Source Code:

Training:

The training was done in Google Colab with following scripts:

```
!git clone https://github.com/WongKinYiu/yolov7.git

%cd yolov7

!python train.py --workers 8 --device 0 --batch-size 2 --epochs 100 --data
data/custom.yaml --img 640 640 --cfg cfg/training/yolov7-custom.yaml --
weights 'yolov7.pt' --name yolov7-custom --hyp data/hyp.scratch.p5.yaml
```

The **train.py** file can be found in the official YOLOv7 repo:
https://github.com/WongKinYiu/yolov7
https://github.com/WongKinYiu/yolov7/blob/main/train.py

data/custom.yaml

```
# train and val data as 1) directory: path/images/, 2) file:
path/images.txt, or 3) list: [path1/images/, path2/images/]
train: ./analog_meter/images/train/
val: ./analog_meter/images/val/
test: ./analog_meter/images/test/

# number of classes
nc: 1

# class names
names: ["meter"]
```

The configuration file **cfg/training/yolov7-custom.yaml** has only one change to the original configuration file found in YOLOv7 repo. All of the networks and configurations are kept same to the original except for the number of classes. The fine-tuned model has only one class. For that, the text *nc:80* has been changed to *nc:1*.

Inference:

This inference code uses components from official YOLOv7 repository. The components are modified to make the prediction with only the inference file and without any dependencies from other program files.
Due to inference programs becoming too large to be accommodated in this programming report, additional files will be provided with the programs developed along with test images and evaluation result files.

The program source files added are detection.py and test_results.py, the first one for the inference and the later one for the evaluation.

Command to run the program file:
python detection.py -i <image_folder>/<image_name> -v0 <value_at_0_degree> -v1 <value_at_180_degree> -f <boolean_value_to_write_result_in_file>

For example:
python detection.py -i meter_images/analog_meter_0.png -v0 3.25 -v1 0.6 -f True

Command to run the evaluation script:
python test_results.py

The error report will be saved in test_results.txt

The test_results.txt file generated after running the program for a set of 30 images along with the label.txt file will be added along with this report. Additionally, sample inputs and output files will also be provided.

Program Testing and Run Examples:

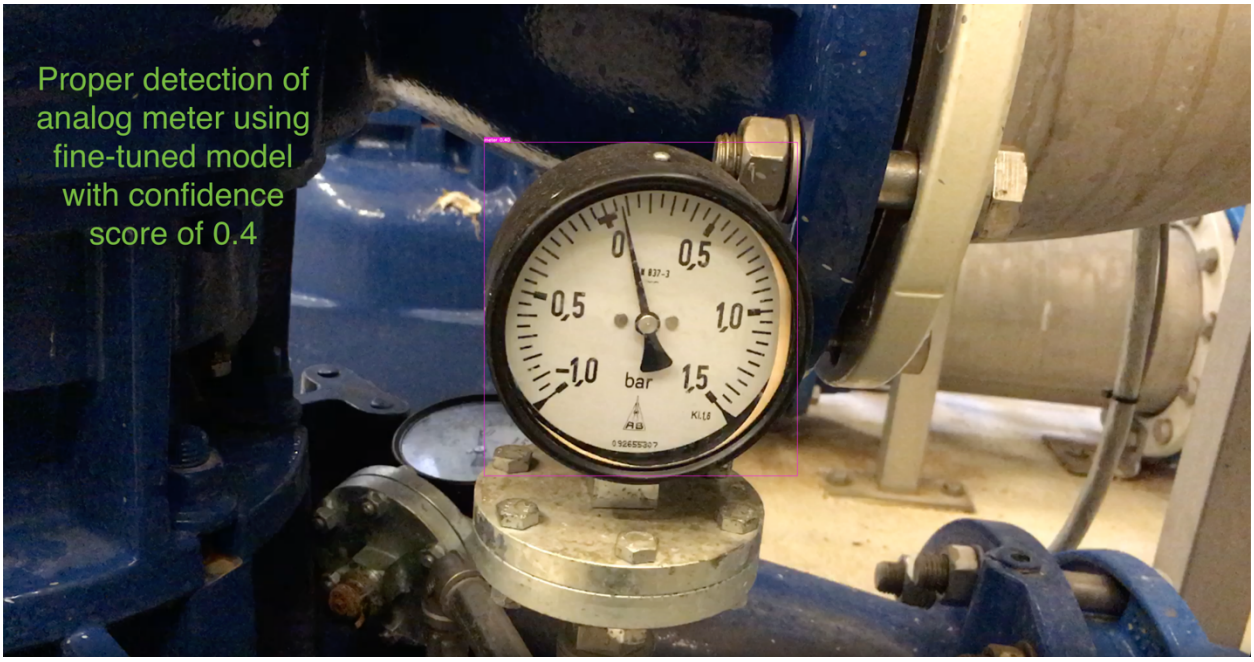

Fig: Proper detection of analog meter with confidence of 0.4

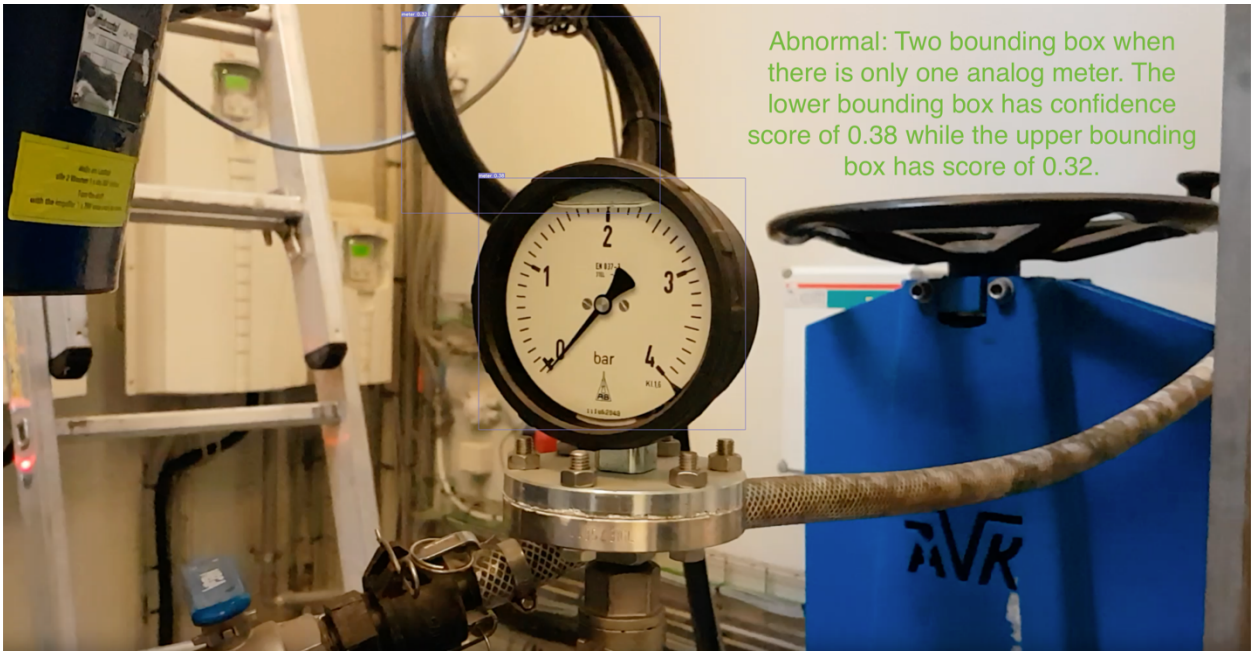

Fig: Abnormal detection of two bounding boxes

Fig: Abnormal case of wall clock detection as meter

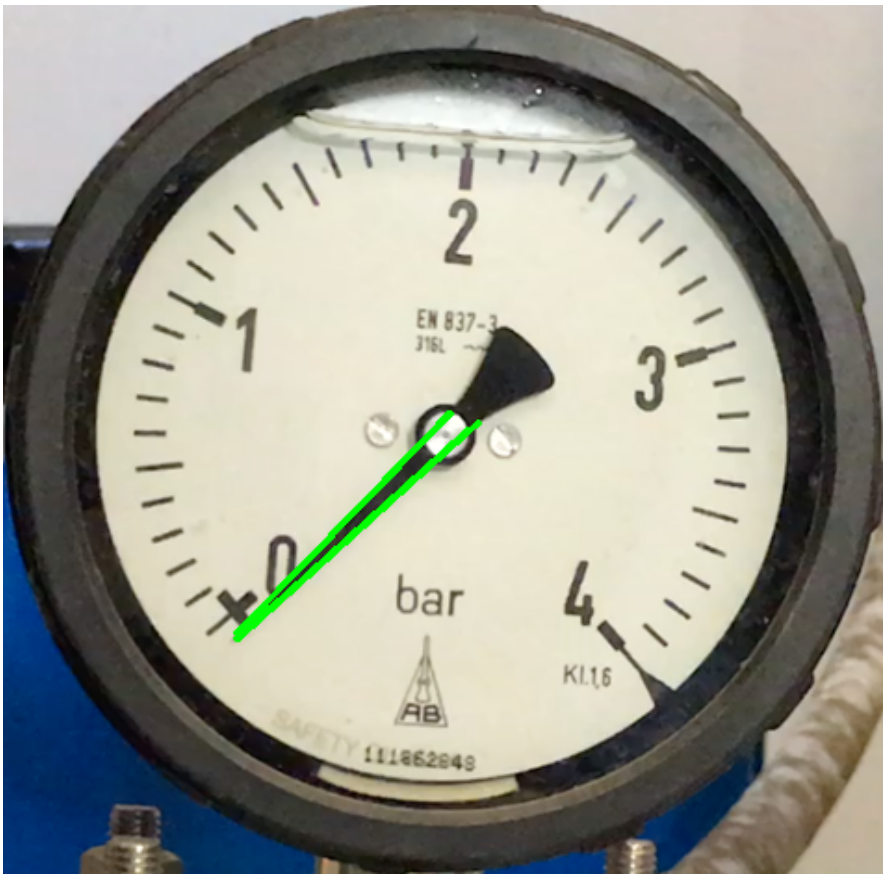

Fig: Proper identification of the analog meter pointer needle

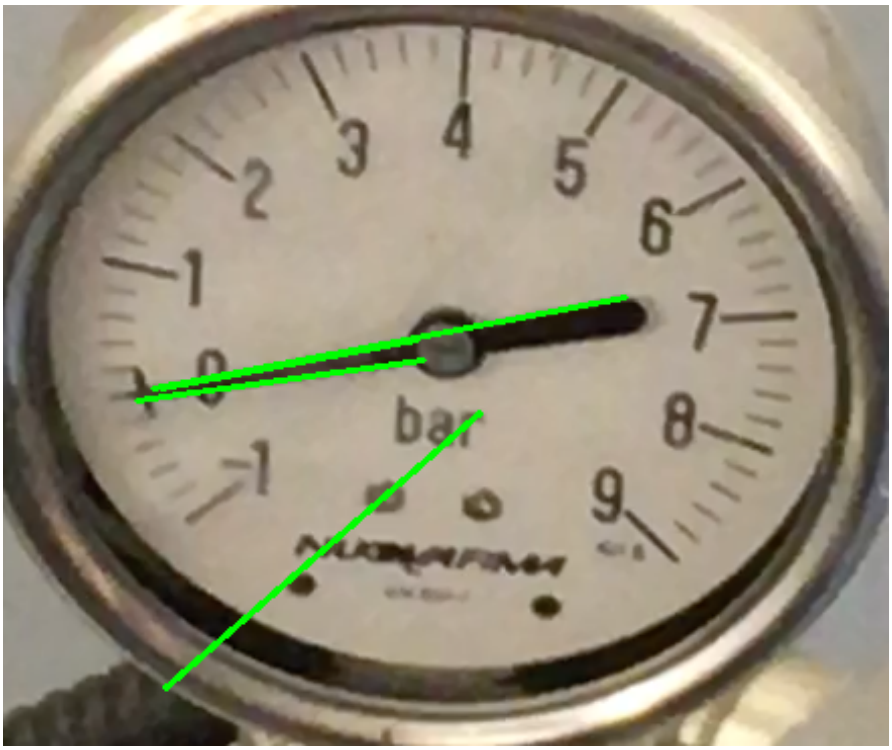

Fig: Improper identification of the analog meter pointer needle

Fig: Partial line identification for pointer needles (with slope being preserved, hence no affect in results)

Fig: Final reading after identification of angle for meter 1 (compared with true value)

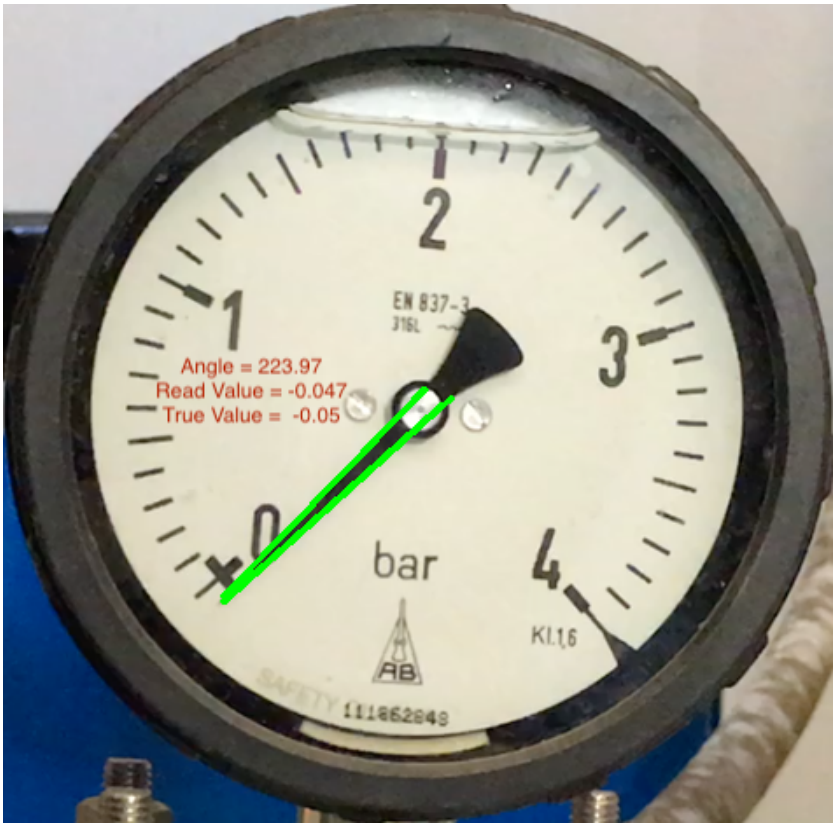

Fig: Final reading after identification of angle for meter 2(compared with true value)

Evaluation Results:

image_name, true_reading, read_value, read_angle, deviation
analog_meter_0,-0.05,-0.04730729493180963, 223.96804267461349, 0.0026927050681903736
analog_meter_1, 0.4, 0.3872280209833674, 207.92378638556806, 0.012771979016632629
analog_meter_2, 0.7, 0.7307369296824855, 104.61753802116762, 0.030736929682485536
analog_meter_3, 0.075, 0.0784888246413269, 103.98377482003544, 0.003488824641326896
analog_meter_4, 0.125, 0.12249763103587796, 90.25738652202399, 0.002502368964122037
analog_meter_5, 3.15, 3.148024646892631,-53.201643126175384, 0.00197535310736896
analog_meter_6, 0.0,-0.0459382707606899, 229.31722218423104, 0.0459382707606899
analog_meter_7, 1.0, 0.9692503251889768, 155.38331165406822, 0.030749674811023198
analog_meter_8,-0.05,-0.07436057605759805, 224.95737173717322, 0.024360576057598046
analog_meter_9, 1.0, 1.0743960999699906, 151.17309509637798, 0.07439609996999064
analog_meter_10, 0.0, 0.006678586081897642, 193.05850801547916, 0.006678586081897642
analog_meter_11, 1.55, 1.5746811483327987, 45.02125677781342, 0.024681148332798664
analog_meter_12, 1.55, 1.552054134516192, 50.80387010833412, 0.0020541345161919633
analog_meter_13, 1.55, 1.5506644675968044, 49.955702160213036, 0.0006644675968043856
analog_meter_14, 0.0,-0.08602196994324895, 200.8064639102625, 0.08602196994324895
analog_meter_15,-0.4,-0.3641732919920404, 242.16485887558665, 0.03582670800795962
analog_meter_16, 0.0, 0.2705513307537153, 188.91740758862122, 0.2705513307537153
analog_meter_17,-1.1,-1.1151052429683186, 221.00701619788788, 0.0151052429683185
analog_meter_18,-1.1,-1.0800756779205232, 226.1938196323374, 0.01992432207947692
analog_meter_19, 0.0, 0.03643601697986654, 195.1405249169588, 0.03643601697986654
analog_meter_20,-0.6,-0.6062613164322288, 197.08408776214858, 0.006261316432228825
analog_meter_21, 0.0, 0.0057623987706270086, 195.72981885607163, 0.0057623987706270086
analog_meter_22, 0.0,-0.0025074446274162554, 190.57683796101242, 0.0025074446274162554
analog_meter_23, 0.0, 0.05923657894799028, 189.02020820431792, 0.05923657894799028
analog_meter_24,-0.45,-0.47236423173103503, 188.0092951678496, 0.022364231731035022
analog_meter_25,-0.5,-0.41544772726446233, 187.55657851185575, 0.08455227273553767
analog_meter_26,-0.1, 0.08050517723654638, 215.28644079147986, 0.18050517723654638
analog_meter_27,-0.1,-0.0483192286685723, 224.03677779635586, 0.051680771331427705
analog_meter_28,-0.1,-0.110040912900915, 231.62542049893005, 0.010040912900914994
analog_meter_29, 0.7, 0.76946783272354, 103.96067551311802, 0.06946783272354007

Root Mean Squared Error: 0.06981656859349669

Table: Comma separated values after running the evaluation script test_results.py

Discussion Page:

Hardware used by the program:

- Laptop: MacBook Air 2020
- CPU: 1.1 GHz Quad-Core Intel Core i5
- GPU: Intel Iris Plus Graphics 1536 MB
- RAM: 16 GB 3733 MHz LPDDR4X
- Google Colaboratory: T4 GPU

Software:

- OS: macOS Ventura Version 13.6
- Programming Language: Python 3.9.10
- Packages: NumPy, OpenCV, PyTorch

Features:

- Data Structures: List, Dictionary, NumPy 2D Array
- Algorithms: Object detection (YOLOv7), Hough Transformation, Canny Edge Detection, BGR to Gray Image conversion, Gaussian Smoothing
- Programming Style: Object Oriented Programming, Functional programming paradigm

Problems encountered:

- Nothing as such.