• **Project**   Real time Drowsiness Detection.

→

Gameplan: ① installing ultralytics YOLO (Latest version of YOLO ultralytics YOLOv8)
② Making detections
③ Fine tuning a drowsiness model
④ Real time performance.

Code: 1) Install dependencies like numpy, matplotlib, cv2
2) Load mode → ultralytics / yolov5 from torchhub.
3) Real time detection.
4) we will store some realtime images and label them. (till now)

5) goal → Real time video drowsyness detection (will show in final phase.

6) Improvement scope: ma'am will suggest + time for drowsyness (like what you thought).

// classification vs object detection.

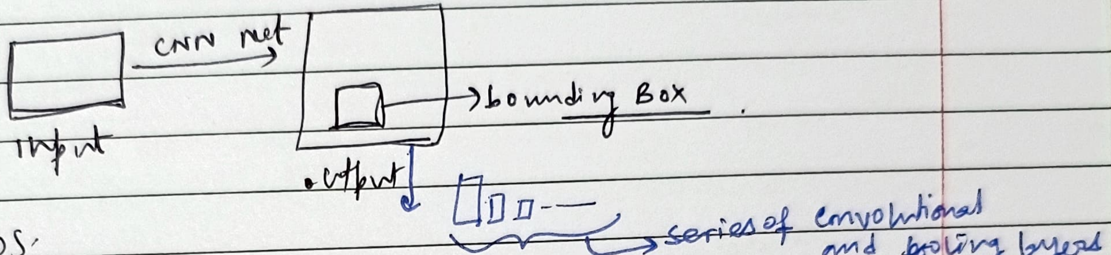
- classification → one object and label per image.
- Detection → Multiple objects per image }
Determine location

= naive approach

I/P img $\xrightarrow{CNN}$ label.

YOLO (How works)

instead of making predictions on many regions of an image, YOLO passes the entire image at once.



Steps:

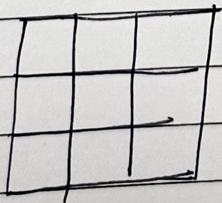
① Divide the image into cells with SXS grid.

② · Each cell produces B bounding boxes.

③ Return bounding boxes above confidence threshold (All other bounding boxes have a confidence probability less than threshold say 0.90)

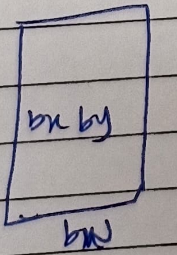
· ~~How~~ Bounding boxes are encoded?

⇒



Suppose have 3x3 image & Each cell produces one bounding box.

For each cell the CNN predicts a vector y



$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \end{bmatrix}$$

p_c → P(bounding box contain object)

b_x, b_y } → coordinate of bounding box.

b_h, b_w } width of bounding box as a ~~cell~~ % of the cells width/height

c_1, c_2 } → P(object contain class c_i).

④ Note: What happens if we predicts multiple bounding boxes / cell → we augment y.