MechDefect Solutions

ME781 Course Project - Group 15

The code for the entire project has been made available on Github

Table of contents

01

Project title and objective

Advanced AI-Driven Defect Detection Solutions for Manufacturing Excellence

Develop AI-powered defect detection and classification solutions for industries that cater to the initial stages of manufacturing

Leverage transfer learning by training our model on different processes (like casting, mining, etc) gaining knowledge from one and applying it to other processes

02

Model details

Models

We have leveraged three different models for the following tasks :

(a) <u>Object Detection Model</u>

For detecting and segmenting out relevant components (by creating bounding boxes) from an image (or video), we chose to fine-tune the **RetinaNet** (with **ResNet50** Backbone) architecture on our our custom dataset. This is a transfer learning based approach where we instantiate and fine-tune the RetinaNet for catering to the requirements of every different industry that we plan to collaborate with based on the component images that we receive from the industries. For the sake of this project however, we intend to fine-tune the model for detecting pump impellers in an image.

(b) <u>Defect Detection Model</u>

For the core task of defect detection, we select the VGG16 architecture (since it was the better performing model compared to ResNet50 for our tasks) as our base model for the transfer learning process.

(c) <u>Defect Classification Model</u>

For the core task of defect classification too, we select the VGG16 architecture (since it was the better performing model compared to ResNet50 for our tasks) as our base model for the transfer learning process. Based on the industry's requirements, the number of output classes of our final model would vary. For the demo, we have 6 types of defects.

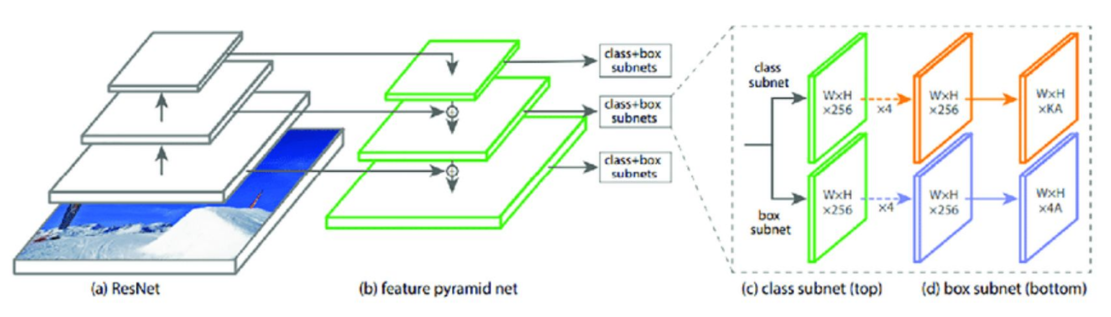
Object Detection Model

Dataset Preparation

An augmented subset consisting of 26 images belonging to Submersible Pump Impeller Defect Dataset proved to be sufficient for object detection owing to the fact that only 1 class is involved. All these images were annotated using an online annotation tool.
This dataset was then split into training & testing sets. For training, two CSV files were needed. The first one contained the path, bounding box and the class name for each image (train.csv). The second file only contained the class name and their corresponding mapping (class.csv).
The CSV file with annotations contains one annotation per line. Note that indexing for pixel values starts at 0.

Model Architecture :

The architecture of the original RetinaNet model (with ResNet50 Backbone) is shown. We leveraged the dataset prepared to fine-tune this model.



Defect Detection Model

Dataset Preparation :

We used the Submersible Pump Impeller Defect Dataset to demonstrate the working of this model. In order to achieve robustness, instead of using the images of pump impellers as they are, we preprocessed them to introduce some rescaling, shearing, rotation & zooming (which could happen during actual usage due to the positioning of the camera positioning, etc).
The dataset contains 6633 training images & 715 test images.

Model Architecture :

We enabled Transfer Learning by freezing the weights of the base VGG16 Model and followed it by adding the following layers to the top of the base model for the detection task with 2 classes (Defective & OK).

Out of the 40,668,738 total parameters, we had 25,954,050 trainable parameters.

Layer Type	Output Shape	Number of Parameter
VGG (base model)	(BATCH_SIZE, 7, 7, 512)	14714688
Flatten	(BATCH_SIZE, 25088)	0
Fully Connected with Dropout	(BATCH_SIZE, 1024)	25691136
Fully Connected with Dropout	(BATCH_SIZE, 256)	262400
Fully Connected with Dropout	(BATCH_SIZE, 2)	514

Defect Classification Model

Dataset Preparation :

We used the Metal Surface Defect Dataset to demonstrate the working of this model. In order to achieve robustness, instead of using the images of metal surfaces as they are, we preprocessed them to introduce some rescaling, shearing, rotation & zooming (which could happen during actual usage due to the positioning of the camera positioning, etc).
The dataset contains 1656 training images, 72 validation images & 72 test images (equal number of images per class).

Model Architecture :

We enabled Transfer Learning by freezing the weights of the base VGG16 Model and followed it by adding the following layers to the top of the base model for the classification task with 6 classes (Crazing, Inclusion, Patches, Pitted, Rolled & Scratches):

Layer Type	Output Shape	Number of Parameter
VGG (base model)	(BATCH_SIZE, 7, 7, 512)	14714688
Flatten	(BATCH_SIZE, 25088)	0
Fully Connected with Dropout	(BATCH_SIZE, 1024)	25691136
Fully Connected with Dropout	(BATCH_SIZE, 256)	262400
Fully Connected with Dropout	(BATCH_SIZE, 6)	1542

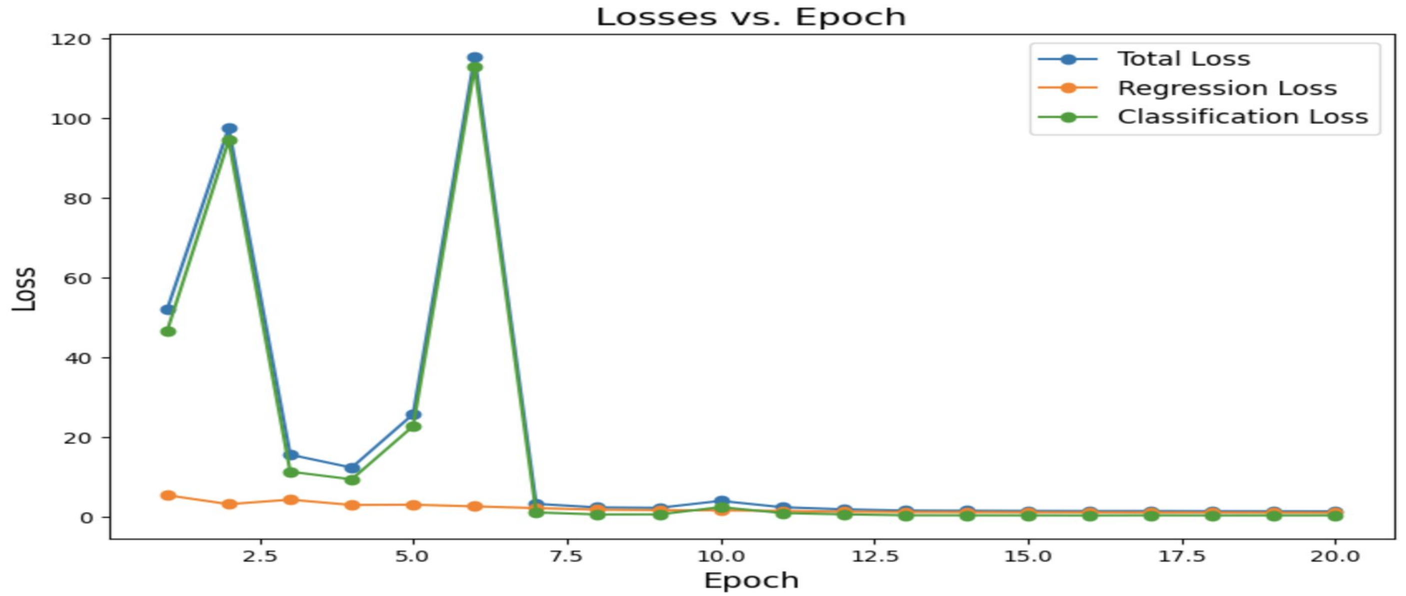
Out of the 40,669,766 total parameters, we have 25,955,078 trainable parameters.

03

Model training

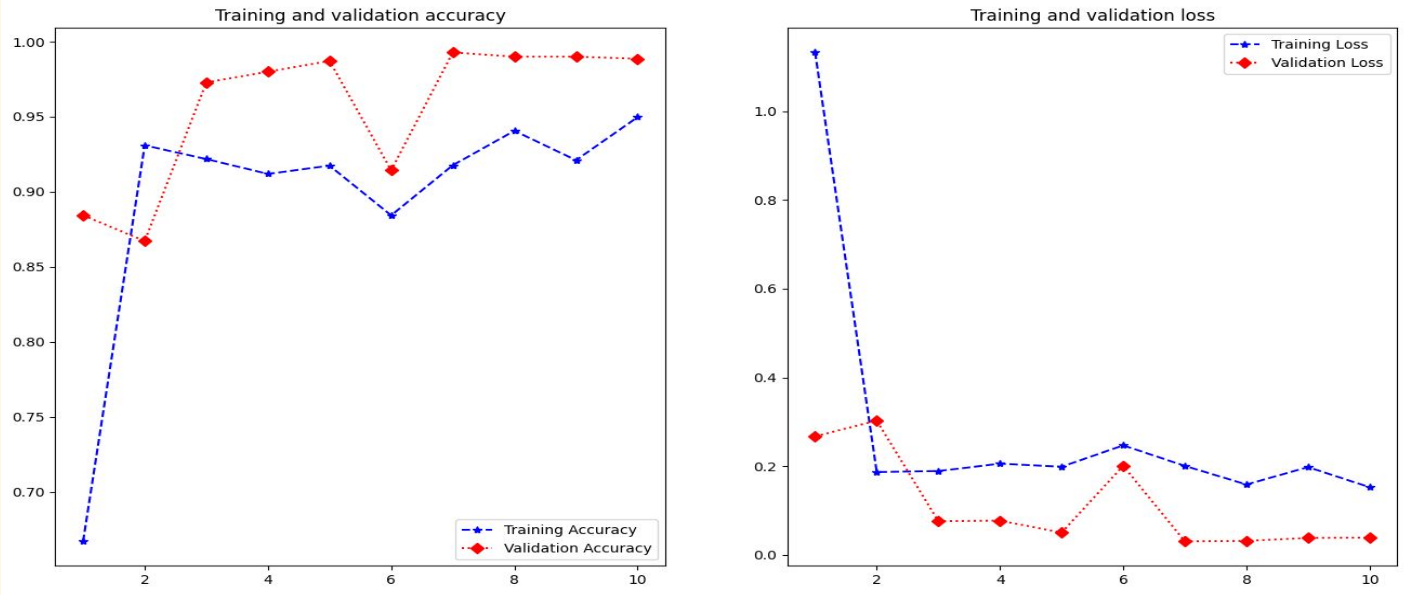
Object Detection Model

Our model is instantiated & trained for 20 epochs with the images from our augmented dataset. The plot of the training loss with each epoch of training is presented below:



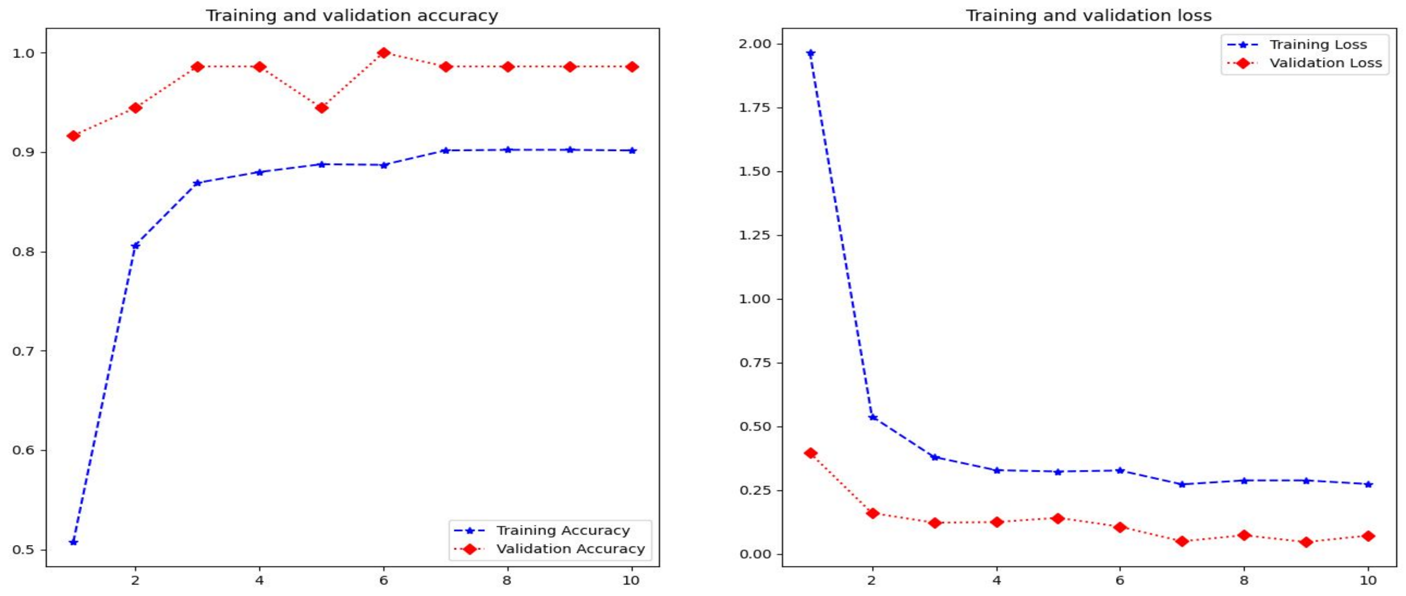
Defect Detection Model

We used a batch size of 36 along with an Adam Optimizer (with an initial learning rate of 0.001) to train the model for 10 epochs. We achieved a validation accuracy of 98.61 %. The plot for the performance of the model over training period is presented below:



Defect Classification Model

We used a batch size of 50 along with an Adam Optimizer (with an initial learning rate of 0.001) to train the model for 10 epochs. We achieved a validation accuracy of 98.86 %. The plot for the performance of the model over training period is presented below:

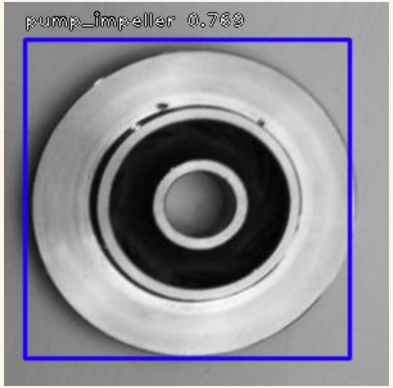


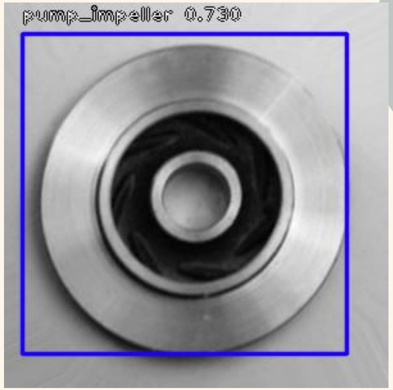
04

Testing results

Object Detection Model

For detecting pump impellers in an image, we tune the detection confidence threshold to 0.7 (this value gave us the best results on our dataset) for creating bounding boxes around the detected impellers in an image. Following are some of the results of our detector:





Defect Detection Model

The **confusion matrix** for this model on the test images is shown below :

		Actual Label	
		OK	Defective
Predicted Label	OK	445	8
	Defective	0	262

The **classification report** for the model obtained is shown here:

	Precision	Recall	F1 score	Support
Defective	1.00	0.98	0.99	453
OK	0.97	1.00	0.98	262
Accuracy			0.99	715
Macro Average	0.99	0.99	0.99	715
Weighted average	0.99	0.99	0.99	715

Defect Classification Model

The **confusion matrix** for this model on the test images is shown below :

		Actual Label					
		Crazing	Inclusion	Patches	Pitted	Rolled	Scratches
Predicted Label	Crazing	12	0	0	0	0	0
	Inclusion	0	12	0	0	0	0
	Patches	0	0	11	1	0	0
	Pitted	0	0	0	12	0	0
	Rolled	0	0	0	0	12	0
	Scratches	0	0	0	0	0	12

Defect Classification Model

The **classification report** for the model obtained is shown here:

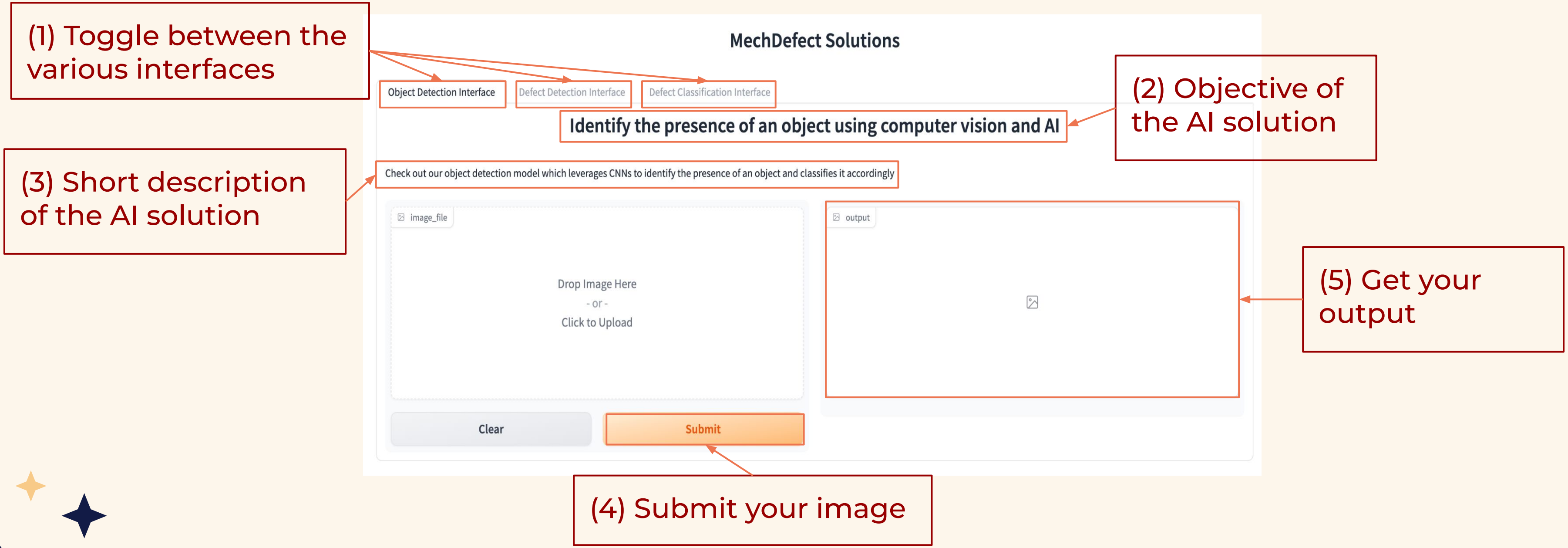
	Precision	Recall	F1 Score	Support
Crazing	1.00	1.00	1.00	12
Inclusion	1.00	1.00	1.00	12
Patches	1.00	0.92	0.96	12
Pitted	0.92	1.00	0.96	12
Rolled	1.00	1.00	1.00	12
Scratches	1.00	1.00	1.00	12
Accuracy			0.99	72
Macro Average	0.99	0.99	0.99	72
Weighted Average	0.99	0.99	0.99	72

05

Usage Guide

User Manual

In order to provide seamless deployment of our AI solutions, we have wrapped it around with Gradio and have hosted the same on the following Hugging Face space : <u>MechDefect Solutions Web Interface</u>

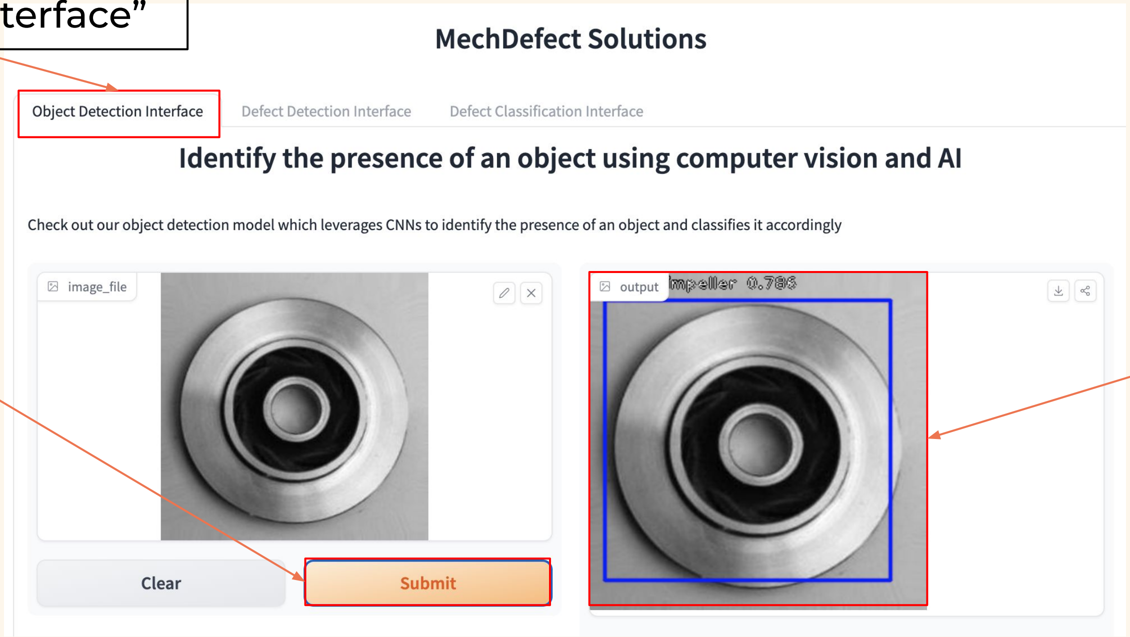


06

User Interface

Object Detection Interface

(1) Select the “Object Detection Interface”

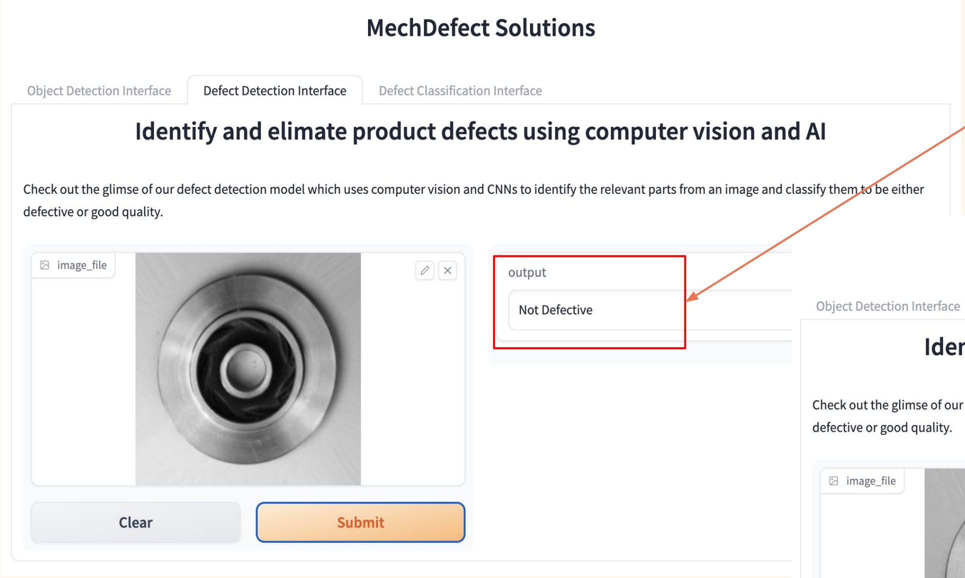


(2) Select and upload the image in which you want to detect and localise the 'pump impeller

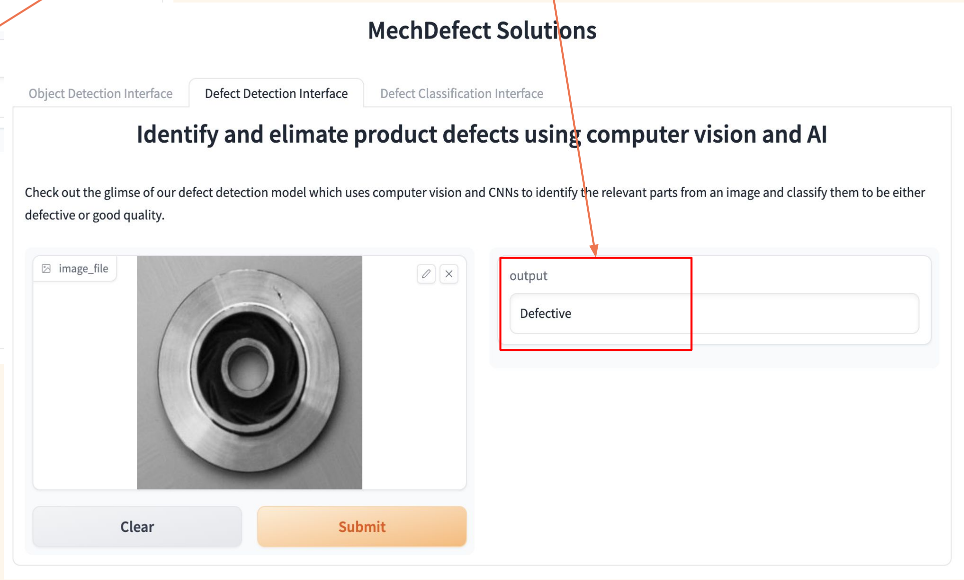
(3) Obtain the output image containing the bounding box along with the model’s confidence score

[MechDefect Solutions Web Interface]()

Defect Detection Interface



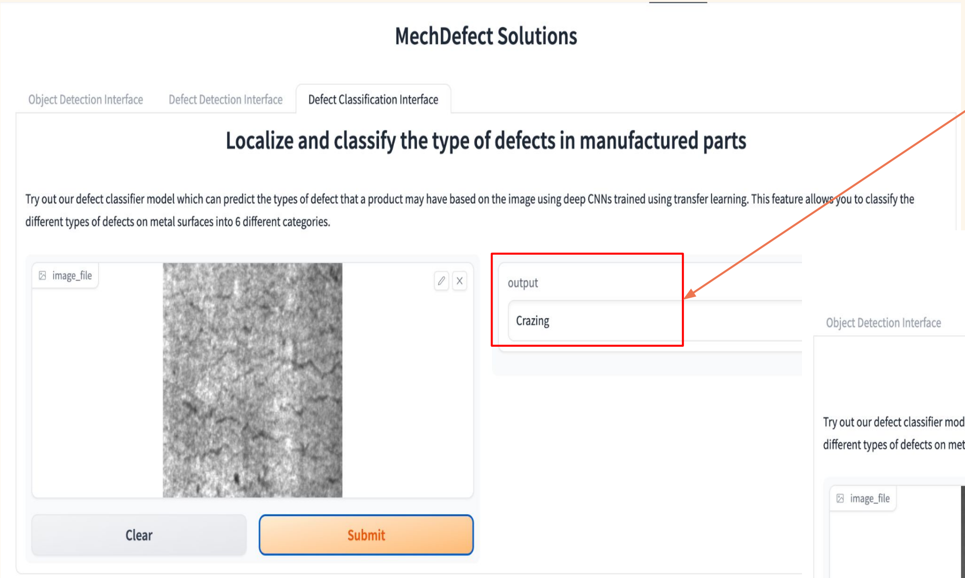
Be able to identify and eliminate product defects within seconds using our interactive UI



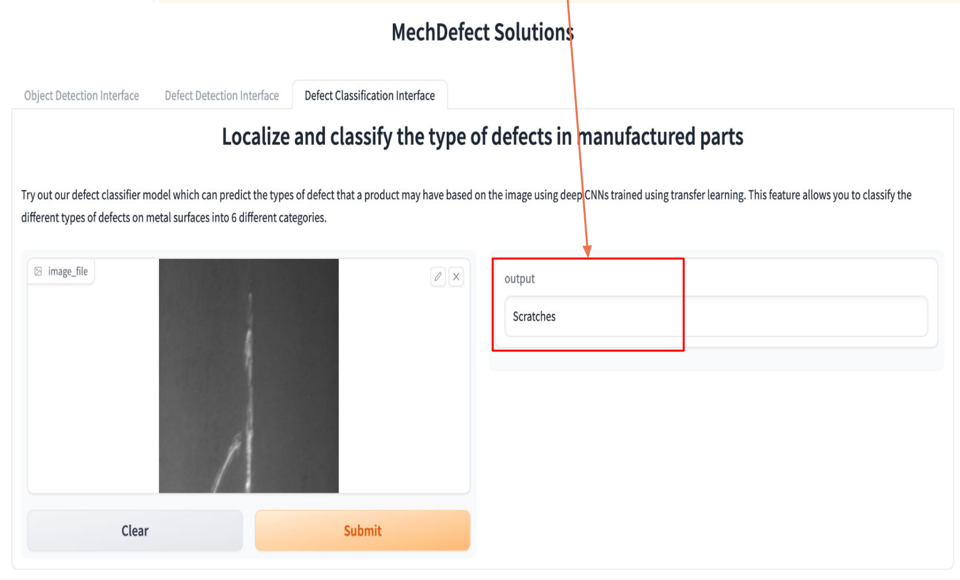
Average processing time per image = 2.9 seconds

MechDefect Solutions Web Interface

Defect Classification Interface



Be able to localize and classify defects in manufactured parts within seconds using our interactive UI



Average processing time per image = 3.2 seconds

MechDefect Solutions Web Interface

Our team

Varad Patil

22B2270

Swapnoneel Kayal

200100154

Kanika Banjare

210260027

Anand Bhaskar

200100024

Shraman Santara

21D100019