Inter IIT Tech Meet 11.0

The Vitals Extraction Challenge

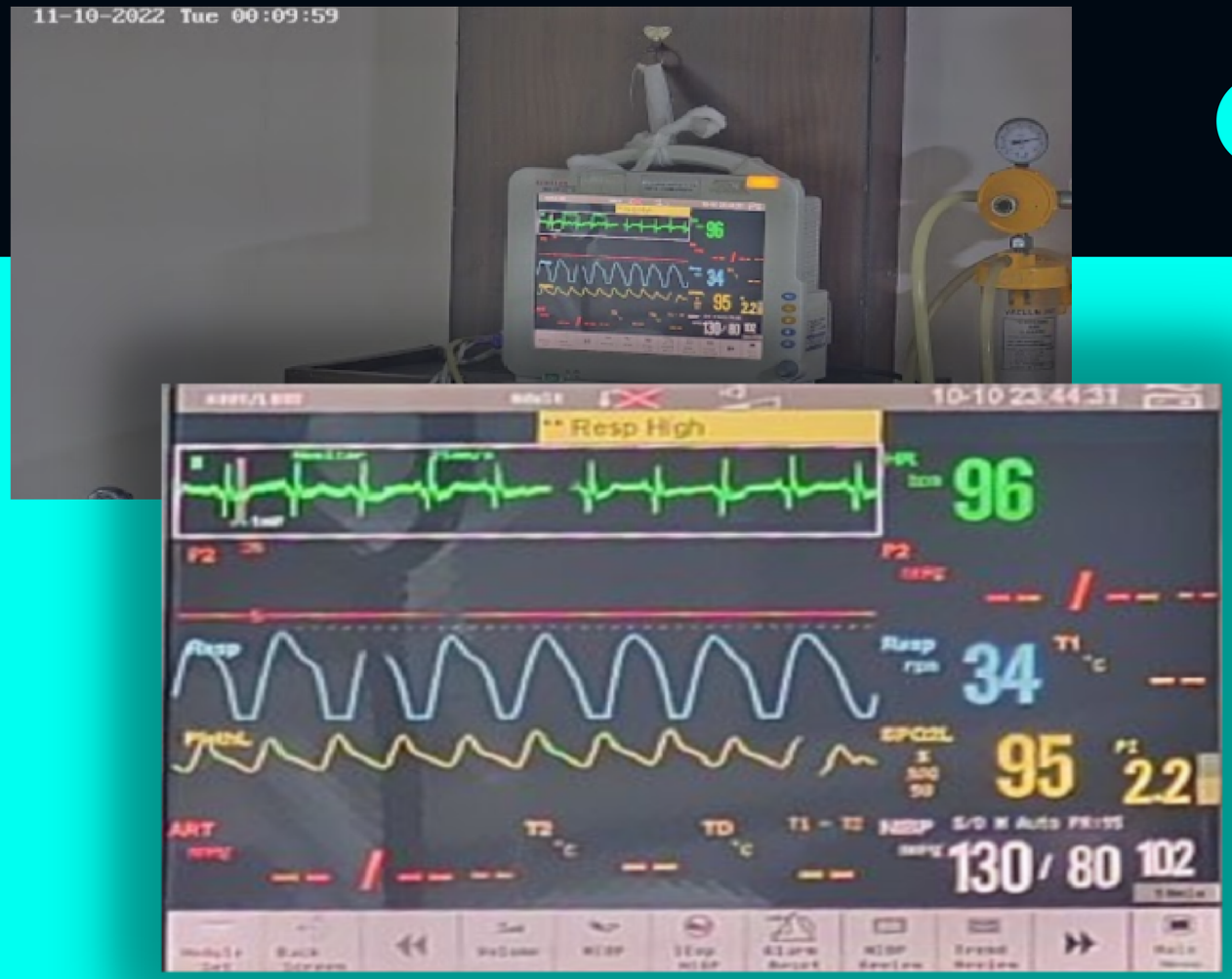
(by CloudPhysician)

Stay Ahead of Health Concerns with Vital Sign Monitoring

Created By : Team 19

Introduction

Patient monitoring is critical in healthcare as it enables healthcare providers to closely observe and track vital signs, thereby detecting any potential health issues before they worsen. Monitoring vital signs such as heart rate, blood pressure, and oxygen saturation provides insightful information on the patient's overall health and well-being.



Problem Statement

The core problem statement is to extract Heart Rate, SpO2, RR, Systolic Blood Pressure, Diabolic Blood Pressure, and MAP from the provided images.

Initial Observations -

- Initially our team analyzed the 2000 images given in the segmentation dataset, along with the other datasets given.
- We Found a wide variety of monitors present in the segmentation as well as the unlabelled dataset -

Observations Continued...

We Found a particular set of observations for every essential :

- For HR : Color is green(rare cases of blue and yellow) and the value is present mostly in top 30% of the image
- For SPO2: Color was not consistent as we detected light blue, pink, yellow, red and even white for it. So apart from the keyword 'SPO2' we did not have a consistent determining observation.
- For RR: Again the color was not consistent but we could see that the value of RR was in the range 1-45.
- For BP: Here we could see that the SYS and DIA BP was most of the times seperated by a '/' and nearby there was 'MAP'.
- For MAP: We found that it was written in brackets '(' and ')'.
- Other than this we always found the keywords next to the values.

Tried Pipelines -

Pipeline 1

We would base off all of our solution on the colors present in the image. We observed that colors like green, blue and red associate with certain values in the image, but we soon found a lot of exceptions.

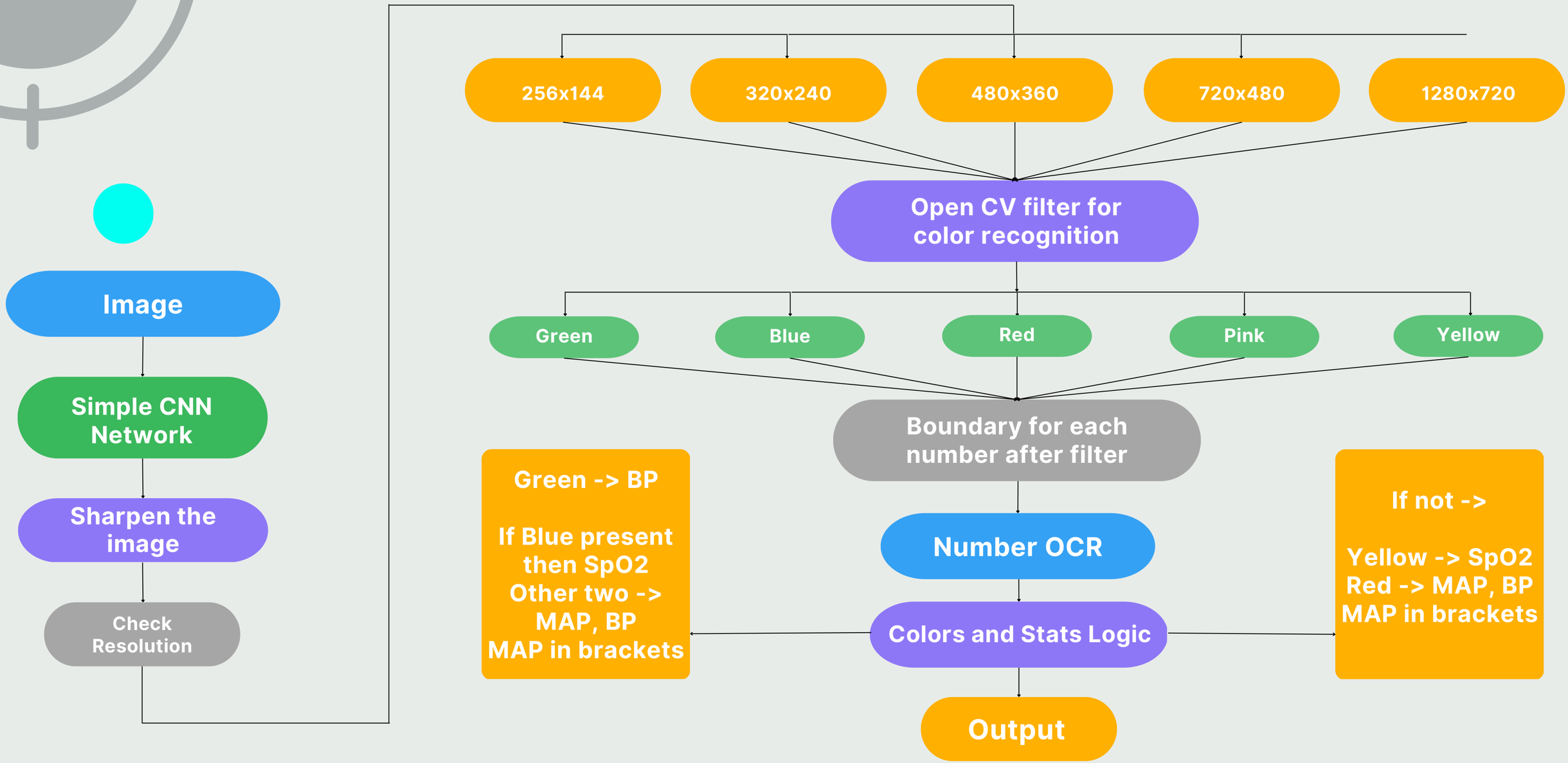
Pipeline 2

After the first FAQ session with CLoudphysician, we tried to approach the problem by classifying it into monitor types, and using location of the numbers as a factor to determine the variables. but the sheer number of classes and lack of data made this strategy inaccurate.

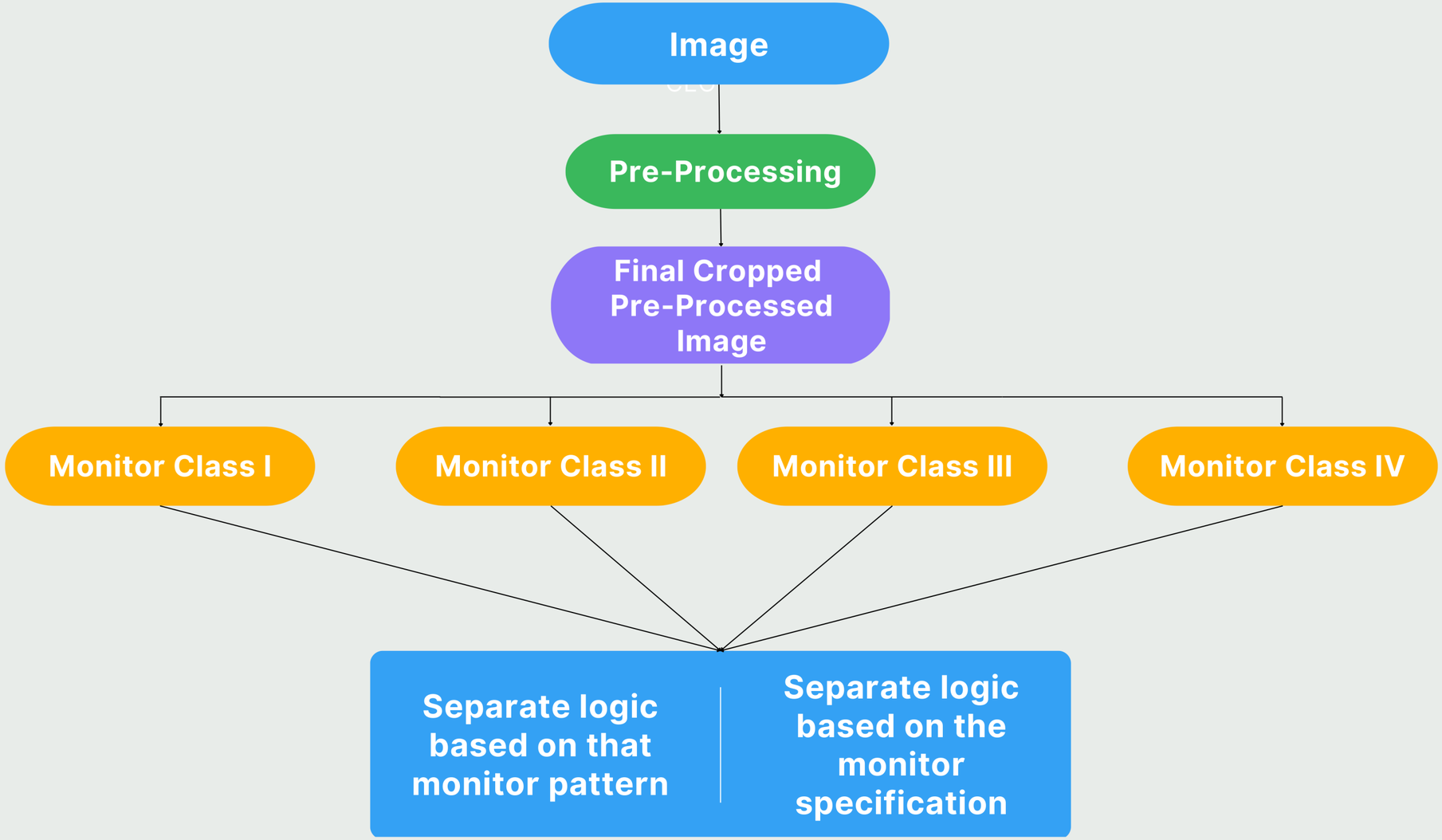
Pipeline 3

We observed that we can use the words present in the images, such as HR, BP, SPO2, and use them to determince which number is which. We broadly classified the images into 2 categories, a monitor with 4 quadrants, and everything else.
This approach worked for a lot of images, but the major problems with this approach was the time taken in processing of the images, inaccuracy of the binary classification model (Overfitting due to less data), and overall greater memory consumption.

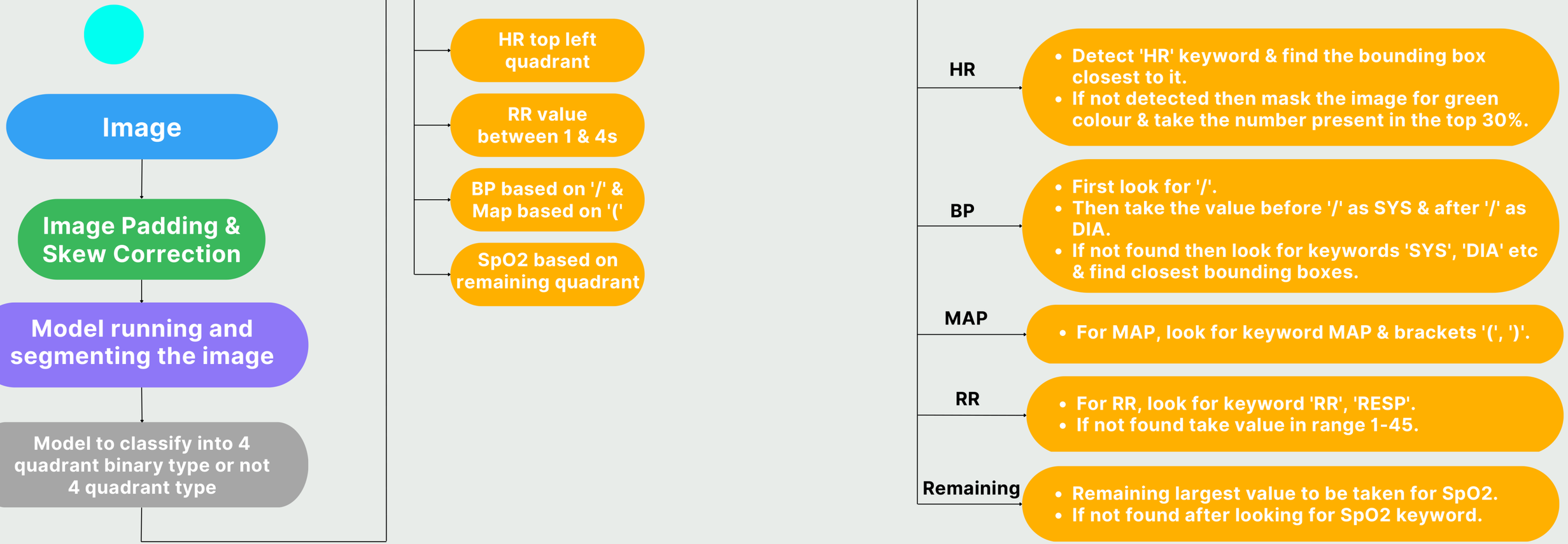
Pipeline 1 flowchart –



Pipeline 2 flowchart –



Pipeline 3 flowchart –



Solution

We Approched the Problem as 2 sub-problems -

1

Segmentation

Identifying the monitors in the image and Cropping them out

2

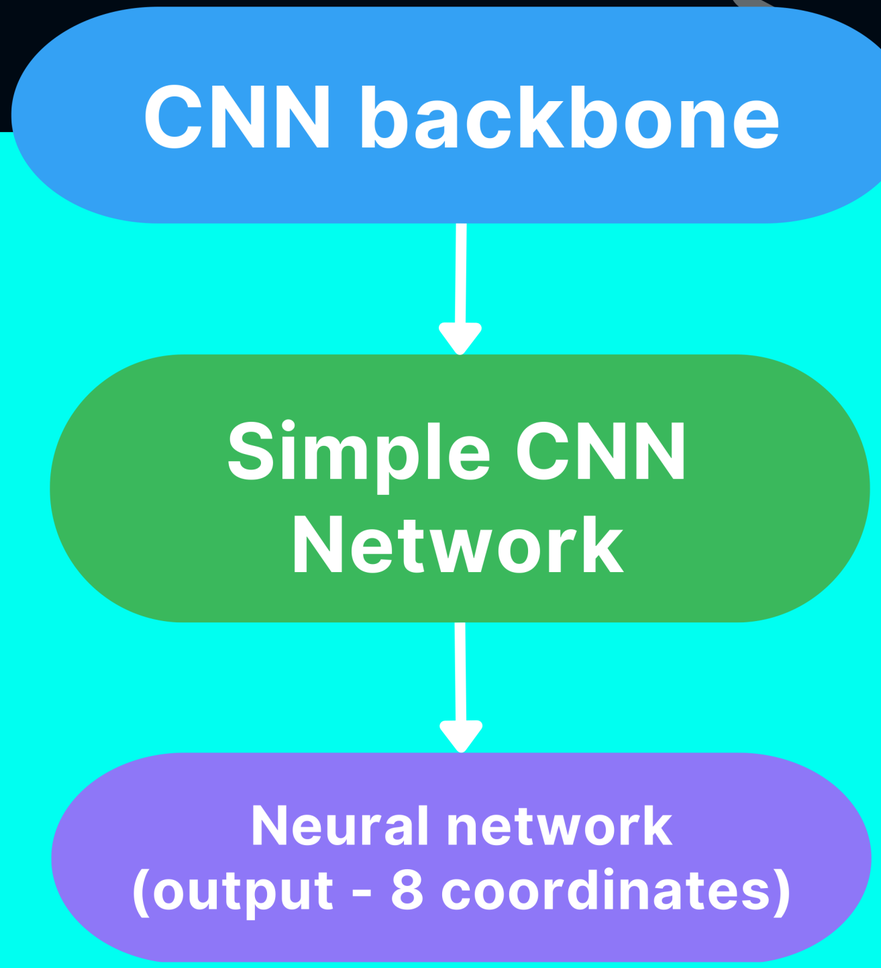
Detection

Identifying the numbers in the image and detecting which number is which

1

Segmentation

For Segmentation of images,
We used CNN networks with different pretrained backbones, along with a custom CNN network and a neural network on top of it.



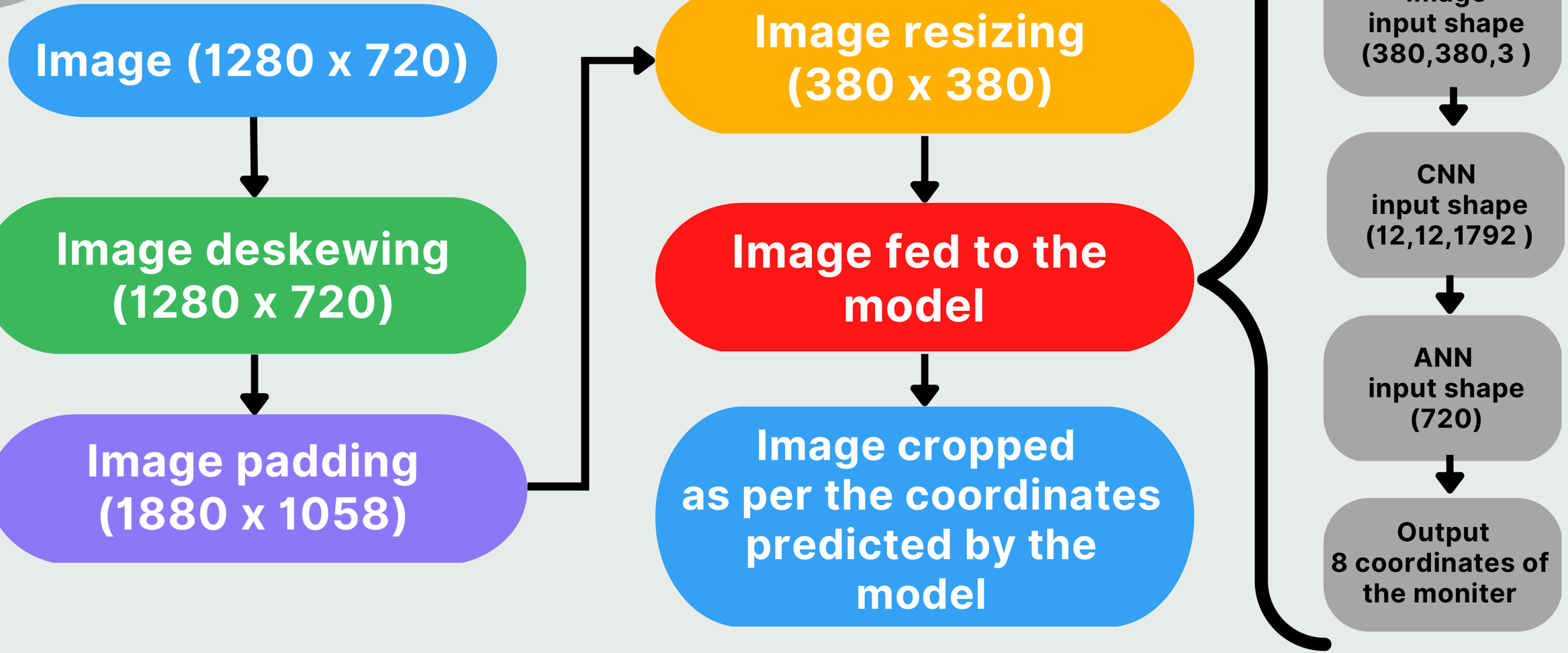
Backbones and their results –

Model name	Nontrainable parameters	Trainable parameters	Total parameters	NO. of images trained on	RMS error on training dataset	RMS error on validation dataset
MobileNetV2	2257984	93874176	96132160	1600	9.01	14.44
EfficientNetB0	4049571	93874176	97923747	1600	8.77	16.03
EfficientNetB1	6575239	72680128	79255367	3200 (after augmentation)	6.36	10.18
EfficientNetB2	7768569	95668860	103437429	3200 (after augmentation)	7.52	8.94
EfficientNetB4	17673823	128106496	145780319	3200 (after augmentation)	5.66	8.17
EfficientNetB4	17673823	128106496	145780319	6400 (after augmentation)	6.87	8.67

Further Techniques to improve performance –

- All the images are padded by 600 pixels of width and 338 pixels by height, before running the model. This ensures that half cropped images give good accuracy in the models.
- an offset of 10 pixels to the width is given after running the model, inorder to eliminate small inaccuracies.
- before running the model, the images are skew-corrected, so that further processing's accuracy is improved (OCR was giving poor accuracy when run on skewed images)

Final segmentation model pipeline –

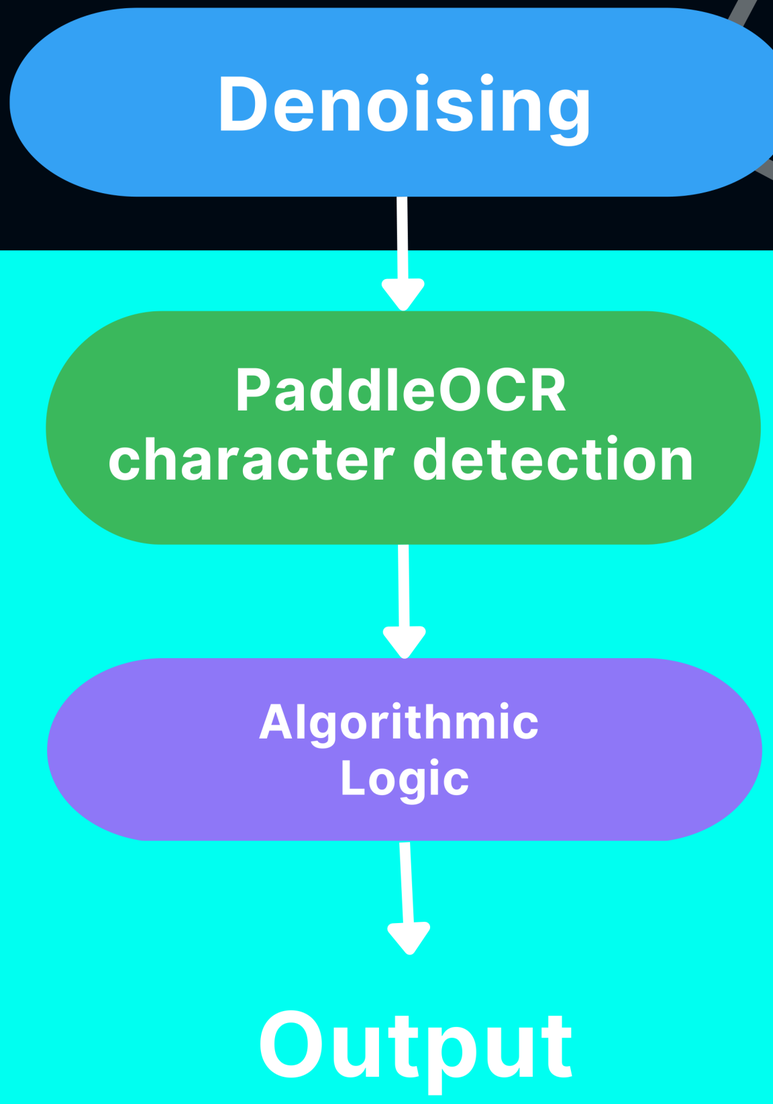


2

Detection

For detection,

We first are denoising the image and running PaddleOCR along with some algorithmic logic to obtain the final output.



Pre-Processing the Cropped Image

- Pre-processing of the image is required so as to get better results from the OCR.
- We tried various pre-processing methods like binarization of image and then noise deletion as well as thinning the image, sharpening the edges etc
- But we didn't get satisfying results so we discarded most of the trials and at last we simply denoised the image and didn't binarize the image as we were using the color factor in the later stages to detect the values.

Why we are using PaddleOCR

- Firstly we are using an OCR(optical character recognition) rather than training our own model as we didn't have sufficient data to train for text identification on LCD monitor and hence using pretrained OCR was the better option.
- We tried various OCRs starting with PyTeseract then EasyOCR similarly KerasOCR, MMOCR but didn't get the desired results.
- But finally we arrived at PaddleOCR which gave us optimal results, we still faced a lot of issues during image detection as the paddleOCR wasn't always perfect at detecting but the results were good enough for us to work with.

Other OCR approaches – EasyOCR

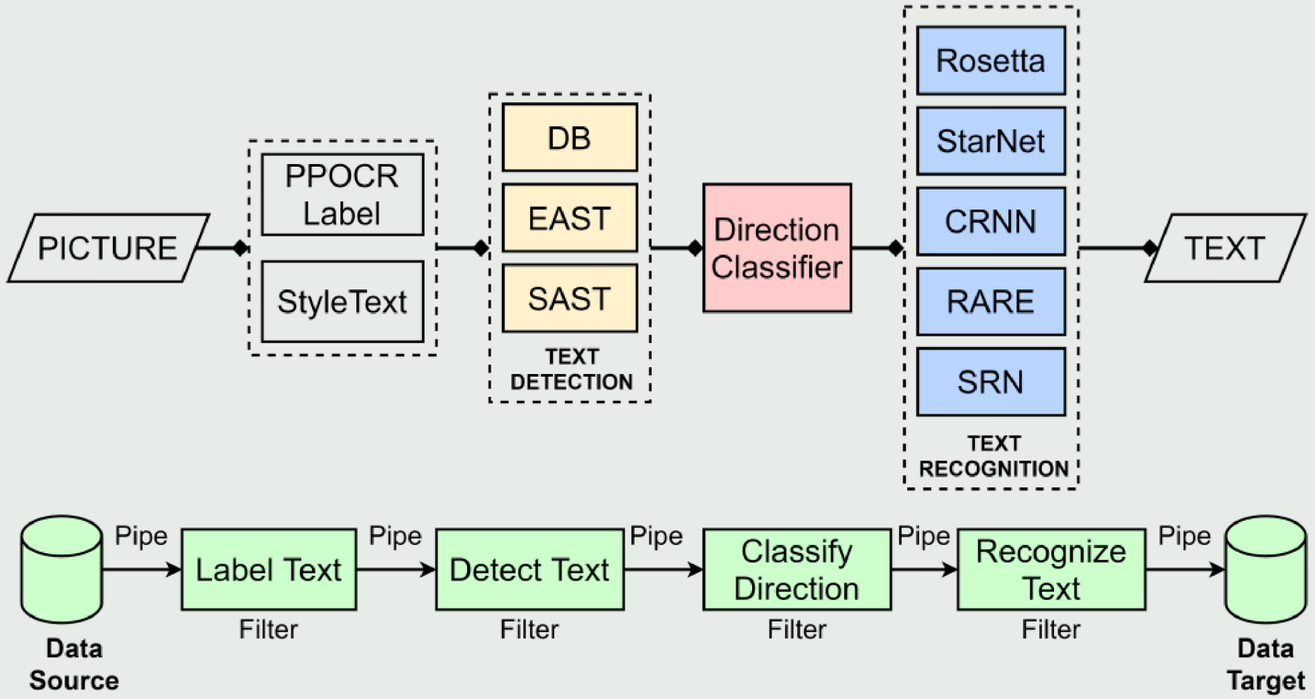
- One of the initial OCR models used when the accuracy of pytesseract tool was not giving good accuracy in value extraction from digital monitors , was EasyOCR.
- EasyOCR, as the name suggests, is a Python package that allows computer vision developers to effortlessly perform Optical Character Recognition.
- When it comes to OCR, EasyOCR is *by far* the most straightforward way to apply Optical Character Recognition:
- The EasyOCR package can be installed with a single pip command.

EasyOCR – Limitations

- Despite being very smooth to use and install in any python environment like the anaconda or Google Colab, Easy OCR has it's own set of limitations.
- As the model is not well curated for digital data. While using it to predict the values, it often failed to find the value of Blood Pressure (BP) with '/' symbol.
- Also, as the Spatial geometry logic to find the nearest keyword to the bounding box of the number (health monitor values) requires to detect the keywords like ('SPO2', 'ECG', 'RR' etc) which are far smaller in font size as compared to the numerical values. The Easy OCR model was unable to detect that accurately.
- Even if the input image was little blurry, the Easy OCR failed to detect the values

PaddleOCR

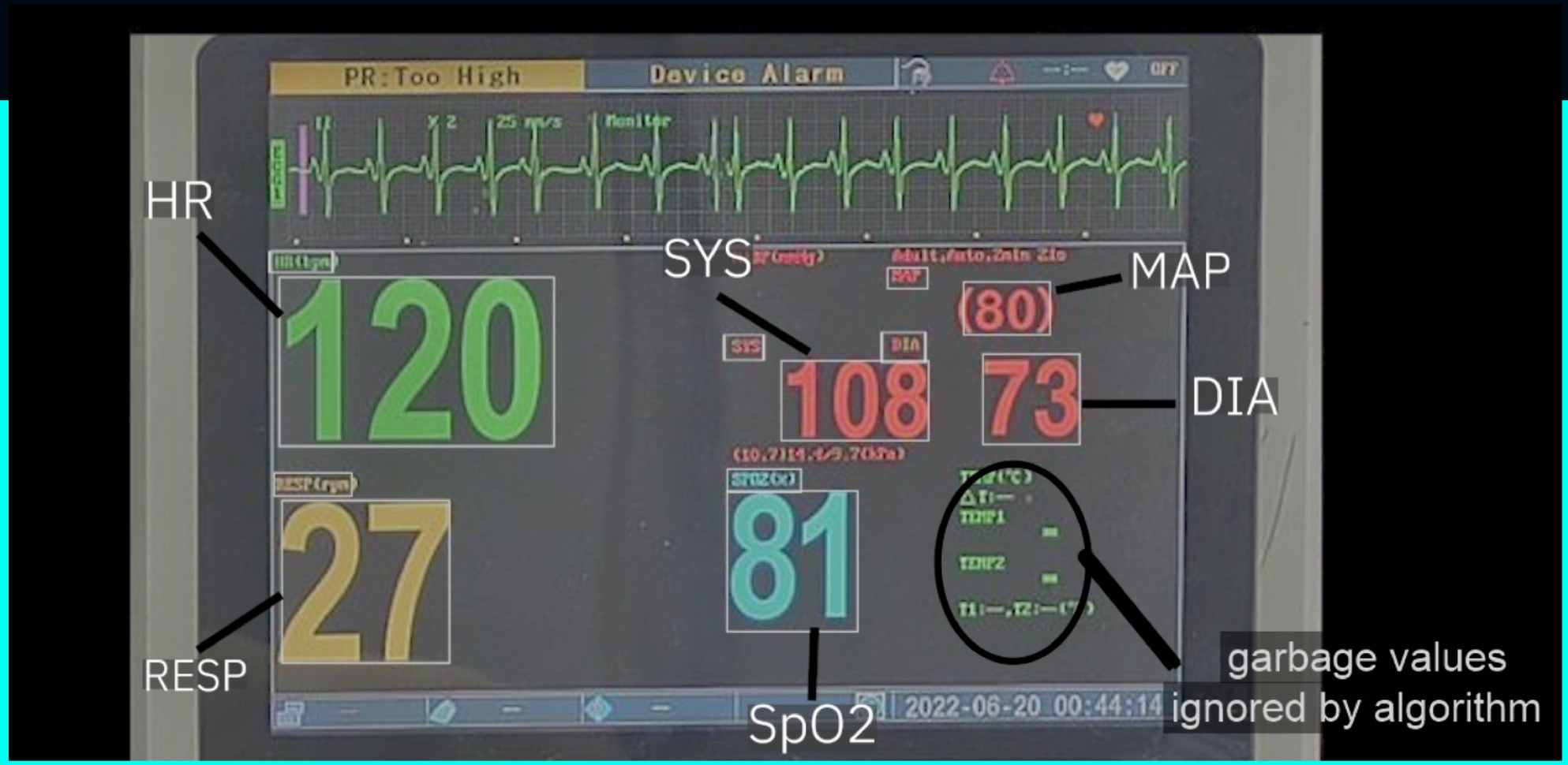
- PaddleOCR is a state-of-the-art Optical Character Recognition (OCR) model, developed by BYDU. It is pre-trained, highly optimized and Light-weight OCR Model.



PaddleOCR

- The output of Paddle OCR model contains:
- The Four Co-ordinates (of all four corners) of the Detection Bounding Boxes in (x,y) format
- Recognition Result - consisting of the recognition string and Confidence Value. Confidence value, basically gives the percentage of the confidence of detection
- The best part about about the Paddle OCR model is the fact that it is light-weight, easy to use and works well for even unclear and blurry input images also.
- It overcomes the limitations of the Easy OCR. Even if the input image is unclear and blurry, the prediction made by the model are excellent and very accurate.

Qauntities to Detect -

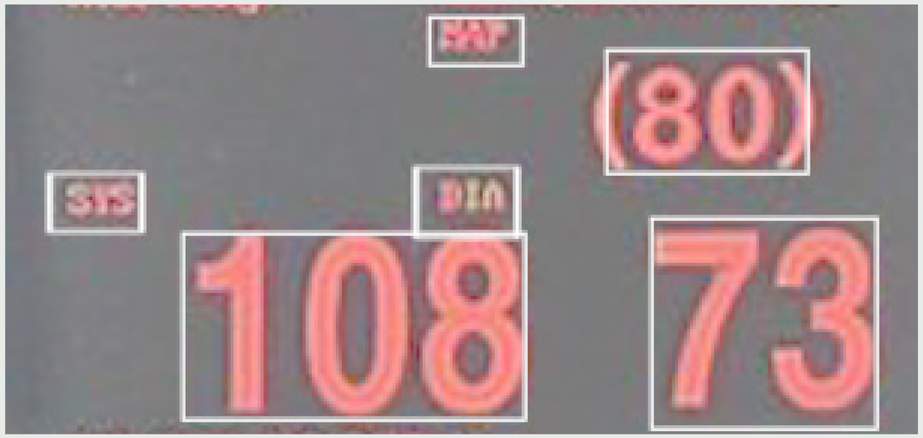


Heart Rate, RR and SpO2 -

- All the 3 values are detected by recognizing the words in the image, and corresponding the closest number to the word related to HR, RR or SpO2 to the relevant quantity.
- Also, if HR detection fails, HR is detected by masking out green color in the image, and extracting the number in it, which is taken to be the HR.

Blood Pressure and MAP -

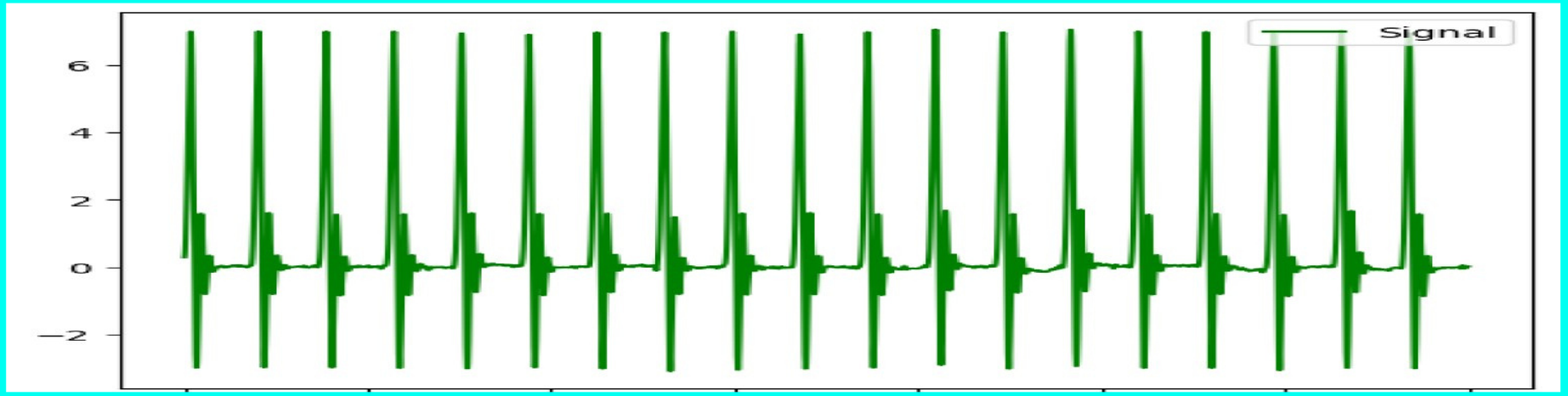
- We are using 4 functions to find the values of BP and MAP, and each function is called only when the function before it fails.
- The first function tries to detect the BP by finding either '/' and if we get the value we are taking the first part before the '/' as SBP and after '/' part as DBP.
- Our second attempt to find the BP is by looking for keywords related to bp, and then finding the nearest bounding boxes to it.



Blood Pressure and MAP –

- We find Map by looking for '(' or ')' and then if we got it we will take the numbers after '(' or before ')'. If not found then we look for the keyword 'map' and find the bounding box nearest to it.
- The last attempt to find out the BP and MAP is by looking for three numbers having closest distance and within distances within a threshold (preset by us) and then trying to find the SBP, DBP and MAP according to the positions of the numbers present.

Graph Detection –



ECG Graph Detection

- To detect the ECG graph we first isolate the green color from the segmented image by masking the image and then we draw the contours around the objects.
- We then look for the contour with the largest perimeter as that will give us the graph.
- Then we simply plot the graph by converting the contour into a numpy array and then removing garbage values and simply plotting it.
- If we are not able to detect the graph then we use neurokit2 library to simulate the graph using heart rate values.

Possible Improvements to the Solution -

Used more colors for better detection of quantities

A lot of colors had associations with different quantites, such as blue being associated with SpO2, and red and pink with BP and MAP. We encountered a problem with certain exceptions, but the information could have been used.

Proper classification could have had been done

The unlabelled dataset contained a lot of classes, but quantification of classes, and then using the location information to get more accurate results.

3rd pipeline

The fallacy of our current methodology is the exclusion of the numbers' location information . This was employed in the 3rd pipeline, by using the quantities' location information into consideration, by partial classification.
The binary classification model could have been developed further and the 3rd pipeline could have been used.

https://www.youtube.com/watch?v=624ykLbOFfk

Thank You