NYCU CV2025 HW2

GitHub Link:
https://github.com/CEJiang/NYCU-Computer-Vision-2025-Spring-HW2

1. Introduction

In this assignment, we aim to develop a robust digit detection model capable of accurately localizing and classifying digits in each image from the HW2 dataset. To achieve strong generalization performance, we adopt a Faster R-CNN framework with a ResNet-50 backbone, a Region Proposal Network (RPN) as the neck, and classification and localization heads. Additionally, we fine-tune the RPN anchor settings to further improve detection accuracy.

Furthermore, we experiment with different loss functions, such as CIoU Loss and DIoU Loss[2], to evaluate their impact on model performance. To further analyze the effect on mean Average Precision (mAP), we also fine-tune various anchor size combinations, comparing their contributions to overall accuracy.

Our best configuration achieves a mAP of 0.38 and an accuracy of 0.84, demonstrating the effectiveness of our model design and optimization strategies.

2. Method

2.1 Data Preprocessing

RandomApply(ColorJitter)	Brightness = 0.2 Contrast = 0.2 Saturation = 0.2 P = 0.3
RandomApply(GaussianBlur)	p = 0.3 kernel size = 3 sigma = (0.3, 0.8)
ToTensor	
Normalize	Mean = [0.485, 0.456, 0.406] Std = [0.229, 0.224, 0.225]

1. RandomApply(ColorJitter)

Randomly adjusts the brightness, contrast, and saturation of the input image to simulate varying lighting conditions. This transformation is applied with a 30% probability, helping the model become more robust to variations in illumination.

2. RandomApply(GaussianBlur)

Applies a Gaussian blur with random strength and kernel size (sigma=(0.3, 0.8)) to simulate out-of-focus or low-quality images. This operation is applied with 30% probability, helping the model learn to be robust to blurry or distorted inputs.

3. ToTensor

Converts the image from PIL format (used by ImageFolder) to a PyTorch tensor. This changes the shape to [C, H, W] and scales pixel values from [0, 255] to [0.0, 1.0].

4. Normalize

All inputs were normalized with the ImageNet mean and std for pretrained compatibility.

2.2 Model Architecture

```python
def faster_rcnn_resnet50(num_classes, device, args):

    weights = models.detection.FasterRCNN_ResNet50_FPN_V2_Weights.DEFAULT
    model = models.detection.fasterrcnn_resnet50_fpn_v2(weights=weights)

    # anchor_generator = AnchorGenerator(
    #     sizes=((4,), (8,), (12,), (24,), (48,)),
    #     aspect_ratios=((0.25, 0.5, 1.0),) * 5
    # )
    anchor_generator = AnchorGenerator(
        sizes=((4,), (8,), (12,), (24,), (48,)),
        aspect_ratios=((0.5, 1.0, 1.5, 2.0),) * 5
    )
    num_anchors = anchor_generator.num_anchors_per_location()[0]

    model.rpn.anchor_generator = anchor_generator

    model.roi_heads.box_roi_pool = MultiScaleRoIAlign(
        featmap_names=['0', '1', '2', '3'],
        output_size=7,
        sampling_ratio=2
    )

    if args.mode == "train" or args.mode == "validate":
        model.roi_heads.score_thresh = 0.5
        model.roi_heads.nms_thresh = 0.5
    elif args.mode == "test":
        model.roi_heads.score_thresh = 0.7
        model.roi_heads.nms_thresh = 0.3

    model.roi_heads.positive_fraction = 0.25

    in_features = model.roi_heads.box_predictor.cls_score.in_features
    model.rpn.head = RPNHead(in_channels=256, num_anchors=num_anchors)
    model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)

    return model.to(device)
```

Backbone: ResNet-50

Pre-trained Weights: FasterRCNN_ResNet50_FPN_V2_Weights.DEFAULT

Neck:

Anchor sizes = ((4,), (8,), (12,), (24), (48))

Anchor aspect ratios = ((0.5, 1.0, 1.5, 2.0)) * 5

We set the anchor sizes to ((4,), (8,), (12,), (24,), (48,)) based on a rough analysis of the bounding box dimensions in both train.json and valid.json. Most of the ground truth boxes in the dataset are smaller than 48 pixels in both width and height. Therefore, we chose relatively small anchor sizes to better match the scale of the digits and improve proposal generation for small objects.

Furthermore, we set the anchor aspect ratios to ((0.5, 1.0, 1.5, 2.0),) * 5 based on a rough analysis of the width-to-height ratios of bounding boxes in the dataset. Most

of the ground truth digits exhibit elongated shapes, especially digits like "1" and "8", which tend to be tall and narrow. Therefore, we included more aspect ratios greater than 1.0 to better capture vertically stretched digits and improve proposal matching for such cases.

Therefore, due to the change in anchor aspect ratios, we also needed to modify the num_anchors parameter in the RPN head. Since each anchor level now contains four aspect ratios — (0.5, 1.0, 1.5, 2.0) — the number of anchors per spatial location becomes 4. Accordingly, the RPN head must be configured with num_anchors = 4 to ensure proper alignment between the anchor generator and the RPN classification and regression layers.

Head:

We adjust the inference thresholds based on the current mode.

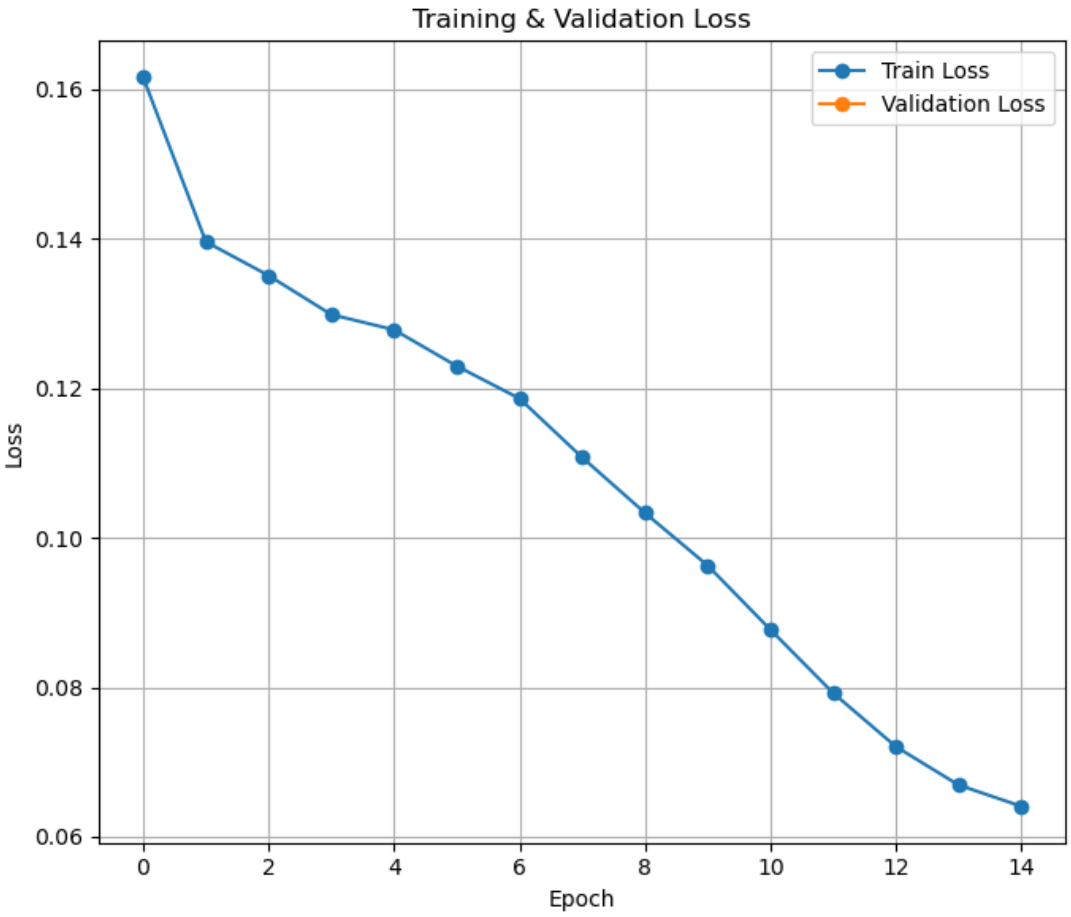
In both training and validation modes, the score threshold is set to 0.5, and the NMS threshold is set to 0.5, which helps remove low-confidence predictions and filter overlapping boxes with IoU > 0.5.

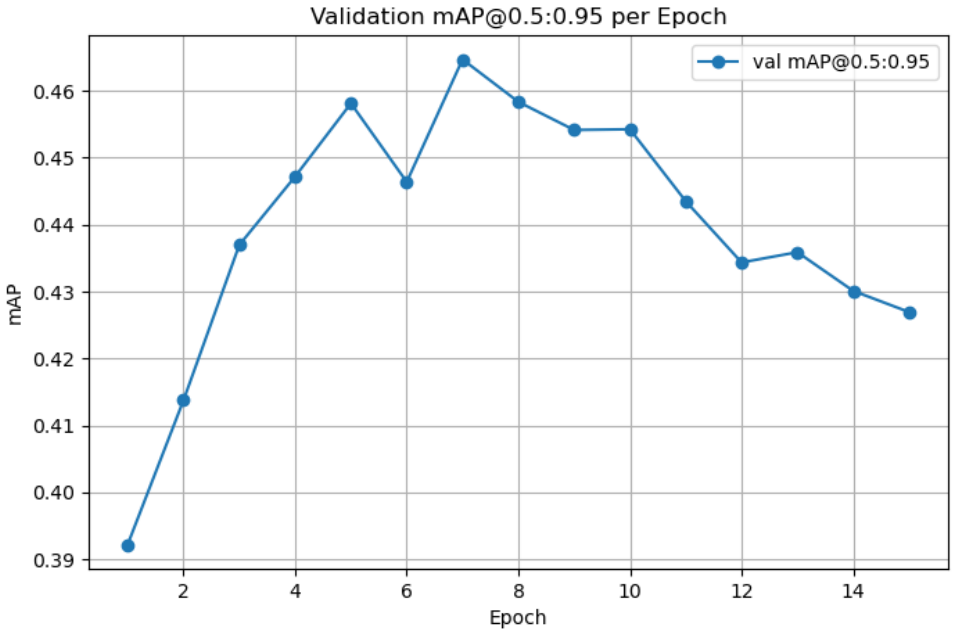
However, in test mode, we increase the score threshold to 0.7 and reduce the NMS threshold to 0.3. These stricter settings make the model more conservative and improve precision by discarding low-confidence boxes and tightening the overlap constraint, resulting in better alignment with ground-truth bounding boxes.

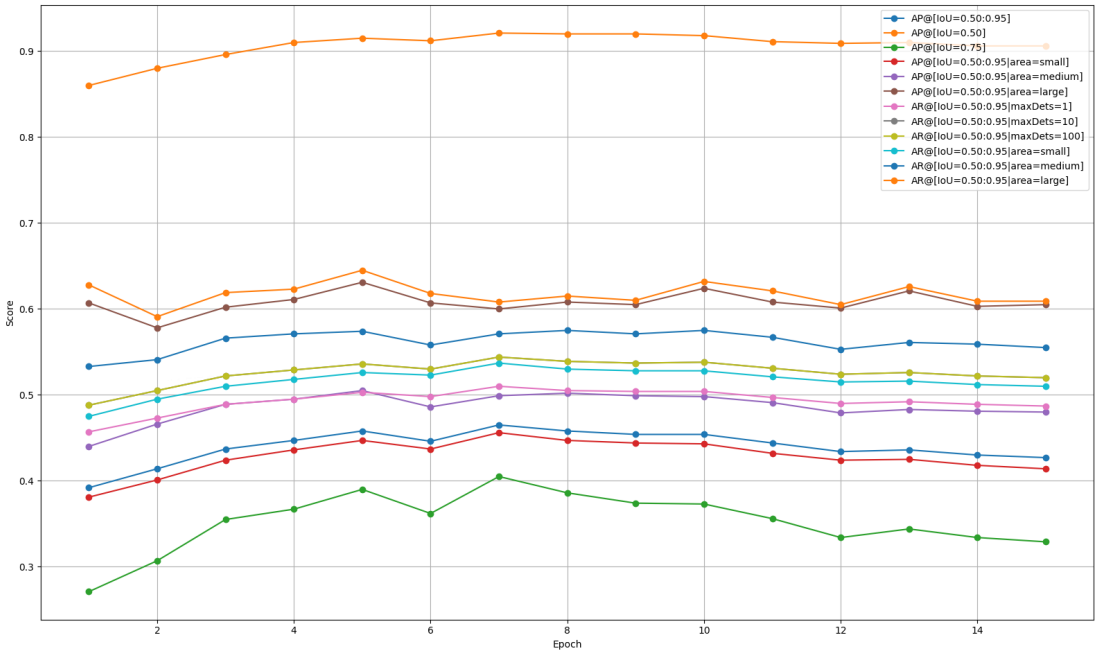
2.3 Hyperparameters

Pretrained Weight	FasterRCNN_ResNet50_FPN_V2
Learning rate	1×10^{-4}
Batch size	8
Epochs	15
decay	1×10^{-4}
Optimizer	AdamW
Eta_min	1×10^{-6}
T_max	15
Scheduler	CosineAnnealing
Criterion	CrossEntropyLoss(Classification) Smooth L1 Loss(Localization)

3. Results




Validation mAP@0.5:0.95 per Epoch
val mAP@0.5:0.95
mAP
Epoch
0.46
0.45
0.44
0.43
0.42
0.41
0.40
0.39
2
4
6
8
10
12
14


AP@[IoU=0.50:0.95]
AP@[IoU=0.50]
AP@[IoU=0.75]
AP@[IoU=0.50:0.95|area=small]
AP@[IoU=0.50:0.95|area=medium]
AP@[IoU=0.50:0.95|area=large]
AR@[IoU=0.50:0.95|maxDets=1]
AR@[IoU=0.50:0.95|maxDets=10]
AR@[IoU=0.50:0.95|maxDets=100]
AR@[IoU=0.50:0.95|area=small]
AR@[IoU=0.50:0.95|area=medium]
AR@[IoU=0.50:0.95|area=large]
Score
Epoch
0.9
0.8
0.7
0.6
0.5
0.4
0.3
2
4
6
8
10
12
14

Training Loss Curve

The training loss curve shows a steady decrease from epoch 1 to epoch 15, indicating that the model is effectively minimizing the loss on the training set. This downward trend suggests that the model is learning well without signs of overfitting in the training phase.

Validation mAP@0.5:0.95 Curve

The validation mAP@0.5:0.95 curve steadily increases during the early epochs and peaks at epoch 7 with a score of 0.465. However, after this point, the mAP begins to fluctuate and slightly decrease, which may indicate early signs of overfitting or that the model has reached a learning saturation point. This divergence between training loss and validation performance suggests that further training might not improve generalization.

mAP / AP / AR curve

The curve shows that AP@[IoU=0.5] remains consistently above 0.90, indicating that the model can detect most objects under looser localization constraints. However, AP@[IoU=0.75] stays between 0.3 and 0.4, which suggests that the model struggles to generate precise bounding boxes that closely match the ground truth. we suspect this may be due to our earlier misunderstanding of the anchor aspect ratios, which could have affected the model's ability to detect narrow objects like the digit "1".

In terms of Average Recall (AR), most metrics show a consistent improvement during the first few epochs and then stabilize around epoch 6 to 8. For instance, AR@[IoU=0.50:0.95|maxDets=100] and AR@[IoU=0.50:0.95|area=large] maintain relatively high and stable values, indicating strong recall performance for large objects. However, AR@[IoU=0.50:0.95|area=small] remains comparatively lower and more fluctuating, which suggests that small digits are more challenging to detect consistently.

4. References

[1] AnchorGenerator
https://github.com/pytorch/vision/blob/main/torchvision/models/detection/anchor_utils.py

[2] DIoU CIoU
https://cdn.aaai.org/ojs/6999/6999-13-10228-1-10-20200525.pdf

[2] DIoU CIoU
https://github.com/Zzh-tju/DIoU-SSD-pytorch/blob/master/utils/box/box_utils.py

[3] COCOeval
https://pyimagesearch.com/2022/05/02/mean-average-precision-map-using-the-coco-evaluator/

[4] RPNHead num_anchors
https://github.com/facebookresearch/detectron2/blob/main/detectron2/modeling/proposal_generator/rpn.py

[5] NMS
https://chih-sheng-huang821.medium.com/機器-深度學習-物件偵測-non-maximum-suppression-nms-aa70c45adffa

5. Additional experiments

In this experiment, we try different loss function combination to evaluate their impact on model performance.

Baseline: CrossEntropyLoss + Smooth L1 Loss

Combination 1: (CrossEntropyLoss(0 ~ 5 epoch) $\rightarrow$ $(1-\lambda)$CrossEntropyLoss + λDIoU Loss(6 ~ 14 epoch), $\lambda = \min(0.1 + (epoch - 6) * 0.05,\ 0.5)$) + Smooth L1 Loss

Combination 2: (CrossEntropyLoss(0 ~ 5 epoch) $\rightarrow$ $(1-\lambda)$CrossEntropyLoss + λCIoU Loss(6 ~ 14 epoch) $\lambda = \min(0.1 + (epoch - 6) * 0.05,\ 0.5)$) + Smooth L1 Loss

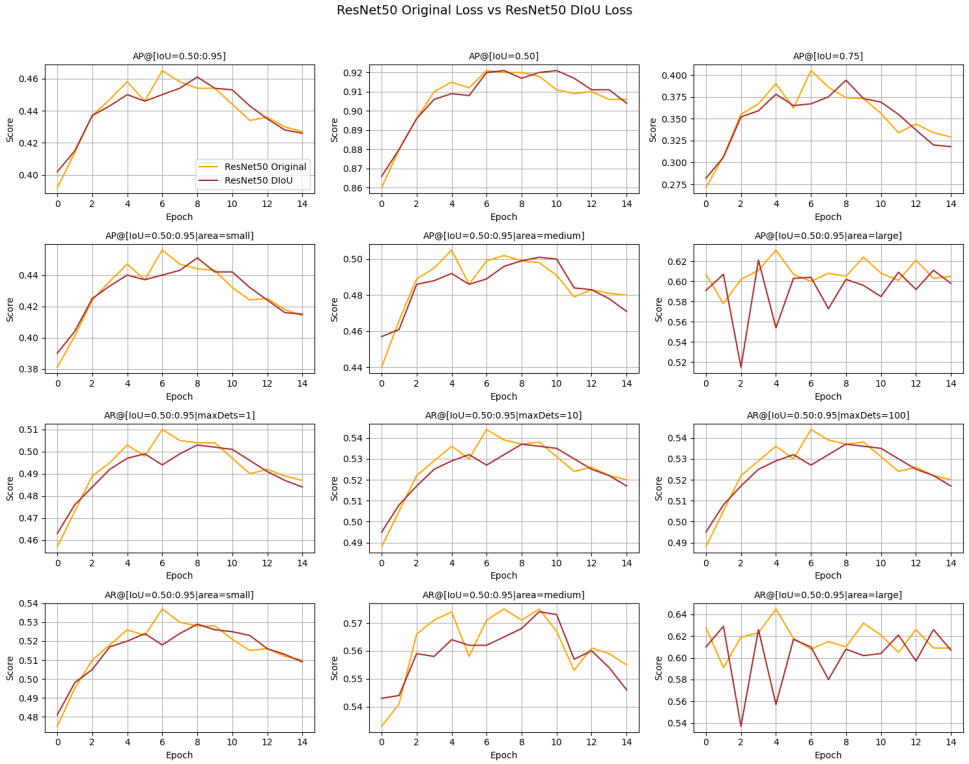
Hypothesis 1: We hypothesize that introducing a DIoU Loss into the classification Loss, as in combination 1, may enhance the model's overall spatial understanding and lead to better detection performance.

How this may work: DIoU Loss considers both the IoU overlap and the center distance between predicted and ground truth boxes. While it is typically used for bounding box regression, we believe that incorporating it into the classification objective may allow the model to learn more geometrically consistent predictions — associating higher classification confidence with spatially aligned bounding boxes. To reduce training instability, DIoU is not applied at the beginning of training. Instead, it is gradually introduced starting from epoch 6, with an increasing weight λ. This progressive integration is expected to let the model first stabilize its classification ability using standard CrossEntropyLoss, then gain additional spatial awareness in later stages. By doing so, we aim to reinforce the link between classification scores and geometric quality.

How this not may work:

Unstable late-stage learning: The progressive increase of λ starting from epoch 7 may disrupt the classification head, particularly in a small-class task like digit detection, where early convergence of class boundaries is crucial. The model may unlearn its previous discriminative features.

Experimental Results and Implications:



The comparison between ResNet50 Original Loss (CrossEntropy + Smooth L1) and ResNet50 DIoU Loss (gradually integrating DIoU into classification) reveals that the original configuration generally performs better or similarly across most metrics.

- mAP (AP@[0.50:0.95]): The original loss achieved a higher peak around epoch 7–8 and remained more stable afterward. The DIoU combination slightly lags in both peak and overall trend.
- AP@[IoU=0.50 and 0.75]: The original model maintains consistently higher scores. The DIoU variant shows minor gains in some early epochs but becomes unstable.
- By object size: For small and medium objects, the two methods are close, but the original has a slight advantage. For large objects, the original loss significantly outperforms DIoU, indicating possible classification interference in the DIoU variant.
- AR (Recall): Across all maxDets (1, 10, 100), the original loss again performs slightly better and is more stable. The DIoU model exhibits sharp fluctuations, especially on large object recal.

Implication:

Although the DIoU-based combination was designed to enhance spatial awareness, its application within the classification head may have introduced gradient interference or task misalignment. This is especially visible in the inconsistent AR and AP scores, suggesting that geometric losses like DIoU may not be suitable for classification tasks without further adaptation. The results also align with our hypothesis that improper mixing of task-specific losses may harm final detection performance.

Hypothesis 2: we hypothesize that introducing a CIoU Loss into the classification Loss, as in combination 2, may enhance the model's overall spatial understanding and lead to better detection performance.

How this may work:

CIoU Loss extends DIoU by incorporating not only IoU and center distance, but also the aspect ratio consistency between predicted and ground truth boxes. This additional term allows the model to capture not just spatial alignment, but also shape alignment.

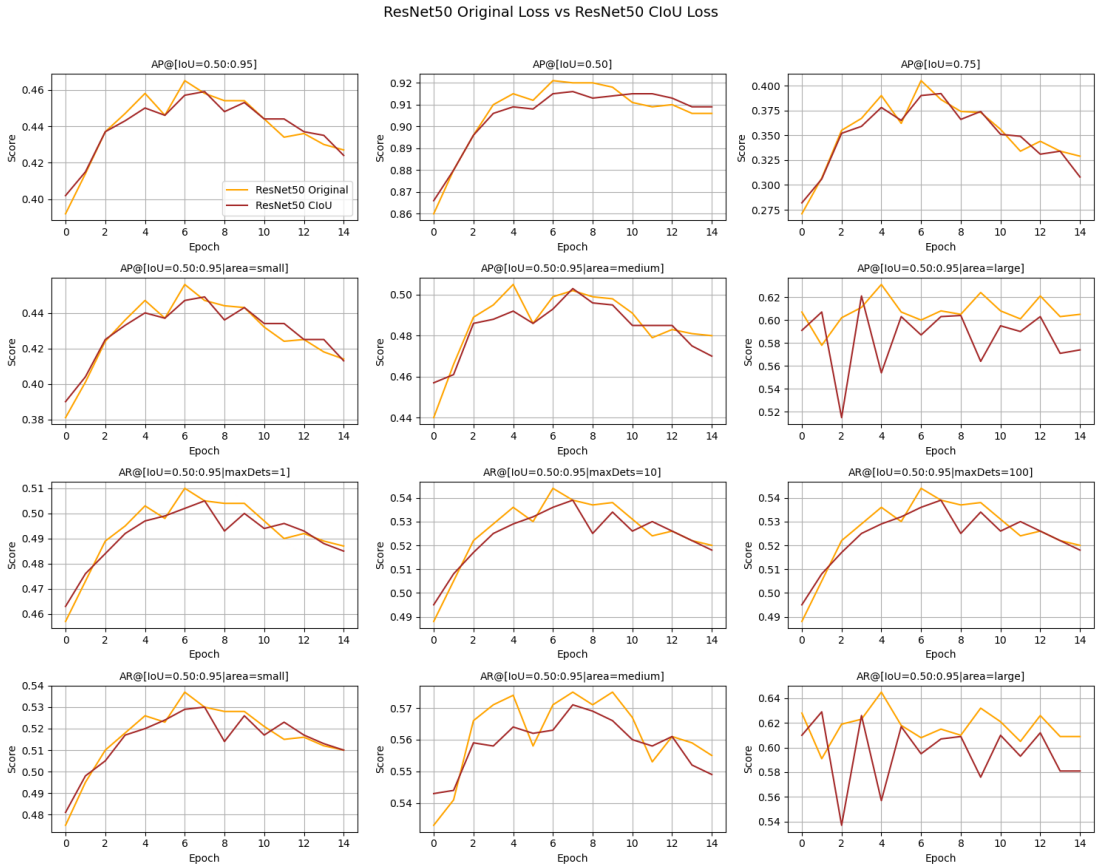
In digit detection, where digits like "1", "0", and "8" have distinguishable aspect ratios, incorporating CIoU into the classification objective may encourage the model to learn class-specific geometric cues.

To maintain training stability, CIoU is introduced progressively starting from epoch 7 with an increasing weight λ. This approach allows the model to first learn clean classification boundaries using CrossEntropyLoss, and then refine them with shape-aware spatial information in later stages.

How this may not work:

Unstable late-stage learning: The increase in λ after epoch 6 may disrupt learned class boundaries, especially in a task with a small number of tightly packed classes. The aspect ratio penalty might introduce unnecessary gradient noise, degrading classification accuracy.

Experimental Results and Implications:



The comparison between ResNet50 Original Loss (CrossEntropy + Smooth L1) and ResNet50 CIoU Loss (gradually integrating CIoU into classification) shows that the original configuration generally performs better or equally well across most evaluation metrics.

- mAP (AP@[0.50:0.95]): The original model reaches a slightly higher peak around epoch 7–8 and shows a smoother curve overall. The CIoU combination follows a similar trend but tends to plateau earlier and drop off slightly more in later epochs.
- AP@[IoU=0.50 and 0.75]: Across both thresholds, the original model consistently maintains higher scores. The CIoU-enhanced model performs closely at first but exhibits more instability as training progresses.
- By object size: For small and medium objects, both configurations perform comparably, though the original has a slight edge. For large objects, the

performance gap widens in favor of the original, suggesting that CIoU's aspect ratio penalty may introduce noise into the classification task.

- AR (Recall): With maxDets = 1, 10, and 100, the original model again demonstrates slightly better and more stable recall. The CIoU model shows more pronounced fluctuations, particularly in large-object recall, which may reflect the added complexity from CIoU's geometric terms.

Implication:

While CIoU theoretically offers a more comprehensive spatial representation by accounting for aspect ratio, its integration into the classification head does not yield noticeable performance improvements. In some cases, it may even interfere with class boundary formation in digit detection, where class distinctions are based more on appearance than shape similarity. This reinforces our hypothesis that geometric losses must be carefully adapted before being applied to non-regression components of object detectors.