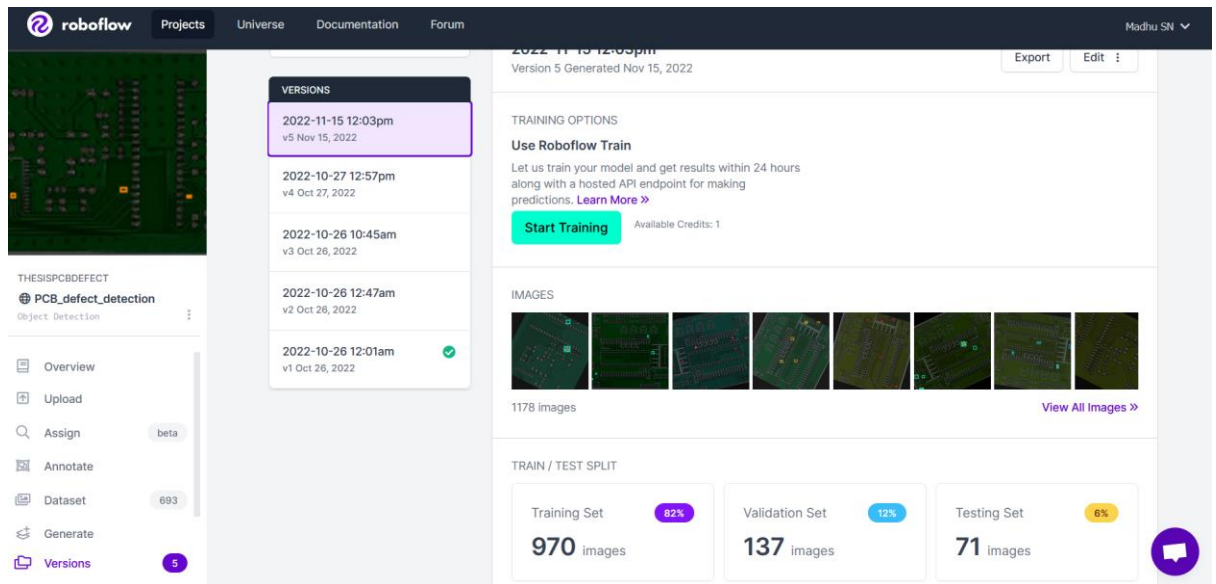
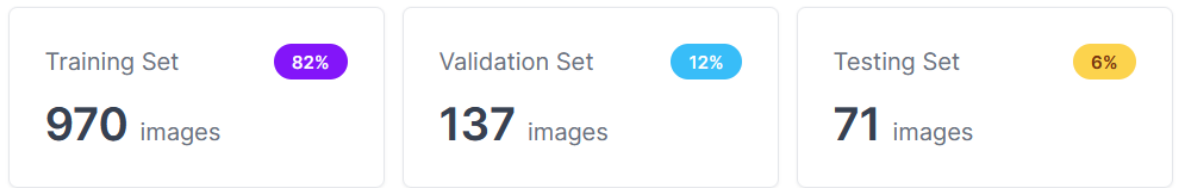


Figure 9 Performing data augmentation and Data pre-processing

- e. The number of images after each augmentation and the operations performed will be displayed as shown in the Figure 10.



PREPROCESSING No preprocessing steps were applied.

AUGMENTATIONS

- Outputs per training example:** 2
- Flip:** Horizontal, Vertical
- 90° Rotate:** Clockwise, Counter-Clockwise, Upside Down
- Crop:** 0% Minimum Zoom, 20% Maximum Zoom
- Rotation:** Between -30° and +30°
- Hue:** Between -91° and +91°
- Brightness:** Between 0% and +30%
- Bounding Box: Brightness:** Between -35% and +35%

DETAILS

- Version Name:** 2022-11-15 12:03pm
- Version ID:** 5
- Generated:** Nov 15, 2022
- Annotation Group:** PCB-Defects

Figure 10 Data Augmentation operations performed

- f. Once the dataset is augmented, the dataset can be exported in the required format, i.e., YOLOv7 PyTorch format in our case as shown in Figure 11.

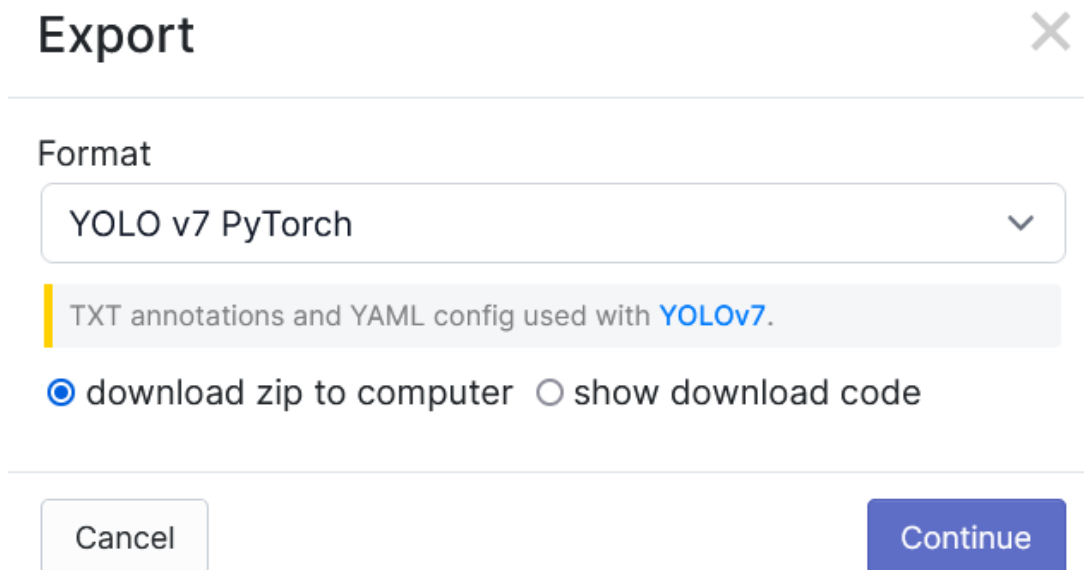


Figure 11 Exporting Data into YOLOv7 format

4.3 Model Training

Once the data is exported into the necessary format, the dataset can be used to train the model and record the outputs. There are certain steps that must be carried out before we train the model. The steps are as follows

- a. The model used to train the dataset must be chosen initially. The 'cfg' folder has a number of models available for training as shown in Figure 12 and any among them could be used for the purpose of training. In this research, YOLOv7 is the chosen model.

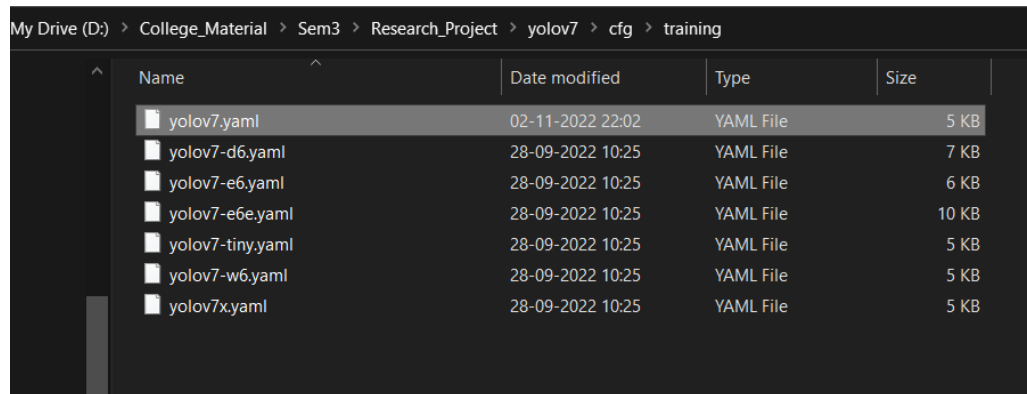


Figure 12 YOLOv7 architecture

- b. In the yaml file of yolov7, the variable 'nc' has to be changed to 6 from the default value of 80 as we are only predicting 6 classes. 'nc' stands for the number of classes to be predicted and is used to define the final output layer in the head.

```
# parameters
nc: 6 # number of classes
depth_multiple: 1.0 # model depth multiple
width_multiple: 1.0 # layer channel multiple
```

Figure 13 Configuring YOLOv7 based on number of classes to be predicted

- c. Once the number of classes are set as per requirement the config yaml file must be configured to read the data from the right folder as shown in Figure 14. In this research the file is named 'data_WA.yaml' and the file has to be placed in the data folder. This config file consists of path location to the training data and validation datasets, the number of classes to be predicted and the names of the classes.

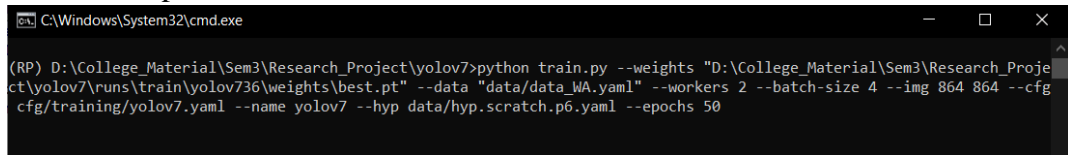
```
train: D:/College_Material/Sem3/Research_Project/Data/PCB_dataset_annotated/train/images
val: D:/College_Material/Sem3/Research_Project/Data/PCB_dataset_annotated/valid/images
test: D:/College_Material/Sem3/Research_Project/Data/PCB_dataset_annotated/test/images

nc: 6
names: ['missing_hole', 'mouse_bite', 'open_circuit', 'short', 'spur', 'spurious_copper']
```

Figure 14 Configuring train data, test data and validation data path

- d. Once the data paths are provided, the dataset provided can be trained using the 'train.py' file by using the following command as shown in Figure 15. The parameters provided with the command are the initial weights to be used before training the model, the path location of data the code should point to, number of

workers for parallel processing, batch size for the input data, the image resolution and size, the model to be used, the hyper parameters to be used and finally the number of epochs.



```
C:\Windows\System32\cmd.exe
(RP) D:\College_Material\Sem3\Research_Project\yolov7>python train.py --weights "D:\College_Material\Sem3\Research_Proje
ct\yolov7\runs\train\yolov736\weights\best.pt" --data "data\data_WA.yaml" --workers 2 --batch-size 4 --img 864 864 --cfg
cfg/training/yolov7.yaml --name yolov7 --hyp data/hyp.scratch.p6.yaml --epochs 50
```

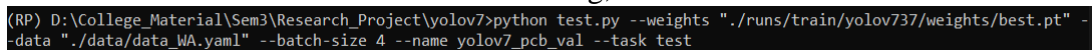
Figure 15 Command for training Custom data on YOLOv7

- e. The results of the training is logged into the ‘runs’ folder under ‘train’.

4.4 Model Testing

The model trained must be validated using an isolated validation set of data to check the integrity of the model and to ensure that the model has been trained properly. The steps to do so are mentioned as follows

- a. The test data path has to configured in the appropriate yaml file present in the data folder.
- b. Figure 16 shows the command to execute the test code. The command includes the location of the model to be used for testing, the location of the test dataset.



```
(RP) D:\College_Material\Sem3\Research_Project\yolov7>python test.py --weights "./runs/train/yolov737/weights/best.pt" --
-data "./data/data_WA.yaml" --batch-size 4 --name yolov7_pcb_val --task test
```

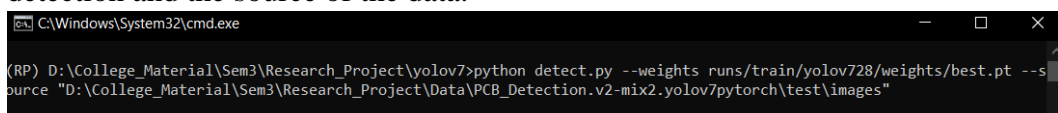
Figure 16 Command for testing on validation set on the trained model

- c. The results of the testing is logged into the ‘runs’ folder under ‘test’.

4.5 Performing defect detection

This section elaborates on the steps to be taken for performing detection. The steps to be followed to perform detection are as follows

- a. The test data path location must be updated in the data file as mentioned in the previous section for training data and must be placed in the appropriate folder.
- b. Figure 17 shows the command to be executed to perform detection on test data. The command includes the location of the weights that are to be used to perform detection and the source of the data.



```
C:\Windows\System32\cmd.exe
(RP) D:\College_Material\Sem3\Research_Project\yolov7>python detect.py --weights runs/train/yolov728/weights/best.pt --s
ource "D:\College_Material\Sem3\Research_Project\Data\PCB_Detection.v2-mix2.yolov7pytorch\test\images"
```

Figure 17 Command for performing detection using YOLOv7

- c. The results of the detecting is logged into the ‘runs’ folder under ‘detect’.