

Aim:

To implement a real time object detection system that can identify and localize multiple objects in an image

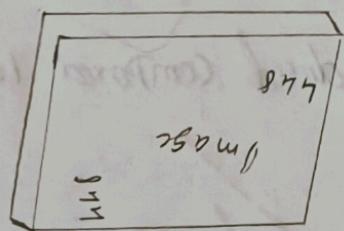
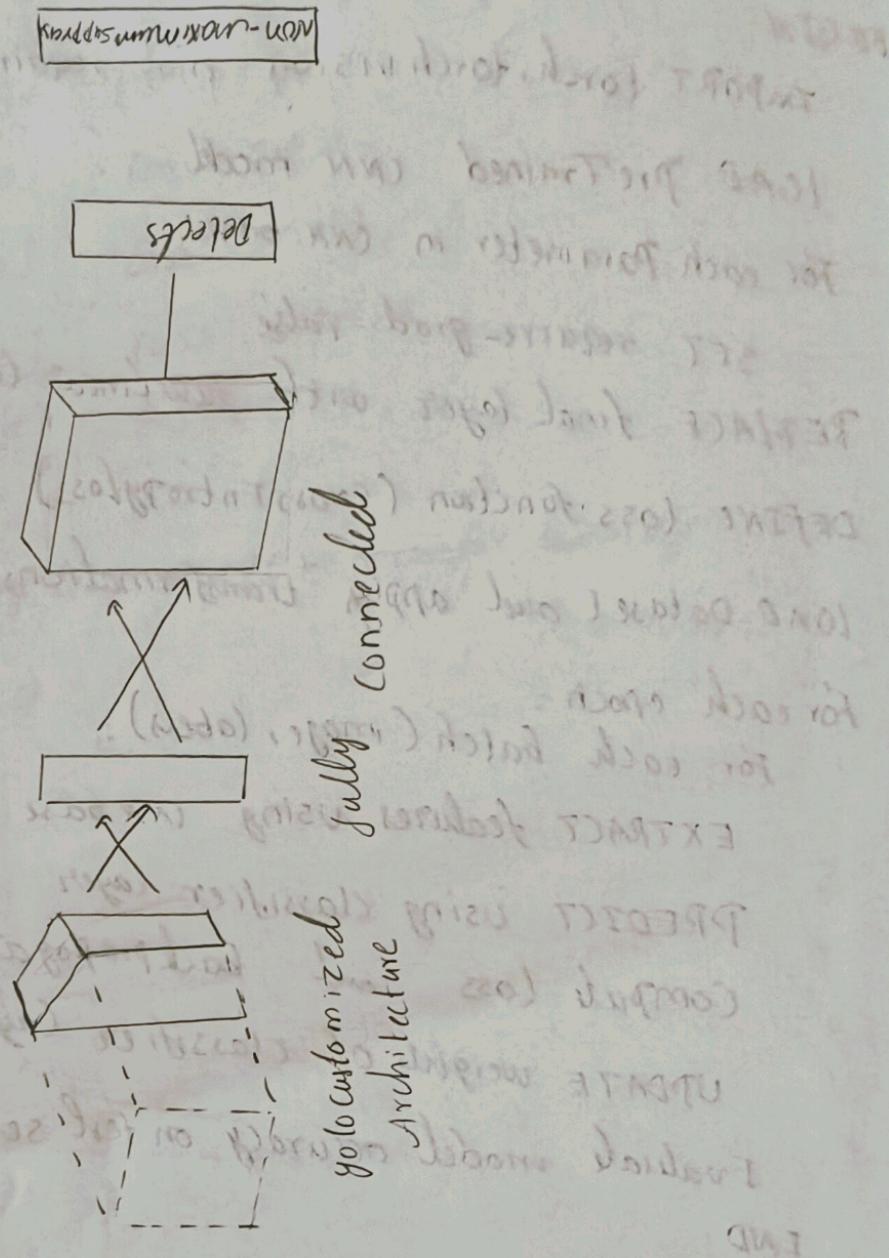
Algorithm:

- * Input processing, Resize input image to fixed dimension
- * Normalize pixel values to [0,1] range
- * Feature Extraction, Pass image through convolutional backbone network
- * Divide input image into $S \times S$ grid
- * Each grid cell is responsible for detecting objects whose center falls within it.
- * Each grid cell predicts B bounding boxes
- * Each box contains: $(x, y, w, h, \text{confidence})$
- * Each grid cell Predicts C class probabilities
- * Conditional Probability for each class given an object is present
- * Apply confidence thresholding

Observation:

- * YOLO detects multiple objects with boundary boxes
- * Inference is fast and efficient
- * Model Accuracy is high

Yolo Architecture Model



pseudo code:

CLASS YOLO :

METHODS :

CONSTRUCTOR (num_classes, anchors) :

self.backbone = DarkNetBackbone()

self.detection_heads = []

self.anchors = anchors

self.num_classes = num_classes

FORWARD (image-tensor) :

features = self.backbone(image-tensor)

detection = []

FOR each feature-level IN features :

detection = self.detection_heads[feature_level]

(features[feature_level])

detections.append(detection)

RETURN detections

PREDICT (image) :

Processed-image, original_dims = PREPROCESS(image)

raw_detections = FORWARD(Processed-image)

FUNCTION DETECT-OBJECTS (image_Path, model) :

image = LOAD-IMAGE (image-Path)

boxes, scores, classes = model.PREDICT (image)

result-image = DRAW-BOXES (image, boxes, scores, classes)

RETURN result-image, boxes, scores, classes

~~Result~~

The yolo model successfully implemented

```
from PIL import Image as PILImage
from IPython.display import Image, display
from ultralytics import YOLO

# Step 1: Load YOLO model (YOLOv8 nano for speed)
model = YOLO('yolov8n.pt')

# Step 2: Convert uploaded image (if .webp)
input_image = "p020rgbg.jpg.webp" # change if your uploaded filename is different
converted_image = "converted_test.jpg"

# Convert webp → jpg
img = PILImage.open(input_image).convert("RGB")
img.save(converted_image, "JPEG")

# Step 3: Run YOLO detection
results = model(converted_image)

# Step 4: Save detection result
results[0].save(filename="result_detected.jpg")

# Step 5: Display result inline
display(Image(filename="result_detected.jpg"))
```

...

image 1/1 /content/converted_test.jpg: 384x640 34 cars, 10.9ms
Speed: 1.7ms preprocess, 10.9ms inference, 18.9ms postprocess per image at shape (1, 3, 384, 640)

