Dynamic Shape-IoU : Addressing Scale Variance

Xiaosai Li (s4019369)

Supervisors:
Prof. Dr. M.S.K. Lew & Dr. E.M. Bakker

Bachelor Thesis– Data Science and Artificial Intelligence

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

July 23, 2026

Abstract

Bounding box regression losses in single-stage object detectors apply the same geometric penalty to every object regardless of its absolute size. Shape-IoU introduced size-aware penalty weighting, but controls it with a single static scale parameter chosen once per dataset. This thesis proposes Dynamic Shape-IoU, which replaces the static parameter with a bounded sigmoid factor $\gamma \in (1, 2)$ computed from the ground-truth area of each object, so that small objects receive up to twice the geometric penalty while the loss reduces exactly to the static baseline for large objects. The factor was developed through three documented design iterations, integrated into the YOLOv11-S bounding box regression loss, and evaluated on the MS COCO 2017 dataset against the CIoU baseline, a static Shape-IoU control and three published detectors, using identical hyperparameters and a single-factor ablation design. Under the official COCO evaluation protocol, the dynamic configuration achieved the highest overall accuracy of the locally trained variants and improved small-object precision by 1.1 points over the baseline without degrading large-object precision. Convergence rate and training stability were unchanged. Within this single-seed, single-dataset setup, the results provide indicative support for instance-aware dynamic scaling in bounding box regression.

Contents

1 Introduction

1.1 The Situation

Computer vision has experienced rapid development in recent years, allowing machines to process and interpret complex visual data. Within this field, object detection has emerged as an important domain, driven by a growing demand in real-world applications such as healthcare, autonomous driving and security monitoring [8]. A primary driver of this progress has been the development of efficient single-stage algorithms, most notably the You Only Look Once (YOLO) architecture [19]. By predicting spatial coordinates and classification probabilities in a single forward pass through a Convolutional Neural Network (CNN), YOLO bypassed the computational bottlenecks of earlier multi-stage models and enabled real-time inference. The trade-off between detection accuracy and computational latency is optimized further by the latest version of the YOLO architecture, including the YOLOv11 [9].

Object localization in YOLO relies on Bounding Box Regression (BBR). The network predicts the center coordinates, width and height of each target object [3, 6]. The spatial accuracy of these predictions is conventionally measured using the Intersection over Union (IoU) metric [2], which computes the ratio between the area of overlap and the area of union of the predicted and ground-truth boxes. During training, this metric is inverted into a loss function, and the localization error is minimized through backpropagation [7]. Standard IoU, however, has an inherent limitation. When the predicted and ground-truth boxes are disjoint, the loss is constant and its gradient is zero, hence the network receives no directional feedback on how to correct the prediction [21]. This zero-gradient condition occurs most frequently in the early phases of training, when predictions rarely overlap their targets, and it slows convergence.

To maintain a usable gradient for disjoint boxes, penalty-based extensions such as Generalized IoU (GIoU), Distance IoU (DIoU) and Complete IoU (CIoU) were introduced [21, 31, 32]. These metrics improved spatial alignment by penalizing center-point distance and relative aspect ratio. However, they also apply the same geometric constraints to every bounding box, regardless of the absolute scale and shape of the object. Zhang and Zhang [28] addressed this with Shape-IoU, a metric that computes decoupled, shape-aware penalty terms from the dimensions of the ground-truth object. In their experiments, Shape-IoU outperformed existing regression losses across several datasets. However, it relies on a static scale parameter, selected according to the standard object scale of the training dataset. In scenes where object sizes vary drastically within a single frame, a single static value cannot fit all instances, which introduces a scaling bias.

To overcome this limitation, this thesis proposes a Dynamic Shape-IoU metric. The static, dataset-level scale parameter is replaced by a bounded sigmoid function of the ground-truth bounding box area, so that the penalty weight adapts to each object instance. The proposed metric is integrated into the YOLOv11 bounding box regression loss and evaluated on the MS COCO 2017 dataset [13] against the standard YOLOv11 baseline, static Shape-IoU, and recent detection architectures.

1.2 Contributions

The main contributions of this thesis are as follows:

- The mathematical formulation of the Dynamic Shape-IoU regression metric, which uses a bounded sigmoid scaling factor to adaptively penalize bounding box scale variance without inducing gradient instability.
- The integration of the proposed dynamic metric into the YOLOv11 architecture [9], demonstrating numerical stability and computational viability under mixed-precision training.
- A comparative evaluation of the dynamic metric against the YOLOv11 baseline, static Shape-IoU, YOLOv10-S [24], RT-DETR-L [30] and RTMDet-S [15] on the MS COCO 2017 dataset [13], reporting accuracy both overall and per object scale (small, medium, large).

1.3 Research Question

This research aims to answer the following main question:

Can a dynamic, instance-aware scale factor in the Shape-IoU loss function improve bounding box regression accuracy across multi-scale datasets compared to a static scale parameter?

To address this, the following sub-questions are investigated:

SQ1: Does this dynamic approach improve the performance on small objects without degrading the performance on larger objects?

SQ2: What is the optimal mapping strategy for transforming the ground-truth bounding box area into a dynamic scale coefficient?

SQ3: How does the implementation of a dynamic scale factor affect the convergence rate and training stability compared to the static Shape-IoU baseline?

1.4 Thesis Overview

The remainder of this thesis is structured as follows. Chapter 2 establishes the theoretical background, covering the fundamentals of computer vision, the evolution of single-stage object detectors, and the core architecture of YOLOv11. Chapter 3 reviews related work, detailing the progression of bounding box regression metrics from standard IoU to later geometry-aware penalties, and identifies the literature gap regarding dynamic scale adaptation. Chapter 4 presents the proposed approach, formulating the Dynamic Shape-IoU metric and its integration into the model. Chapter 5 outlines the experimental methodology, defining the MS COCO dataset, the fixed control variables and the evaluation metrics used for benchmarking. Chapter 6 presents and discusses the experimental results, analyzing the performance of the dynamic metric against the baselines with a focus on

object scale and training stability. Finally, Chapter 7 concludes the research and proposes directions for future work. This work was carried out as a bachelor thesis at the Leiden Institute of Advanced Computer Science (LIACS), Leiden University, under the supervision of Prof. Dr. M.S.K. Lew and Dr. E.M. Bakker.

2 Background

This section establishes the technical foundations for the rest of this thesis: how images are represented computationally, how Convolutional Neural Networks process them, how object detection architectures evolved, the structure of the YOLO architecture, and the metrics used to evaluate detection performance. The evolution of the bounding box regression losses themselves is reviewed separately in Section 3.

2.1 Fundamentals of Computer Vision

In computational systems, visual data is processed as high-dimensional discrete matrices. A standard digital color image is defined as a three-dimensional tensor of shape $H \times W \times C$, where H and W represent the spatial height and width in pixels, and C represents the number of color channels (typically $C = 3$ for red, green and blue) [7]. The main objective of computer vision is to extract meaningful semantic structures from these raw pixel arrays. Within this domain, tasks are categorized by their required level of spatial and semantic detail:

- **Image classification**: assigning a single categorical label to an entire image, producing a probability distribution over a predefined set of classes.
- **Object localization**: determining the spatial location of a primary object within the image plane.
- **Object detection**: the combination of both tasks. A detection algorithm must isolate multiple objects within a single frame, assign categorical probabilities to each and define their precise spatial boundaries.

In modern object detection architectures, spatial localization is parameterized by a bounding box, formally defined as a four-dimensional coordinate vector

$$\mathbf{b} = [x_c,\, y_c,\, w,\, h]^T \tag{1}$$

where x_c and y_c represent the geometric center point of the target object, and w and h represent its width and height. To ensure numerical stability during optimization, these coordinates are typically normalized to the range $[0, 1]$, relative to the spatial dimensions of the input image.

Transforming a high-dimensional pixel matrix into a set of accurate bounding box vectors requires models capable of hierarchical spatial feature extraction. This requirement forms the operational foundation of Convolutional Neural Networks (CNNs).

2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) form the foundational architecture for processing high-dimensional spatial tensors such as digital images [17]. Unlike fully connected networks, which flatten input matrices and remove spatial locality, CNNs maintain the hierarchy of visual data using localized mathematical operations. The architecture achieves this through three primary mechanisms: feature extraction via convolution, non-linear activation, and spatial downsampling via pooling.

The core computational component is the convolutional layer. It applies a set of learnable filters (kernels) across the input tensor to extract spatial features. For a two-dimensional image input I and a two-dimensional kernel K, the discrete convolution operation is defined as [7]

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m,\, j-n)\, K(m,n) \tag{2}$$

where $S(i,j)$ is the resulting value in the output feature map. By sliding these kernels across the image, the network learns to identify rudimentary features, such as edges and sharp contours, in the shallow layers, and aggregates them into complex semantic representations in the deeper layers [27].

The convolution output is passed through a non-linear activation function, such as the Rectified Linear Unit (ReLU) [10] or the Sigmoid Linear Unit (SiLU) [1], which allows the network to model non-linear visual patterns. Following activation, pooling layers (such as max-pooling) perform spatial downsampling. By retaining only the maximum activation within a localized neighborhood, pooling reduces the spatial dimensionality of the feature maps. This reduces computational cost and introduces a degree of translational invariance, where features are recognized regardless of minor shifts in their exact pixel location.

The internal parameters θ, specifically the kernel weights, are optimized through backpropagation [7]. During training, a loss function $\mathcal{L}$ computes the error between the network's predictions and the annotated ground-truth targets, and the gradient of this loss is computed with respect to every weight in the network. The parameters are then updated iteratively by an optimization algorithm such as gradient descent:

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta \mathcal{L}(\theta_t) \tag{3}$$

where α is the learning rate and $\nabla_\theta \mathcal{L}$ is the gradient of the loss function [7].

This update rule requires a non-zero gradient. If the derivative of the loss evaluates to zero while the prediction is still incorrect, the parameter update is also zero and the network receives no corrective signal. This property is central to the limitations of IoU-based losses analyzed in Section 3.

Modern networks are commonly trained with mixed precision: most operations are computed in 16-bit floating point (FP16) to reduce memory usage and training time, while precision-critical steps are kept in 32-bit [16]. FP16 has a narrow representable range, so intermediate values that grow very large overflow to infinity, and values very close to zero are rounded away. Loss functions

used under mixed precision must therefore avoid operations whose outputs or derivatives can leave this range. This constraint shapes the design of the loss proposed in Section 4.

2.3 Evolution of Object Detection

Deep learning based object detection is generally categorized into two paradigms: two-stage and single-stage frameworks. The division between the two is defined by how the network handles the dual requirements of spatial localization and semantic classification.

Object detection was initially dominated by two-stage architectures, most notably the R-CNN family [6] and its real-time successor Faster R-CNN [20]. These networks separate localization and classification into two sequential operations. In Faster R-CNN, a Region Proposal Network (RPN) first scans the extracted feature maps and generates a sparse set of candidate bounding boxes (Regions of Interest, RoIs) that potentially contain objects. In the second stage, these regions are cropped, normalized and passed to separate classification and bounding box refinement heads. This approach achieves high precision, but processing thousands of region proposals makes two-stage architectures computationally expensive and generally unsuitable for real-time detection.

Single-stage architectures, pioneered by the original YOLO framework [19], remove the region proposal stage entirely and reframe object detection as a single unified regression problem: the entire input tensor passes through the convolutional backbone exactly once. The spatial dimensions of the image are divided into an $S \times S$ grid. If the geometric center of a ground-truth object falls within a grid cell, that cell becomes responsible for predicting the bounding box coordinates, the objectness confidence score and the class probabilities [19].

By computing all predictions across the full feature map in a single forward pass, single-stage models reach real-time inference speeds. The original YOLO, for example, processed images at 45 frames per second [19]. This efficiency, however, introduces optimization challenges for localization accuracy, particularly under extreme variations in object scale. Because single-stage detectors lack the secondary refinement stage provided by an RPN, spatial precision depends heavily on the network's internal feature aggregation and on its training loss function. As a result, the design of the bounding box regression loss directly influences the localization quality of a single-stage detector.

2.4 The YOLO Architecture

Since its introduction, the YOLO framework has undergone numerous iterations to optimize computational efficiency. The core structure, however, remains consistent and is divided into three functional modules: the Backbone, the Neck and the Head [9] (Figure 1). This design allows the network to separately optimize feature extraction, multi-scale fusion and final spatial regression.

The Backbone is a CNN that processes the raw input image tensor and functions as the primary feature extractor. As the image passes through successive convolutional and pooling layers, the spatial dimensions are reduced while the channel depth increases. This creates a hierarchy of feature maps: shallow layers retain high-resolution spatial detail (important for detecting small

objects), while deep layers capture abstract semantic information (important for recognizing large objects). Modern YOLO variants employ cross-stage partial architectures to maximize gradient flow during this extraction phase while minimizing computational redundancy [25].

The Neck is the component responsible for handling scale variance, an inherent challenge in single-stage detection. To detect both large and small objects within the same frame, YOLO uses multi-scale feature fusion, largely inspired by Feature Pyramid Networks (FPN) [11] and Path Aggregation Networks (PANet) [14]. The Neck extracts feature maps from multiple depths of the Backbone and merges them through top-down and bottom-up pathways. This sets semantic features to precise spatial coordinates, producing feature maps capable of localizing objects across a wide range of sizes.

The Head translates the fused feature maps into final predictions. Modern YOLO variants use a decoupled head, introduced to the YOLO lineage by YOLOX [4] where instead of computing classification and localization through a single dense layer, the feature maps are split into two parallel branches:

- **Classification branch**: outputs the categorical probability distribution for the detected object.
- **Regression branch**: outputs the bounding box vector $\mathbf{b} = [x_c, y_c, w, h]^T$ of Equation 1.

Because bounding box coordinates are geometric quantities, their accuracy cannot be evaluated with classification losses such as cross-entropy. The network instead relies on specialized bounding box regression losses, which measure the geometric agreement between predicted and ground-truth boxes. The gradients of these losses determine how the network learns spatial localization.

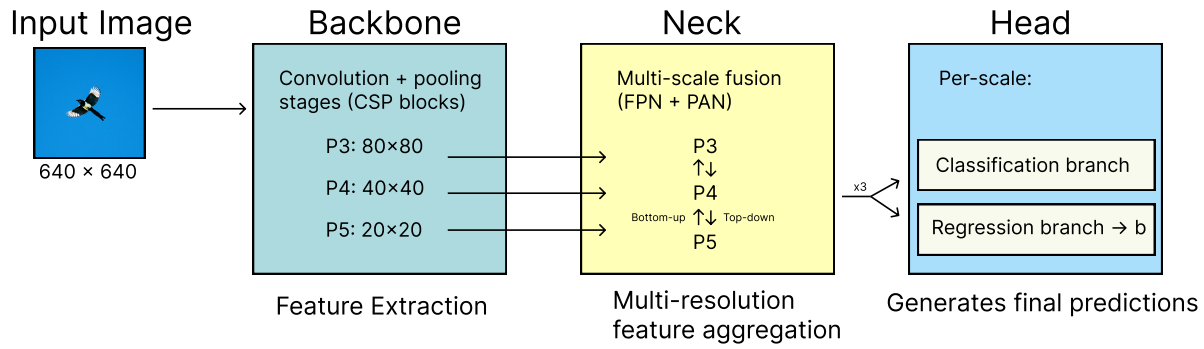

Figure 1: Simplified structure of the YOLO architecture. The Backbone extracts a hierarchy of feature maps at three scales (P3–P5; spatial sizes for a 640×640 input). The Neck fuses them through top-down and bottom-up pathways (FPN + PAN). The decoupled Head outputs, per scale, the class probabilities and the bounding box vector $\mathbf{b}$ of Equation 1. Adapted from [9, 22].

2.5 Evaluation Metrics for Object Detection

Because object detection performs semantic classification and spatial localization simultaneously, evaluating a detector requires metrics that grade both tasks. The standard evaluation protocol for modern architectures relies on Intersection over Union (IoU) and Mean Average Precision (mAP) [2, 13].

2.5.1 Intersection over Union (IoU)

Intersection over Union measures the spatial accuracy of a predicted bounding box against the ground-truth bounding box. It is calculated as the ratio of the overlapping area of the two boxes to their total combined area:

$$IoU = \frac{|B \cap B_{gt}|}{|B \cup B_{gt}|} \tag{4}$$

where B represents the predicted bounding box, B_{gt} the ground-truth bounding box, $\cap$ denotes the intersection (overlapping area) and $\cup$ the union (total combined area) as illustrated in Figure 2.

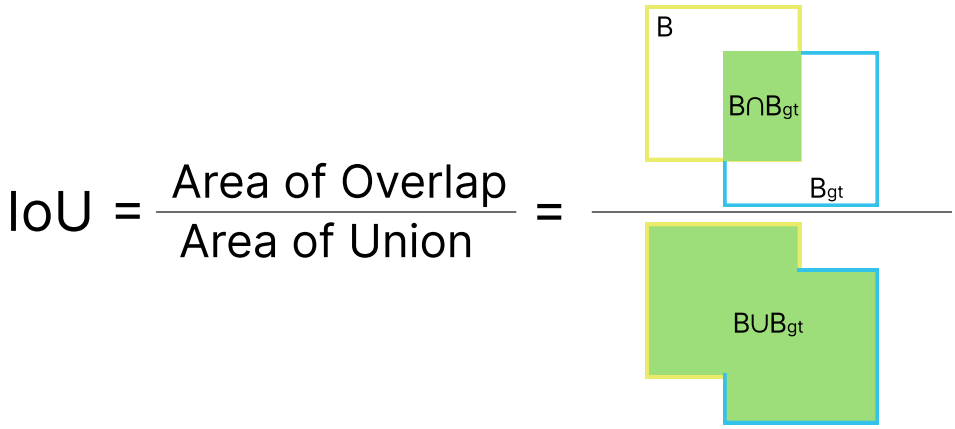

Figure 2: Intersection over Union. The IoU score is the ratio between the area of overlap ($B \cap B_{gt}$) and the area of union ($B \cup B_{gt}$) of the predicted box B and the ground-truth box B_{gt}.

The resulting score is a continuous value between 0 and 1. A score of 0 indicates no overlap and a score of 1 indicates a perfect prediction. In evaluation, a predefined threshold (commonly $IoU \geq 0.5$) classifies each spatial prediction as a True Positive (TP) or a False Positive (FP) [2].

2.5.2 Precision, Recall and Mean Average Precision (mAP)

Once the IoU threshold determines the validity of each predicted box, the network's classification capability is evaluated using Precision and Recall.

Precision measures the network's exactness: the proportion of positive predictions that are correct. It penalizes False Positive detections, such as predicting a class where none exists or

producing a box whose IoU falls below the threshold:

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

Recall measures the network's completeness: the proportion of ground-truth objects that the network successfully detects. It penalizes False Negatives (FN), which occur when the network misses an object entirely:

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

Precision and Recall are inversely related: tuning a network to find every object (high Recall) typically increases the number of incorrect predictions (lower Precision). To summarize this trade-off in a single value, Average Precision (AP) is computed as the area under the Precision–Recall curve for a single class [2], illustrated in Figure 3. To evaluate the network across all object categories, the Mean Average Precision (mAP) averages the AP scores over all N classes:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{7}$$

In modern benchmarks, mAP is reported at a fixed threshold of $IoU = 0.5$ (mAP_{50}) and averaged over ten thresholds from 0.5 to 0.95 in steps of 0.05 ($\text{mAP}_{50:95}$), which is the standard protocol of the MS COCO benchmark [13]. These are the primary metrics used to evaluate model accuracy in Section 5.

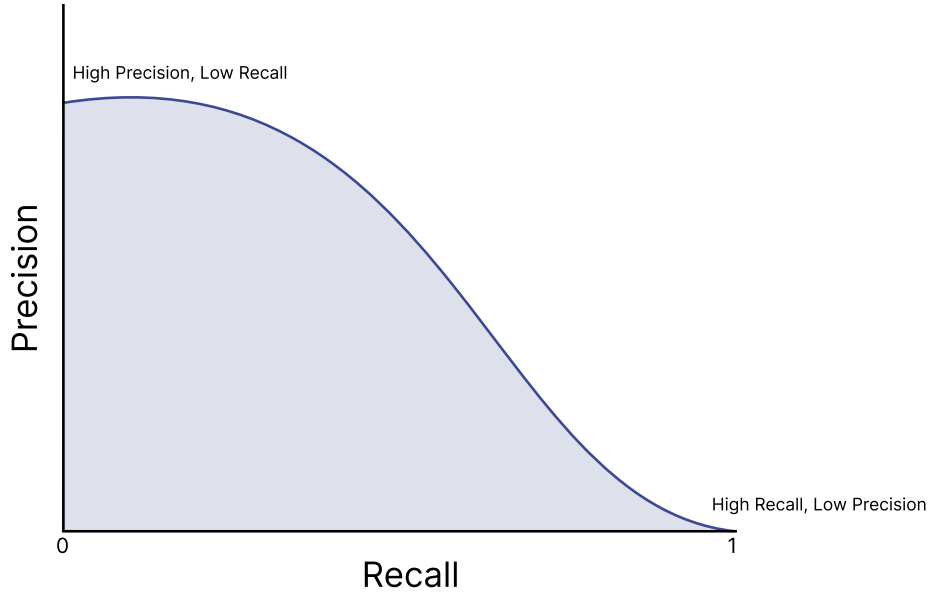

Figure 3: Illustrative precision–recall curve. Average Precision (AP) corresponds to the shaded area under the curve. Raising recall typically lowers precision; a curve that stays high across the full recall range yields an AP close to 1. Measured curves for the evaluated models are shown in Section 6.

3 Related Work

This section reviews the development of bounding box regression losses: the standard IoU loss and its zero-gradient limitation, the penalty-based extensions GIoU, DIoU and CIoU, and the shape- and scale-aware Shape-IoU. It concludes with the static-scaling constraint of Shape-IoU, which defines the gap this thesis addresses.

3.1 Intersection over Union and the Zero-Gradient Problem

Section 2.5 established Intersection over Union as the standard protocol for evaluating spatial predictions. To train a network with it, the evaluation metric must be inverted into a loss function that gradient descent can minimize [7]. Standard IoU evaluates bounding boxes as two-dimensional geometric regions. Let B_p represent the area of the predicted bounding box and B_{gt} the area of the annotated ground-truth target. The standard IoU loss [26] is then defined as

$$\mathcal{L}_{IoU} = 1 - \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|} \tag{8}$$

Under this formulation, a perfect prediction yields an IoU of 1 and a loss of 0, correctly signaling that no further weight updates are necessary. Conversely, a disjoint prediction yields an IoU of 0 and the maximum loss of 1 [21].

3.1.1 The Zero-Gradient Problem

When used as a training loss, $\mathcal{L}_{IoU}$ provides no gradient for disjoint boxes. Whenever the predicted box B_p and the target box B_{gt} do not intersect, the intersection area is zero and the loss takes its maximum value of 1. Because this penalty is constant regardless of the physical distance between the two boxes, the partial derivative of the loss with respect to the bounding box coordinates is zero [21]. This condition is referred to in this thesis as the zero-gradient problem. It is distinct from the standard vanishing gradient problem of deep networks, in which gradients shrink as they propagate backward through many layers [18]. Here the loss surface itself is flat in the disjoint region.

When the gradient is zero, the spatial feedback loop is severed and the network is penalized for an incorrect position but receives no directional information on how to correct it. This stalls the regression optimization, leaving the predicted boxes dependent on stochastic weight updates until they happen to overlap a target. The problem is particularly an issue early in training, when weights are randomly initialized and predictions rarely intersect their targets. This limitation motivated the development of penalty-based regression losses that maintain gradient flow in the absence of overlap.

3.2 Evolution of IoU

To resolve the zero-gradient problem, researchers extended the IoU loss with geometric penalty terms. The common framework appends a penalty $\mathcal{R}$ to the standard loss [31]:

$$\mathcal{L} = 1 - IoU + \mathcal{R}(B_p, B_{gt}) \tag{9}$$

The penalty term $\mathcal{R}$ measures the geometric discrepancy between the two boxes even when they are disjoint, ensuring that the derivative does not evaluate to zero. The structural definition of this penalty evolved through three major iterations, detailed in the following subsections: GIoU, DIoU and CIoU.

3.2.1 Generalized Intersection over Union (GIoU)

The first variant to prevent the zero gradient was Generalized Intersection over Union (GIoU), introduced by Rezatofighi et al. [21]. GIoU calculates the smallest enclosing box, denoted C, that encapsulates both the predicted box B_p and the ground-truth box B_{gt}. Its penalty measures the proportion of empty space between the two boxes relative to this enclosing box:

$$\mathcal{L}_{GIoU} = 1 - IoU + \frac{|C \setminus (B_p \cup B_{gt})|}{|C|} \tag{10}$$

The enclosing box C grows with the distance between B_p and B_{gt}, so the penalty remains active even when $IoU = 0$, providing a continuous gradient that drives the predicted box toward the target.

However, GIoU has a known limitation, identified by Zheng et al. [31]. When one box completely encloses the other, the enclosing box C coincides with the larger box and the penalty term becomes zero, meaning GIoU reduces to the standard IoU. A gradient still exists in this case, because the boxes overlap, but the loss can no longer distinguish *where* inside the target the prediction lies.

Moreover, for disjoint boxes, the penalty does not translate the predicted box towards the target. It expands it until they overlap, which slows convergence.

3.2.2 Distance Intersection over Union (DIoU)

To accelerate convergence and resolve these constraints, Zheng et al. proposed Distance Intersection over Union (DIoU) [31], which explicitly penalizes the Euclidean distance between the geometric center points of the two boxes. Let b_p and b_{gt} represent the center coordinates of the predicted and ground-truth boxes, respectively. The DIoU loss is defined as

$$\mathcal{L}_{DIoU} = 1 - IoU + \frac{\rho^2(b_p, b_{gt})}{c^2} \tag{11}$$

where ρ^2 is the squared Euclidean distance between the two center points and c is the diagonal length of the smallest enclosing box C.

By penalizing the center points directly, DIoU resolves both weaknesses of GIoU. For enclosed boxes, the penalty stays active because the center points are rarely perfectly aligned. For disjoint boxes, the gradient translates the predicted box toward the target instead of inflating its area, resulting in significantly faster convergence [31]. However, DIoU does not take into account the shape of the bounding box. Two boxes can share the same center point while having very different widths and heights, so center alignment alone does not guarantee a correct boundary.

3.2.3 Complete Intersection over Union (CIoU)

In the same work, Zheng et al. proposed Complete Intersection over Union (CIoU) [31], elaborated further in [32], which appends an aspect ratio penalty to the DIoU formulation:

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(b_p, b_{gt})}{c^2} + \alpha v \tag{12}$$

The parameter v measures the consistency of the aspect ratio between the target and the prediction:

$$v = \frac{4}{\pi^2}\left(\arctan\frac{w_{gt}}{h_{gt}} - \arctan\frac{w_p}{h_p}\right)^2 \tag{13}$$

and α is a positive trade-off parameter that balances the penalty weight:

$$\alpha = \frac{v}{(1 - IoU) + v} \tag{14}$$

Because it simultaneously optimizes overlap area, center-point distance and aspect ratio consistency, CIoU became the default regression loss in state-of-the-art architectures, including YOLOv8, YOLOv10 and YOLOv11 [9], the primary baseline model of this research.

CIoU does not, however, account for absolute scale. The parameter v depends only on the relative aspect ratio w/h, so the loss cannot distinguish between a large and a small bounding box with the same proportions. By ignoring absolute dimensions, CIoU applies uniform penalties regardless of

an object's physical size, which limits localization refinement for objects of drastically different scales.

Several further IoU variants followed. Zhang et al. proposed EIoU [29], which replaces the aspect ratio term with direct width and height penalties. Gevorgyan introduced SIoU [5], which adds an angle-aware component to the distance penalty. Tong et al. developed Wise-IoU [23], which dynamically re-weights the penalty according to the quality of each prediction, not the geometry of the target. These variants refine how the geometric error is measured or weighted, but none of them conditions the penalty on the absolute size of the ground-truth object. The first loss to do so explicitly was Shape-IoU.

3.3 The Shape-IoU Metric

To address this scaling constraint, Hao Zhang and Shuaijie Zhang introduced Shape-IoU, a regression loss designed to weight the penalty by the absolute shape and scale of the ground-truth box rather than its relative proportions [28].

3.3.1 The Mechanism of Shape-IoU

The core idea of Shape-IoU is to decouple the width and height penalties, weighting each dimension by the shape of the ground-truth target. Zhang and Zhang achieve this by defining shape-aware weighting parameters ww and hh from the ground-truth width w_{gt} and height h_{gt}:

$$ww = \frac{2 \times (w_{gt})^{scale}}{(w_{gt})^{scale} + (h_{gt})^{scale}}, \qquad hh = \frac{2 \times (h_{gt})^{scale}}{(w_{gt})^{scale} + (h_{gt})^{scale}} \tag{15}$$

where *scale* is a non-negative hyperparameter chosen per dataset according to the typical size of its targets. The parameter controls how strongly the target's shape influences the weighting: for $scale = 0$ both weights reduce to 1, and larger values skew the penalty toward the target's dominant dimension.

These weights are embedded into a center-distance penalty, normalized by the squared diagonal c^2 of the smallest enclosing box:

$$dist^{shape} = \frac{hh \cdot (x_p - x_{gt})^2 + ww \cdot (y_p - y_{gt})^2}{c^2} \tag{16}$$

The shape penalty compares the predicted and ground-truth dimensions directly:

$$\Omega^{shape} = \sum_{t \in \{w,h\}} \left(1 - e^{-\omega_t}\right)^{\theta}, \qquad \theta = 4 \tag{17}$$

with the relative dimensional errors

$$\omega_w = hh \cdot \frac{|w_p - w_{gt}|}{\max(w_p, w_{gt})}, \qquad \omega_h = ww \cdot \frac{|h_p - h_{gt}|}{\max(h_p, h_{gt})} \tag{18}$$

The complete Shape-IoU loss combines both penalties:

$$\mathcal{L}_{Shape\text{-}IoU} = 1 - IoU + dist^{shape} + \frac{1}{2}\,\Omega^{shape} \tag{19}$$

By embedding the target's absolute dimensions into the penalty, Shape-IoU optimizes targets differently according to their shape and size (tall, narrow targets differently than short, wide targets). In the evaluation of Zhang and Zhang, this weighting produced faster convergence and more precise localization than CIoU and SIoU across multiple detection and segmentation benchmarks [28].

3.3.2 The Constraint of Static Scaling

Although the decoupled weighting improves localization, the Shape-IoU formulation is constrained by its use of a static *scale* parameter. The value is chosen once per dataset, according to the typical size of the targets it contains, and is then applied to every object instance identically.

This dataset-level parameter introduces a scaling bias in environments where object areas vary drastically within single images. For example, a distant pedestrian occupying a few dozen pixels receives exactly the same scale treatment as a nearby vehicle occupying half the frame. Applying one fixed setting to both extremes of the size spectrum prevents the loss from adapting its penalty to individual object instances.

Eliminating this bias requires replacing the static scalar with a function that computes instance-specific penalty weights from the absolute geometric area of each individual ground-truth box. Addressing this gap is the primary motivation behind the Dynamic Shape-IoU loss proposed in this thesis.

4 Approach

This section develops the Dynamic Shape-IoU loss. It first defines the requirements that the dynamic scaling function must satisfy, then documents the three design iterations that led to the final formulation: an unbounded logarithmic mapping, a bounded sigmoid applied inside the penalty weights, and the final bounded-sigmoid penalty multiplier. This section covers only the design and its offline verification. training results are reported in Section 6.

4.1 Design Requirements

Replacing the static *scale* parameter of Shape-IoU with an instance-dependent function requires the function to adhere to the following points:

1. **Bounded output:** The function's output and its intermediate values must remain within the representable range of FP16 mixed-precision training (Section 2.2) [16].
2. **Inverse size dependence:** Small objects must receive a higher penalty weight than large objects, matching the goal of SQ1.
3. **Resolution in the small-object regime:** The function must vary meaningfully across the MS COCO small/medium boundary (roughly 16–96 pixels) [13], where SQ1 is evaluated.
4. **Single-factor reduction to the static baseline:** With the dynamic component disabled, the loss must reduce exactly to the static Shape-IoU baseline, so that any performance difference is attributable to dynamic scaling alone.

5. **Independence from the detection level:** Object size must be measured in image pixels, not in the grid units of the feature-pyramid level that predicts the object, so that the same physical object receives the same weight regardless of which level detects it.

6. **No gradient through the scaling factor:** The factor is computed from ground-truth dimensions only, which carry no gradient. it rescales the penalty without adding a new backpropagation path.

4.2 First Iteration: Logarithmic Mapping

Because bounding box areas in detection datasets span several orders of magnitude, a natural logarithm was initially hypothesized as the mapping to compress this range. Following the logarithmic encoding of box dimensions that are standard in bounding box regression targets [6, 20]. The ground-truth area was normalized against the input canvas and mapped through a scaled logarithm with a lower clamp:

$$scale_{dyn} = \max(\beta \cdot \ln(area_{norm} + \epsilon) + 2,\ 1)\,, \qquad \beta = 8 \tag{20}$$

which was then used as a divisor inside exponential penalty weights with a fractional exponent $\theta = 0.5$.

Training with this mapping destabilized. The bounding box loss grew to the order of 3×10^3 and the model did not converge, consistent with the exploding-gradient failure mode [18]. Subsequent analysis identified the logarithm's unbounded negative output for small normalized areas as the most plausible cause. For the box sizes the loss function actually receives, $\ln(area_{norm})$ is strongly negative, so the mapping either produces extreme scale values or, where the clamp intervenes, is pinned at the constant floor of 1. The result is unstable dynamics on one side of the clamp and no dynamics on the other.

Two conclusions were drawn from this iteration. First, the mapping must be *intrinsically* bounded. A hard clamp on an unbounded function either fails to prevent the instability or removes the size-dependent variation. Second, this variant had also replaced the full Shape-IoU penalty with a distance-only term. A difference measured against the static baseline would therefore not have been attributable to dynamic scaling alone, which reinforces requirement 4 in the list above.

4.3 Second Iteration: Sigmoid on Raw Area

Following the conclusion of the first iteration, the logarithm was replaced by a sigmoid $\sigma(x) = 1/(1 + e^{-x})$, which is inherently bounded in $(0, 1)$ [7], applied to the normalized ground-truth area as a dynamic divisor inside the penalty weights:

$$scale_{dyn} = 1 + \sigma(10 \cdot area_{norm}), \qquad ww_{\sigma} = 2\left(1 - e^{-w_{norm}/scale_{dyn}}\right)^{\theta}, \quad \theta = 0.5 \tag{21}$$

with hh_{σ} defined from h_{norm}. These weights replaced the ratio weights of Section 3.3.1. This variant trained stably, confirming the bounded mapping as the correct family. However, a subsequent verification of the implementation against the actual training call path revealed three defects.

First, a unit mismatch. The loss function receives box coordinates in feature-grid units (pixels divided by the detection level's stride), while the normalization assumed image pixels. At runtime, $area_{norm}$ therefore peaked at approximately 0.016 instead of 1.0, and $scale_{dyn}$ spanned only $[1.500, 1.539]$ over the entire physically possible object range. The dynamic mechanism was effectively constant, and the trained model behaved as a second static variant. The same mismatch made the weighting depend on which pyramid level detected the object. Breaking requirement 5.

Second, the direction was inverted: the weights of Equation 21 increase with object size, assigning larger penalties to larger objects, which is the opposite of requirement 2.

Third, raw area is quadratic in object size, so even with correct units the sigmoid input barely varies below approximately 100 pixels, meaning a 16-pixel and a 64-pixel object would receive nearly identical factors, breaking requirement 3.

In addition, the sigmoid variant restructured the penalty itself relative to the static baseline. It altered the weight form, the axis pairing, the distance normalization and the shape-cost form. As with the first iteration, a performance difference measured against the static Shape-IoU baseline could therefore not have been attributed to dynamic scaling alone.

This iteration yielded three conclusions: the sigmoid input must be the relative side length $\sqrt{area_{norm}}$, a square-root transform comparable to the one YOLO applies to box dimensions to increase resolution for small objects [19]. The mapping must decrease with size, and the dynamic term must rescale exactly the penalty of the static baseline to satisfy requirement 4.

4.4 Final Formulation: Bounded-Sigmoid Penalty Multiplier

In the final design, boxes are first converted to image-pixel units at the loss call site (multiplying by the per-anchor stride), and the normalized area of the ground-truth box is computed against the 640×640 input canvas:

$$area_{norm} = \frac{w_{gt} \cdot h_{gt}}{640^2}, \qquad area_{norm} \in [0, 1] \tag{22}$$

The dynamic factor γ is a bounded sigmoid of the relative side length, decreasing with object size:

$$\gamma(area_{norm}) = 1 + \sigma(k \cdot (\tau - \sqrt{area_{norm}})), \qquad k = 25, \quad \tau = 0.1 \tag{23}$$

so that $\gamma \in (1, 2)$. The threshold $\tau = 0.1$ places the sigmoid midpoint at objects of 64 pixels (0.1×640), centering the transition across the COCO small/medium boundary, and $k = 25$ controls its steepness. The complete loss multiplies the geometric penalty of the static baseline by this factor:

$$\mathcal{L}_{Dynamic} = 1 - IoU + \gamma(area_{norm}) \cdot \left(dist^{shape} + \frac{1}{2}\, \Omega^{shape} \right) \tag{24}$$

where $dist^{shape}$ and Ω^{shape} are computed exactly as in the static Shape-IoU of Section 3.3.1 [28] with $scale = 0$ (uniform weights $ww = hh = 1$). This form follows the loss-modulation paradigm introduced by Lin et al. for focal loss, which rescales a base loss by a bounded factor computed per

Object size (square, px)	8	32	64	128	320
γ	1.90	1.78	1.50	1.08	1.00

Table 1: Dynamic factor γ (Equation 23) for square ground-truth objects of increasing size. Small objects receive up to twice the geometric penalty. For large objects the loss approaches the static baseline.

instance [12]. In focal loss the factor is driven by prediction confidence, whereas in Equation 24 it is driven by ground-truth size. Table 1 shows the resulting factors across object sizes.

This formulation satisfies all six points. The sigmoid bounds γ in $(1, 2)$, keeping the penalty within twice the static baseline's magnitude (the same order as CIoU), so no loss-weight retuning is required (requirement 1). γ decreases with size (requirement 2), and its transition is centered in the small-object size range (requirement 3). Setting $\gamma \equiv 1$ recovers the static baseline, making the static-vs-dynamic comparison a single-factor ablation (requirement 4). The pixel-unit conversion removes the dependence on the detection level (requirement 5). Finally, γ is computed from ground-truth dimensions produced by the label assigner, which carry no gradient, so the factor rescales the existing gradients without introducing a new backward path (requirement 6). Numerical safety clamps on the forward value guard against degenerate zero-size boxes under FP16.

Before training, the implementation was verified through the full training call path: γ spans $[1.00, 1.90]$ over COCO-sized objects and decreases with size (the dynamic loss equals γ times the static penalty to within floating-point precision). The gradients remain finite on degenerate boxes. A subsequent five-epoch training run confirmed the same behavior on real training batches: γ spanned $[1.000, 1.872]$ with no numerical instability. Overall, the final formulation retains the bounded-sigmoid family identified in the second iteration while satisfying all six design requirements.

5 Methodology

This section defines the experimental setup used to evaluate the Dynamic Shape-IoU loss: the dataset, the trained model configurations, the training protocol, the benchmark models and their source, and the evaluation protocol.

5.1 The Dataset

All experiments were conducted on the MS COCO 2017 dataset [13], consisting of 118,287 training images, 5,000 validation images and approximately 860,000 annotated object instances across 80 categories. COCO was selected for two reasons. First, it is the standard evaluation benchmark for object detection, so the reported values are directly comparable with published work, including the original Shape-IoU evaluation [28]. Second, its objects span the full size spectrum, and objects of very different sizes co-occur within single images. This size variance is the condition under which a single dataset-level *scale* parameter cannot fit all instances (Section 3.3.2), which makes it the appropriate test environment for an instance-level scale factor. The size annotations additionally support the size-stratified evaluation that SQ1 requires.

5.2 Model Configurations

Three YOLOv11-S configurations were trained, differing only in the IoU variant of the bounding box regression loss:

- **Baseline (CIoU)**: the default YOLOv11 regression loss [9].
- **Static Shape-IoU**: the loss of Zhang and Zhang (Section 3.3.1) [28] at the official implementation's default $scale = 0$. At this setting the shape weights reduce to $ww = hh = 1$. This configuration represents Shape-IoU with neutral weighting and not a dataset-tuned scale value.
- **Dynamic Shape-IoU**: the final formulation of Section 4.4.

As established in Section 4.4, the dynamic loss reduces to the static configuration for $\gamma \equiv 1$, so the static and dynamic runs form a single-factor comparison. All other components of the training pipeline are identical across the three configurations: the label assigner, the classification loss, the distribution focal loss, the loss gains and the data augmentation. The IoU variant is exchanged at a single line in the loss module of the training framework.

5.3 Training Protocol

Each configuration was trained for 100 epochs at an input resolution of 640×640 pixels with a batch size of 16, under mixed-precision training (Section 2.2). Optimizer selection was left to the framework's automatic mode, which selects stochastic gradient descent for runs of this length, with an initial learning rate of 0.01 decaying linearly to 0.0001, momentum 0.937, weight decay 0.0005 and three warmup epochs. Mosaic augmentation is disabled for the final 10 epochs (close_mosaic = 10). Each run stores its complete training configuration. Each configuration was trained once, with a fixed random seed of 0 and deterministic mode enabled. Training was performed on a NVIDIA GeForce RTX 4050 laptop GPU, at approximately 42 hours per 100-epoch run.

All configurations were initialized from the official COCO pretrained YOLOv11-S checkpoint (9.4 million parameters). Training from random initialization was rejected as it requires substantially longer schedules and introduces larger run-to-run variance. Initializing every configuration from the same checkpoint gives all loss variants an identical starting capability. The bias direction of this choice is stated in Section 5.6.

5.4 Benchmark Models

Three published real-time detectors are included to position the YOLOv11 results within the current accuracy landscape:

- **YOLOv10-S** (7.2 million parameters): the previous YOLO generation, which removes Non-Maximum Suppression from inference through a dual-assignment training strategy [24]. Its inclusion isolates generational architecture changes within the YOLO lineage.

- **RT-DETR-L** (32 million parameters): a transformer-based real-time detector with a hybrid encoder [30]. Its larger capacity makes it an upper accuracy reference from outside the YOLO family.
- **RTMDet-S** (8.9 million parameters): a single-stage detector of comparable size from the MMDetection ecosystem [15].

YOLOv10-S and RT-DETR-L were not trained locally. Their official pretrained checkpoints were evaluated on the COCO 2017 validation split under the protocol of Section 5.5. RTMDet-S could not be executed locally. Installing its native MMDetection framework on the available machines failed repeatedly. Its values are therefore taken from Lyu et al. [15], who evaluate RTMDet-S at the same 640×640 resolution on the same COCO 2017 validation split, after a 300-epoch training schedule (three times the schedule used here).

5.5 Evaluation Protocol

Accuracy is reported as mAP_{50} and $\text{mAP}_{50:95}$, as defined in Section 2.5.2. $\text{mAP}_{50:95}$ is the primary metric. Because it averages over strict overlap thresholds, it is the most sensitive measure of the exact box placement (the property the compared loss functions change). For each trained configuration, the best-performing checkpoint on the validation split was selected. Its detections were exported in the COCO results format and scored with the official COCO evaluation implementation (pycocotools) against the validation annotations [13].

To address SQ1, the evaluation is additionally stratified by object size, following the COCO size categories [13]:

- AP_S: instances with an area below 32^2 pixels;
- AP_M: instances with an area between 32^2 and 96^2 pixels;
- AP_L: instances with an area above 96^2 pixels.

mAP can be computed by two different tools: the training framework's built-in validator and the official pycocotools implementation. The two produce slightly different values for the same model. To keep all reported numbers comparable, every accuracy value in this thesis comes from pycocotools, which is also the tool behind the published values of the benchmark models.

For SQ3, the per-epoch training and validation box-loss curves and the per-epoch validation mAP of the static and dynamic runs were recorded and are compared in Section 6.

Table 2 additionally lists the parameter count and FLOPs of each architecture, taken from their official specifications. These are properties of the architecture and are independent of the training loss.

5.6 Fairness and Scope of the Comparison

Three properties of this setup constrain what the results can claim. They are stated here and taken into account in Section 6.

Pretrained initialization on the target dataset. The YOLOv11 configurations are fine-tuned on the same dataset their initialization checkpoint was pretrained on, and that checkpoint was pretrained with the CIoU loss. This setting compresses differences between loss functions and, if anything, favors the CIoU baseline, which continues training under the loss it was pretrained with.

Single run per configuration. Each configuration was trained once (Section 5.3). Seed-to-seed variance was not measured, so differences of a few tenths of a point, particularly in the size-specific metrics, cannot be distinguished from run-to-run noise with certainty.

Mixed benchmark provenance. Table 2 mixes three provenance classes: locally trained models, official checkpoints evaluated locally, and literature values. The externally trained models were optimized by their authors under longer and different training schedules. The benchmark rows therefore provide context for the absolute accuracy level. The controlled comparison of this thesis is exclusively the one among the three locally trained YOLOv11 configurations.

6 Results and Discussion

This section reports the experimental results and interprets them, addressing the main research question and each sub-question in turn.

6.1 Overall Accuracy

Table 2 reports the accuracy of all evaluated models. Among the three locally trained configurations, the Dynamic Shape-IoU run reaches the highest overall accuracy: 46.0 $mAP_{50:95}$ and 63.0 mAP_{50}. Relative to the static Shape-IoU control, the dynamic factor adds 0.2 points of $mAP_{50:95}$ and 0.5 points of mAP_{50}. Relative to the CIoU baseline, it adds 0.1 and 0.4 points, respectively. The static configuration alone scores 0.1 points below the baseline on both overall metrics.

Model	mAP_{50}	$mAP_{50:95}$	AP_S	AP_M	AP_L	Params (M)	FLOPs (G)	$mAP_{50:95}$/FLOPs	AP_S/FLOPs
YOLOv11-S, CIoU (baseline)	62.6	45.9	25.6	51.0	63.1	9.4	21.5	2.135	1.191
YOLOv11-S, Shape-IoU (*scale* = 0)	62.5	45.8	26.4	50.9	63.2	9.4	21.5	2.130	1.228
YOLOv11-S, Dynamic Shape-IoU (ours)	63.0	46.0	26.7	51.2	63.1	9.4	21.5	2.140	**1.242**
YOLOv10-S°	62.0	46.2	26.8	51.0	63.6	**7.2**	21.6	2.139	1.241
RT-DETR-L°	**70.6**	**52.4**	**33.6**	**57.0**	**70.5**	32	110	0.476	0.305
RTMDet-S•	61.9	44.6	-	-	-	8.9	**14.8**	**3.014**	-

Table 2: Accuracy on the COCO 2017 validation split under the evaluation protocol of Section 5.5 (values in %). °Official pretrained checkpoint, evaluated locally. •Values reported by Lyu et al. [15]. Parameter counts and FLOPs are official specifications and independent of the training loss. Dashes mark values the cited source does not report.

In Table 2 FLOPs denotes the number of floating-point operations required for one forward pass at the 640×640 input resolution in billions. The two rightmost columns divide $\text{mAP}_{50:95}$ and AP_S by FLOPs (in %/GFLOPs), expressing the accuracy achieved per unit of computation. Bold marks the best value in each column. For parameters and FLOPs, where lower is better, bold marks the lowest value.

The margins are small. Within this setup, however, the dynamic configuration is the only variant that improves on the baseline on every overall metric, and every difference in the single-factor pair points in the direction predicted by the design. A consistent direction across all reported metrics is less likely to result from seed noise than an isolated difference would be. With one run per configuration (Section 5.6), these results should be read as an indicative trend and not a conclusive effect size.

6.2 Accuracy by Object Size (SQ1)

The size-stratified columns of Table 2 show the following pattern. Small-object precision rises from 25.6 (baseline) to 26.4 (static) and 26.7 (dynamic). Medium-object precision is 51.0, 50.9 and 51.2, respectively. Large-object precision is 63.1, 63.2 and 63.1, a spread of 0.1 points.

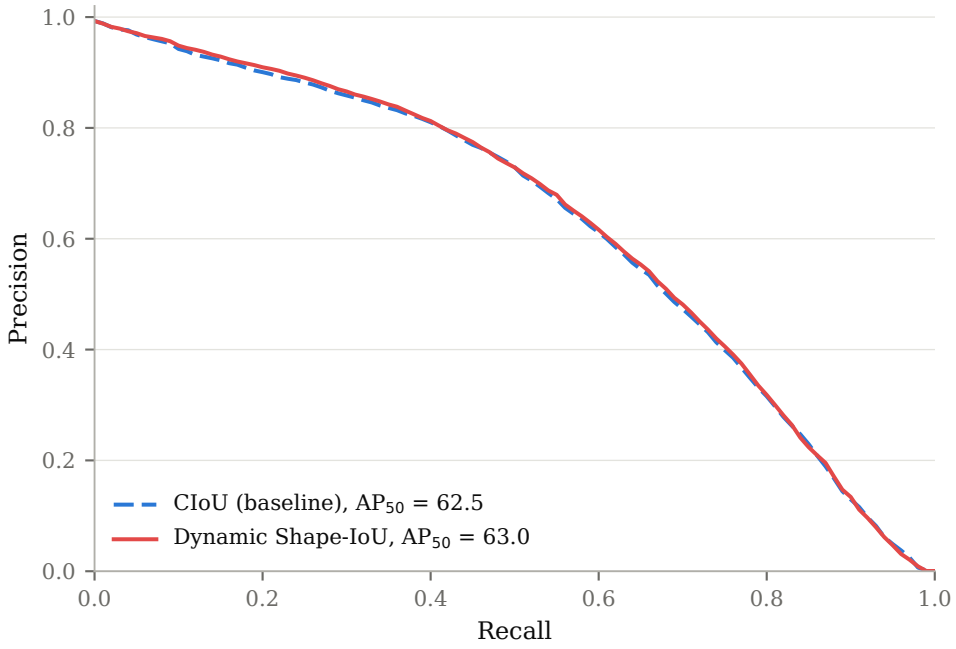

Figure 4: Measured precision–recall curves of the baseline and the Dynamic Shape-IoU configuration on the COCO 2017 validation split at $IoU = 0.5$. The dynamic curve matches or exceeds the baseline over most of the recall range.

Three observations follow. First, both Shape-IoU variants improve small-object precision over CIoU. The static structure alone adds 0.8 points, and the dynamic factor adds a further

0.3 points on top of it. Second, the dynamic configuration is the only variant that also improves medium-object precision (0.2 points over the baseline), whereas the static configuration sits 0.1 points below it. Third, large-object precision is largely unchanged across all three configurations.

A plausible interpretation of this pattern is that the Shape-IoU penalty structure itself already favors small-object localization relative to CIoU. The γ factor adds a further, smaller improvement that is attributable to dynamic scaling alone through the single-factor design. The flat large-object results are consistent with the bounded construction, in which γ approaches 1 for large objects and the loss reduces to the static baseline (Section 4.4). The 1.1-point small-object gain over the baseline is the largest accuracy difference among the local runs. The 0.3-point increment over the static control is within the range that seed noise can produce.

Overall, within this dataset and setup, SQ1 is answered in the affirmative direction: small-object accuracy improves and large-object accuracy does not degrade. The attribution of the improvement specifically to dynamic scaling rests on the smaller 0.3-point single-factor increment, and is therefore indicative not conclusive.

Figure 4 shows the measured precision–recall curves of the baseline and the dynamic configuration at $IoU = 0.5$. The dynamic curve lies on or above the baseline curve for 84% of the evaluated recall range, with the largest separation at low recall (approximately 0.6 points of precision for recall below 0.3).

6.3 Training Behavior (SQ3)

Figure 5 shows the per-epoch validation mAP of the three configurations and Figure 6 their box-loss curves. All three runs converge smoothly. The validation mAP of the dynamic run drops briefly during the three warmup epochs (from 0.340 to a minimum of 0.321 at epoch 3) and then rises steadily to its best value at epoch 96, without oscillation or collapse. The baseline and static runs show the same warmup dip. The training box loss of the dynamic run is approximately twice that of the baseline throughout training (values around 1.6 to 2.0 versus around 1.0). All runs show a step change at epoch 91, where mosaic augmentation is switched off (Section 5.3). No non-finite loss values occurred in any run.

The per-epoch curves are produced by the training framework's internal validator and are used here only to compare convergence behavior. The accuracy comparison of Section 6.1 and Section 6.2 rests exclusively on the protocol of Section 5.5, and the two sets of values are not directly comparable. The elevated absolute loss of the dynamic run is the designed consequence of the multiplier. $\gamma \in (1, 2)$ scales the geometric penalty, so the dynamic run optimizes a larger-valued objective while representing the same geometry. Absolute loss values are therefore not comparable across loss functions, and the roughly factor-of-two offset indicates that the mechanism operated at its intended magnitude. The indicators that are comparable across runs (monotone validation loss, smooth mAP growth, absence of non-finite values) show no difference between the dynamic run and the static or baseline runs. This stability contrasts with the unbounded first design iteration, which diverged (Section 4.2). The computational overhead of the factor is limited by construction to a few element-wise operations per assigned ground-truth box, with no new gradient path (Section 4.4) A

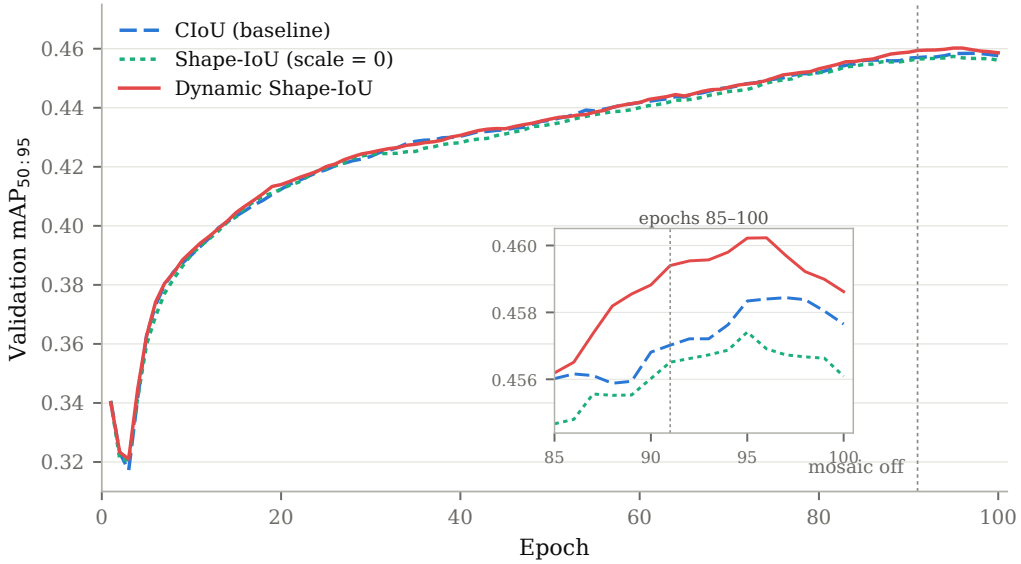

Figure 5: Per-epoch validation $\mathrm{mAP}_{50:95}$ of the three YOLOv11 configurations, as reported by the training framework's validator. All three runs converge smoothly. The step at epoch 91 coincides with mosaic augmentation being disabled.

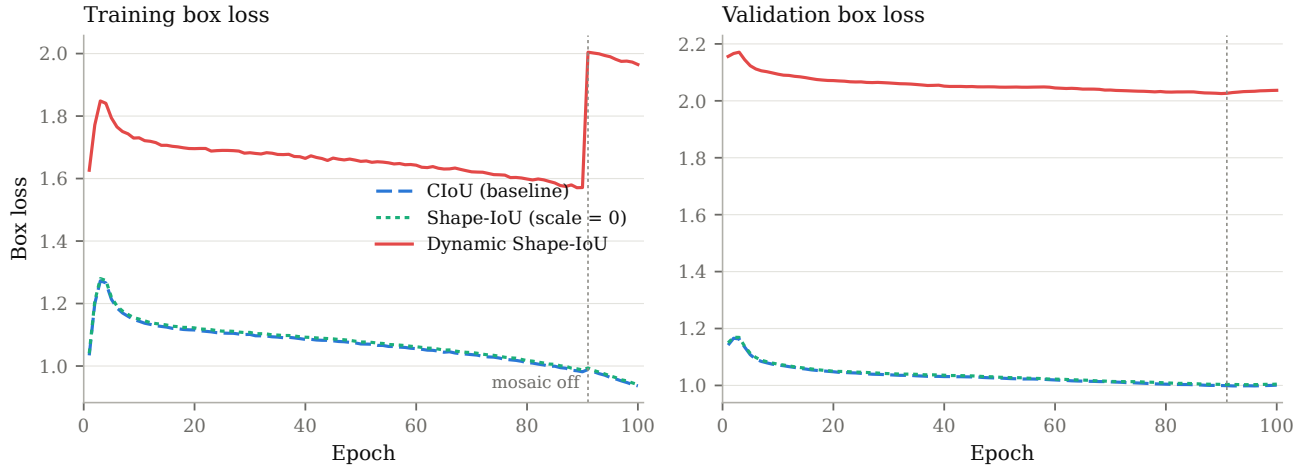

Figure 6: Training (left) and validation (right) box loss per epoch. The dynamic run optimizes a loss that is by construction up to twice the static penalty ($\gamma \in (1, 2)$), so its higher absolute values do not indicate instability. Absolute loss values are not comparable across loss functions.

separate runtime measurement was not performed.

Overall, under the evaluated conditions, the dynamic factor left convergence rate and training stability unchanged relative to the static baseline, which answers SQ3.

6.4 Mapping Strategy (SQ2)

The evidence on SQ2 comes from the design iterations of Section 4 together with the training outcomes. Of the two mapping families implemented, the unbounded logarithmic mapping destabilized training, whereas the bounded sigmoid trained stably. The corrected bounded-sigmoid multiplier additionally satisfies the direction and resolution requirements and produced the accuracy pattern of Section 6.1 and Section 6.2. The results support the bounded sigmoid of the relative side length as a mapping that satisfies the stability and resolution requirements of Section 4.1, not as the optimal mapping.

6.5 Comparison with Benchmark Models

In the context of the benchmark models, the dynamic configuration is competitive with detectors of comparable size: it scores 0.2 points below YOLOv10-S and 1.4 points above the literature value of RTMDet-S on $\text{mAP}_{50:95}$. On small objects, the dynamic configuration (26.7 AP_S) is on par with YOLOv10-S (26.8). RT-DETR-L scores 6.4 points higher overall, at more than three times the parameter count and five times the FLOPs. Following Section 5.6, these rows provide context for the absolute accuracy level and are not evidence of a ranking: the external models were trained by their authors under different and longer schedules, and the RTMDet-S values originate from its publication.

The normalized columns of Table 2 add the computational perspective. RTMDet-S achieves the highest overall accuracy per unit of computation. This is due to its low FLOPs count. The accuracy advantage of RT-DETR-L, however, disappears under normalization (0.476 against values above 2 for every smaller detector). On the small-object measure, the dynamic configuration achieves the highest normalized value (1.242 AP_S per GFLOP), approximately equal to YOLOv10-S (1.241). No comparison with RTMDet-S is possible, as its AP_S is not reported.

6.6 Qualitative Examples

Figure 7 shows detections of the baseline and the dynamic configuration on validation images containing many small objects, at a fixed confidence threshold. In the harbor scene, the dynamic configuration additionally detects two small objects the baseline misses: the distant boat at the left image edge and the person at the right shoreline. In the kite-field scene, the two outputs differ only in isolated detections near the horizon.

Taken together, three findings emerge from this section. First, the dynamic configuration is the only locally trained variant that improves on the CIoU baseline on every overall metric, with the largest gain on small objects and no degradation on large objects. Second, the single-factor comparison attributes a consistent but small increment specifically to dynamic scaling, on top of a

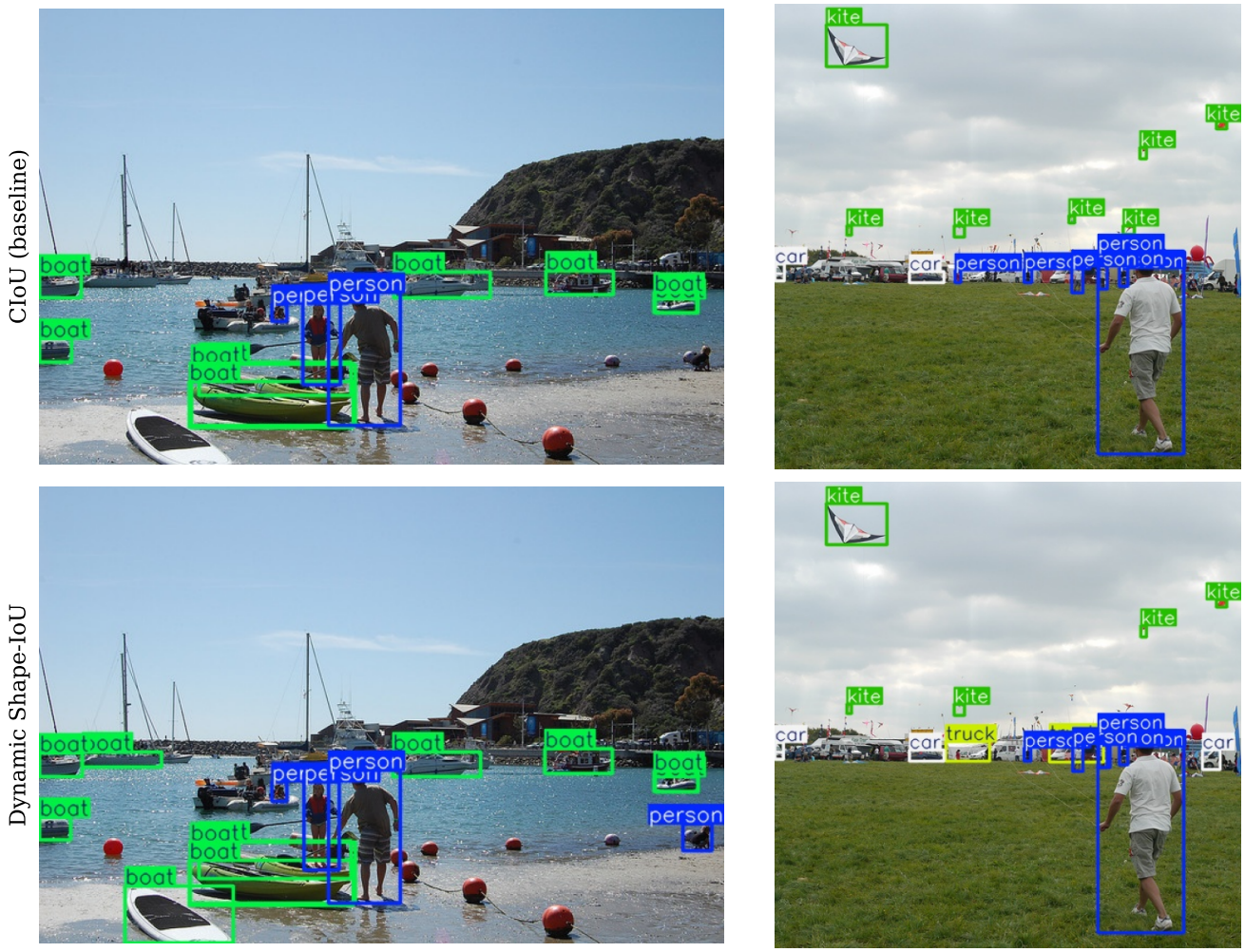

Figure 7: Detections of the CIoU baseline (top) and the Dynamic Shape-IoU configuration (bottom) on COCO 2017 validation images with many small objects (images 000000160728 and 000000432898, confidence threshold 0.25). The examples are illustrative and were selected to visualize small-object behavior. They are not evidence of aggregate performance.

larger gain contributed by the Shape-IoU structure itself. Third, the bounded multiplier achieved this without any measurable cost to convergence or stability. These findings should be interpreted as specific to this dataset, architecture and training setup: one dataset, and a pretrained initialization that compresses differences between loss functions (Section 5.6). The concluding answers to the research questions are given in Section 7.

7 Conclusion

This thesis investigated whether the static scale parameter of the Shape-IoU bounding box regression loss can be replaced by a dynamic, instance-aware factor. Our novel contribution is a bounded sigmoid multiplier $\gamma \in (1, 2)$, computed from the ground-truth area of each object. This assigns small objects up to twice the geometric penalty while the loss reduces exactly to the static Shape-IoU baseline for large objects. The multiplier introduces no new gradient path and left training stability unchanged. It was developed through three documented design iterations, integrated into the YOLOv11-S loss. It was evaluated on MS COCO 2017 against the CIoU baseline and a static Shape-IoU control under a single-factor ablation design. This section answers the research questions and outlines future work.

Within the evaluated setting, the answer to the main research question is a qualified yes. The dynamic configuration improved on the static control on every reported metric (0.2 points of $\text{mAP}_{50:95}$, 0.5 points of mAP_{50}, 0.3 points of AP_S) and was the only variant to improve on the CIoU baseline on every overall metric. The margins are small and derive from one training run per configuration, so they constitute solely as an indicative trend.

Regarding SQ1, small-object precision improved by 1.1 points over the baseline while large-object precision remained unchanged, matching the intended behavior of the size-decreasing factor. The larger share of this gain appears already with the static Shape-IoU structure. The share attributable to dynamic scaling alone is the 0.3-point increment over the static control.

Regarding SQ2, no optimal mapping can be identified from this study. Of the mappings implemented, the unbounded logarithm destabilized training, and only the corrected bounded sigmoid of the relative side length was evaluated end to end. The supported claim is therefore that a bounded, size-decreasing mapping with resolution in the small-object regime satisfies the design requirements of Section 4.1. Whether this mapping is optimal within that family remains open.

Regarding SQ3, the dynamic factor left convergence rate and training stability unchanged. All runs converged smoothly, and the higher absolute loss of the dynamic run is the designed consequence of the multiplier rather than an instability.

These conclusions are bounded by the constraints of Section 5.6: one dataset, one architecture, and a COCO-pretrained initialization that compresses differences between loss functions.

Future work would follow directly from these limits. First, repeating the dynamic and static runs under additional random seeds would establish whether the observed margins persist beyond

run-to-run noise. Second, evaluating alternative bounded mappings (varying the midpoint τ and slope k, or replacing the sigmoid with a linear function of the relative side length) would allow SQ2 to be answered comparatively. Third, training from random initialization, or on datasets dominated by small objects, would remove the compression introduced by pretraining and test the factor in the regime its design targets. Finally, the multiplier is loss-agnostic by construction: γ rescales a geometric penalty computed from ground-truth size only, so applying it to other regression losses such as CIoU is a direct extension.

References

[1] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

[2] Mark Everingham, Luc Van Gool, Christopher Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88:303–338, 2010.

[3] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[4] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: Exceeding YOLO series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[5] Zhora Gevorgyan. SIoU loss: More powerful learning for bounding box regression. *arXiv preprint arXiv:2205.12740*, 2022.

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[8] Marwa Hameed and Zainab Khalaf. A survey study in object detection: A comprehensive analysis of traditional and state-of-the-art approaches. *Basrah Researches Sciences*, 50, 2024.

[9] Rahima Khanam and Muhammad Hussain. YOLOv11: An overview of the key architectural enhancements. *arXiv preprint arXiv:2410.17725*, 2024.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[11] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2125, 2017.

[12] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[14] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8759–8768, 2018.

[15] Chengqi Lyu, Wenwei Zhang, Haian Huang, Yue Zhou, Yudong Wang, Yanyi Liu, Shilong Zhang, and Kai Chen. RTMDet: An empirical study of designing real-time object detectors. *arXiv preprint arXiv:2212.07784*, 2022.

[16] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations (ICLR)*, 2018.

[17] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[18] Razvan Pascanu, Tomáš Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *arXiv preprint arXiv:1211.5063*, 2012.

[19] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, pages 91–99, 2015.

[21] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 658–666, 2019.

[22] Juan Terven, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González. A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Machine Learning and Knowledge Extraction*, 5(4):1680–1716, 2023.

[23] Zanjia Tong, Yuhang Chen, Zewei Xu, and Rong Yu. Wise-IoU: Bounding box regression loss with dynamic focusing mechanism. *arXiv preprint arXiv:2301.10051*, 2023.

[24] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. YOLOv10: Real-time end-to-end object detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[25] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. CSPNet: A new backbone that can enhance learning capability of CNN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 390–391, 2020.

[26] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. UnitBox: An advanced object detection network. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 516–520, 2016.

[27] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833, 2014.

[28] Hao Zhang and Shuaijie Zhang. Shape-IoU: More accurate metric considering bounding box shape and scale. *arXiv preprint arXiv:2312.17663*, 2024.

[29] Yi-Fan Zhang, Weiqiang Ren, Zhang Zhang, Zhen Jia, Liang Wang, and Tieniu Tan. Focal and efficient IOU loss for accurate bounding box regression. *arXiv preprint arXiv:2101.08158*, 2021.

[30] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. DETRs beat YOLOs on real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16965–16974, 2024.

[31] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-IoU loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12993–13000, 2020.

[32] Zhaohui Zheng, Ping Wang, Dongwei Ren, Wei Liu, Rongguang Ye, Qinghua Hu, and Wangmeng Zuo. Enhancing geometric factors in model learning and inference for object detection and instance segmentation. *IEEE Transactions on Cybernetics*, 52(8):8574–8586, 2022.