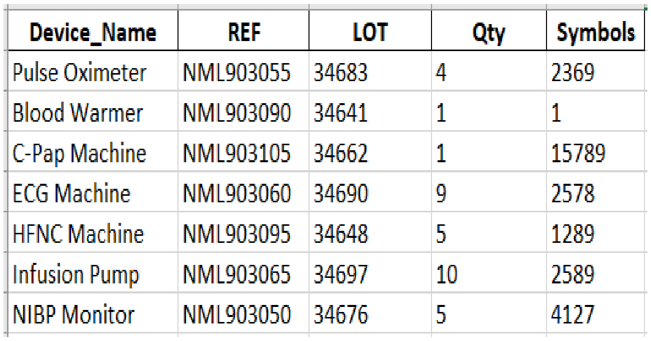
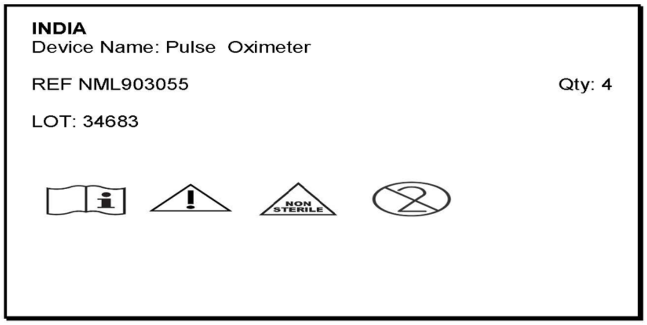
Project Report

The goal of the project is to extract the data from the images, including any symbols that may be present on the images. This data should be extracted and organized in an Excel spreadsheet.

Sample output:



Device_Name	REF	LOT	Qty	Symbols
Pulse Oximeter	NML903055	34683	4	2369
Blood Warmer	NML903090	34641	1	1
C-Pap Machine	NML903105	34662	1	15789
ECG Machine	NML903060	34690	9	2578
HFNC Machine	NML903095	34648	5	1289
Infusion Pump	NML903065	34697	10	2589
NIBP Monitor	NML903050	34676	5	4127



INDIA
Device Name: Pulse Oximeter

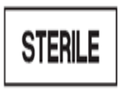
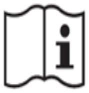
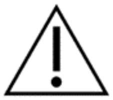
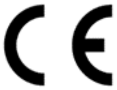
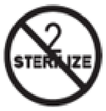
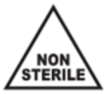
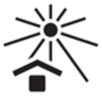
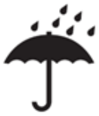
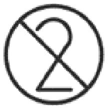
REF NML903055 Qty: 4

LOT: 34683

NON STERILE

Excel file generated by the Model	Sample input image

Index of the symbols:

								
1	2	3	4	5	6	7	8	9

Description:

"Each image contains certain information at specific locations, including the device name, reference, lot, and quantity. If symbols are present on the image, the order in which they occur should be taken into account, and their name and ID should be concatenated in the appropriate order."

The project involves extracting data from images and organizing it into an Excel spreadsheet in order to automate the manual task of data entry. This can improve efficiency and accuracy in the data collection process by **eliminating the need for manual data entry, which can be time-consuming**. The data extracted from the images includes device name, reference, lot, quantity, and any symbols that may be present, with the name and ID of the symbols being concatenated in the appropriate order.

https://colab.research.google.com/drive/1L9e73IY65zR4ySQTDIpm2nKAQxZnu437?usp=sharing

https://drive.google.com/drive/folders/1YyJ5zdsiS6eifEjtj5LIxo44_cjZk7jr

Problems to tackle:

- Extraction of text from images, where noise will be added while using **tesseract-OCR.** Pre-processing is required to extract the details.
- Proper Symbol detection and issue is their order in the image.

The project is divided into five sections.

Libraries	This section imported all the necessary libraries.
Text Extraction	Here, the details were extracted and processed.
YOLO v5 model	In this section, the model was trained and tested using custom data.
Symbol Extraction	This section focused on the task of ordering the symbols..
Final	The final results obtained were consolidated into a single Excel file.

Text Extraction:

Tesseract-OCR was the library used to export the text, the problem is that the output text provided has lots of noise in it. Here is an example of the output

'INDIA\nDevice Name: Pulse Oximeter\n\nREF NML903055\n\nLOT: 34683\n\nCl Ay Zo @\n\n \n\x0c'

The following function was written to handle text preprocessing and remove extraneous noise such as spaces, while extracting the relevant details based on specified features.A array named as Features were used as a template.

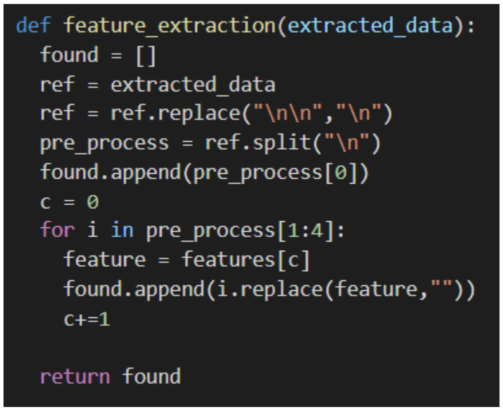
Features = [“Device Name: “, “REF”, “LOT:”]
A function was created to preprocess the noisy output text by performing the following steps

- Replacing all double spaces with single spaces.
- Splitting the text into lines using the parameter \n as the separator.
- Extracting relevant details from the lines using a list of specified features

Result: ['INDIA', 'Pulse Oximeter', 'NML903055', '34683']

To extract the quantity I have used the method findall in the cropped image. cmd: temp = re.findall(r'\d+', cropped[i])

Final results after appending were stored in the values_found list.
['INDIA', 'Pulse Oximeter', 'NML903055', '34683', '4']



```
def feature_extraction(extracted_data):
  found = []
  ref = extracted_data
  ref = ref.replace("\n\n","\n")
  pre_process = ref.split("\n")
  found.append(pre_process[0])
  c = 0
  for i in pre_process[1:4]:
    feature = features[c]
    found.append(i.replace(feature,""))
    c+=1

  return found
```

YOLO V5

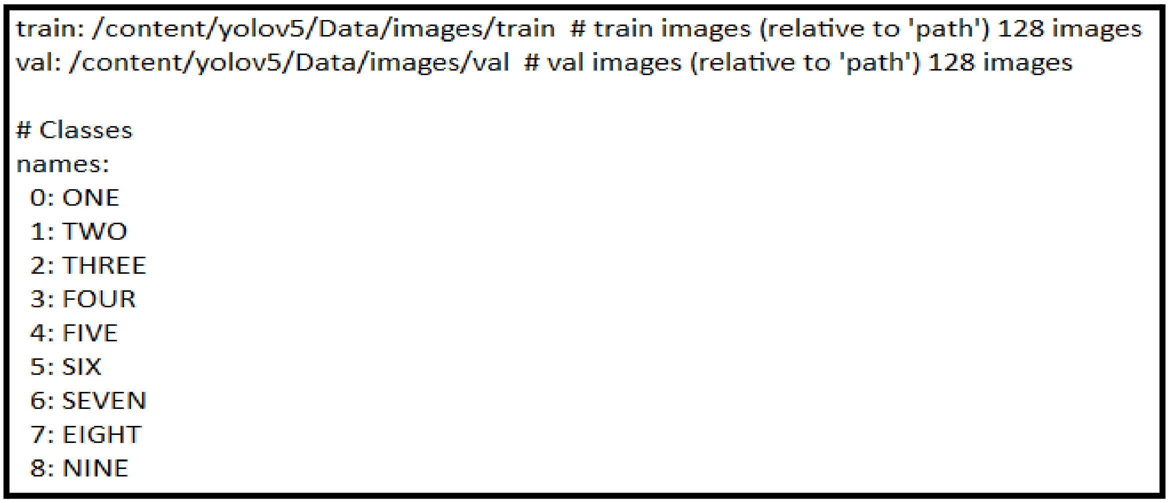
- **YOLO v5 model:**

I have used the Yolo v5 model for object detection tasks, a clone of the official Yolo v5 GitHub was done to use the model, once all the requirements were completed.

To generate the training data for this project, I used data augmentation techniques on the provided images and labeled each training image. The data augmentation included zooming and rotating the images to create variations.

The model was trained using a batch size of 8 and 300 epochs, with pre-yolo v5 weights. The resulting model, config file, and labels.txt were uploaded to the drive. The config file was generated to meet the specific requirements of this project

Config file:



```
train: /content/yolov5/Data/images/train  # train images (relative to 'path') 128 images
val: /content/yolov5/Data/images/val  # val images (relative to 'path') 128 images

# Classes
names:
  0: ONE
  1: TWO
  2: THREE
  3: FOUR
  4: FIVE
  5: SIX
  6: SEVEN
  7: EIGHT
  8: NINE
```

The symbols found in the product descriptions were divided into 9 different classes for the purpose of this project.

After the YOLO V5 model was trained and tested, it was used to detect the symbols in the test images. The model generated a label.txt file containing the object classes and their coordinates for each test image.

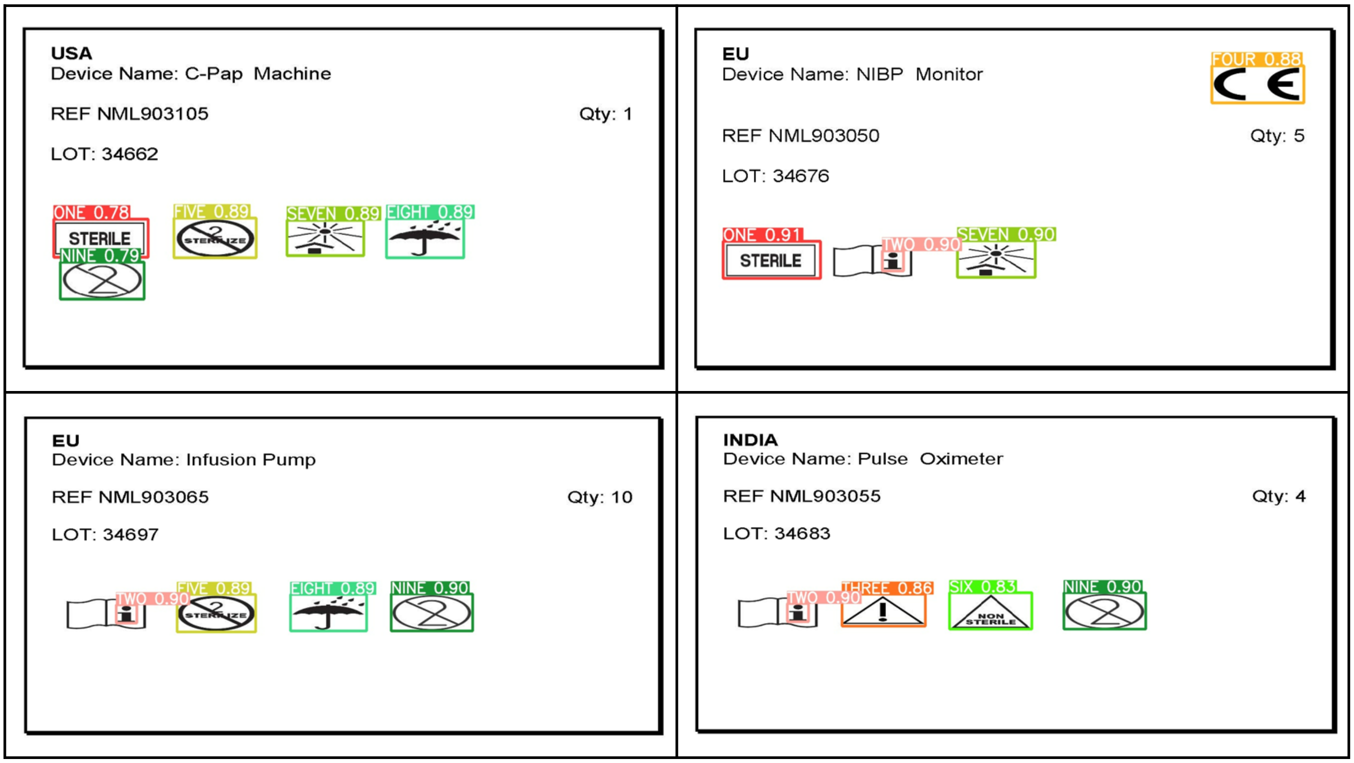
The detect.py script in yolov5 was used to generate label.txt files for all of the test images. The model was able to accurately detect the symbols in the test images.An example of the format of label.txt for an image looks like this:

General template in the label.txt file

[**Class // x-coordinate // y-coordinate // width // height // probability**] of each bounding box

```
5 0.46601 0.596154 0.113528 0.0948718 0.826414
2 0.318491 0.594231 0.114888 0.0833333 0.8596
8 0.623385 0.596154 0.112848 0.0948718 0.899456
1 0.200204 0.600641 0.0292318 0.0474359 0.904421
```

Results of the YOLO v5:



USA
Device Name: C-Pap Machine
REF NML903105 Qty: 1
LOT: 34662

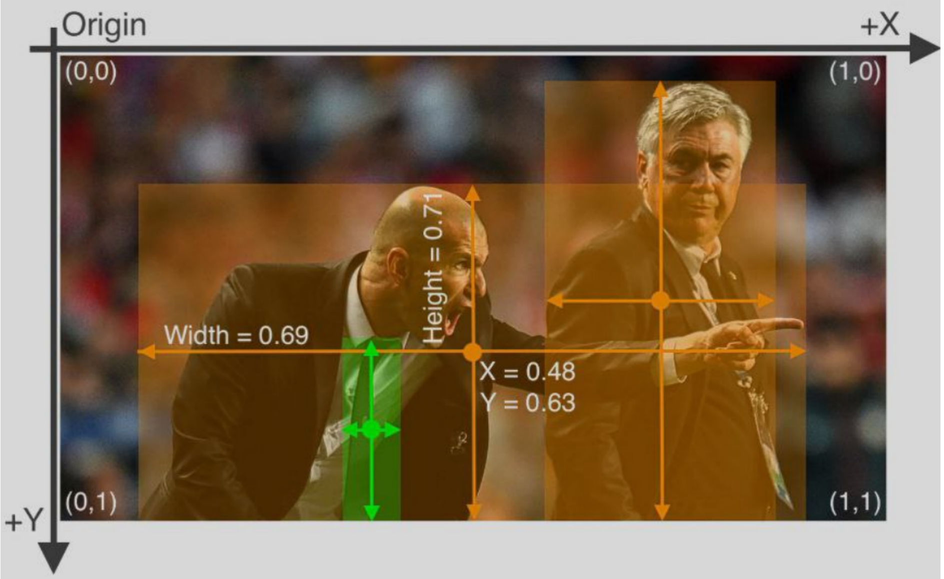
EU
Device Name: NIBP Monitor
REF NML903050 Qty: 5
LOT: 34676

EU
Device Name: Infusion Pump
REF NML903065 Qty: 10
LOT: 34697

INDIA
Device Name: Pulse Oximeter
REF NML903055 Qty: 4
LOT: 34683

For each test image, the model generated a corresponding label.txt file containing the dimensions of the bounding boxes around the detected symbols, as well as the percentages of the symbols within those boxes. This information allowed for accurate identification and localization of the symbols in the test images.

Example:



[Class // x-coordinate // y-coordinate // width // height // probability] of each bounding box

Symbol Extraction:

To determine the order of occurrence of symbols, the x and y coordinates were used as the deciding factors. The positions of the symbols were then calculated using these coordinates and were sorted in the appropriate order.

In the section Symbol extraction in colab, extracting the classes and the order of occurrence were proceeded over there.

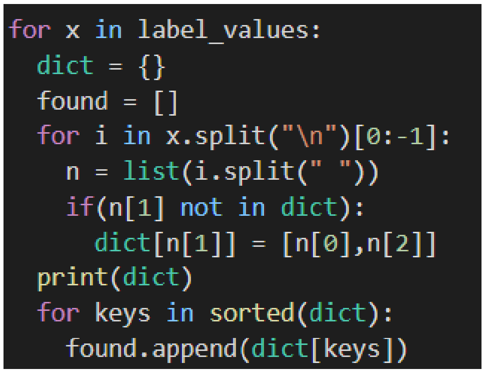
Position sorting is based on the x-coordinates and y-coordinates.

To sort the positions, I first used the x-coordinates as keys in a dictionary and the corresponding [class, y-coordinate] as values. The positions were then sorted based on their x-coordinates. If there are multiple positions with the same x-coordinate, they are sorted based on their y-coordinates.

Example: dictionary {'0.46601': ['5', '0.596154']}

At first, I sorted all the classes based on the x-coordinate using a dictionary,
Given the x-coordinates as dictionary keys and [class, y-coordinate] as values for the dictionary.
Example:
{'0.46601': ['5', '0.596154'],
'0.318491': ['2', '0.594231']}

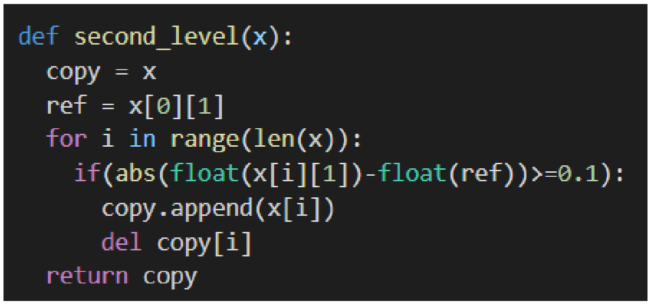
After sorting we get the results found as
[['1', '0.600641'], ['2', '0.594231'],
['5', '0.596154'], ['8', '0.596154']]



```
for x in label_values:
  dict = {}
  found = []
  for i in x.split("\n")[0:-1]:
    n = list(i.split(" "))
    if(n[1] not in dict):
      dict[n[1]] = [n[0],n[2]]
  print(dict)
  for keys in sorted(dict):
    found.append(dict[keys])
```

Once sorting based on the x-coordinate was completed, later second level using y-coordinates sorting was performed using the function,

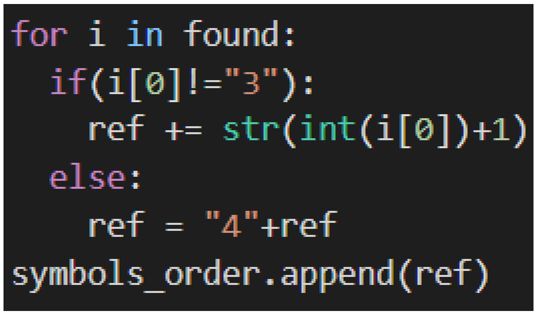
I have assigned a y-coordinate of the first symbol to a variable called ref, later if the difference between the two symbols y-coordinates vary more than 0.1 which means the symbol is below the first image itself, I have appended the image to last indicating the second line.



```
def second_level(x):
  copy = x
  ref = x[0][1]
  for i in range(len(x)):
    if(abs(float(x[i][1])-float(ref))>=0.1):
      copy.append(x[i])
      del copy[i]
  return copy
```

Once sorting both x-coordinate and y-coordinate sorting gets completed we will be having the result as

[['1', '0.600641'], ['2', '0.594231'], ['5', '0.596154'], ['8', '0.596154']]

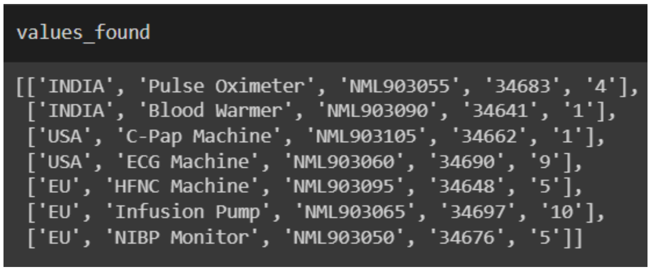
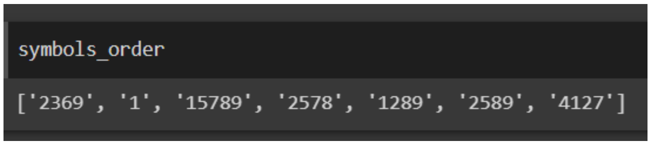


```
for i in found:
  if(i[0]!="3"):
    ref += str(int(i[0])+1)
  else:
    ref = "4"+ref
symbols_order.append(ref)
```

Finally extracting the first value of the list using i[0], which consists of the class.
Indexing starts with zero. I have added 1 to the predictions.

Whenever I have encountered class 4 which is “CE” I have added it to the first of the order
Result:
['2369', '1', '15789', '2578', '1289', '2589', '4127']

Finally every detail got extracted:

Text data extracted	Symbols order
	

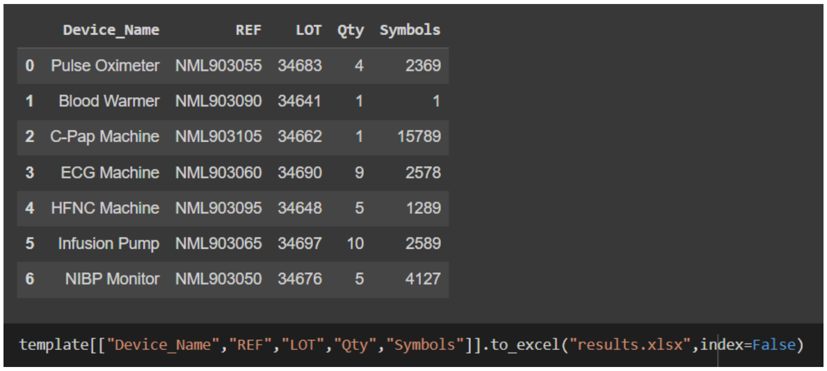
```
values_found
[['INDIA', 'Pulse Oximeter', 'NML903055', '34683', '4'],
 ['INDIA', 'Blood Warmer', 'NML903090', '34641', '1'],
 ['USA', 'C-Pap Machine', 'NML903105', '34662', '1'],
 ['USA', 'ECG Machine', 'NML903060', '34690', '9'],
 ['EU', 'HFNC Machine', 'NML903095', '34648', '5'],
 ['EU', 'Infusion Pump', 'NML903065', '34697', '10'],
 ['EU', 'NIBP Monitor', 'NML903050', '34676', '5']]
```

```
symbols_order
['2369', '1', '15789', '2578', '1289', '2589', '4127']
```

FINAL SECTION:

Using pandas data frame, at last, I have converted the data type to pandas DataFrame.

Created a template file which consists of the device_name, ref, lot , qty and the symbols as the column names.Then each of the respective values was added to their respective columns and turned them into an excel sheet.



```
       Device_Name        REF    LOT  Qty  Symbols
0   Pulse Oximeter  NML903055  34683    4     2369
1     Blood Warmer  NML903090  34641    1        1
2    C-Pap Machine  NML903105  34662    1    15789
3      ECG Machine  NML903060  34690    9     2578
4     HFNC Machine  NML903095  34648    5     1289
5    Infusion Pump  NML903065  34697   10     2589
6     NIBP Monitor  NML903050  34676    5     4127
```

```
template[["Device_Name","REF","LOT","Qty","Symbols"]].to_excel("results.xlsx",index=False)
```