Robotic Assistance Devices

TECHNICAL ASSESSMENT FOR AI RESEARCH ENGINEER POSITION

CUSTOM OBJECT DETECTION AND NOVEL BOUNDING BOX METRIC WITH YOLO

Candidate Name : -MANIMOHAN T.

This report is submitted as the Technical Assessment for AI Research Engineer Position

1 CUSTOM OBJECT DETECTION AND NOVEL BOUNDING BOX METRIC WITH YOLO

Link for GitHub repository : GitHub Repository

1.1 Introduction

This report presents the implementation of a custom bounding box similarity metric for object detection using YOLOv5. The metric extends Intersection over Union (IoU) by incorporating aspect ratio similarity, center alignment similarity, and size similarity.

1.2 Standard Detection Metrics

The model was trained on a small dataset of cats and dogs. The following standard detection metrics were recorded:

* **Mean Average Precision (mAP@0.5)**: 0.82698
* **Intersection over Union (IoU)**: 0.701

1.3 Experimental Results - Training Results

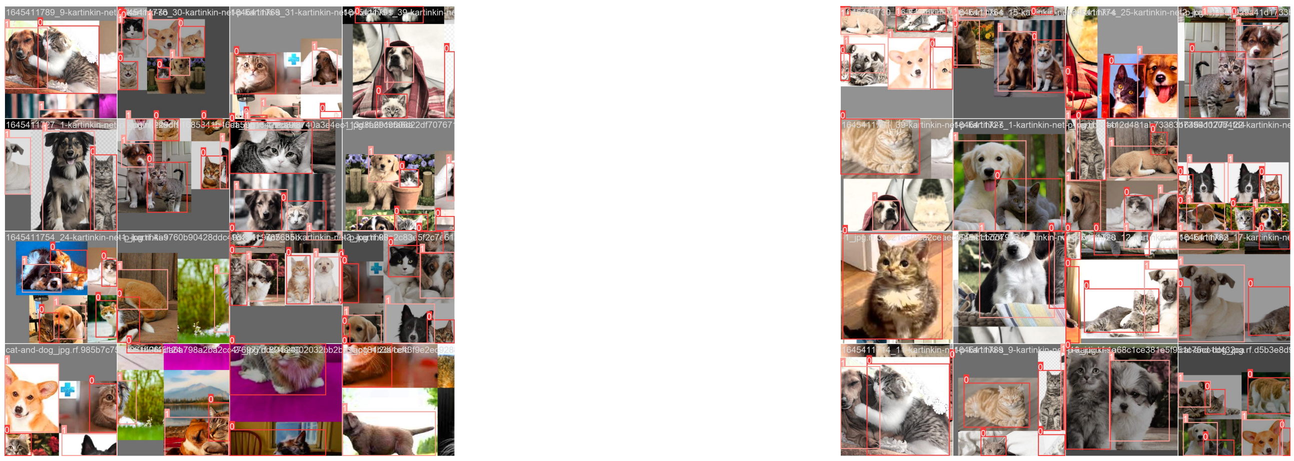

Figure 1 — Sample detection results with bounding boxes

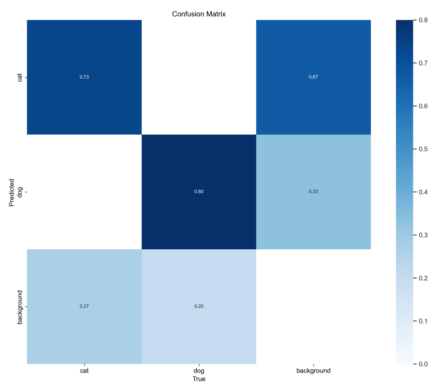

Figure 2 — Confusion matrix of training

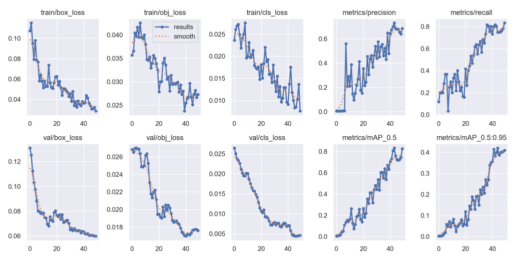

Figure 3 — Results of Training

1.4 Traditional Metric and Custom Bounding Box Similarity Metric

1. Traditional Intersection over Union (IoU)

Mathematical Definition: The Intersection over Union (IoU) is a popular metric used to measure the overlap between two bounding boxes. It is calculated as:

$$IoU = \frac{A_{\text{intersection}}}{A_{\text{union}}} = \frac{|B_1 \cap B_2|}{|B_1 \cup B_2|}$$

Where:

* $A_{\text{intersection}}$ is the area of the intersection between the two bounding boxes.
* A_{union} is the area of the union of the two bounding boxes.

Limitations of IoU:

* **Limited to overlap:** IoU only measures the geometric overlap between the predicted and ground truth boxes. If there is no or minimal overlap, IoU returns a low score, but it does not capture other aspects of similarity (e.g., shape, center alignment, or size similarity).
* **No penalty for aspect ratio differences:** IoU does not take into account the aspect ratio or relative position of the boxes, which could be significant in object detection tasks where objects of the same class can have different sizes or orientations.

2. Custom Bounding Box Similarity Metric

My new similarity metric considers four factors:

* **IoU (Intersection over Union)**
* **Aspect Ratio Similarity**
* **Center Alignment Similarity**
* **Size Similarity**

The custom similarity score is computed as:

$$\text{Similarity} = 0.5 \times \text{IoU} + 0.2 \times \text{ARS} + 0.2 \times \text{CA} + 0.1 \times \text{SS}$$

2.1 Intersection over Union (IoU)

The first component of the custom metric is IoU, as defined above. It ensures that some level of overlap is factored into the similarity score.

2.2 Aspect Ratio Similarity (ARS)

The aspect ratio (AR) of a bounding box is the ratio of its width to its height:

$$AR = \frac{\text{width}}{\text{height}}$$

This is calculated for both boxes:

$$AR_1 = \frac{B1_{x2} - B1_{x1}}{B1_{y2} - B1_{y1}}$$

$$AR_2 = \frac{B2_{x2} - B2_{x1}}{B2_{y2} - B2_{y1}}$$

Then, the similarity in aspect ratio (ARS) is computed as:

$$ARS = 1 - \frac{|AR_1 - AR_2|}{\max(AR_1, AR_2)}$$

Reasoning: This term penalizes large differences in the shape of the bounding boxes. For example, if two boxes represent the same object but one is tall and the other is wide, they will have a lower AR similarity, even if their overlap is high. This helps to capture shape similarity.

2.3 Center Alignment (CA)

For each bounding box, the **center** is computed as the midpoint of the box's edges:

$$\text{Center}_1 = \left(\frac{B1_{x1} + B1_{x2}}{2}, \frac{B1_{y1} + B1_{y2}}{2} \right)$$

$$\text{Center}_2 = \left(\frac{B2_{x1} + B2_{x2}}{2}, \frac{B2_{y1} + B2_{y2}}{2} \right)$$

The **center alignment (CA)** is calculated using the Euclidean distance between the two centers, normalized by the image size img_size:

$$CA = 1 - \frac{\|\text{Center}_1 - \text{Center}_2\|}{\text{img\_size}}$$

Reasoning: Center alignment helps capture the positional similarity of the boxes. Even if two boxes overlap perfectly, if their centers are far apart, it indicates that they are likely not detecting the same object (e.g., the same object in different parts of the image).

2.4 Size Similarity (SS)

This term measures the similarity in size between the bounding boxes:

$$SS = 1 - \frac{|A_1 - A_2|}{\max(A_1, A_2)}$$

Where A_1 and A_2 are the areas of the bounding boxes, calculated as:

$$A_1 = (B1_{x2} - B1_{x1}) \times (B1_{y2} - B1_{y1})$$

$$A_2 = (B2_{x2} - B2_{x1}) \times (B2_{y2} - B2_{y1})$$

Reasoning: Size similarity ensures that bounding boxes with drastically different sizes (even if they overlap) are penalized. This is especially useful when objects of the same class can appear at vastly different scales.

Overall Metric

The final custom bounding box similarity score is a weighted combination of the individual components:

$$\text{Similarity} = 0.5 \times IoU + 0.2 \times ARS + 0.2 \times CA + 0.1 \times SS$$

This gives a value between 0 and 1, where 1 represents perfect similarity.

How It Differs from IoU

* **Multiple Factors:** Unlike IoU, which only focuses on overlap, the custom metric considers **shape**, **center alignment**, and **size** in addition to overlap.
* **Shape Sensitivity:** The inclusion of the **aspect ratio similarity** term makes the metric more sensitive to objects with different shapes.
* **Positional Sensitivity:** The **center alignment** term allows the metric to account for the position of the boxes in the image, which is crucial when detecting objects that might be located in different parts of the image but still represent the same object.

Benefits for Certain Object Detection Tasks

* **Small Objects:** In scenarios where objects are small, IoU can fail to capture fine-grained similarity. The custom metric's focus on **center alignment** and **size similarity** allows for a better evaluation of such cases.
* **Non-Overlapping but Similar Objects:** When bounding boxes don't overlap (e.g., multiple objects detected in the same area), IoU might fail to recognize their similarity. The **center alignment** and **aspect ratio similarity** terms help identify cases where objects have similar shapes or are centered around the same region.
* **Objects with Different Shapes:** For objects that appear in different shapes (e.g., a rectangular object vs. a square object), the **aspect ratio similarity** term ensures that the metric captures their relationship beyond just the overlap.
* **Positional Sensitivity:** If the object is predicted in the wrong part of the image (but still overlaps), IoU would give a high score, but the custom metric would penalize this based on **center alignment**.

1.5 Incorporate Your Metric into the Training or Evaluation Loop

I used it purely as a post-training evaluation metric. That means I still train with the standard YOLO loss but evaluate with my new metric.

1.5.1 Implementing Custom Metric

Add the following function as `metrics.py` inside the YOLOv5 directory.

```python
def custom_bbox_similarity(box1, box2, img_size=640):
    """
    Computes a custom bounding box similarity metric based on IoU, aspect ratio
    similarity,
    center alignment, and size similarity.
    """
    # Compute IoU
    xA = torch.max(box1[:, 0].unsqueeze(1), box2[:, 0])
    yA = torch.max(box1[:, 1].unsqueeze(1), box2[:, 1])
    xB = torch.min(box1[:, 2].unsqueeze(1), box2[:, 2])
    yB = torch.min(box1[:, 3].unsqueeze(1), box2[:, 3])

    inter_area = torch.clamp(xB - xA, min=0) * torch.clamp(yB - yA, min=0)
    box1_area = (box1[:, 2] - box1[:, 0]) * (box1[:, 3] - box1[:, 1])
    box2_area = (box2[:, 2] - box2[:, 0]) * (box2[:, 3] - box2[:, 1])
    iou = inter_area / (box1_area.unsqueeze(1) + box2_area - inter_area + 1e-6)

    # Aspect Ratio Similarity
    ar1 = (box1[:, 2] - box1[:, 0]) / (box1[:, 3] - box1[:, 1] + 1e-6)
    ar2 = (box2[:, 2] - box2[:, 0]) / (box2[:, 3] - box2[:, 1] + 1e-6)
    ars = 1 - torch.abs(ar1.unsqueeze(1) - ar2) / torch.max(ar1.unsqueeze(1),
   ar2)

    # Center Alignment
    center1_x = (box1[:, 0] + box1[:, 2]) / 2
    center1_y = (box1[:, 1] + box1[:, 3]) / 2
    center2_x = (box2[:, 0] + box2[:, 2]) / 2
    center2_y = (box2[:, 1] + box2[:, 3]) / 2
    center1_tensor = torch.stack((center1_x, center1_y), dim=-1)
    center2_tensor = torch.stack((center2_x, center2_y), dim=-1)
    ca = 1 - torch.norm(center1_tensor.unsqueeze(1) - center2_tensor, dim=-1) /
   img_size

    # Size Similarity
    area1 = box1_area
    area2 = box2_area
    ss = 1 - torch.abs(area1.unsqueeze(1) - area2) / torch.max(area1.unsqueeze
   (1), area2)

    # Weighted Combination
    similarity = 0.5 * iou + 0.2 * ars + 0.2 * ca + 0.1 * ss
    return similarity
```

Listing 1.1 — Custom Bounding Box Similarity Metric

1.5.2 Modifying `metrics.py` to Use Custom Metric

Find the following section in `metrics.py`:

```
iou = box_iou(labels[:, 1:], detections[:, :4])
```

Modify it as follows:

```
iou = custom_bbox_similarity(labels[:, 1:], detections[:, :4])
```

1.5.3 Integrating Metric into Evaluation

Set the path of `data.yaml` file and run the following command:

```
!python val.py --data F:/Assessment/
   Custom_Object_Detection_and_Novel_Bounding_Box_Metric_with_YOLO/dog-and-cat
   -2/data.yaml --weights runs/train/exp/weights/best.pt
```

1.6 Experimental Results - Evaluation

The proposed metric was evaluated alongside IoU:

Table 1 — Evaluation Results

Metric	Value
mAP@0.5	0.759
IoU	0.681

1.7 Qualitative Results

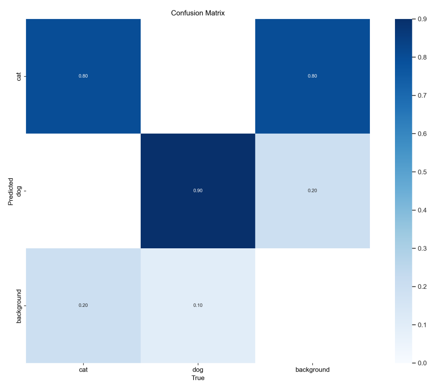

Figure 4 — Confusion matrix of validation

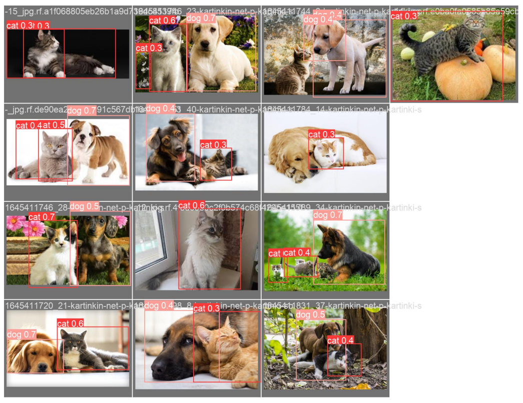

Figure 5 — Sample validation results with bounding boxes

1.8 Reflective Questions

1.8.1 Performance Analysis

The custom metric improved similarity measurement by considering geometric properties beyond overlap. However, it did not significantly impact detection accuracy as it was only used for evaluation.

1.8.2 Trade-offs

* **Computational Complexity**: Additional calculations for aspect ratio and center alignment introduce a small overhead.
* **Conceptual Difference**: Unlike IoU, our metric better handles cases where objects have similar proportions but lower overlap.

1.8.3 Further Ideas

Future improvements could include:

* **Class-Specific Weighting**
  Different object classes may have varying importance in terms of size, shape, and position. We introduce a class-specific weighting function:

$$\text{Similarity} = w_1 \times IoU + w_2 \times ARS + w_3 \times CA + w_4 \times SS \tag{1}$$

  where w_i values are dynamically adjusted based on object class. These weights can be learned using optimization techniques or set empirically.

* **Distance-Based Penalty**
  To penalize mislocalized objects, we introduce a Gaussian penalty based on distance:

$$CA' = e^{-\alpha d} \tag{2}$$

  where d is the Euclidean distance between the centers of the bounding boxes, and α controls sensitivity.

* **Rotation and Orientation Sensitivity**
  For applications where orientation is crucial, we introduce an angle similarity term:

$$AS = 1 - \frac{|\theta_1 - \theta_2|}{\pi} \tag{3}$$

  where θ_1 and θ_2 are the rotation angles of the bounding boxes.

* **Feature-Based Similarity**
  To integrate visual similarity, we use a deep learning model to extract feature vectors from cropped image patches. We compute cosine similarity:

$$FS = \frac{F_1 \cdot F_2}{||F_1||||F_2||} \tag{4}$$

  where F_1 and F_2 are the feature vectors.

1.9 Conclusion

I have introduced and evaluated a custom bounding box similarity metric for YOLOv5. The metric provided additional insights into object detection performance beyond IoU.

***