BUCKETING RANKING-BASED LOSSES FOR EFFICIENT TRAINING OF OBJECT DETECTORS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

FEYZA YAVUZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JANUARY 2025

Approval of the thesis:

BUCKETING RANKING-BASED LOSSES FOR EFFICIENT TRAINING OF OBJECT DETECTORS

submitted by **FEYZA YAVUZ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Naci Emre ALTUN
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Mehmet Halit S. Oğuztüzün
Head of Department, **Computer Engineering** _____

Prof. Dr. Sinan Kalkan
Supervisor, **Computer Engineering, METU** _____

Assoc. Prof. Dr. Emre Akbaş
Co-supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Gökberk Cinbiş
Computer Engineering, METU _____

Prof. Dr. Sinan Kalkan
Computer Engineering, METU _____

Assoc. Prof. Dr. Hacer Yalım Keleş
Computer Engineering, Hacettepe University _____

Date:10.01.2025

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Feyza Yavuz

Signature :

ABSTRACT

BUCKETING RANKING-BASED LOSSES FOR EFFICIENT TRAINING OF OBJECT DETECTORS

Yavuz, Feyza
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. Sinan Kalkan
Co-Supervisor: Assoc. Prof. Dr. Emre Akbaş

January 2025, 72 pages

Object detection is a fundamental computer vision task that focuses on classifying and locating objects in an image. Classification and localization of objects are commonly supervised with score-based loss functions, e.g., Cross-entropy Loss for classification and L1 Loss for localization. On the other hand, ranking-based loss functions, such as Average Precision Loss and Rank & Sort Loss, better align with the evaluation criteria, have fewer hyperparameters, and offer robustness against the imbalance between positive and negative samples. However, they require pairwise comparisons among P positive and N negative predictions, introducing a time complexity of $\mathcal{O}(PN)$, which is prohibitive since N is often large. Despite their advantages, the widespread adoption of ranking-based losses has been hindered by their high time and space complexities.

In this thesis, we focus on improving the efficiency of ranking-based loss functions. To this end, we propose Bucketed Ranking-based (BR) Losses which group negatives into B buckets ($B \ll N$) to reduce the number of pairwise comparisons. Thanks to bucketing, our method reduces the time complexity to $\mathcal{O}(\max(N \log(N), P^2))$.

To validate our approach, we conducted experiments on two different tasks, three different datasets, seven different detectors. We show that BR Losses yield the same accuracy with their unbucketed versions and provide $2\times$ faster training on average. Lower complexity of BR Losses enable us to train, for the first time, transformer-based object detectors using a ranking-based loss. When we train CoDETR, a state-of-the-art transformer-based object detector, we consistently outperform its original results over several different backbones.

Keywords: Object Detection, Detection Transformers, Loss Functions, Ranking-based Losses, Efficient Ranking-based Losses

ÖZ

NESNE TESPİTÇİLERİNİN ETKİN EĞİTİMİ İÇİN SIRALAMA BAZLI KAYIP FONKSİYONLARINI GRUPLAMA

Yavuz, Feyza
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Sinan Kalkan
Ortak Tez Yöneticisi: Doç. Dr. Emre Akbaş

Ocak 2025, 72 sayfa

Nesne tespiti, bir görüntüdeki nesneleri tanımlama ve konumlandırmaya odaklanan bir bilgisayarlı görü problemidir. Nesnelerin sınıflandırılması ve sınırlayıcı kutu koordinatlarının tahmin edilmesi genellikle puan tabanlı kayıp fonksiyonları ile denetlenir. Örneğin, sınıflandırma için Cross-entropy Loss ve konumlandırma için L1 Loss kullanılır. Buna karşın, Average Precision Loss ve Rank&Sort Loss gibi sıralama tabanlı (ranking-based) kayıp fonksiyonları, değerlendirme kriterleriyle daha iyi uyuşur, daha az hiperparametreye sahiptir ve pozitif-negatif örnekler arasındaki dengesizliğe karşı daha dayanıklıdır. Ancak bu fonksiyonlar, (P) pozitif ve (N) negatif tahmin arasında ikili karşılaştırmalar gerektirdiğinden $\mathcal{O}(PN)$ zaman karmaşıklığı ortaya çıkar; bu da N genellikle büyük olduğu için engelleyici boyutlara ulaşabilir.

Bu tezde, sıralama tabanlı kayıp fonksiyonlarının verimliliğini artırmaya odaklanıyoruz. Bu doğrultuda, negatif tahminleri B gruba ($B \ll N$) ayırarak çift yönlü karşılaştırmaların sayısını azaltmayı hedefleyen Bucketed Ranking-based Losses (Gruplandırılmış Sıralama Tabanlı Kayıplar) yöntemini öneriyoruz. Yöntemimiz, zaman

karmaşıklığını $\mathcal{O}(\max(N\log(N), P^2))$ seviyesine düşürmektedir. Yöntemimizin geçerliliğini göstermek ve genelliğini doğrulamak için iki farklı problem, üç farklı veri seti ve yedi farklı tespitçi üzerinde deneyler gerçekleştirdik. Sonuçlarımız, Gruplandırılmış Sıralama Tabanlı Kayıpların gruplandırılmamış versiyonlarla aynı doğruluğu sağladığını ve ortalama olarak iki kat daha hızlı bir eğitim sunduğunu göstermektedir. Ayrıca, bu verimlilik sayesinde, ilk kez sıralama tabanlı kayıplar kullanılarak transformer tabanlı nesne tespitçileri eğitilmiştir. CoDETR adlı, transformer tabanlı nesne tespitçiyi Bucketed Ranking-based Loss ile eğittiğimizde, farklı omurgalar üzerinde orijinal sonuçlarından sürekli olarak daha iyi performans elde ettik.

Anahtar Kelimeler: Nesne Tespiti, Tespit Transformerları, Kayıp Fonksiyonları, Sıralama Bazlı Kayıplar, Verimli Sıralama Tabanlı Kayıplar

This thesis is dedicated to my family and friends, who supported me in every aspect throughout my studies.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisors, Prof. Dr. Sinan Kalkan and Assoc. Prof. Dr. Emre Akbaş, for their tireless support throughout my graduate studies and research. I also want to thank Kemal Öksüz and Barış Can Çam for their contributions. Their patience, motivation, and extensive knowledge have been invaluable to my journey. I feel truly fortunate to have worked under their guidance, and their mentorship has played a crucial role in my academic development.

First, I want to express my heartfelt gratitude to Can, whose support, patience, and understanding have been a true source of strength for me during this journey. Even during the long hours I spent studying, Can's constant encouragement made all the difference.

To my amazing friends Aybüke, Deniz, Erce, İpek, Sude, Yiğit, Yunus and Yusuf, I am deeply grateful for your friendship and support. Whether through late-night study sessions or simply being there when I needed a break, your encouragement has made this journey unforgettable. I would like to thank the people of ImageLab for sharing this academic experience and for navigating the ups and downs of graduate life together.

I would also like to extend my gratitude to TÜBİTAK ULAKBİM High Performance and Grid Computing Center (TRUBA) for providing the computational resources that played a vital role in my research.

Lastly, I wish to thank my family. My parents have always encouraged me to pursue excellence in education and instilled in me the values of hard work and resilience.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	
AP	Average Precision
BAP	Bucketed Average Precision
BRS	Bucketed Rank & Sort
CE	Cross Entropy
IoU	Intersection over Union
RB	Ranking-based
RS	Rank & Sort

CHAPTER 1

INTRODUCTION

1.1 Motivation

Figure 1.1: Object detection pertains to localizing and classifying objects in images. [Image from the COCO dataset]

Object detection is a fundamental problem in computer vision, requiring the classification and localization of objects within an image (Figure 1.1). Training an object detector is more challenging than that of a classifier since it requires more sophisticated learning algorithms, usually including a combination of (i) dynamically assigning a large number of hypotheses to objects [7–13], (ii) sampling among these hy-

potheses to ensure that the background class does not dominate training [10, 14–16] and (iii) optimizing a multi-task objective function [10, 17–19]. While the choices on assignment and sampling generally vary depending on the trained object detector, it is very common to combine Cross Entropy or Focal Loss [19] with a regression loss [20,21] as the multi-task loss function. However, the imbalance between positive and negative examples during training affects score-based classification losses. The background class, i.e., negatives, constitutes up to 99.9% of the hypotheses generated during training [22]. Recently proposed ranking-based loss functions [1,2,17,18] offer an alternative approach for addressing these challenges by formulating the training objective based on the rank of positive examples over negative examples. Recently proposed ranking-based loss functions [1, 2, 17, 18] offer an alternative approach for addressing these challenges by formulating the training objective based on the rank of positive examples over negative examples. Figure 1.2 illustrates the difference between common score-based classification losses and ranking-based losses in terms of computation.

Benefits of ranking-based losses. First, they are inherently robust to imbalance [17] and hence, do not require any sampling mechanism under very challenging scenarios [18], e.g., even when the background-foreground ratio is 10K for LVIS [23]. Second, they are shown to generalize over different detectors with diverse architectures – with the exception of transformer-based ones [4–6], since ranking-based losses further slow down the training of transformer-based detectors, which we address in this paper. Such losses also offer significant performance gain over their score-based counterparts, and having less hyperparameters, they are much easier to tune [17, 18].

The drawback of ranking-based losses. Compared to score-based losses, ranking-based losses are less efficient as the ranking operation inherently requires each pair of object hypotheses to be compared against each other (Figure 1.3(a)), inducing a quadratic complexity on the large number of object hypotheses (e.g., 10^8 for ATSS [7] on COCO [24]). As a result, vectorized implementations for parallel GPU processing are infeasible as such large matrices (e.g., with $\sim 10^{16}$ entries for ATSS) do not fit into GPU memories. This has driven researchers towards alternative ways of computing these matrices, which, in the end, saves from the storage complexity but results in more inefficient algorithms [1, 2] in terms of time complexity. Two techniques were

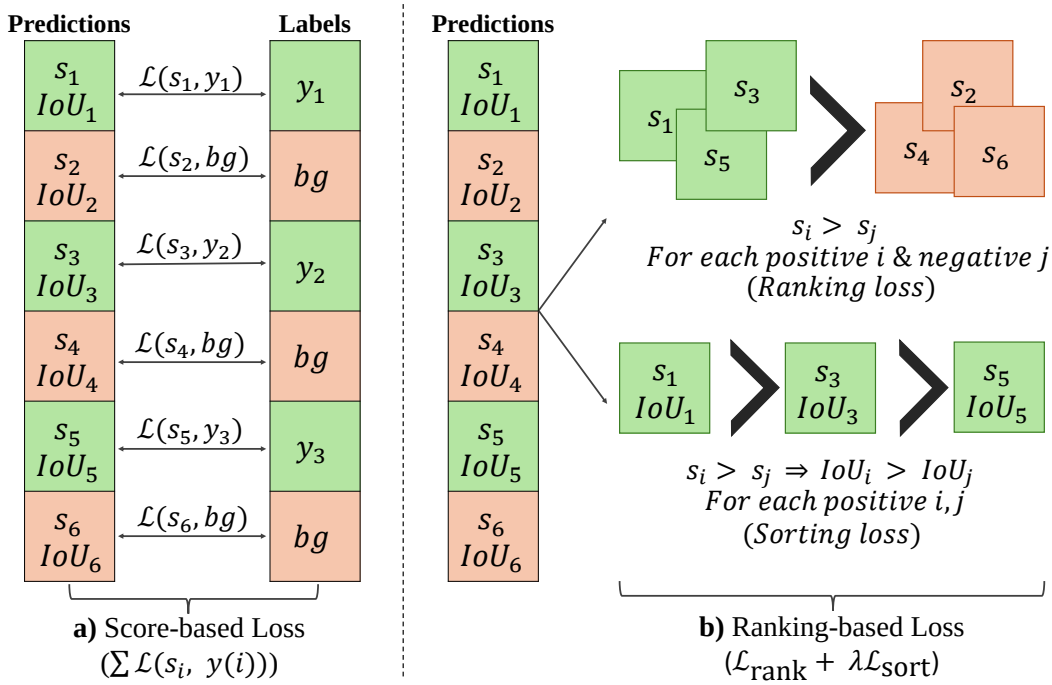


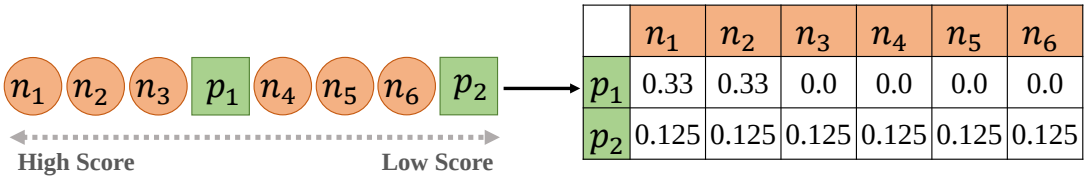
Figure 1.2: **a)** Score-based classification losses compute the loss according to each prediction and its label for training object detectors. **b)** Ranking-based losses offer a different approach by ranking positives over negatives and sorting positives concerning their IoU scores. s_i, IoU_i denote respectively the classification score and the Intersection-over-Union (IoU) score for box i.

specifically proposed to address these drawbacks: implementing a loop over positive examples and discarding trivial negatives. By introducing these techniques, storage complexity is significantly reduced; however, time complexity remains an important drawback.

1.2 Problem Definition and Scope of the Thesis

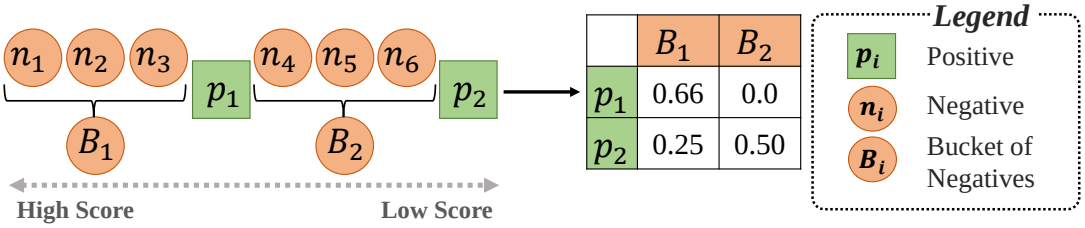
In this thesis, we aim to address the inefficiencies in ranking-based loss functions. Specifically, we focus on reducing their computational complexity while preserving their theoretical robustness and advantages. To achieve this, we introduce a novel bucketing approach that efficiently handles negative predictions.

The core idea of our approach is to sort all examples by their scores and group the



	n_1	n_2	n_3	n_4	n_5	n_6
p_1	0.33	0.33	0.0	0.0	0.0	0.0
p_2	0.125	0.125	0.125	0.125	0.125	0.125

a) Required errors in AP Loss



	B_1	B_2
p_1	0.66	0.0
p_2	0.25	0.50

b) Required errors in BAP Loss (ours)

Figure 1.3: (a) Existing ranking-based losses (i.e., AP Loss [1, 2]) incur significant overhead owing to pairwise comparisons between positives and negatives. (b) We propose bucketing negatives to decrease the number of comparisons, and hence the complexity. Under certain assumptions, our bucketing approach provides equal gradients with conventional ranking-based losses such as AP Loss in (a). [Figure from our ECCV 2024 paper [3]]

negatives located between successive positives into buckets. Each bucket of negatives is treated as a single negative prediction, represented by their average score (Figure 1.3(b)). This restructuring reduces the computational overhead associated with ranking-based loss functions.

To demonstrate the effectiveness of our approach, we integrate this bucketing mechanism into established ranking-based loss functions such as AP Loss [1, 2] and RS Loss [18]. Furthermore, we propose a gradient computation that ensures the gradients remain the same with their original formulations, under certain assumptions.

From a theoretical perspective, our approach provides exact gradients compared to conventional ranking-based losses while significantly reducing time complexity. Practically, we achieve up to a $40\times$ **reduction in loss computation time** and a $6\times$ **reduction in training time** for object detectors.

These improvements eliminate the primary drawback of ranking-based loss functions, that is, their computational cost. By bridging the gap between score-based and ranking-based loss functions in terms of training efficiency, we enable, for the first time, the training of high-performing transformer-based object detectors such as Co-DETR [6] using our loss functions.

As a result, ranking-based loss functions not only maintain their advantages, but also close the gap in terms computational efficiency. This thesis contributes to advancing object detection by making ranking-based loss functions more efficient for object detection frameworks.

1.3 Contributions

In this thesis, we make the following contributions:

- We propose a novel bucketing approach to improve the efficiency of computationally expensive ranking-based losses to train object detectors. Theoretically, our approach yields the same gradients with the original ranking-based losses while decreasing their time complexity. Practically, we enable up to $6\times$ faster training using our bucketing approach with no accuracy loss.
- For the first time, we incorporate ranking-based loss functions to transformer-based detectors. Specifically, we construct an object detector called BRS-DETR by replacing the training objective of the state-of-the-art transfomer-based object detector Co-DETR [6] by our Bucketed RS (BRS) Loss.
- Our comprehensive experiments on detection and instance segmentation on three different challenging datasets, five backbones and seven detectors show the effectiveness and generalizability of our approach. Our BRS-DETR reaches 50.3 AP on COCO val set with only 12 epochs and ResNet-50 backbone, outperforming all existing detectors using the same backbone and 300 queries. BRS-DETR also outperforms CoDETR with the Swin-T and Swin-L backbones, reaching 57.2 AP on the COCO dataset.

The main contributions of the thesis have been published at the European Conference

on Computer Vision (ECCV) 2024 [3].

1.4 The Outline of the Thesis

In Chapter 2, we provide a detailed explanation of the object detection problem and discuss various aspects of existing detectors. Additionally, we present well-known loss functions used in state-of-the-art detectors. We also include a brief overview of evaluation metrics and datasets used for experiments.

In Chapter 3, we begin by providing a detailed explanation of AP Loss and RS Loss in terms of their formulation and optimization, as these are the core methods of our research. Next, we introduce our novel approach and describe how it can be integrated with these losses. We discuss both the theoretical and practical aspects of our proposed bucketing method.

In Chapter 4, we describe the experiments we conducted to validate our method and present results with various architectures and datasets.

Finally, in Chapter 5, we summarize the effects and limitations of our method while outlining future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Object Detection

Object detection is a fundamental task in computer vision that focuses on recognizing and localizing objects within an image or video frame. This involves not only classifying the objects but also predicting their spatial bounding box coordinates. Formally, given an input image I, the objective is to predict a set of N objects, where each object is represented by its class label $c_i \in \mathcal{C}$ and its bounding box $\mathbf{b}_i = (x_i, y_i, w_i, h_i)$. Here, $\mathcal{C}$ denotes the set of predefined object classes, (x_i, y_i) are the coordinates of the top-left corner of the bounding box, and w_i, h_i represent its width and height.

A major challenge in object detection is matching the predicted bounding boxes to the actual objects during training. This matching process, known as assignment in object detection, aligns each predicted anchor with the correct class label and real bounding box. Object detection uses different assignment techniques that adapt to the properties of the predictions and ground truth. These techniques may use overlap-based criteria, such as Intersection over Union (IoU), or more advanced methods like adaptive thresholding and probabilistic matching.

Once assignments are identified, the object detection task involves jointly optimizing two objectives: the classification loss, which ensures the correct assignment of class labels c_i, and the localization loss, which minimizes the error between predicted bounding boxes $\mathbf{b}_i$ and their corresponding ground-truth boxes. Together, these losses guide the model in learning both the semantic and spatial characteristics of objects. The overall loss can be expressed as:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda \mathcal{L}_{\text{loc}}, \tag{2.1}$$

where $\mathcal{L}_{\text{cls}}$ measures the classification error, $\mathcal{L}_{\text{loc}}$ measures the localization error, and λ balances the contributions of the two losses.

Object detection has progressed significantly, shifting from traditional handcrafted methods [25–27] to approaches based on deep learning [14, 28–30]. Early techniques used fixed feature extractors but struggled with scalability and efficiency, especially on large datasets. The introduction of convolutional neural networks (CNNs) was a significant success because of the use of learnable features that adapt to various tasks. CNNs became essential for capturing spatial and semantic details in images, laying the groundwork for improvements in object detection. While CNNs dominated the field for years [7,10–12,28,31–37], their reliance on local receptive fields caused challenges in capturing global context, particularly in complex scenes with overlapping or distant objects. Initially developed for natural language processing tasks, transformers outperform other works of modeling relationships across sequences. Adapting transformers to vision tasks with Visual Transformers (ViT) [38] improved object detection by addressing these limitations through global attention mechanisms.

Modern object detectors rely on numerous hand-crafted components carefully designed to improve detection accuracy and efficiency. These components often include predefined anchor boxes for region proposals, filtering overlapping predictions, and heuristics for matching predictions with ground-truth objects and they require extensive hyperparameter tuning and limit the model's scalability to diverse datasets.

2.1.1 Object Detectors

Object detectors in the literature can be broadly categorized into two classes: anchor-based and anchor-free detectors.

Anchor-based methods [7, 8, 10, 11, 14, 16, 19, 30, 39–44] rely on predefined anchor boxes to suggest regions of interest and predict objects within those regions. These methods are further divided into one-stage and two-stage detectors. One-stage detectors [7, 8, 14, 30, 42, 45, 46] directly predicts object classes and bounding boxes. In contrast, two-stage detectors [9–11, 16, 29, 39, 41, 47, 48] follow a multi-step approach where they first generate object proposals and then classify and localize objects within

those proposals. Two-stage detectors often achieve higher accuracy but comes with increased computational complexity.

On the other hand, anchor-free detectors [12, 31, 49–53] focus on detecting objects based on keypoints, object centers, or other features directly from the image. They can simplify the detection process and reduce the need for the extensive hyperparameter tuning generally associated with anchor-based methods, providing an alternative framework for object detection.

2.1.2 Anchor based detectors

One-stage Object Detectors simplify the object detection process by directly predicting object locations and class probabilities from the input image in a single pass. The general pipeline begins with a backbone network extracting features from the image, often enhanced by multi-scale feature maps for detecting objects of various sizes. Prediction heads simultaneously perform classification to identify object categories and regression to predict bounding box coordinates. Finally, a post-processing step like Non-Maximum Suppression (NMS) eliminates duplicate detections, ensuring only the most confident predictions remained.

YOLO (You Only Look Once) [30] was one of the first one-stage object detectors. It was designed to divide the input image into a grid and predict both bounding boxes and class probabilities in a single pass. Over the years, the original YOLO has been enhanced with various improvements, including multi-stage detection [39, 41], advanced backbone networks for better feature extraction [54], and anchor-free techniques that boost accuracy, efficiency, and adaptability [55, 56].

Introduced in the same year as YOLO [30], SSD [14, 46] (Single Shot MultiBox Detector) builds on the one-stage paradigm by introducing multi-scale feature maps and predefined anchor boxes, improving the detection of objects of varying sizes. SSD balances speed and accuracy, outperforming YOLO for small and medium objects but still behind two-stage detectors. Over the years, various improvements to SSD have been proposed.

Numerous studies have explored ways to enhance the assignment process between

predictions and ground-truth objects [7, 8, 42, 57, 58] . In 2020, PAA (Probabilistic Anchor Assignment) [8] introduced a probabilistic method for assigning anchors, eliminating the need for fixed Intersection over Union (IoU) thresholds. PAA calculates anchor scores based on the model's predictions and fits a probability distribution to these scores. Anchors are then classified as positive or negative based on their probabilities. Furthermore, to bridge the gap between training and testing objectives, PAA [8] predicts the IoU values of the detected boxes to assess localization quality.

Published in the same year, ATSS (Adaptive Training Sample Selection) [7] also introduced a mechanism to dynamically select positive and negative samples based on the properties of the dataset. It establishes a threshold using the mean and standard deviation of IoU values from a group of the nearest k anchors corresponding to each ground-truth object. To determine candidate anchors, ATSS selects pre-defined number of anchors per FPN level whose centers are closest to the ground-truth bounding box and evaluates IoU for these anchors. Samples with IoU above the threshold and center inside the ground-truth bounding box are defined as positive. Both methods significantly improved training stability and overall detection performance by addressing the limitations of traditional assignment techniques.

In summary, one-stage detectors are known for their speed thanks to their architectural design; however, they often lag behind two-stage detectors in terms of performance. Recent advancements in anchor assignment and post-processing strategies have further improved the performance of one-stage detectors. Techniques such as PAA and ATSS are helping one-stage methods to narrow the performance gap, providing a better balance between speed and accuracy.

Two-stage Object Detectors work in two separate phases to achieve high accuracy. In the first stage, candidate regions (proposals) that are likely to contain objects are generated using various methods. These methods may include a Region Proposal Network (RPN) or alternatives such as selective search, edge boxes, or attention-based mechanisms . In the second stage, these proposals are refined, and their class labels are predicted through classification and regression networks. Two-stage detectors utilize shared convolutional features for efficiency and employs several techniques to extract region-specific features for precise localization. Although two-stage detectors

are more computationally intensive than one-stage detectors, the two-stage approach outperforms in accuracy.

R-CNN (Region-Based Convolutional Neural Network) [29], introduced in 2014, was a significant approach that integrated deep learning into object detection by combining region proposals with deep learning. It generates around 2,000 candidate regions using Selective Search method and processes each region individually. Features are extracted from these regions using a pre-trained CNN, and the extracted features are then sent to an SVM classifier to predict object classes, while a bounding box regressor is used for localization. Although R-CNN significantly improved detection performance compared to traditional methods, it remains computationally expensive because each region is processed independently. To solve the inefficiency problem of R-CNN, Fast R-CNN [59] was introduced in 2015. Instead of processing each region proposal independently, the entire image is passed through a CNN once to produce a feature map. Region proposals are then projected onto this feature map, and Region of Interest (ROI) pooling extracts fixed-size feature representations for each proposal. Then produced features are fed into fully connected layers for classification and bounding box regression. Fast R-CNN significantly reduces computational overhead, allowing faster training and inference while maintaining high accuracy. However, it still relies on Selective Search to generate regional proposals.

Faster R-CNN [10] was developed shortly after Fast R-CNN. Introducing the Region Proposal Network (RPN) created a fully end-to-end trainable system and established a benchmark for two-stage object detection models. The RPN is integrated into the CNN, allowing it to generate region proposals directly from the feature map in a fully trainable manner. These proposals are then refined using the same ROI pooling and fully connected layers as in Fast R-CNN [59]. By unifying the region proposal and detection processes, Faster R-CNN significantly improves speed and efficiency. Several works [9, 11, 16, 47, 48, 60, 61] have been introduced to extend the performance of Faster R-CNN. Some of the most acknowledged works are Mask R-CNN [61] and Cascade R-CNN [11]. Mask R-CNN [61], introduced by He et al., adds a segmentation branch to Faster R-CNN, enabling instance segmentation alongside object detection. It uses ROIAlign instead of ROI pooling for better alignment of features and outputs a binary mask for each detected object, making it suitable for tasks re-

quiring pixel-level accuracy. Cascade R-CNN [11], proposed by Cai and Vasconcelos, improves detection performance by introducing a multi-stage refinement process. Each stage applies a higher IoU threshold to refine the quality of bounding boxes and classification progressively. While Mask R-CNN focuses on integrating detection and segmentation, Cascade R-CNN emphasizes robust, high-quality detection through sequential refinement.

2.1.3 Anchor-free detectors

Anchor-free methods differ from anchor-based methods in defining and localizing bounding boxes. As described above, anchor-based methods rely on predefined anchor boxes and refine them through regression; anchor-free methods directly predict bounding box coordinates or key points without anchors. Anchor-free methods do not require matching anchors and need less fine-tuning. On the other hand, anchor-based methods usually provide better performance.

Many prominent anchor-free methods have been introduced over the years. A well-known example, CornerNet [12], formulates object detection as the task of finding the corners of the boxes containing. It is implemented using a single CNN. CornerNet further introduces a novel corner pooling layer to enhance the localization of these corners.

CenterNet [31], introduced in 2019, enhances the CornerNet model by focusing on detecting the centers of objects rather than just their corners. It predicts the central point of an object, its dimensions, and offset values, which simplifies the detection process. By focusing on object centers, CenterNet achieves faster inference and improved accuracy, making it more efficient than CornerNet. Similarly, FCOS (Fully Convolutional One-Stage Object Detection) [49], also released in 2019, adopts a different anchor-free approach. It treats object detection as a dense regression problem by predicting the distances from each point in the feature map to the edges of the object's bounding box while simultaneously classifying the object. FCOS includes a centerness scoring mechanism that prioritizes central locations, enhancing localization accuracy and minimizing duplicate detections.

2.1.4 Transformer-based Object Detectors

Initially developed for Natural Language Processing (NLP), transformers are neural network architectures that utilize self-attention mechanisms to model global dependencies. Their ability to capture complex relationships across sequences has led to state-of-the-art performance in machine translation and text understanding tasks. Building on their success in NLP, Vision Transformers (ViTs) [38] adapt the transformer architecture for image analysis by dividing images into patches, treating them as sequences, and processing them with the same attention-based approach. The attention mechanism enables ViTs to capture long-range spatial dependencies and global context more effectively than CNNs, constrained by local receptive fields. The first significant implementation of transformers in object detection was DETR (DEtection TRansformer) [4]. CNN-based methods in object detection suffer from several problems. They rely on components like anchor boxes, region proposal networks, and NMS, which means non-end-to-end pipelines in object detection.

As illustrated in Figure 2.1, DETR [4] is the first work that simplifies the pipeline into a fully end-to-end trainable system by introducing assignment problem as set-matching problem. DETR starts by passing the input image through a CNN to extract feature maps. Next, using a self-attention mechanism, the encoder processes these feature maps to capture global dependencies across all spatial regions. DETR uniquely approaches object detection as a set prediction problem and introduces a fixed set of learnable embeddings known as object queries, which are processed by the decoder. The number of object queries determines the maximum number of objects that model can predict. Each object query represents a potential object in the image, and the decoder refines these queries based on the global context provided by the encoder. It employs a Hungarian matching algorithm during training to assign ground-truth objects to the predictions. This method ensures that each predicted object uniquely corresponds to a ground truth object, preventing any overlap in predictions. After decoding, each object query outputs a class label and a bounding box. DETR achieves outstanding results across various datasets and sets a new benchmark in detection research. DETR has one significant issue: slow convergence. Typically, DETR requires between 50 to 500 epochs to converge, whereas general detection

models usually only need about 12 epochs for training. Numerous studies have proposed analyses and solutions have been proposed to address DETR's slow convergence [5, 6, 62–70].

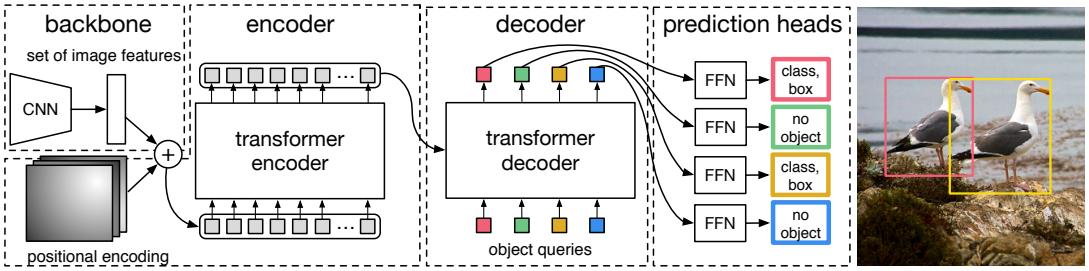


Figure 2.1: Overview of the DETR pipeline. The input image is passed through a CNN backbone to produce feature maps, which are then processed by the Transformer encoder to capture global context. Simultaneously, a set of learnable object queries is refined by the Transformer decoder, producing bounding box coordinates and class labels in the prediction heads. [Figure taken from [4]]

Specifically, Sun et al. [62] proposed a detailed analysis of the optimization difficulty in DETR training. Their analysis focuses mainly on three issues: Instability of Bipartite Matching, Attention Modules, and Cross-attention. They stated that Hungarian matching, which is bipartite, could cause instability due to the randomness in the initialization of the Hungarian matching and noisy conditions in different training epochs. Their experiments showed that the instability in the bipartite matching component of DETR only contributes partially to the slow convergence (especially in the early training stage) but not necessarily the primary cause. Next, they examine how much the sparsity dynamics of transformer attention modules in DETR contribute to convergence. Transformer attention maps are nearly uniform in the initialization stage but gradually become more and more sparse during the training process toward the convergence. They stated that cross-attention sparsity consistently increases even after 100 epochs. This means that the cross-attention part of DETR is a more significant factor for the slow convergence compared to the early-stage bipartite-matching instability. Lastly, they studied the effect of cross-attention by removing the cross-attention module from DETR and designing an encoder-only version. Initially, the decoder produces the detection results per object query in DETR. In contrast, the encoder-only version of DETR directly uses the outputs of the transformer encoder

for object prediction. Their experiments showed that the encoder-only DETR outperforms the original DETR significantly on small and partially on medium but underperforms on large objects. Overall, the study suggests that all three issues can be further improved, and many proposed studies have begun to address these challenges.

Some studies have focused on improving the assignment [6, 63, 69, 71] process by using auxiliary heads [6,63] or transforming one-to-one assignment into one-to-many assignment [71]. Other studies have concentrated on the transformer architecture [72], attention modules [5, 63, 65, 73], or enhancements to the learnable queries [64, 67, 70]. Deformable DETR [5], DINO [70], and Co-DETR [6] are some of the most significant studies among these works.

Zhu et al. [5] introduce a deformable attention module that targets a small set of sampling locations to help identify key elements from all pixels in the feature map. This module can be easily expanded to combine features from different scales without needing Feature Pyramid Networks (FPN) [43]. They replace the transformer attention modules with multi-scale deformable attention modules for processing feature maps. The approach achieves convergence ten times faster. Additionally, they propose iterative refinement for bounding boxes and a two-stage Deformable DETR. This two-stage process generates region proposals using an encoder and sent to a decoder for further bounding box refinement, specifically step-by-step box refinement.

DINO [70] presents three techniques built upon various robust methods from different DETRs. Following DAB-DETR [67], queries in the decoder are formulated as dynamic anchor boxes, refining them progressively across decoder layers. Ground truth labels and bounding boxes with added noise are incorporated into the transformer decoder layers by DN-DETR [64] to stabilize the bipartite matching. Different from these models, DINO [70] proposes three key methods: First, contrastive denoising training is employed to mitigate duplicate predictions by introducing both positive and negative samples for the same ground truth. Noise is added to the ground truth box, with the box having smaller noise marked as positive and the other as negative. Second, DINO addresses the limitations of step-by-step box refinement from Deformable DETR by introducing a look-forward-twice scheme. Look-forward-twice refines parameters iteratively while including gradients from later decoder layers.

Lastly, DINO adopts a mixed query selection strategy, initializing positional queries using anchor boxes derived from the encoder's output while leaving content queries as learnable embeddings.

The most recent state-of-the-art model is Co-DETR [6], which employs a novel training scheme called collaborative hybrid assignments to introduce additional positive queries. Co-DETR can be easily integrated with Deformable DETR [5] and DINO [70]. Zong et al. [6] increase the number and variation of positive examples in the transformer head by utilizing one-to-many assignment strategies in anchor-based detectors as auxiliary heads. In our thesis, we will be integrating ranking-based losses into Co-DETR, and a detailed explanation of Co-DETR will be provided in Section 3.5.

2.2 Loss Functions for Object Detection

Loss functions play an important role in object detection due to the need for multi-task optimization. Described earlier, this task involves two main goals: classification, which identifies object categories, and localization, which predicts accurate bounding box coordinates. Each objective presents unique challenges that require specific loss functions to optimize performance and improve accuracy in both classification and localization tasks.

2.2.1 Score-based Loss Functions

2.2.1.1 Classification Losses

Cross-Entropy Loss. Cross-entropy loss [74] is a widely used loss function for classification tasks, including object detection. It measures the dissimilarity between the predicted probability distribution and the true class labels, penalizing incorrect predictions. Although Cross-Entropy Loss is a simple and well-known loss function, it struggles with a class imbalance in object detection, particularly for dense predictions, often favors common classes.

Formally, Cross-Entropy Loss for binary classification can be expressed as:

$$\mathcal{L}(p_t) = -\log(p_t), \tag{2.2}$$

where p_t represents the predicted probability for the ground-truth class and $p \in [0, 1]$ is the model's estimated probability for the class, defined as:

$$p_t = \begin{cases} p & \text{if the sample is positive,} \\ 1 - p & \text{if the sample is negative (background).} \end{cases} \tag{2.3}$$

Binary Cross-entropy can be adapted for multi-class classification, where each sample can belong to one of C classes by modifying the target vector into a one-hot vector that represents one positive class and $C - 1$ negative classes.

Cross-entropy penalizes all misclassifications equally, making it less effective in addressing problems with significant class imbalance.

Focal Loss. RetinaNet [19] introduced Focal Loss to tackle the problem of class imbalance in dense predictions, enabling the model to concentrate on hard-to-classify objects. Since its introduction, Focal Loss has become one of object detection's most widely used classification loss functions. It is known for its effectiveness in improving model performance across imbalanced datasets and its generalizability. Over time, several variants of Focal Loss have been proposed, such as Adaptive Focal Loss [75], Generalized Focal Loss [76] and it's variants [77], to further refine its application and adapt it to diverse detection tasks. Focal Loss dynamically scales the weight of each prediction based on how difficult it is, prioritizing hard examples.

Formally, Focal Loss can be defined as:

$$\mathcal{L}(p_t) = -\alpha_t(1 - p_t)^{\gamma} \log(p_t), \tag{2.4}$$

where p_t defined as it is in Cross-entropy loss, α_t is a balancing factor to address class imbalance, assigning different weights to positive and negative samples and $\gamma \geq 0$ is a parameter that reduces the loss for well-classified examples, allowing the model to focus on harder examples ($\gamma \neq 0$ differs Focal Loss from Cross-Entropy Loss).

2.2.1.2 Localization Losses

Localization losses measure the difference between the predicted and ground-truth bounding boxes. One of the most popular losses is the L1 loss and Smooth L1 Loss, which evaluates the absolute differences between the predicted and ground-truth bounding box coordinates. It is defined as:

$$\mathcal{L}_{L1} = |x_i - \hat{x}_i| + |y_i - \hat{y}_i| + |w_i - \hat{w}_i| + |h_i - \hat{h}_i|, \tag{2.5}$$

where (x_i, y_i, w_i, h_i) represents the ground-truth bounding box coordinates and the quadruple $(\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i)$ represents the predicted bounding box coordinates. L1 loss only considers the distance between two bounding boxes, which causes all errors to be treated equally. This can be suboptimal for object detection tasks where the quality of the predicted bounding box is critical. To address these limitations and measure the quality, several losses considering Intersection over Union (IoU) metric have been introduced. Specifically, for predicted bounding box $\hat{b}$ and ground-truth bounding box b, IoU Loss , defines the localization error as:

$$\mathcal{L}_{IoU} = 1 - \frac{\text{Intersection}(\hat{b},\, b)}{\text{Union}(\hat{b},\, b)}. \tag{2.6}$$

Several losses have also been introduced to improve IoU-based losses further. Namely, GIoU Loss (Generalized IoU Loss) [20], DIoU Loss (Distance IoU Loss) [21], and CIoU Loss (Complete IoU Loss) [78] extend the basic IoU metric by addressing limitations such as non-overlapping boxes, central point alignment, and aspect ratio consistency.

2.2.2 Ranking-based Loss Functions

Ranking-based losses differ from score-based losses in their optimization focus, particularly classification in object detection tasks. While score-based losses, such as Cross-Entropy or Focal Loss, aim to maximize the confidence of correct predictions and minimize it for incorrect ones independently, ranking-based losses emphasize the relative ordering of predictions. This means they prioritize ensuring that foreground objects are ranked higher than background objects. Unlike score-based losses, which treat predictions individually, ranking-based losses consider the relationships between

predictions, making them particularly effective for tasks like Non-Maximum Suppression (NMS), where the ranking of detections is crucial for performance. Furthermore, ranking-based loss functions offer significant advantages, such as easing the hyperparameter tuning process and directly optimizing the Average Precision (AP) performance metric. Though they have shown outstanding results in various object detection and instance segmentation tasks, it is important to recognize that they are computationally expensive than score-based losses.

Mohapatra et al. [79] was one of the first to focus on improving the efficiency of ranking-based losses specifically for Support Vector Machines (SVMs). They introduced a quicksort-inspired algorithm to optimize ranking losses with reduced computational complexity for AP and NDCG. However, their solutions have been primarily limited to training linear SVMs and consequently to problems that can be addressed using linear functions.

One of the main challenges with ranking-based losses is their non-differentiability due to the discrete nature of ranking. Several significant works have addressed this issue, proposing different solutions. Rolinek et al. [80, 81] introduced a framework based on black box backpropagation, leveraging combinatorial optimization. This method allows gradient computation through a non-differentiable black box implementation, enabling robust optimization of ranking losses. Additionally, DR Loss [82] presents a differentiable approach that utilizes Hinge Loss to maintain a margin between positive and negative predictions. Introduced by Chen et al., AP Loss [1, 2] is a ranking-based loss function that directly optimizes the AP using an error-driven update algorithm inspired by perceptron learning and backpropagation algorithms. A detailed explanation of AP Loss can be found in Section 3.1.2. Smooth AP [83] introduces a smooth, differentiable approximation of AP, with the assumption that $|\mathcal{N}|$ is not large.

Although Average Precision (AP) Loss is an important step in ranking-based losses, it primarily focuses on optimizing the ranking of predictions based on confidence scores and does not explicitly consider the quality of bounding box localization. To address this limitation, aLRP Loss [17] extends AP loss by incorporating localization quality into the optimization process. Specifically, aLRP builds upon the Localization-Recall-Precision (LRP) performance metric [84], just as AP loss expands the concept

of precision within a ranking-based loss function for classification. In 2021, Oksuz et al. [18] further enhanced the concept of ranking by considering not only the distinctions between positive and negative predictions but also among positive predictions themselves. With this approach, Rank&Sort Loss addresses a gap in both AP and aLRP Loss. By sorting positive predictions based on their continuous Intersection over Union (IoU) values, models can receive several advantages in terms of alignment with localization. As a result of prioritizing positives during training, detectors trained with RS Loss do not require an auxiliary head. Detailed explanation of RS Loss is provided in Section 3.1.5.

Rank&Sort Loss obtains state-of-the-art results in both one-stage and two-stage detectors, offering a great alternative to score-based losses. However, AP, aLRP, and RS Losses suffer significantly from time complexity due to the requirement for pairwise ranking. This complexity causes such a disadvantage of ranking-based loss functions prevent their applicability to larger visual detectors and recently proposed state-of-the-art transformer based-methods. In this thesis, we focus on improving AP Loss and RS Loss [1, 2, 18] that can be used for training complex deep networks such as object detectors.

2.3 Performance Evaluation

2.3.1 Average Precision

We evaluate our experiments with Average Precision (AP), the primary metric used to evaluate the performance of object detection models. It measures the area under the precision-recall curve across various confidence thresholds. For a more comprehensive evaluation, AP is reported at different Intersection over Union (IoU) thresholds, such as AP@50 and AP@75, and the mean AP across a range of IoU values from 0.5 to 0.95.

2.4 Datasets

We utilize several datasets to evaluate our proposed method and provide various insights. Initially, we conducted experiments using artificially generated toy data. A detailed description of the synthetic data generation process can be found in Section 4.1.1. For the majority of our experiments involving different detectors, we employed the COCO dataset [24], which is a large-scale dataset extensively used for object detection, segmentation, and image captioning. Additionally, we demonstrate the performance of our method on the instance segmentation task using the LVIS dataset [23], which highlights its effectiveness with long-tailed data. Furthermore, we include results from the Cityscapes dataset [85]. Detailed descriptions of these datasets are provided in the following sections.

COCO. The COCO (Common Objects in Context) [24] dataset is a large-scale dataset used for object detection, segmentation, and captioning tasks. We used the COCO dataset for object detection experiments. It includes 330,000 images, of which 200,000 are annotated specifically for object detection, segmentation, and captioning. The dataset features 80 different object categories.

Cityscapes. For our instance segmentation experiments, we utilized the Cityscapes [85] dataset. Cityscapes is a large-scale database designed for semantic understanding of urban street scenes. It offers semantic, instance-wise, and dense pixel annotations for 30 classes that are categorized into eight groups. The dataset contains approximately 5,000 finely annotated images.

LVIS. LVIS (Large Vocabulary Instance Segmentation) [23] is a dataset designed for long-tail instance segmentation tasks. It contains annotations for over 1,000 object categories across 164,000 images and emphasizes rare object classes. We extend our instance segmentation experiments further with LVIS to emphasize the effect of our method on the imbalanced dataset.

2.5 Comparative Summary

Object detection has advanced significantly, from traditional methods with hand-crafted components to modern deep learning-based models. CNN-based detectors [10,11,59,61] introduced end-to-end training and improved detection accuracy. Transformer-based models [4] simplified the pipeline with self-attention mechanisms and global feature extraction. However, these models faced challenges such as slow convergence and optimization difficulties, which were addressed in subsequent works [5, 63, 64, 67, 70].

In this thesis, we focus on ranking-based losses applied to both CNN and Transformer architectures. Our work is the first to address the issue of computational complexity associated with known ranking-based losses. We improved the efficiency of these losses, allowing for their integration into state-of-the-art transformer-based models, namely Co-DETR [6].

In Chapter 3, we first examine AP and RS Losses [1, 2, 18] and provide a detailed analysis explaining how we integrate our proposed method into ranking-based losses. Then, we extend the bucketed version of these losses to Co-DETR. Meanwhile, Chapter 4 demonstrates the comparative results of our experiments.

CHAPTER 3

BUCKETING RANKING-BASED LOSS FUNCTIONS

In this chapter, we present our novel approach of grouping negative samples into buckets to minimize the number of pairwise comparisons, thus addressing the time inefficiency issues associated with existing ranking-based losses. Our method reduces the time complexity to $\mathcal{O}(\max(N \log(N), P^2))$. First, we provide a background description of Average Precision (AP) Loss and Rank&Sort (RS) Loss. Next, we introduce our method in relation to these losses and discuss the theoretical foundations of our approach. Finally, we explain how we integrate our method with Co-DETR.

Some material in this chapter is adapted from our ECCV 2024 paper [3].

3.1 Background on Ranking-based Losses

In this section, we first define the problem with a suitable notation and provide the necessary background for our approach by reviewing Average Precision Loss [1, 2] and Rank&Sort Loss [18].

3.1.1 Problem Definition and Notation

We have an object detection problem wherein we are interested in finding the class label $c_i \in \mathcal{C}$ and the bounding box coordinates $\mathbf{b}_i \in \mathbb{R}^4$ for each object i in an input image. Given an image, an object detector generally provides many candidate object predictions where each prediction i has an associated confidence score over classes $\mathbf{s}_i \in \mathbb{R}^{|\mathcal{C}|}$ in addition to the box coordinates $\mathbf{b}_i$. An object detector is conventionally trained to minimize a weighted combination of a classification loss $\mathcal{L}_{cls}$ (e.g.,

Cross Entropy Loss [74], Focal Loss [19]) and a localization loss $\mathcal{L}_{loc}$ (e.g., Smooth-L1 Loss, Intersection-over-Union-based Losses [20, 21, 78]). During training in this conventional approach, each prediction is penalized based only on its closeness to its target.

Ranking-based approaches can be used for $\mathcal{L}_{cls}$ (e.g., in Average Precision Loss [1, 2]) or both $\mathcal{L}_{cls}$ and $\mathcal{L}_{loc}$ (in Rank&Sort Loss [18]). Unlike the conventional losses, ranking-based losses aim to rank the scores of positive detections above negative detections.

Ranking-based losses for object detection rely on pairwise comparisons between the scores of different detections to determine the rank of a detection among positives and negatives. Denoting the score of the i^{th} detection by s_i, we can compare the scores of two detections i and j with a simple difference transform: $x_{ij} = s_j - s_i$. By counting the number of instances with $x_{ij} > 0$, we can determine i^{th} detection's rank among positives ($\mathcal{P}$) and all detections (positives and negatives: $\mathcal{P} \cup \mathcal{N}$) as follows:

$$\text{rank}^+(i) = \sum_{j \in \mathcal{P}} \bar{H}(x_{ij}), \text{ and } \text{rank}(i) = \sum_{j \in \mathcal{P} \cup \mathcal{N}} \bar{H}(x_{ij}), \tag{3.1}$$

where $\bar{H}(x)$ is one if $x > 0$ and zero otherwise. Since $\mathrm{d}\bar{H}(x)/\mathrm{d}x$ is either infinite (at $x = 0$) or zero (for $x \neq 0$), a smoothed version is used (δ: a hyper-parameter):

$$H(x) = \begin{cases} 0\,, & x < -\delta \\ \dfrac{x}{2\delta} + 0.5\,, & -\delta \leq x \leq \delta \\ 1\,, & \delta < x \end{cases} \tag{3.2}$$

An illustration of this smoothed Heaviside step function can be found in Figure 3.1.

3.1.2 Revisiting Average Precision (AP) Loss

Given the definitions in Equation 3.1, AP Loss [1, 2] can be defined as:

$$\mathcal{L}_{AP} = 1 - \text{AP} = 1 - \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \text{precision}(i) = 1 - \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \frac{\text{rank}^+(i)}{\text{rank}(i)}. \tag{3.3}$$

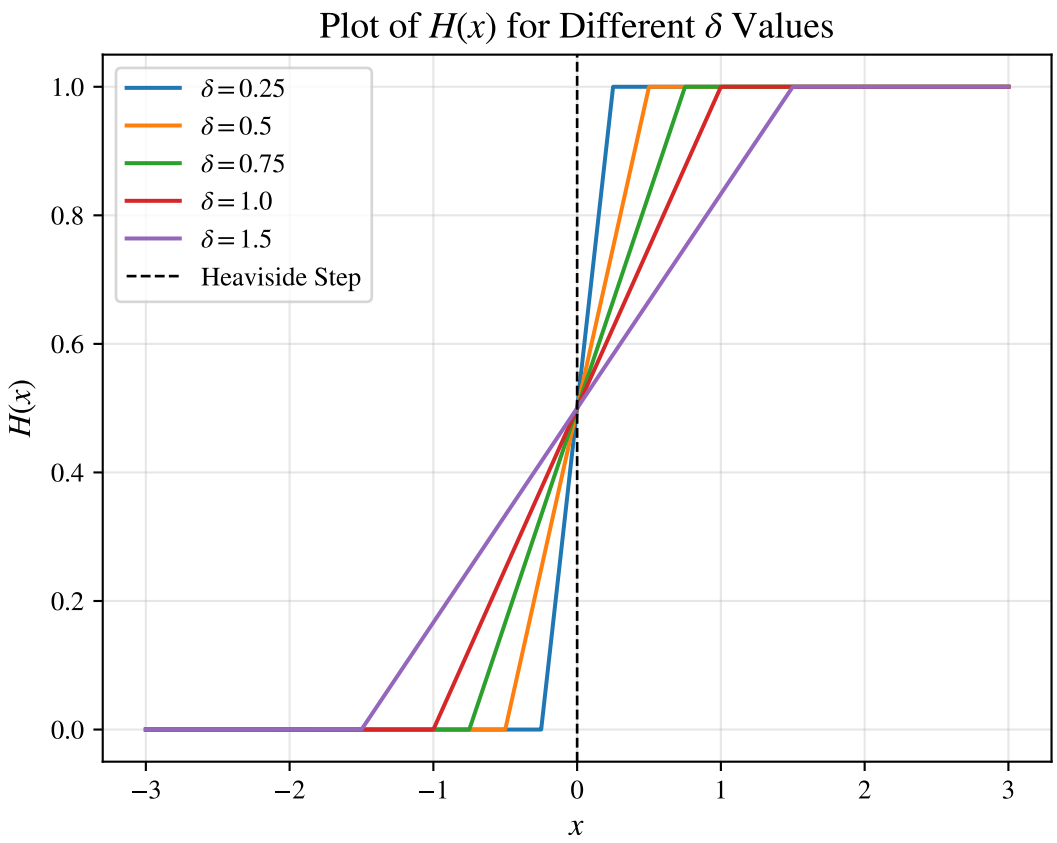


Figure 3.1: Plot of smoothed Heaviside step function with different delta values.

Chen et al. [1, 2] simplified Equation 3.3 by rewriting it in terms of pairwise comparisons between positive and negative detections:

$$\mathcal{L}_{AP} = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} \frac{H(x_{ij})}{\text{rank}(i)} = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} L_{ij}^{AP}, \tag{3.4}$$

where $L_{ij}^{AP} = H(x_{ij})/\text{rank}(i)$ is called the primary term. Note that L_{ij}^{AP} is non-differentiable since the step function ($H(\cdot)$) applied on x_{ij} is non-differentiable, which will be discussed next.

3.1.3 Revisiting Identity Update

Oksuz et al. [18] showed that various ranking-based losses (including AP Loss in Equations 3.3 & 3.4) can be written in a general form as:

$$\mathcal{L} = \frac{1}{Z} \sum_{i \in \mathcal{P} \cup \mathcal{N}} (\ell(i) - \ell^*(i)), \tag{3.5}$$

where $\ell(i)$ is the ranking-based error (e.g., precision error) computed on the ith detection, $\ell^*(i)$ is the target ranking-based error (the lowest error possible) and Z is the normalization constant.

The loss in Equation 3.5 can be computed and optimized as follows [18]:

1. Computation of the Loss. First, each pair of logits (s_i and s_j) are compared by calculating their difference transforms as $x_{ij} = s_j - s_i$. With the step function $H(\cdot)$ (Equation 3.2), the number of detections higher than s_i (and therefore its precision error, $\ell(i)$) can be easily calculated (see Section 3.1.2). Given x_{ij}, the loss can be re-written and calculated in terms of primary terms L_{ij} as:

$$\mathcal{L} = \frac{1}{Z} \sum_{i \in \mathcal{P} \cup \mathcal{N}} \sum_{j \in \mathcal{P} \cup \mathcal{N}} L_{ij}, \tag{3.6}$$

by taking:

$$L_{ij} = \left(\ell(i) - \ell^*(i)\right) p(j|i), \tag{3.7}$$

where $p(j|i)$ is a probability mass function (pmf) that distributes the error computed on the ith example over the jth example in order to determine the pairwise primary term L_{ij}. $p(j|i)$ is commonly taken as a uniform distribution [17, 18].

2. Optimization of the Loss. The gradient of the primary term wrt. the difference transform ($\partial L_{ij}/\partial x_{ij}$) is non-differentiable. Denoting this term by Δx_{ij}, we have [1, 2]:

$$\frac{\partial \mathcal{L}}{\partial s_i} = \sum_{j,k} \frac{\partial \mathcal{L}}{\partial L_{jk}} \frac{\partial L_{ij}}{\partial x_{ij}} \frac{\partial x_{jk}}{\partial s_i} = \sum_{j,k} \frac{\partial \mathcal{L}}{\partial L_{jk}} \Delta x_{jk} \frac{\partial x_{jk}}{\partial s_i} = \frac{1}{Z} \Big(\sum_j \Delta x_{ji} - \sum_j \Delta x_{ij} \Big). \tag{3.8}$$

Therefore, optimizing a ranking-based loss function reduces to determining Δx_{ij}. Inspired by Chen et al. [1, 2], Oksuz et al. [18] employ Perceptron Learning [86] and show that Δx_{ij} in Equation 3.8 is simply the primary term itself: $\Delta x_{ij} = -(L^*_{ij} - L_{ij}) = -(0 - L_{ij}) = L_{ij}$, hence the name Identity Update. Plugging this into Equation 3.8 yields (see Oksuz et al. [18] for the steps of the derivation):

$$\frac{\partial \mathcal{L}}{\partial s_i} = \frac{1}{Z} \Big(\sum_j L_{ji} - \sum_j L_{ij} \Big). \tag{3.9}$$

Therefore, both computing and optimizing the loss reduces determining the primary term L_{ij}.

3.1.4 Revisiting AP Loss with Identity Update

For defining AP Loss with identity update, the errors in Equation 3.5 can be derived as $\ell_{\mathrm{R}}(i) = \frac{N_{\mathrm{FP}}(i)}{\mathrm{rank}(i)}$, $\ell^*_{\mathrm{R}}(i) = 0$ and $Z = |\mathcal{P}|$. Furthermore, defining $p_R(j|i)$ as a uniform pmf, that is $p_R(j|i) = \frac{H(x_{ij})}{N_{\mathrm{FP}}(i)}$, we can obtain the primary terms of AP Loss using Identity Update, completing the derivation for computation and optimization (Equation 3.9):

$$L_{ij}^{AP} = \begin{cases} (\ell_{\mathrm{R}}(i) - \ell^*_{\mathrm{R}}(i))\, p_R(j|i), & \text{for } i \in \mathcal{P},\ j \in \mathcal{N}, \\ 0, & \text{otherwise}, \end{cases} \tag{3.10}$$

Replacing L_{ij} in Equation 3.9, the gradient of AP Loss [1, 2] is defined as follows:

$$\frac{\partial \mathcal{L}_{AP}}{\partial s_i} = \begin{cases} \frac{1}{|\mathcal{P}|} \sum_{j \in \mathcal{P}} \ell_R(j) p_R(i|j), & \text{for } i \in \mathcal{N} \\ -\frac{1}{|\mathcal{P}|} \ell_{\mathrm{R}}(i), & \text{for } i \in \mathcal{P}, \end{cases}$$

3.1.5 Revisiting RS Loss with Identity Update

In addition to AP Loss [1, 2], RS Loss [18] includes an additional sorting objective to promote better-localized positives to be ranked higher than other positives. RS Loss is the average difference between the current ($\ell_{\mathrm{RS}}(i)$) and the target ($\ell^*_{\mathrm{RS}}(i)$) RS errors over positives where labels for positives are defined as their IoU values, IoU_i:

$$\mathcal{L}_{\mathrm{RS}} := \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \left(\ell_{\mathrm{RS}}(i) - \ell^*_{\mathrm{RS}}(i) \right), \tag{3.11}$$

where $\ell_{\mathrm{RS}}(i)$ is a summation of the current ranking error and current sorting error and ($\ell^*_{\mathrm{RS}}(i)$) is a summation of the target ranking error and target sorting error:

$$\ell_{\mathrm{RS}}(i) := \underbrace{\frac{\mathrm{N}_{\mathrm{FP}}(i)}{\mathrm{rank}(i)}}_{\ell_{\mathrm{R}}(i)\text{: Current Ranking Error}} + \underbrace{\frac{\sum\limits_{j \in \mathcal{P}} \mathrm{H}(x_{ij})(1 - IoU_j)}{\mathrm{rank}^+(i)}}_{\ell_{\mathrm{S}}(i)\text{: Current Sorting Error}}. \tag{3.12}$$

When i belongs to the positive set $\mathcal{P}$, the current ranking error is calculated as the precision error. The current sorting error, on the other hand, assigns a penalty to the positives with logits greater than s_i. This penalty is proportional to the average of

Algorithm 1 AP Loss and RS Loss algorithms. Grey shows the additional operations of RS Loss compared to AP Loss.

Require: $\{s_i\}$, predicted logits and $\{t_i\}$, corresponding labels

Ensure: $\{g_i\}$, Gradient of loss wrt. input

$\forall i,\ \ g_i \leftarrow 0, \mathcal{P} \leftarrow \{i \mid t_i = 1\},\ \ \mathcal{N} \leftarrow \{i \mid t_i = 0\}$

$s_{\text{min}} \leftarrow \min_{i \in \mathcal{P}} \{s_i\}, \widehat{\mathcal{N}} \leftarrow \{i \in \mathcal{N} \mid s_i > s_{\text{min}} - \delta\}$

for $i \in P$ **do**

  $\forall j \in \mathcal{P} \cup \widehat{\mathcal{N}}, x_{ij} = s_j - s_i$

  Ranking error $\ell_{\text{R}}(i) = H(x_{ij})/N_{\text{FP}}(i)$ and $\ell^*_{\text{R}}(i) = 0$

  $\forall j \in \widehat{\mathcal{N}}, L_{ij} = \ell_{\text{R}}(i) \cdot p_{\text{R}}(j|i)$ ▷ Equation 3.10

  $\forall j \in \mathcal{P}$, Current sorting error $\ell_{\text{S}}(j)$ ▷ Equation 3.12

  $\forall j \in \mathcal{P}$, Target sorting error $\ell^*_{\text{S}}(j)$ ▷ Equation 3.13

  $\forall j \in \mathcal{P}, L_{ij} = (\ell_{\text{S}}(i) - \ell^*_{\text{S}}(i)) \cdot p_{\text{S}}(j|i)$ ▷ Equation 3.14

  Obtain gradient g_i for ith positive ▷ Equation 3.9

  Obtain gradients g_j for $\forall j \in \widehat{\mathcal{N}}$ ▷ Equation 3.9

end for

$\forall i,\ \ g_i \leftarrow g_i/|\mathcal{P}|$ ▷ Normalization

their inverted labels, $1 - IoU_j$:

$$\ell^*_{\text{RS}}(i) = \cancelto{0}{\ell^*_{\text{R}}(i)} + \underbrace{\frac{\sum_{j \in \mathcal{P}} \text{H}(x_{ij})[IoU_j \geq IoU_i](1 - IoU_j)}{\sum_{j \in \mathcal{P}} \text{H}(x_{ij})[IoU_j \geq IoU_i]}}_{\ell^*_{\text{S}}(i)\text{:Target Sorting Error}}, \tag{3.13}$$

where $[\cdot]$ denotes the Iverson Bracket. The target ranking of i, which is based on its desired position in the ranking, is compared to this measure. The target sorting error is calculated by averaging over the inverted IoU values of $j \in \mathcal{P}$ with larger logits and labels than $i \in \mathcal{P}$, corresponding to the desired sorted order.

Please note that the current ranking error and the target ranking error are the same when using the AP Loss definition.

Based on these definitions, we presented the following primary terms of RS Loss:

$$L_{ij} = \begin{cases} (\ell_{\mathrm{R}}(i) - \ell^*_{\mathrm{R}}(i))\, p_R(j|i), & \text{for } i \in \mathcal{P}, j \in \mathcal{N} \\ (\ell_{\mathrm{S}}(i) - \ell^*_{\mathrm{S}}(i))\, p_S(j|i), & \text{for } i \in \mathcal{P}, j \in \mathcal{P}, \\ 0, & \text{otherwise}, \end{cases} \tag{3.14}$$

where ranking and sorting errors on i are uniformly distributed with ranking pmf $p_R(j|i)$ and sorting pmf $p_S(j|i)$:

$$p_R(j|i) = \frac{\mathrm{H}(x_{ij})}{\sum\limits_{k \in \mathcal{N}} \mathrm{H}(x_{ik})}, \tag{3.15}$$

$$p_S(j|i) = \frac{\mathrm{H}(x_{ij})[IoU_j < IoU_i]}{\sum\limits_{k \in \mathcal{P}} \mathrm{H}(x_{ik})[IoU_k < IoU_i]}. \tag{3.16}$$

Similar to what we did for AP Loss, replacing L_{ij} in Equation 3.9, we obtain the gradient of RS Loss [18] for $i \in \mathcal{P}$ as follows:

$$\frac{\partial \mathcal{L}_{\mathrm{RS}}}{\partial s_i} = \frac{1}{|\mathcal{P}|}\Big(\underbrace{\ell^*_{\mathrm{RS}}(i) - \ell_{\mathrm{RS}}(i)}_{\text{Update signal to promote } i} + \underbrace{\sum_{j \in \mathcal{P}} (\ell_{\mathrm{S}}(j) - \ell^*_{\mathrm{S}}(j))\, p_S(i|j)}_{\text{Update signal to demote } i} \Big). \tag{3.17}$$

Doing the same for $i \in \mathcal{N}$ yields:

$$\frac{\partial \mathcal{L}_{\mathrm{RS}}}{\partial s_i} = \frac{1}{|\mathcal{P}|} \sum_{j \in \mathcal{P}} \ell_{\mathrm{R}}(j) p_R(i|j), \tag{3.18}$$

which completes the derivation of the gradients for RS Loss.

3.1.6 Complexity of Ranking-based Loss Functions

These loss functions originally have the space complexity of $\mathcal{O}((|\mathcal{P}|+|\mathcal{N}|)^2)$ [1,2] as they need to compare each pair. This quadratic space complexity makes it infeasible compute these loss functions using modern GPUs as the number of logits in object detection is very large. To alleviate that, Chen et al. [1, 2] introduced two tricks at the expense of making the computation inefficient: (1) Loop on Positives: By implementing a loop over the positive examples, Chen et al. managed to reduce the time complexity to $\mathcal{O}(|\mathcal{P}|(|\mathcal{P}| + |\mathcal{N}|)) \approx \mathcal{O}(|\mathcal{P}||\mathcal{N}|)$ since $\mathcal{N} \gg \mathcal{P}$ and the space complexity to $\mathcal{O}(|\mathcal{P}| + |\mathcal{N}|)$. (2) Discard Trivial Negatives: AP Loss considers

negative examples that meet the condition $s_j - s_i \leq -\delta$ for all $i \in \mathcal{P}$ and $j \in \mathcal{N}$, categorizing these instances as trivial negative samples. This may not lead to a significant improvement in speed during the initial stages of training, as almost all negative samples are non-trivial at that point. However, as the detector's performance improves,authors argued that the algorithm will gradually become faster.

3.2 Limitations of Existing Ranking-based Losses

Training object detectors with existing ranking losses require computing an error between each pair of examples (Figure 1.3), referred to as primary terms (as reviewed in Section 3.1.2). Ideally, these primary terms should be obtained in parallel, however, due to the large number of examples in object detection, it requires a huge amount of GPU memory to the extent that all primary terms do not fit into the memory of modern GPUs. For that reason, Chen et al. [1, 2] suggested the two aforementioned tricks for making AP Loss computation feasible. As one of the main tricks, instead of obtaining the errors between each pair, they employ a loop over the positive examples as indicated in the red box of 1. This loop decreases the space complexity of the algorithm, however, it makes the computation significantly inefficient, which results in up to $6\times$ slower training as in the case of Co-DETR [6].

- The number of negatives $|\mathcal{N}|$ is a prohibitive factor in these losses. For example, in Faster R-CNN [10], in the first epoch, $|\mathcal{P}| \sim 5.10^2$ whereas $|\mathcal{N}| \sim 10^6$.
- As $|\mathcal{N}|$ is large, the positive-negative pair-wise comparisons (to obtain L_{ij}) and subsequent derivations cannot be stored as matrices in memory. Therefore, ranking-based loss implementations use iterations, making them significantly slower compared to score-based losses.

3.3 Bucketed Ranking-based Loss Functions

Our main intuition is to bucket the sequential negative examples considering that they will have very similar or equal gradients. Figure 1.3(a) demonstrates this phenomenon in which n_1, n_2 both are assigned equal gradients as well as each nega-

tive among n_3, n_4, n_5, n_6 in the standard AP Loss. Formally, we sort all given logits $s_i \in \mathcal{P} \cup \mathcal{N}$ using a conventional sorting methods. Let us denote this sorted permutation of the logits by $\hat{s}_1$, $\hat{s}_2$, ..., $\hat{s}_{|\mathcal{P}\cup\mathcal{N}|}$, i.e., $\hat{s}_1 > \hat{s}_2 > .... > \hat{s}_{|\mathcal{S}|}$. Given these sorted logits and denoting the ith positive logit in the ordering by $\hat{s}_i^+$, the buckets of negatives $B_1, ..., B_{|\mathcal{P}+1|}$ can be obtained by:

$$B_1 > \hat{s}_1^+ > B_2 > \hat{s}_2^+ > ... > \hat{s}_{|\mathcal{P}|}^+ > B_{|\mathcal{P}+1|}. \tag{3.19}$$

We will denote the size of the ith negative bucket by b_i. Having created the buckets, we now determine a single logit value for each bucket, which we call as the prototype logit as it is necessary while assigning the ranking error. We denote the prototype logit for the i^{th} bucket by s_i^b. Note that if $\delta = 0$ in Equation 3.2, then any logit satisfying $\hat{s}_{i-1}^+ > s_i^b > \hat{s}_i^+$ can be a prototype logit as the ranking stays the same. However, in the case of $\delta > 0$, the boundaries between the logits are smoothed. That's why, practically, we find it effective to use the mean logit of a bucket as its prototype logit.

Note that this bucketing approach reduces the number of logits (positive and prototype negative) to a maximum of $2|\mathcal{P}| + 1$. As a result, the pairwise errors can now fit into the memory. Consequently, the loop in the red box of Algorithm 1 is no longer necessary, which gives rise to efficient ranking-based loss functions which we discuss in the following.

3.3.1 Bucketed Ranking-based Loss Functions

Here, we introduce Bucketed versions of AP and RS Losses. Please refer to Algorithm 2 for the algorithm.

3.3.1.1 Bucketed AP Loss

Definition. To define Bucketed version of AP Loss, we need to define current ($\ell_{\text{R}}^b(i)$) and target ranking errors ($\ell_{\text{R}}^{b,*}(i)$) for the ith positive. Similar to previous works [1, 2, 18], we use a target ranking error of 0, i.e., $\ell_{\text{R}}^{b,*}(i) = 0$. Defining the current ranking error, similar to Equation 3.3, requires $\text{rank}(i)$ and $N_{\text{FP}}(i)$. These two quantities can easily be defined using the bucket size b_i and the prototype logit. Different from the

Algorithm 2 Bucketed RS and AP Losses. Grey shows the additional operations of BRS Loss compared to BAP Loss.

Require: All scores $\{s_i\}$ and corresponding labels $\{t_i\}$

Ensure: Gradient of input $\{g_i\}$, ranking loss ℓ_{R}, sorting loss ℓ_{S}

Sort logits to obtain $\hat{s}_1$, $\hat{s}_2$, ..., $\hat{s}_{|S|}$.
Bucket consecutive negative logits to obtain $B_1, ..., B_{|\mathcal{P}+1|}$.
$\forall i \in \mathcal{P}, \forall j^b \in \tilde{\mathcal{N}}$, Calculate x_{ij}^b
$\forall i \in \mathcal{P}$, Calculate bucketed ranking error ℓ_{R}^b ▷ Equation 3.20
$\forall i \in \mathcal{P}, \forall j^b \in \tilde{\mathcal{N}}$, Calculate L_{ij}^b ▷ Equation 3.21
Calculate current sorting error ℓ_{S}^b ▷ Equation 3.12
Calculate target sorting error ℓ_{S}^{*b} ▷ Equation 3.13
$\forall i \in \mathcal{P}, \forall j \in \mathcal{P}$, $L_{ij} = (\ell_{\mathrm{S}}^b(i) - \ell_{\mathrm{S}}^{*b}(i)) \cdot p_{\mathrm{S}}(j|i)$ ▷ Equation 3.14
$\forall i \in \mathcal{P}$, obtain gradients g_i ▷ Equation 3.9
$\forall j \in \tilde{\mathcal{N}}$, find gradients for each prototype negative g_j^b ▷ Equation 3.9
$\forall j \in \tilde{\mathcal{N}}$, normalize g_j^b by bucket size b_j to obtain $g_i, \forall i \in \mathcal{N}$
$\forall i, \;\; g_i \leftarrow g_i/|\mathcal{P}|$ ▷ Normalization

conventional AP Loss, we obtain the pairwise relation $H(x_{ij}^b)$ using the prototype logit s_i^b. Particularly, with the ordering in Equation 3.19, $\mathrm{rank}(i) = \sum_{j=1}^{i} H(x_{ij}) + H(x_{ij}^b)b_j$ and $N_{\mathrm{FP}}(i) = \sum_{j=1}^{i} H(x_{ij}^b)b_j$ where $x_{ij}^b = s_j^b - s_i$. Then, the resulting ranking error computed on ith positive is:

$$\ell_{\mathrm{R}}^b(i) = \frac{N_{\mathrm{FP}}(i)}{\mathrm{rank}(i)} = \frac{\sum_{j=1}^{i} H(x_{ij}^b)b_j}{\sum_{j=1}^{i} H(x_{ij}) + H(x_{ij}^b)b_j}, \quad i \in \mathcal{P}. \tag{3.20}$$

Optimization. Here, we need to define the primary terms, as the product between the error and the pmf following Identity Update [18]. Unlike the previous work, the weights of each prototype negative are not equal as a bucket includes varying number of negatives. Therefore, we use a weighted pmf while distributing the ranking error over negatives. Formally, if j is the jth prototype negative, then we define the pmf as $p(j^b|i) = b_j/N_{\mathrm{FP}}(i)$. The resulting primary term between the ith positive and j^bth negative is then:

$$L_{ij}^b = \frac{\sum_{j=1}^{i} H(x_{ij}^b)b_j}{\sum_{j=1}^{i} H(x_{ij}) + H(x_{ij}^b)b_j} \times \frac{b_j}{N_{\mathrm{FP}}(i)}, \quad i \in \mathcal{P}, j \in \tilde{\mathcal{N}} \tag{3.21}$$

where $\tilde{\mathcal{N}}$ is the set of prototype negatives. However, we need the primary terms for the ith negative, not for the prototype ones. Given L_{ij}^b, the gradients of the prototype negatives and actual negatives can easily be obtained following Identity Update. However, we still need to find the gradients of the actual negatives. To do so, we simply normalise the prototype gradient by its bucket size b_j and distribute the gradients to actual negatives, completing our method.

The Gradients of Bucketed AP Loss. While obtaining the gradients of Bucketed AP (BAP) Loss, we follow the Identity Update framework, which basically requires the definition of the primary terms. The primary terms in Equation 3.21 can be more compactly stated as:

$$L_{ij}^b = \begin{cases} \left(\ell_R^b(i)p(j^b|i)\right), & i \in \mathcal{P}, j \in \tilde{\mathcal{N}}, \\ 0, & \text{otherwise}, \end{cases} \tag{3.22}$$

in which

$$\ell^b(i) = \frac{\sum_{j=1}^{i} \mathrm{H}(x_{ij}^b)b_j}{\sum_{j=1}^{i} \mathrm{H}(x_{ij}) + \mathrm{H}(x_{ij}^b)b_j}, \tag{3.23}$$

and $p(j^b|i) = \frac{b_j}{N_{\mathrm{FP}}(i)}$.

Following Identity Update, the gradients for the ith logit is defined as:

$$\frac{\partial \mathcal{L}}{\partial s_i} = \frac{1}{Z}\Big(\sum_j L_{ji} - \sum_j L_{ij}\Big). \tag{3.24}$$

Therefore, we need to replace L_{ij} by the primary term of the BAP Loss and set $Z = |\mathcal{P}|$ to find the gradients. First, we do this for the positive logit s_i in the following. Replacing L_{ij} by L_{ij}^b and for $i \in \mathcal{P}$ we obtain:

$$\frac{\partial \mathcal{L}_{\mathcal{BAP}}}{\partial s_i} = \frac{1}{|\mathcal{P}|}\left(\sum_j L_{ji}^b - \sum_j L_{ij}^b\right) \tag{3.25}$$

$$= \frac{1}{|\mathcal{P}|}\left(\sum_{j\in\mathcal{P}} L_{ji}^b + \sum_{j\in\tilde{\mathcal{N}}} L_{ji}^b - \left(\sum_{j\in\mathcal{P}} L_{ij}^b + \sum_{j\in\tilde{\mathcal{N}}} L_{ij}^b\right)\right) \tag{3.26}$$

Since there is no error defined among positives in AP and BAP loss, $\sum_{j\in\mathcal{P}} L_{ji}^b$ and $\sum_{j\in\mathcal{P}} L_{ij}^b = 0$. L_{ji}^b is also equal to zero from our primary term definition (Equation

3.22). Hence, Equation 3.26 becomes:

$$\frac{1}{|\mathcal{P}|}\left(\cancelto{0}{\sum_{j\notin\mathcal{P}} L^b_{ji}} + \cancelto{0}{\sum_{j\notin\tilde{\mathcal{N}}} L^b_{ji}} - \left(\cancelto{0}{\sum_{j\notin\mathcal{P}} L^b_{ij}} + \sum_{j\in\tilde{\mathcal{N}}} L^b_{ij}\right)\right) \tag{3.27}$$

$$= -\frac{1}{|\mathcal{P}|}\sum_{j\in\tilde{\mathcal{N}}} L^b_{ij} = -\frac{1}{|\mathcal{P}|}\sum_{j\in\tilde{\mathcal{N}}} \ell^b_R(i)p(j^b|i) \tag{3.28}$$

$$= -\frac{1}{|\mathcal{P}|}\ell^b_R(i)\sum_{j\in\tilde{\mathcal{N}}} p(j^b|i). \tag{3.29}$$

Considering that $\sum_{j\in\tilde{\mathcal{N}}} p(j^b|i) = 1$ as $p(j^b|i)$ is a probability mass function,

$$\frac{\partial\mathcal{L}_{\mathcal{BAP}}}{\partial s_i} = -\frac{1}{|\mathcal{P}|}\ell^b_R(i), \quad \text{if } i \in \mathcal{P} \tag{3.30}$$

Now we follow Identity Update considering that s_i is a *prototype-negative logit*, in which case we have:

$$\frac{\partial\mathcal{L}_{\mathcal{BAP}}}{\partial s_i} = \frac{1}{|\mathcal{P}|}\left(\sum_{j\in\mathcal{P}} L^b_{ji} + \cancelto{0}{\sum_{j\notin\tilde{\mathcal{N}}} L^b_{ji}} - \left(\cancelto{0}{\sum_{j\notin\mathcal{P}} L^b_{ij}} + \cancelto{0}{\sum_{j\notin\tilde{\mathcal{N}}} L^b_{ij}}\right)\right). \tag{3.31}$$

The three terms in the equation cancel to 0, considering that the primary term L_{ij} can be positive only in the case when i is a positive and j is a negative (more precisely, prototype negative in this case). Then, the following expression yields the gradients for the prototype negative:

$$\frac{\partial\mathcal{L}_{\mathcal{BAP}}}{\partial s_i} = \frac{1}{|\mathcal{P}|}\sum_{j\in\mathcal{P}} L^b_{ji} = \frac{1}{|\mathcal{P}|}\sum_{j\in\mathcal{P}} \ell^b_R(j)p(i^b|j). \tag{3.32}$$

However, we need to obtain the gradient for kth negative. To this end, we simply distribute the gradient considering the size of the bucket containing the kth negative, that is the ith bucket. Representing the bucket size by b_i, and with a uniform distribution from the ith prototype negative over the kth negative in the ith bucket, the gradient for the kth negative is then:

$$\frac{\partial\mathcal{L}_{\mathcal{BAP}}}{\partial s_k} = \frac{1}{|\mathcal{P}|}\sum_{j\in\mathcal{P}} \ell^b_R(j)p(i^b|j)\frac{1}{b_i}, \tag{3.33}$$

concluding the derivation of the gradients for Bucketed AP Loss.

3.3.1.2 Bucketed RS Loss

Given that we obtained Bucketed AP Loss, converting RS Loss into a bucketed form is more straightforward. This is because (i) the primary term and its gradients of

RS Loss are equal to those of AP Loss if $i \in \mathcal{P}$, $j \in \mathcal{N}$ and we simply use L_{ij}^b in such cases; and (ii) if $i \in \mathcal{P}$, $j \in \mathcal{P}$, then an additional term is included in RS Loss to estimate the pairwise relations between positives (Equation 3.14), which is not affected by bucketing.

3.3.2 The Gradients of Bucketed RS Loss

In this section, we describe how we apply our bucketing approach to RS Loss, first for the negative logits and then for the positive logits, similar to what we did for AP Loss in the previous section. Comparing Equation 3.18 and Equation 3.1.4, one can easily note that the gradient of the RS Loss is equal to the gradient of AP Loss if s_k is a negative logit. In this case, one can refer to our derivation for Bucketed AP Loss, as they are identical. Consequently, for $i \in \mathcal{N}$, using the same notation from the previous section, the gradient of the Bucketed RS Loss is:

$$\frac{\partial \mathcal{L}_{\mathcal{BRS}}}{\partial s_k} = \frac{1}{|\mathcal{P}|} \sum_{j \in \mathcal{P}} \ell_R^b(j) p(i^b|j) \frac{1}{b_i}. \tag{3.34}$$

For $i \in \mathcal{P}$, the gradient of RS Loss is expressed in Equation 3.17

$$\frac{1}{|\mathcal{P}|} \Big(\ell_{\mathrm{RS}}^*(i) - \ell_{\mathrm{RS}}(i) + \sum_{j \in \mathcal{P}} \left(\ell_{\mathrm{S}}(j) - \ell_{\mathrm{S}}^*(j) \right) p_S(i|j) \Big), \tag{3.35}$$

which can be decomposed as:

$$= \frac{1}{|\mathcal{P}|} \Big(\ell_{\mathrm{R}}^*(i) - \ell_{\mathrm{R}}(i) + \ell_{\mathrm{S}}^*(i) - \ell_{\mathrm{S}}(i) + \sum_{j \in \mathcal{P}} \left(\ell_{\mathrm{S}}(j) - \ell_{\mathrm{S}}^*(j) \right) p_S(i|j) \Big), \tag{3.36}$$

$$= \frac{1}{|\mathcal{P}|} \Big(- \ell_{\mathrm{R}}(i) + \ell_{\mathrm{S}}^*(i) - \ell_{\mathrm{S}}(i) + \sum_{j \in \mathcal{P}} \left(\ell_{\mathrm{S}}(j) - \ell_{\mathrm{S}}^*(j) \right) p_S(i|j) \Big). \tag{3.37}$$

Please note that calculating the sorting errors above ($\ell_{\mathrm{S}}(j)$, $\ell_{\mathrm{S}}^*(j)$) requires only positives, which can be efficiently computed. Then, the only term requiring the relations between the positives and the negatives is $\ell_{\mathrm{R}}(i)$. We simply replace this term by the bucketed ranking error $\ell_{\mathrm{R}}^b(i)$ in Equation 3.23. Consequently, the resulting gradient for the ith positive is:

$$\frac{1}{|\mathcal{P}|} \Big(- \ell_{\mathrm{R}}^b(i) + \ell_{\mathrm{S}}^*(i) - \ell_{\mathrm{S}}(i) + \sum_{j \in \mathcal{P}} \left(\ell_{\mathrm{S}}(j) - \ell_{\mathrm{S}}^*(j) \right) p_S(i|j) \Big). \tag{3.38}$$

This concludes the derivation of the gradients for Bucketed RS Loss. Please see Figure 3.2 for an example in which we illustrate obtaining the gradient of BRS Loss.

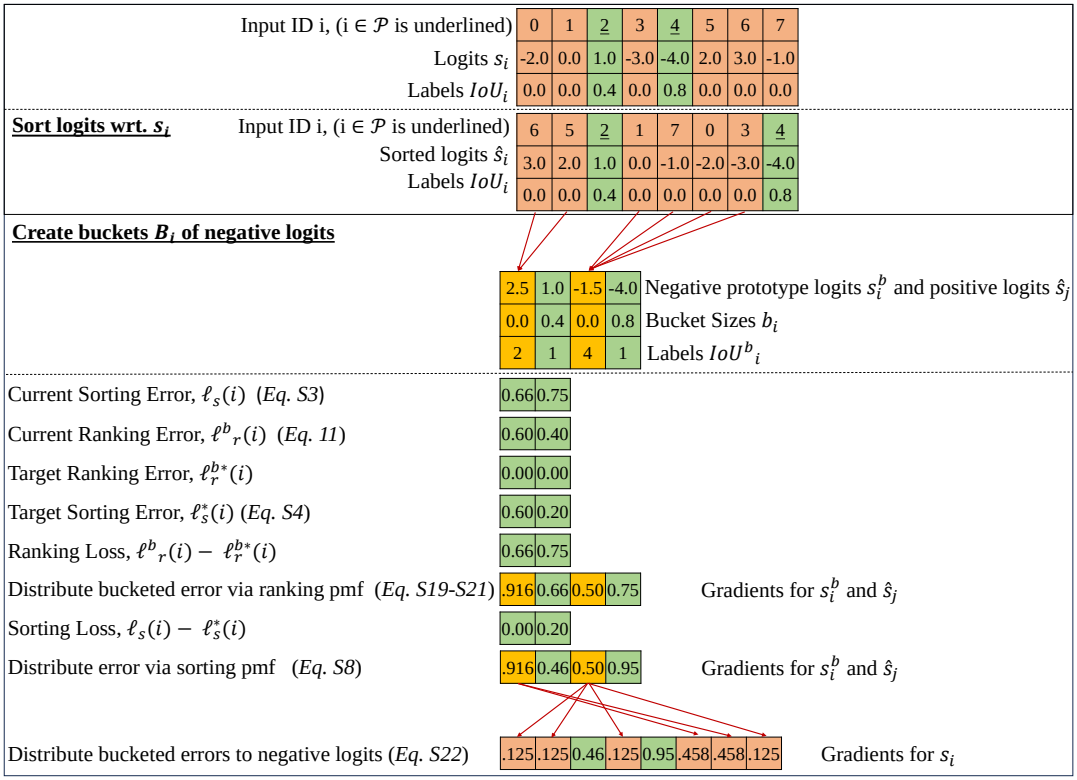


Figure 3.2: An example illustrating the computation of gradients in BRS Loss. The colors green, red, and orange represent positive, negative, and prototype negative logits, respectively. [Figure from our ECCV 2024 paper [3]]

3.4 Theoretical Discussion

The first theorem ensures that the gradients provided by our loss functions are identical with their conventional counterparts under certain circumstances. The second states that our algorithm is theoretically faster than the conventional AP and RS Losses.

Theorem. *Bucketed AP Loss and Bucketed RS Loss provide exactly the same gradients with AP and RS Losses respectively when* $\delta = 0$.

Proof. We divide the theorem into two parts, in which we first show that the gradients of Bucketed AP Loss reduce to those of AP Loss once $\delta = 0$. And this will be followed by RS Loss.

Case 1. On the equality of the gradients for AP Loss and Bucketed AP Loss. Similar

to how we obtained the gradients of AP Loss, we investigate this for positive and negative logits respectively:

Case 1a. s_i is a positive logit. In this case, the gradient of Bucketed AP Loss is defined in Equation 3.30 as follows:

$$\frac{\partial \mathcal{L}_{\mathcal{BAP}}}{\partial s_i} = -\frac{1}{|\mathcal{P}|}\ell_R^b(i), \quad \text{if } i \in \mathcal{P}, \tag{3.39}$$

where, by denoting the pair-wise relations between the ith positive and the jth prototype negative by $\mathrm{H}(x_{ij}^b)$ and the size of the jth bucket b_j, the bucketed ranking error is:

$$\ell_R^b(i) = \frac{\sum_{j=1}^{i} \mathrm{H}(x_{ij}^b) b_j}{\sum_{j=1}^{i} \mathrm{H}(x_{ij}) + \mathrm{H}(x_{ij}^b) b_j}, \tag{3.40}$$

$$= \frac{\sum_{j=1}^{i} \mathrm{H}(x_{ij}^b) b_j}{\sum_{j=1}^{i} \mathrm{H}(x_{ij}) + \sum_{j=1}^{i} \mathrm{H}(x_{ij}^b) b_j}. \tag{3.41}$$

Considering that $\mathrm{H}(x_{ij}^b) = 0$ for $j > i$, we can express the bucketed ranking error as follows:

$$\ell_R^b(i) = \frac{\sum_{j \in \tilde{\mathcal{N}}} \mathrm{H}(x_{ij}^b) b_j}{\sum_{j \in \mathcal{P}} \mathrm{H}(x_{ij}) + \sum_{j \in \tilde{\mathcal{N}}} \mathrm{H}(x_{ij}^b) b_j}. \tag{3.42}$$

Please note that in the case that $\delta = 0$, the step function $\mathrm{H}(\cdot)$ corresponds to Equation 3.2 in the following form:

$$H(x) = \begin{cases} 0\,, & x < 0 \\ 1\,, & 0 < x, \end{cases} \tag{3.43}$$

where we assume that no logits are exactly equal (i.e., $x \neq 0$) for the sake of simplicity. As a result, the term $\sum_{j \in \tilde{\mathcal{N}}} \mathrm{H}(x_{ij}^b) b_j$ both in the numerator and the denominator of Equation 3.42 can be expressed as:

$$\sum_{j \in \tilde{\mathcal{N}}} \mathrm{H}(x_{ij}^b) b_j = \sum_{j \in \mathcal{N}} H(x_{ij}). \tag{3.44}$$

This is because, for any prototype logit s_j^b, $s_{j-1} > s_j^b > s_j$ holds and the cardinality of the jth bucket is b_j. In other words, counting the number of negatives one-by-one as in the right-hand side of the equation versus counting each bucket size of negatives and summing over the bucket sizes (in the left-hand side expression) are equal. Furthermore, these expressions both yield the number of false positives on

the ith positive examples, i.e., $N_{\text{FP}}(i)$. Consequently, replacing this term in Equation 3.42 shows that the bucketed ranking error is equal to the ranking error when $\delta = 0$:

$$\ell_R^b(i) = \frac{\sum_{j \in \tilde{\mathcal{N}}} \text{H}(x_{ij}^b) b_j}{\sum_{j \in \mathcal{P}} \text{H}(x_{ij}) + \sum_{j \in \tilde{\mathcal{N}}} \text{H}(x_{ij}^b) b_j}, \tag{3.45}$$

$$= \frac{\sum_{j \in \mathcal{N}} H(x_{ij})}{\sum_{j \in \mathcal{P}} \text{H}(x_{ij}) + \sum_{j \in \mathcal{N}} H(x_{ij})}, \tag{3.46}$$

$$= \ell_R(i). \tag{3.47}$$

As a result:

$$\frac{\partial \mathcal{L}_{\mathcal{BAP}}}{\partial s_i} = -\frac{1}{|\mathcal{P}|} \ell_R^b(i) = -\frac{1}{|\mathcal{P}|} \ell_R(i) = \frac{\partial \mathcal{L}_{\mathcal{AP}}}{\partial s_i}, \quad \text{if } i \in \mathcal{P}, \tag{3.48}$$

Case 1b. s_i is a negative logit. For this case, the gradient expression of the Bucketed AP Loss is presented in Equation 3.33 as follows:

$$\frac{\partial \mathcal{L}_{\mathcal{BAP}}}{\partial s_i} = \frac{1}{|\mathcal{P}|} \sum_{j \in \mathcal{P}} \ell_R^b(j) p(k^b|j) \frac{1}{b_k}, \tag{3.49}$$

where k represents the bucket containing the ith negative. In case 1a, we have already shown that $\ell_R^b(j) = \ell_R(j)$ when $\delta = 0$. Hence, now we focus on the $p(i^b|j)\frac{1}{b_i}$, which reduces to:

$$p(k^b|j) \frac{1}{b_k} = \frac{b_k}{N_{\text{FP}}(j)} \frac{1}{b_i} = \frac{1}{N_{\text{FP}}(j)}. \tag{3.50}$$

Please note that, for a negative logit s_i with a higher value than s_j, which is the case for the probability mass function of AP Loss, $1 = \text{H}(x_{ij})$ holds. As a result:

$$\frac{1}{N_{\text{FP}}(j)} = \frac{\text{H}(x_{ij})}{N_{\text{FP}}(j)} = p(i|j). \tag{3.51}$$

Finally, replacing these terms in Equation 3.49, we have

$$\frac{\partial \mathcal{L}_{\mathcal{BAP}}}{\partial s_i} = \frac{1}{|\mathcal{P}|} \sum_{j \in \mathcal{P}} \ell_R^b(j) p(k^b|j) \frac{1}{b_k} \tag{3.52}$$

$$= \frac{1}{|\mathcal{P}|} \sum_{j \in \mathcal{P}} \ell_R(j) p(i|j) = \frac{\partial \mathcal{L}_{\mathcal{AP}}}{\partial s_i}, \tag{3.53}$$

completing the proof for Bucketed AP Loss.

Case 2. On the equality of the gradients for RS Loss and Bucketed RS Loss. Similar to how we obtained the gradients of AP Loss, we investigate this for the positive and negative logits respectively:

Case 2a. s_i is a positive logit. Please note that the gradient of RS Loss in this case is defined in Equation 3.38. As we have already shown in Case 1a that $\ell_R^b(j) = \ell_R(j)$ when $\delta = 0$ and we do not modify the remaining terms, this case holds.

Case 2b. s_i is a negative logit. Similarly, when the logit is negative, the gradient of RS Loss is equal to the gradient of AP Loss as we discussed in Section 3.3.2. Following from Case 1b, this case also holds, completing proof of the theorem. □

Theorem. *Bucketed RS and Bucketed AP Losses have* $\mathcal{O}(\max((|\mathcal{P}|+|\mathcal{N}|)\log(|\mathcal{P}|+|\mathcal{N}|), |\mathcal{P}|^2))$ *time complexity.*

Proof. We first summarize the time complexity for each line of 2 and then provide the derivation details.

Line in Algorithm 2	Time Complexity
Line 1	$\mathcal{O}((\lvert\mathcal{P}\rvert+\lvert\mathcal{N}\rvert)\log(\lvert\mathcal{P}\rvert+\lvert\mathcal{N}\rvert))$
Line 2	$\mathcal{O}(\lvert\mathcal{P}\rvert+\lvert\mathcal{N}\rvert)$
Line 3	$\mathcal{O}(\lvert\mathcal{P}\rvert^2)$
Line 4	$\mathcal{O}(\lvert\mathcal{P}\rvert^2)$
Line 5	$\mathcal{O}(\lvert\mathcal{P}\rvert^2)$
Line 6	$\mathcal{O}(\lvert\mathcal{P}\rvert^2)$
Line 7	$\mathcal{O}(\lvert\mathcal{P}\rvert^2)$
Line 8	$\mathcal{O}(\lvert\mathcal{P}\rvert^2)$
Line 9	$\mathcal{O}(\lvert\mathcal{P}\rvert^2)$
Line 10	$\mathcal{O}(\lvert\mathcal{P}\rvert^2)$
Line 11	$\mathcal{O}(\lvert\mathcal{P}\rvert+\lvert\mathcal{N}\rvert)$
Line 12	$\mathcal{O}(\lvert\mathcal{P}\rvert+\lvert\mathcal{N}\rvert)$
Overall	$\mathcal{O}(\max((\lvert\mathcal{P}\rvert+\lvert\mathcal{N}\rvert)\log(\lvert\mathcal{P}\rvert+\lvert\mathcal{N}\rvert), \lvert\mathcal{P}\rvert^2))$

Now we provide the derivation details for each line:

Line 1: Sorting logits. For sorting the logits, we use **torch.sort**, which internally employs the Quicksort implementation of NumPy (**np.sort**). Therefore, the time complexity of Line 1 is $\mathcal{O}\left(|\mathcal{S}|\log|\mathcal{S}|\right)$, where $|\mathcal{S}| = |\mathcal{P}\cup\mathcal{N}| = |\mathcal{P}|+|\mathcal{N}|$ is

the length of the array to be sorted. Since $|\mathcal{P}|$ is negligible ($|\mathcal{P}| \ll |\mathcal{N}|$), the time complexity at 1 can be approximated as $\mathcal{O}\left(|\mathcal{N}| \log |\mathcal{N}|\right)$.

Line 2: Bucketing. Bucketing the sorted array of $\hat{s}_1$, $\hat{s}_2$, ..., $\hat{s}_{|\mathcal{S}|}$ can be computed in one pass. Therefore, its time complexity is $\mathcal{O}(|\mathcal{S}|) \approx \mathcal{O}(|\mathcal{N}|)$ since the length of the pass is $\mathcal{S}| = |\mathcal{P} \cup \mathcal{N}| = |\mathcal{P}| + |\mathcal{N}|$ and $|\mathcal{P}| \ll |\mathcal{N}|$.

Line 3: Difference Terms. For one $i \in \mathcal{P}$ and $\forall j^b \in \tilde{\mathcal{N}}$, the time complexity of the term $x_{ij}^b = s_j^b - s_i$ is constant, i.e. $\mathcal{O}(1)$. Therefore, the worst-case time complexity of Line 3 is $\mathcal{O}(|\mathcal{P}| \cdot |\tilde{\mathcal{N}}|) \approx \mathcal{O}(|\mathcal{P}|^2))$ since $|\tilde{\mathcal{N}}| \leq |\mathcal{P}| + 1$.

Line 4: Current Bucketed Ranking Error. The worst-case scenario for the current ranking error ℓ_{R}^b is when $i = |\tilde{\mathcal{S}}| = 2|\mathcal{P}| + 1$ since the worst-case scenarios for both $\mathrm{N_{FP}}(i) = \sum_{j=1}^{i} \mathrm{H}(x_{ij}^b) b_j$ and $\mathrm{rank}(i) = \sum_{j=1}^{i} \mathrm{H}(x_{ij}) + \mathrm{H}(x_{ij}^b) b_j$ are when $i = |\tilde{\mathcal{S}}| = 2|\mathcal{P}| + 1$. That is, the time complexity of $\mathrm{N_{FP}}(|\tilde{\mathcal{S}}|)$ and $\mathrm{rank}(|\tilde{\mathcal{S}}|)$ is $\mathcal{O}(|\tilde{\mathcal{S}}|)$ = $\mathcal{O}(|\mathcal{P}|)$, so is the time complexity of the current ranking error. Therefore, computing the current ranking loss for $|\mathcal{P}|$ positive logits has a time complexity of $\mathcal{O}(|\mathcal{P}|^2)$.

Line 5: Bucketed Primary Terms. The target ranking error $\ell_{\mathrm{R}}^{*b} = 0$, making its complexity constant, i.e. $\mathcal{O}(1)$. Therefore, the computation of the primary terms L_{ij}^b = $\left(\ell_{\mathrm{R}}(i) - \ell_{\mathrm{R}}^{*}(i)\right) p_R(j|i)$ for every $i \in \mathcal{P}$ and every $j \in \tilde{\mathcal{N}}$ has a worst-case time complexity of $\mathcal{O}(|\mathcal{P}| \cdot |\tilde{N}|)$. Since $|\tilde{\mathcal{N}}| \leq |\mathcal{P}| + 1$, the complexity of Line 5 can be approximated as $\mathcal{O}(|\mathcal{P}|^2)$.

Line 6 and Line 7: Bucketed Sorting Error. The time complexity of the current sorting error ℓ_{S}^b at the worst case is the maximum of time complexities of the numerator and denominator terms in Equation 3.12. The numerator is the summation of terms with constant time complexities. The denominator, $\mathrm{rank}^{+}(i)$, has a time complexity of $\mathcal{O}(|\mathcal{P}|)$ in the worst case, similar to $\mathrm{rank}(i)$. Therefore, the worst-case time complexity of the current sorting error is $\mathcal{O}(|\mathcal{P}|)$. Similar to the current sorting error, both the numerator and the denominator of Equation 3.13 are summations of the terms with constant time complexities. Therefore, the worst-case time complexity of the target sorting error is $\mathcal{O}(|\mathcal{P}|)$ as well. Lastly, computing the current sorting loss ℓ_{S}^b and the target sorting loss ℓ_{S}^{*b} for every positive logits corresponds to $\mathcal{O}(|\mathcal{P}|^2)$.

Line 8: Primary Terms. Similar to Line 5, the computation of the primary terms $L_{ij} = \left(\ell_{\mathrm{S}}^{b}(i) - \ell_{\mathrm{S}}^{*b}(i)\right) p_S(j|i)$ for every $i, j \in \mathcal{P}$ has a worst-case time complexity of $\mathcal{O}(|\mathcal{P}| \cdot |\mathcal{P}|) = O(|\mathcal{P}|^2)$.

Line 9: Final Gradients for Positive Logits. Calculating the final gradients $\frac{\partial \mathcal{L}}{\partial s_i} = \frac{1}{Z} \sum_{j \in \tilde{\mathcal{S}}} \left(L_{ji} - L_{ij}\right)$ in Equation 3.24 has a time complexity of $\mathcal{O}(|\tilde{\mathcal{S}}|) = \mathcal{O}(|\mathcal{P}|)$ since the primary terms are already calculated in Line 8 and $|\tilde{\mathcal{S}}| \leq 2|\mathcal{P}| + 1$. Therefore, computing the final gradients for every positive logits is $\mathcal{O}(|\mathcal{P}|^2)$.

Line 10: Final Gradients for Negative Prototypes. Similar to Line 9, the time complexity of computing final gradients $\frac{\partial \mathcal{L}}{\partial s_i}$ for every negative prototype logits is $\mathcal{O}(|\tilde{\mathcal{N}}| \cdot |\mathcal{P}|) = \mathcal{O}(|\mathcal{P}|^2)$ since $|\tilde{\mathcal{N}}| \leq |\mathcal{P}| + 1$.

Line 11: Final Gradients for Negative Logits. Distributing the final gradients computed for every negative prototype logits, i.e. $\tilde{\mathcal{N}}$, to negative logits, i.e. $\mathcal{N}$, has a time complexity of $\mathcal{O}(|\mathcal{N}|)$, since it can be done in one pass over the sorted array $\hat{s}_1$, $\hat{s}_2$, ..., $\hat{s}_{|\mathcal{S}|}$ and related gradients for negative prototype logits are computed already in Line 11.

Line 12: Normalization. Applying normalization operation, with time complexity $\mathcal{O}(1)$, for each gradient corresponding to the logits in the sorted array $\hat{s}_1$, $\hat{s}_2$, ..., $\hat{s}_{|\mathcal{S}|}$ has a time complexity of $\mathcal{O}(|\mathcal{S}|) = \mathcal{O}(|\mathcal{N}|)$ since $|\mathcal{S}| = |\mathcal{P} \cup \mathcal{N}| = |\mathcal{P}| + |\mathcal{N}|$ and $|\mathcal{P}| \ll |\mathcal{N}|$.

Overall time complexity of Algorithm 2. The time complexity of the whole algorithm is the sum of the time complexity at each line, which equals to the time complexity of the line with the maximum time complexity. That is, $\mathcal{O}(\max((|\mathcal{P}| + |\mathcal{N}|)\log(|\mathcal{P}| + |\mathcal{N}|), |\mathcal{N}|, |\mathcal{P}|^2)) = \mathcal{O}(\max((|\mathcal{P}| + |\mathcal{N}|)\log(|\mathcal{P}| + |\mathcal{N}|), |\mathcal{P}|^2))$. □

3.5 Bucketed Rank-Sort (BRS) DETR

While transformer-based detectors [4–6, 63, 64, 66, 67, 70] have been providing the best performance in several object detection benchmarks, optimizing such detectors using ranking-based losses based on performance measures has not been investigated. Here, we address this gap by incorporating our BRS Loss into Co-DETR [6] as the

current SOTA detector on the common COCO benchmark [24].

Co-DETR [6] introduces a collaborative hybrid assignment training scheme. Zong et al. [6] focus on the intuitive drawback of one-to-one set matching in DETR-based models: it explores less positive queries. This will lead to severe inefficient training issues. They analyze this from two aspects, the latent representation generated by the encoder and the attention learning in the decoder. The key insight of Co-DETR is to use one-to-many label assignments to improve the training efficiency and effectiveness of both the encoder and decoder. By using auxiliary heads, Co-DETR increases the number and variation of the positive examples in the transformer head by leveraging one-to-many assignment strategies in anchor-based detectors (such as ATSS [7] and Faster R-CNN [10]) as auxiliary heads. As the number of proposals is large in such detectors, they can easily provide more and diverse examples, resulting in better performance of the transformer head. Through extensive experiments, the number of auxiliary heads is set to 2, specifically using ATSS [7] and Faster R-CNN [10] as the auxiliary heads. To enhance clarity, one can refer to Figure 3.3. Please note that auxiliary heads are only used during the training phase.

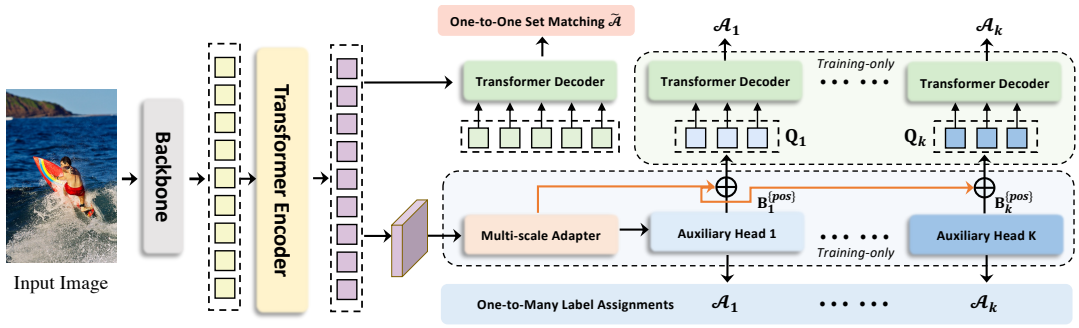


Figure 3.3: Architecture of Co-DETR. Elements shared with DETR [4] and Deformable DETR [5] (e.g., the backbone, Transformer encoder-decoder, and set matching) remain unchanged. Key additions include a multi-scale adapter to better capture features at different resolutions, as well as new auxiliary heads that support extra positive queries. [Figure from [6]]

The Co-DETR pipeline can be outlined as follows: Like other DETR-based detectors, it begins by extracting features from a CNN backbone, after which the encoder generates representations based on these features. These representations are then trans-

formed and fed into auxiliary heads to create additional positive sets known as extra positive queries. Queries are subsequently introduced to the decoder for extra supervision. It is important to note that Co-DETR employs a two-stage approach followed by Deformable-DETR [5]. Given that the pipeline includes several optimization objectives, the overall loss function can defined as:

$$\sum_{l=1}^{L}\left(\tilde{\mathcal{L}}_{l}^{Dec}+\lambda_{1}\sum_{k=1}^{K}\tilde{\mathcal{L}}_{k,l}^{Aux}+\lambda_{2}\sum_{k=1}^{K}\mathcal{L}_{k,l}^{Aux}\right), \tag{3.54}$$

where λ_1 and λ_2 weigh each loss component. The first two components, $\tilde{\mathcal{L}}_{l}^{Dec}$ and $\tilde{\mathcal{L}}_{k,l}^{Aux}$, are the losses of the l^{th} decoder layer in the following form:

$$\lambda_{cls}\mathcal{L}_{cls}+\lambda_{bbox}\mathcal{L}_{bbox}+\lambda_{IoU}\mathcal{L}_{IoU}, \tag{3.55}$$

and $\mathcal{L}_{cls}$, $\mathcal{L}_{bbox}$ and $\mathcal{L}_{IoU}$ are Focal Loss, L1 Loss and GIoU Loss respectively [6]. $\tilde{\mathcal{L}}_{l}^{Dec}$ and $\tilde{\mathcal{L}}_{k,l}^{Aux}$ differ in their input queries. That is, while $\tilde{\mathcal{L}}_{l}^{Dec}$ follows standard DETR-based models [4,5] with one-to-one matching of the queries, $\tilde{\mathcal{L}}_{k,l}^{Aux}$ is computed based on positive anchors of k^{th} auxiliary head with one-to-many assignment. And $\mathcal{L}_{k,l}^{Aux}$ is the conventional loss of the k^{th} auxiliary head, e.g., the weighted sum of classification, localisation and centerness losses for ATSS [7].

In order to align the training and evaluation objectives better, we replace $\tilde{\mathcal{L}}_{l}^{Dec}$ and $\tilde{\mathcal{L}}_{k,l}^{Aux}$ in Equation 3.55 by:

$$\mathcal{L}_{BRS}+\lambda_{bbox}\mathcal{L}_{bbox}+\lambda_{IoU}\mathcal{L}_{IoU}, \tag{3.56}$$

and set λ_{bbox} and λ_{IoU} dynamically during training to $\mathcal{L}_{BRS}/\mathcal{L}_{bbox}$ and $\mathcal{L}_{BRS}/\mathcal{L}_{IoU}$ to simplify the hyperparameter tuning following Oksuz et al. [18]. Similarly, we replace the loss functions of the auxiliary ATSS [7] and Faster R-CNN [10] heads ($\mathcal{L}_{k,l}^{Aux}$ in Equation 3.54) by our BRS Loss. This modification also aligns with the objective of the auxiliary heads by the performance measure, hence results in more accurate auxiliary heads. Furthermore, as using BRS Loss with Faster R-CNN does not require sampling thanks to its robustness to imbalance, no limitation is imposed on the number of positives in Faster R-CNN. Hence, the main aim of Co-DETR, that is to introduce more positive examples to the transformer head, is corroborated. As a result, our BRS DETR enables significantly efficient training compared to RS Loss also by improving the performance of Co-DETR.

CHAPTER 4

EXPERIMENTS

In this chapter, we evaluate the effectiveness of our bucketing method across different backbones and datasets. To comprehensively analyze the effectiveness of our bucketing method, we design experiments on real-world and synthetic data. For the former, we aim to present that our method significantly decreases the training time of the detection and segmentation methods (up to $\sim 5\times$). As training the entire detector does not isolate the runtime of the loss function, on which our main contribution is, we also design an experiment with synthetic data. This set of experiments show that our bucketing approach decreases the loss function runtime by up to $40\times$, thereby resulting in shorter training time of the detectors.

We analyze our contributions in three primary sections. First, we evaluate the effectiveness of our bucketing approach by contrasting it with RS Loss [18] and AP Loss [1, 2] across different CNN-based visual detectors, on object detection and instance segmentation tasks. Next, we compare score-based loss functions and other ranking-based loss functions in the literature. Finally, we thoroughly examine how our bucketed RS Loss performs on transformer-based object detectors, specifically Co-DETR [6] with different backbones. This is the first time a ranking-based loss is applied to transformer-based object detectors, thanks to the time efficiency of our bucketing method. Our experiments demonstrate that our BRS loss is preferable to any existing loss function in training object detectors in terms of training time, accuracy, and ease of tuning. Some material in this chapter is adapted from our ECCV 2024 paper [3].

4.1 Training Details

4.1.1 Dataset and Performance Measures

We initially evaluated our method using synthetic data. Next, for object detection, unless otherwise noted, we train all models on COCO *trainval35k* (115k images) and test them on *minival* (5k images). We use COCO-style Average Precision (AP) and also report AP_{50}, AP_{75} as the APs at IoUs 0.50 and 0.75; and AP_{S}, AP_{M} and AP_{L} to present the accuracy on small, medium and large objects. Additionally, we also perform experiments on the LVIS and Cityscapes datasets, specifically for instance segmentation. Please note that AP_{75} is not reported in the Cityscapes dataset.

Synthetic Data Generation. In our analyses using the synthetic data, we generate logits with different cardinalities $L = \{10K, 100K, 1M\}$ also for various percentages of positives $m = \{0.1, 1.0, 2.0, 5.0\}$. Specifically, we sample positive and negative logits from the Gaussian distributions, $s_i^+ \sim \mathcal{N}(-1, 1)$ and $s_i^- \sim \mathcal{N}(1, 1)$ respectively. Please note that the mean of the negative logits is higher than that of the positive logits. This ensures that the sampled set of logits is likely to include trivial cases in which all the positives have higher confidence than all the negatives, and consequently, the set of negative logits is empty and the loss computation is trivial. For RS and Bucketed RS-Loss experiments, we generated uniformly distributed random IoU values for positive logits.

CNN-based Visual Detectors. In order to comprehensively demonstrate the efficiency of our approach, we use five different detectors: Faster R-CNN [10], Cascade R-CNN [11], ATSS [7] and PAA [8] for object detection, as well as Mask R-CNN [61] for instance segmentation. For all methods evaluated on both COCO [24] and LVIS [23], we adopt the experimental setup from RSLoss [18], only replacing the loss function with our bucketed loss function to maintain comparability and consistency of results.

Specifically, similar to RS Loss [18], while training multi-stage detection and segmentation methods, i.e., Faster R-CNN [10], Cascade R-CNN [11] and Mask R-CNN [61], we do not use random sampling different from the original architecture.

We use all anchors for RPN and top-1000 proposals for Faster R-CNN [10] and top-2000 proposals for Cascade R-CNN [11]. Again similar to RS Loss, we replace softmax classifier with class-wise binary sigmoid classifiers. For multi-stage methods, we start with an initial learning rate of 0.012, while for one-stage methods, we set it at 0.008. After the 8th and 11th epochs, we reduce the learning rate by a factor of 10, as the models are trained for a total of 12 epochs. We train the ATSS [7] and PAA [8] methods without using centerness head. During the training of PAA, we maintain the scoring function while splitting positives and negatives. For the RS Loss and AP Loss, we use the same scheduling and configuration to ensure a fair comparison.

We utilize the MMDetection [87] framework and distribute our computations across 4 Tesla A100 GPUs. Each GPU processes 4 images concurrently, leading to a total batch size of 16.

Additionally, for experiments with Cityscapes [85], we follow the Mask R-CNN [61] configuration provided in MMDetection and replace the loss function with Bucketed RS Loss and RS Loss.

BRS-DETR. We incorporate our BRS Loss into the official Co-DETR repository to be aligned with the original implementation and keep its settings unless otherwise explicitly stated. We train our BRS-DETR for 12 epochs on 8 Tesla A100 GPUs, with each GPU processing 2 images, resulting in a total batch size of 16. Testing is conducted at a single scale with images of 1333×800 size. We initialize the learning rate to 0.0002 and decrease it by a factor of 5 after epochs 10 and 11. We employ the setting of Co-DETR with 300 queries. As for the auxiliary heads, we similarly use ATSS [7] and Faster-RCNN [10]. One key difference when using ranking-based losses is that they do not rely on sampling, since they are robust to long-tailed data. In line with RS-R-CNN, we eliminate sampling in Faster R-CNN, and similarly with RS-ATSS, we remove the centerness head from ATSS. For both auxiliary heads, we replace the classification loss with BRS Loss and incorporate the self-balancing techniques proposed by Oksuz et al. [18].

The original Co-DETR employed a two-stage method from Deformable DETR [5] in the transformer head, so they have encoder and decoder losses. In both the encoder and each layer of the decoder, we change the classification losses (Focal Loss) to

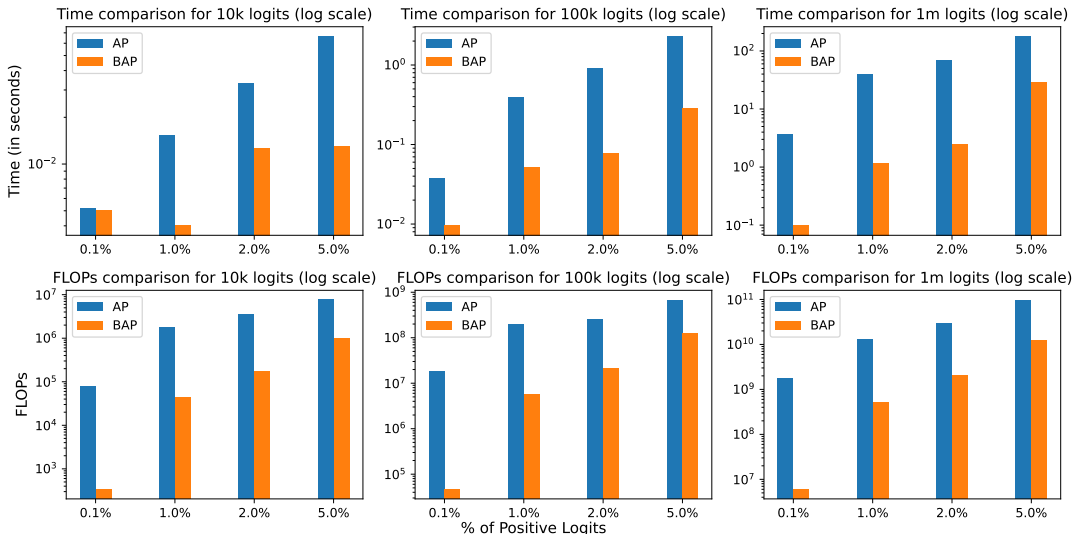


Figure 4.1: Performance Comparison of AP and BAP Loss Functions: Log-Scale Analysis of Computational Time and number of floating point operations (FLOPs) across various data cardinalities and percentages. [Figure from our ECCV 2024 paper [3]]

BRS-Loss. Co-DETR [6] employs L1 Loss and GIoU Loss for localization so that we keep these losses for localization. As described in Section 3.5, we also employ self-balancing in Co-DETR localization losses. Finally, once our loss function is used, we find it useful (i) to increase the classification loss weight used for the Hungarian assignment of the proposals as positives or negatives from 2 to 4 and (ii) to decrease the auxiliary head weights from 12 to 5.

4.2 Experiments on Bucketed Losses with Synthetic Data

Since our method only focuses on the loss computation step, it is essential to demonstrate its effectiveness independently from the full pipeline of the detectors, as the training pipeline includes several steps. Hence, an analysis that considers only this part can reveal our contribution more clearly. Therefore, we design a simple experiment using synthetic data. As described in Training Details, we randomly generate L logits such that $m\%$ of these logits are positive. For the sake of simplicity, here we compare our BAP Loss with AP Loss in terms of time efficiency and number of floating point operations (FLOPs) and report the mean iteration time of three inde-

pendent runs with each setting. To illustrate the extreme case, one might consider $L = 1M$ logits with $m = 0.1$ correspond to anchor-based detector making dense predictions such as RetinaNet [19], the smaller number of logits mimics the R-CNN head of two-stage detectors, and finally $L = 10K$ with higher m approximates recent transformer-based detectors.

Figure 4.1 shows that our bucketing approach, in fact, improves loss efficiency by up to $\sim 40\times$, a very significant improvement. This enhancement becomes increasingly evident as the total number of logits increases. Furthermore, when we focus on the FLOPs of the AP loss, it decreases from 76M to 900k with our BAP loss, introducing more than $80\times$ less FLOPs and further validating the effectiveness of our approach.

Table 4.1: Bucketed Losses (BRS, BAP) vs. RS & AP Losses on COCO object detection for one-stage object detectors.

Detector	Backbone	Loss	Time	Accuracy		
			$T_{iter}(s) \downarrow$	AP $\uparrow$	AP_{50} $\uparrow$	AP_{75} $\uparrow$
ATSS [7]	ResNet50	AP	0.32	38.1	58.2	41.0
		BAP	**0.15** 2.1x $\downarrow$	38.5	58.6	41.3
		RS	0.36	39.8	58.9	42.6
		BRS	**0.15** 2.4x $\downarrow$	39.8	58.8	42.9
PAA [8]		AP	0.45	37.3	54.3	41.2
		BAP	**0.30** 1.5x $\downarrow$	37.2	56.2	40.2
		RS	0.57	40.8	58.8	44.6
		BRS	**0.30** 1.9x $\downarrow$	40.9	59.0	44.4

4.3 Experiments on Bucketed Losses with One Stage Detectors

To show that our gains generalize to one-stage detectors, we train the common ATSS and PAA detectors with our BAP and BRS Losses and compare them with AP and RS Losses. To show that our gains generalize to one-stage detectors, we train the common ATSS and PAA detectors with our BAP and BRS Losses and compare them with AP and RS Losses.

Table 4.2: BRS Loss vs. RS Loss on COCO object detection for multi-stage object detectors. Models with R-101 are trained 36 epochs.

Detector	Backbone	Loss	Time	Accuracy		
			$T_{iter}(s) \downarrow$	AP $\uparrow$	AP_{50} $\uparrow$	AP_{75} $\uparrow$
Faster R-CNN [10]	ResNet50	RS	0.50	39.4	59.5	43.0
		BRS	**0.17** 3.0x $\downarrow$	39.5	59.5	42.8
	ResNet101	RS	0.75	47.3	67.4	51.2
		BRS	**0.38** 2.0x $\downarrow$	47.7	67.8	51.5
Cascade R-CNN [11]	ResNet50	RS	1.28	41.1	58.7	44.1
		BRS	**0.24** 5.3x $\downarrow$	41.1	58.8	44.2

In Table 4.1, we compare both AP Loss with its bucketed version (BAP) and RS Loss with its bucketed version (BRS)on one-stage object detectors. Table 4.1 validates our previous claims: Our bucketed losses obtain similar performance in around half of the training time required for AP and RS Losses. Compared to AP Loss, BAP Loss consistently speeds up training (from $1.5\times$ to $2.1\times$) while preserving AP scores.

Similar to the AP Loss experiments, leveraging our tailored gradient adjustment method, we have secured Average Precision (AP) scores on par with RSLoss [18]. As can be seen from Table 4.1, we have also managed to cut down the training time by about 2.3 times for ATSS [7] and nearly 1.9 times for PAA [8].

4.4 Experiments on BRS Losses with Multi Stage Detectors

Among multi-stage detectors, we train the commonly-used Faster R-CNN [10] and Cascade R-CNN [11] with RS Loss [18] and BRS Loss, and present average iteration time as well as AP of the trained models. In order to evaluate our contribution comprehensively, we train Faster R-CNN also with a stronger setting, in which, we use ResNet-101, train it for 36 epochs using multi-scale training similar to RS Loss [18]. The results are presented in Table 4.2, in which we can clearly see that BRS Loss consistently reduces the training time of all three detectors as well as preserves (or slightly improve) their performance. Especially for Cascade R-CNN, which is still

a very popular and strong object detector along with its variants such as HTC [88], the training time decreases by $5.3\times$. This is because the loss function is applied to the logits for three times based on the cascaded nature of this detector, and therefore, our contribution can be easily noticed. Moreover, it might seem that the efficiency gain decreases once the size of the backbone increases from R-50 to R-101 in Faster R-CNN. However, this is an expected result as the overall feature extraction time increases due to the larger number of parameters in the backbone, and hence, as we will discuss in the synthetic experiments, this is independent from our bucketing approach, operating on the loss function after the feature extraction.

4.5 Experiments on Instance Segmentation Methods

Given that our loss functions is efficient in object detectors, one simple extension is to see their generalization to instance segmentation methods. To show that, we train Mask R-CNN [61], a common baseline, with our BRS Loss on three different dataset from various domains: (i) COCO (Table 4.3), (ii) Cityscapes [85] as an autonomous driving dataset (Table 4.4) and LVIS [23] a long-tailed dataset with more than 1K classes (Table 4.4). The results confirm our earlier findings that using BRS Loss significantly reduces training time by approximately 2.3 times compared to RS Loss, while maintaining accuracy across both backbones on the COCO dataset. Additionally, when we extended our experiments to the Cityscapes and LVIS datasets, we observed a similar training time reduction of around 2.5 times, even with the imbalanced LVIS dataset. In Table 4.4, it is evident that BRS Loss narrows the training time gap compared to Cross-entropy Loss. Furthermore, BRS Loss outperforms Cross-entropy Loss on both datasets with respect to +1.5AP and +3.3AP.

Given Similar Training Budget, Bucketed Ranking Based Loss Functions Significantly Improve the Accuracy. There might be several benefits of reducing the training time of the detector as more GPU time is saved. Here, we show a use-case in which we ask the following question: what would happen if we allocate the similar amount of training time to both bucketed and non-bucketed losses? To answer that, we train Cascade R-CNN [11], a detector, and Mask R-CNN [61], an instance segmentation method, using our BRS Loss. Considering our speed-ups on these models

Table 4.3: Comparison with RS Loss on instance segmentation task on COCO val using Mask R-CNN.

Backbone	Loss	Efficiency	Accuracy		
		$T_{iter}(s) \downarrow$	AP $\uparrow$	AP_{50} $\uparrow$	AP_{75} $\uparrow$
ResNet 50	RS	0.56	36.3	57.2	38.8
	BRS	**0.24** 2.3x $\downarrow$	36.2	57.2	38.8
ResNet 101	RS	0.59	40.2	61.8	43.5
	BRS	**0.27** 2.2x $\downarrow$	40.3	62.0	43.8

Table 4.4: Comparison on different instance segmentation datasets using Mask R-CNN. AP_{75} is N/A as it is not used for Cityscapes.

Dataset	Loss	Efficiency	Accuracy		
		$T_{iter}(s) \downarrow$	AP $\uparrow$	AP_{50} $\uparrow$	AP_{75} $\uparrow$
Cityscapes [85]	Cross Entropy	0.18	41.8	67.1	N/A
	RS	0.43	43.5	68.1	N/A
	BRS	**0.19** 2.3x $\downarrow$	43.3	67.7	N/A
LVIS [23]	Cross Entropy	0.32	22.5	36.9	23.8
	RS	0.87	25.6	39.2	27.3
	BRS	**0.35** 2.5x $\downarrow$	**25.8**	39.6	27.4

($6.0\times$ and $2.3\times$), we simply increased their training epochs from 12 to 36 and 27 respectively. Note that, for Cascade R-CNN, we still spend significantly less amount of training time. Table 4.5 shows that models trained with the BRS Loss outperform (with $+1.2$ and 1.0 AP) their counterparts trained with the RS Loss in both cases.

4.6 Comparison of BRS Loss with Score-based and Other Ranking-based Losses

In previous sections, we showed the efficiency of our loss functions compared to AP and RS losses as their counterparts. This might give rise to the question whether Bucketed loss functions can close the gap between ranking-based losses and score-based losses such as Focal Loss [19]. To see that, we compare our bucketed losses

Table 4.5: BRS Loss further improves performance with similar training budget. We report mask AP for instance segmentation

Task	Detector	Loss	AP ↑	AP_{50} ↑	AP_{75} ↑
Object Detection	Cascade R-CNN [11]	RS	41.1	58.7	44.1
		BRS	**42.3** +1.2	**60.2**	**45.2**
Instance Segmentation	Mask R-CNN [61]	RS	36.3	57.2	38.8
		BRS	**37.3** +1.0	**58.4**	**40.2**

with the score-based losses, i.e., cross-entropy and Focal Loss [19], and other ranking-based losses including DR loss [82] and aLRP Loss [17].

To do so, in Tables 4.6 and 4.7, we report the average training time, AP for accuracy, and number of hyperparameters to capture the simplicity of tuning the loss functions. We note that our BRS and BAP Losses take similar training time with cross entropy and focal loss, closing the gap between ranking-based losses and score-based losses. Furthermore, our loss functions maintain the performance gain and tuning simplicity of ranking-based loss functions. Having addressed the main shortcoming of ranking-based losses, the performance margin between two popular score-based methods, Cross-Entropy [74] and Focal Loss [19]. Our findings show that BRS Loss reduces the gap between these losses, resulting in only a marginal increase in training time for one-stage object detectors, which is approximately 1.05 times longer than Focal Loss [19]. Furthermore, BRS Loss can improve the average precision (AP) by roughly 0.5 points, whilst requiring only one hyper-parameter, compared to Focal Loss [19] which requires 5. In the case of two-stage detectors, we have noticed a time speed-up of $1.2\times$ when using BRS Loss in comparison to Cross-Entropy Loss [74]. However, the use of BRS Loss results in a significant boost of $+1.9$ AP points in performance and requires fewer hyper-parameters than the setup with Cross-Entropy Loss [74] in Faster R-CNN [10]. Hence, our BRS Loss is either superior or on par with existing score-based losses. As for other ranking-based losses in Table 4.7, our BRS Loss is the most efficient and accurate ranking-based loss function for all three detectors. As an example, compared to ATSS trained with DR Loss, our BRS Loss yields 1.8AP better accuracy with 25% less training time and significantly less num-

Table 4.6: Comparison with the score-based losses. The chosen score based loss functions are the commonly used ones for each detector. Ours has very similar $T_{iter}(s)$ also by being more accurate and simple-to-tune. #H: Number of hyperparameters.

Detector	Loss	$T_{iter}(s) \downarrow$	AP $\uparrow$	#H
Faster R-CNN [10]	Cross Entropy+L1	**0.14**	37.6	9
	RS	0.50	39.4	**3**
	BRS (Ours)	0.17	**39.5**	**3**
ATSS [7]	Focal Loss+GIoU	**0.14**	39.3	5
	RS	0.36	**39.8**	**1**
	BRS (Ours)	0.15	**39.8**	**1**

Table 4.7: Comparison with other ranking-based losses. Our approach performs better on both accuracy and efficiency.

Detector	Loss	$T_{iter}(s) \downarrow$	AP $\uparrow$	#H
Faster R-CNN [10]	aLRP [17]	0.28	37.4	**3**
	BRS (Ours)	**0.17**	**39.5**	**3**
ATSS [7]	AP [1]	0.32	38.1	5
	DR [82]	0.20	38.1	5
	aLRP [17]	0.32	37.7	**1**
	BRS (Ours)	**0.15**	**39.8**	**1**
RetinaNet [19]	DR [82]	0.29	37.4	5
	BRS (Ours)	**0.23**	**38.3**	**1**

ber of hyperparameters.

Therefore, our bucketed loss functions are now promising alternatives to train object detectors. Figure 4.2 provides a comprehensive overview of our contributions across a range of detectors.

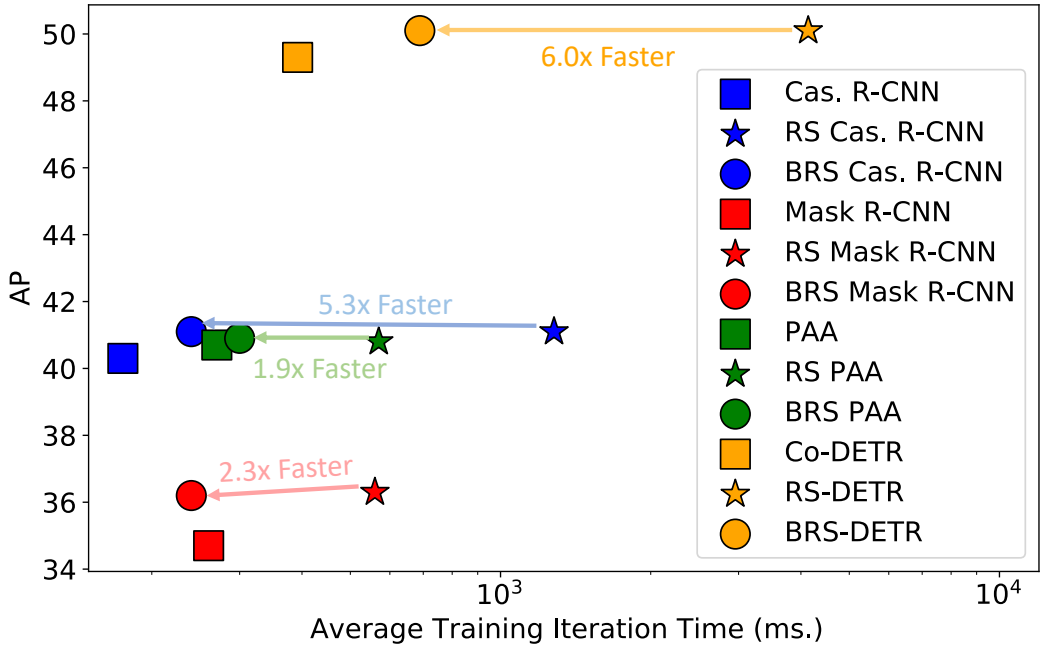


Figure 4.2: Accuracy and efficiency comparison across various detectors. Our BRS formulation facilitates faster (between $1.9\times$ and $6.0\times$) training of visual detectors with similar AP. [Figure from our ECCV 2024 paper [3]]

4.7 Experiments of BRS Loss with Co-DETR

In previous experiments, we present the efficiency of BRS Loss in CNN-based visual detectors. Most recent methods have shifted to using transformer-based approaches. This switch is crucial for understanding the effect of ranking-based losses for a couple of reasons: (i) Unlike CNN-based detectors, DETR variants propose a fixed number of queries, typically ranging from 300 to 900. This results in fewer logits or predictions compared to the dense detections produced by other models. (ii) The decoder layers are stacked, and separate loss calculations are performed at each decoder layer, with the number of layers usually set to six.

BRS-DETR Outperforms Existing Transformer-based Detectors Consistently. To analyze whether ranking-based loss functions can also enhance the DETR variants, we first employ the ResNet-50 backbone, as it has been used by many detectors, enabling us to compare our method with many different DETR-based manners in a consistent manner. In this fair comparison, our BRS-DETR outperforms all existing

DETR variants, achieving an AP of 50.1, as demonstrated in Table 4.8. For instance, our BRS-DETR surpasses DN-DETR [64] by 1.5 AP and DINO [70] by 0.9 AP, while requiring less training epochs or less queries. Additionally, it shows improvements over Co-DETR, which we will discuss next.

BRS-DETR Improves Co-DETR over Different Backbones Consistently. In order to show the effectiveness of our BRS Loss, we compare our results with Co-DETR in Table 4.9 using different backbones. We note that we improve the baseline Co-DETR consistently in all settings. For example, our improvement on ResNet-50 is 0.8AP, which is a notable improvement. However, as the backbone gets larger (e.g., Swin-L [89]), our gains decrease, which is expected and commonly observed in the literature, e.g., [8].

Training Co-DETR with RS Loss takes 6x less time compared to using RS Loss. Finally, we also compare the training efficiencies of BRS Loss and RS Loss on Co-DETR [6]. When we train Co-Deformable DETR with RSLoss on 300 query settings, we observe that training takes nearly 4 seconds per iteration. Hence, it's not practical to train Co-Deformable DETR. Furthermore, we observe that our BRS Loss with the same training setup decreases the training time of Co-DETR by $6.0\times$ (from 4.14s per iteration to 0.69s) compared to RS Loss. We note that this is the largest training time gain of our BRS Loss. This is because Co-DETR [6] consists of multiple transformer-based as well as auxiliary heads, requiring multiple loss estimations.

Table 4.8: Comparison of our BRS-DETR with DETR variants (trained with their original loss functions) w ResNet-50 on COCO val set.

Detector	Query	Epoch	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
DETR [4]	100	300	42.0	62.4	44.2	20.5	45.8	61.1
DN-DETR [64]	300	50	48.6	67.4	52.7	31.0	52.0	63.7
DINO [70]	900	12	49.4	66.9	53.8	32.3	52.5	63.9
H-DETR [63]	300	12	48.7	66.4	52.9	31.2	51.5	63.5
Co-DETR [6]	300	12	49.3	67.2	54.0	32.1	52.6	63.8
BRS-DETR	300	12	**50.1**	67.4	**54.6**	31.9	**53.9**	**65.0**

Table 4.9: Comparison of our BRS-DETR with Co-DETR (trained with its original loss function) on different backbones on COCO val set.

Backbone	Detector	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
ResNet50	Co-DETR [6]	49.3	67.2	54.0	**32.1**	52.6	63.8
	BRS-DETR	**50.1**	**67.4**	**54.6**	31.9	**53.9**	**65.0**
Swin-T	Co-DETR [6]	51.7	**69.6**	56.4	34.4	54.9	66.8
	BRS-DETR	**52.3**	69.5	**57.1**	**34.6**	**55.7**	**68.0**
Swin-L	Co-DETR [6]	56.9	**75.5**	**62.6**	**40.1**	61.2	73.3
	BRS-DETR	**57.2**	75.0	62.5	39.4	**61.7**	**74.1**

4.8 Discussion

In summary, we demonstrate the efficiency and generalizability of our bucketed ranking-based loss functions across two different tasks, six detectors, and three datasets. Notably, we observe a significant improvement by integrating ranking-based losses into Co-DETR [6]. Furthermore, since we propose an efficient loss function, we may achieve additional performance enhancements in other DETR variants as well.

CHAPTER 5

CONCLUSION

In this thesis, we tackle an important problem in object detection, which involves identifying and localizing objects in images. While score-based loss functions have traditionally been used for classification, ranking-based losses more closely align with evaluation criteria and are particularly effective in handling imbalanced data. However, their reliance on pairwise comparisons introduces computational inefficiencies, limiting their broader applicability. To address this gap, we introduced a novel method to improve the efficiency of ranking-based loss functions by binning negatives into buckets and implementing positive-negative pairwise comparisons between positives and these buckets. We showed that this approach reduces the computational complexity to a level where pairwise comparisons can be stored as a matrix in memory, bringing the running time close to that of score-based loss functions and paving the way for more efficient large-scale object detection.

We thoroughly evaluated the performance of our method through comprehensive experiments that include two different tasks, three different datasets, and six different detectors. This wide range of experiments highlights the general applicability of our approach. We first showcase the efficiency and generalizability of our method across various tasks and datasets. We demonstrate that Bucketed Ranking-based Losses achieve the same accuracy as their unbucketed counterparts while consistently reducing training time for ranking-based losses in both one-stage and multi-stage CNN-based object detectors. Furthermore, we demonstrate that bucketed ranking-based losses are also effective in instance segmentation tasks across different datasets. With these improvements, ranking-based losses emerge as a strong alternative to score-based losses, as they not only outperform them but also help close the training time

gap.

Also, for the first time, we integrated a ranking-based loss to Co-DETR [6], a DETR-based state-of-the-art detector, which was possible thanks to our method's lower complexity and reported improvements on different backbones. With our BRS Loss, we are able to obtain consistent performance gains over Co-DETR with different backbones, both on ResNet50, SwinT, and SwinL. Our experiments showed that bucketed ranking-based losses can be incorporated into various DETR-based detectors, providing a valuable opportunity for further performance improvements.

5.1 Limitations and Future Work

The primary purpose of this thesis is to address the inefficiencies associated with ranking-based losses. We close the performance gap and surpass score-based losses by utilizing our bucketing ranking losses. This work is the first to integrate ranking-based losses with detection transformers. The BRS Loss significantly enhances the performance of Co-DETR. However, several factors could be explored in future research.

First, it is important to note that there is a relationship between the cost function used in Hungarian assignment and the loss function in a general adapted structure for DETR [4]. Following the original DETR model, its variants have adapted the same structure, ensuring that the cost function comprises the same components as the loss function. Our integration of BRS Loss currently focuses solely on the loss function. Replacing the Focal Cost with a Ranking-based Cost is not a straightforward task, and introducing Ranking-based Cost functions could be a valuable contribution that further enhances the effectiveness of BRS Loss. Additionally, the impact of each component of the cost function can be analyzed in detail to reduce the number of hyperparameters.

One minor limitation to consider is the need for training and fine-tuning large detectors. While detailed fine-tuning has been conducted for the ResNet50 backbone using a 12-epoch and 300-query setup, training times for the SwinL backbone, particularly with approaches like Co-DINO-DETR, can be excessively long. This issue

may be addressed in future research. Additionally, there are numerous DETR variants available, and our loss generalizability allows for the potential integration of more detectors using ranking losses.

REFERENCES

[1] K. Chen, J. Li, W. Lin, J. See, J. Wang, L. Duan, Z. Chen, C. He, and J. Zou, "Towards accurate one-stage object detection with ap-loss," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[2] K. Chen, W. Lin, J. li, J. See, J. Wang, and J. Zou, "Ap-loss for accurate one-stage object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pp. 1–1, 2020.

[3] F. Yavuz, B. C. Cam, A. H. Dogan, K. Oksuz, E. Akbas, and S. Kalkan, "Bucketed ranking-based losses for efficient training of object detectors," in *European Conference on Computer Vision (ECCV)*, p. 93–109, Springer-Verlag, 2024.

[4] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision (ECCV)*, pp. 213–229, Springer, 2020.

[5] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable {detr}: Deformable transformers for end-to-end object detection," in *International Conference on Learning Representations (ICLR)*, 2021.

[6] Z. Zong, G. Song, and Y. Liu, "Detrs with collaborative hybrid assignments training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6748–6758, 2023.

[7] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li, "Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[8] K. Kim and H. S. Lee, "Probabilistic anchor assignment with iou prediction for object detection," in *The European Conference on Computer Vision (ECCV)*, 2020.

[9] P. Sun, R. Zhang, Y. Jiang, T. Kong, C. Xu, W. Zhan, M. Tomizuka, L. Li, Z. Yuan, C. Wang, and P. Luo, "SparseR-CNN: End-to-end object detection with learnable proposals," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, no. 6, pp. 1137–1149, 2017.

[11] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *The European Conference on Computer Vision (ECCV)*, 2018.

[13] K. Oksuz, B. C. Cam, F. Kahraman, Z. S. Baltaci, S. Kalkan, and E. Akbas, "Mask-aware iou for anchor assignment in real-time instance segmentation," in *The British Machine Vision Conference (BMVC)*, 2021.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," in *The European Conference on Computer Vision (ECCV)*, 2016.

[15] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin, "Libra R-CNN: Towards balanced learning for object detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[17] K. Oksuz, B. C. Cam, E. Akbas, and S. Kalkan, "A ranking-based, balanced loss function unifying classification and localisation in object detection," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[18] K. Oksuz, B. C. Cam, E. Akbas, and S. Kalkan, "Rank & sort loss for object detection and instance segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3009–3018, 2021.

[19] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 42, no. 2, pp. 318–327, 2020.

[20] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[21] Z. Zheng, P. Wang, W. Liu, J. Li, Y. Rongguang, and R. Dongwei, "Distance-iou loss: Faster and better learning for bounding box regression," in *AAAI Conference on Artificial Intelligence*, 2020.

[22] K. Oksuz, B. C. Cam, S. Kalkan, and E. Akbas, "Imbalance problems in object detection: A review," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 10, pp. 3388–3415, 2020.

[23] A. Gupta, P. Dollar, and R. Girshick, "Lvis: A dataset for large vocabulary instance segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[24] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *The European Conference on Computer Vision (ECCV)*, 2014.

[25] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I–I, 2001.

[26] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, 2005.

[27] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[29] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[30] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[31] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[32] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang, "Acquisition of localization confidence for accurate object detection," in *The European Conference on Computer Vision (ECCV)*, 2018.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[35] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *The International Conference on Learning Representations (ICLR)*, 2015.

[37] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural net-

works for mobile vision applications," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[38] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[39] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, vol. 1804.02767, 2018.

[40] Q. Zhao, T. Sheng, Y. Wang, Z. Tang, Y. Chen, L. Cai, and H. Ling, "M2det: A single-shot object detector based on multi-level feature pyramid network," in *AAAI Conference on Artificial Intelligence*, 2019.

[41] J. Choi, D. Chun, H. Kim, and H.-J. Lee, "Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[42] C. Feng, Y. Zhong, Y. Gao, M. R. Scott, and W. Huang, "Tood: Task-aligned one-stage object detection," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[43] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[44] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[45] Z. Shen, Z. Liu, J. Li, Y.-G. Jiang, Y. Chen, and X. Xue, "Dsod: Learning deeply supervised object detectors from scratch," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[46] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD: Deconvolutional single shot detector," *arXiv*, vol. 1701.06659, 2017.

[47] Z. Sun, S. Cao, Y. Yang, and K. M. Kitani, "Rethinking transformer-based set prediction for object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3611–3620, October 2021.

[48] H. Zhang, H. Chang, B. Ma, N. Wang, and X. Chen, "Dynamic r-cnn: Towards high quality object detection via dynamic training," in *The European Conference on Computer Vision (ECCV)*, 2020.

[49] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[50] C. Zhu, Y. He, and M. Savvides, "Feature selective anchor-free module for single-shot object detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[51] X. Zhou, J. Zhuo, and P. Krahenbuhl, "Bottom-up object detection by grouping extreme and center points," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[52] Z. Yang, S. Liu, H. Hu, L. Wang, and S. Lin, "Reppoints: Point set representation for object detection," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[53] Y. Chen, Z. Zhang, Y. Cao, L. Wang, S. Lin, and H. Hu, "Reppoints v2: Verification meets regression for object detection," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[54] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020.

[55] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022.

[56] G. Jocher, J. Qiu, and A. Chaurasia, "Ultralytics YOLO," Jan. 2023.

[57] X. Zhang, F. Wan, C. Liu, R. Ji, and Q. Ye, "Freeanchor: Learning to match anchors for visual object detection," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[58] H. Li, Z. Wu, C. Zhu, C. Xiong, R. Socher, and L. S. Davis, "Learning from noisy anchors for one-stage object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10588–10597, 2020.

[59] R. Girshick, "Fast R-CNN," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015.

[60] J. Li, X. Liang, S. Shen, T. Xu, J. Feng, and S. Yan, "Scale-aware fast r-cnn for pedestrian detection," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 985–996, 2018.

[61] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[62] Z. Sun, S. Cao, Y. Yang, and K. M. Kitani, "Rethinking transformer-based set prediction for object detection," in *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, pp. 3611–3620, 2021.

[63] D. Jia, Y. Yuan, H. He, X. Wu, H. Yu, W. Lin, L. Sun, C. Zhang, and H. Hu, "Detrs with hybrid matching," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 19702–19712, 2023.

[64] F. Li, H. Zhang, S. Liu, J. Guo, L. M. Ni, and L. Zhang, "Dn-detr: Accelerate detr training by introducing query denoising," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 13619–13627, 2022.

[65] X. Dai, Y. Chen, J. Yang, P. Zhang, L. Yuan, and L. Zhang, "Dynamic detr: End-to-end object detection with dynamic attention," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2968–2977, 2021.

[66] Z. Dai, B. Cai, Y. Lin, and J. Chen, "Unsupervised pre-training for detection transformers," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, p. 1–11, 2022.

[67] S. Liu, F. Li, H. Zhang, X. Yang, X. Qi, H. Su, J. Zhu, and L. Zhang, "DAB-DETR: Dynamic anchor boxes are better queries for DETR," in *International Conference on Learning Representations (ICLR)*, 2022.

[68] S. Liu, T. Ren, J. Chen, Z. Zeng, H. Zhang, F. Li, H. Li, J. Huang, H. Su, J. Zhu, *et al.*, "Detection transformer with stable matching," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6491–6500, 2023.

[69] Q. Chen, X. Chen, J. Wang, S. Zhang, K. Yao, H. Feng, J. Han, E. Ding, G. Zeng, and J. Wang, "Group detr: Fast detr training with group-wise one-to-many assignment," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6633–6642, 2023.

[70] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. Ni, and H.-Y. Shum, "DINO: DETR with improved denoising anchor boxes for end-to-end object detection," in *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[71] J. Ouyang-Zhang, J. H. Cho, X. Zhou, and P. Krähenbühl, "Nms strikes back," *arXiv preprint arXiv:2212.06137*, 2022.

[72] J. Lin, X. Mao, Y. Chen, L. Xu, Y. He, and H. Xue, "D^2etr: Decoder-only detr with computationally efficient cross-scale attention," 2022.

[73] G. Zhang, Z. Luo, Y. Yu, K. Cui, and S. Lu, "Accelerating detr convergence via semantic-aligned matching," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 949–958, 2022.

[74] S. Mannor, D. Peleg, and R. Rubinstein, "The cross entropy method for classification," in *The International Conference on Machine Learning (ICML)*, 2005.

[75] A. Ghosh, T. Schaaf, and M. Gormley, "Adafocal: Calibration-aware adaptive focal loss," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1583–1595, 2022.

[76] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21002–21012, 2020.

[77] X. Li, W. Wang, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection,"

in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11632–11641, 2021.

[78] Z. Zheng, P. Wang, D. Ren, W. Liu, R. Ye, Q. Hu, and W. Zuo, "Enhancing geometric factors in model learning and inference for object detection and instance segmentation," *IEEE Transactions on cybernetics*, vol. 52, no. 8, pp. 8574–8586, 2021.

[79] P. Mohapatra, C. Jawahar, and M. P. Kumar, "Efficient optimization for average precision svm," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[80] M. V. Pogančić, A. Paulus, V. Musil, G. Martius, and M. Rolinek, "Differentiation of blackbox combinatorial solvers," in *International Conference on Learning Representations (ICLR)*, 2020.

[81] M. Rolínek, V. Musil, A. Paulus, M. Vlastelica, C. Michaelis, and G. Martius, "Optimizing rank-based metrics with blackbox differentiation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[82] Q. Qian, L. Chen, H. Li, and R. Jin, "Dr loss: Improving object detection by distributional ranking," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[83] A. Brown, W. Xie, V. Kalogeiton, and A. Zisserman, "Smooth-ap: Smoothing the path towards large-scale image retrieval," in *The European Conference on Computer Vision (ECCV)*, pp. 677–694, Springer, 2020.

[84] K. Oksuz, B. C. Cam, E. Akbas, and S. Kalkan, "Localization recall precision (LRP): A new performance metric for object detection," in *The European Conference on Computer Vision (ECCV)*, 2018.

[85] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[86] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, pp. 65–386, 1958.

[87] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "MMDetection: Open mmlab detection toolbox and benchmark," *arXiv*, vol. 1906.07155, 2019.

[88] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "Hybrid task cascade for instance segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[89] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10002, 2021.