TRAINING OBJECT DETECTORS BY DIRECTLY OPTIMIZING LRP METRIC

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

BY

BARIŞ CAN ÇAM

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

SEPTEMBER 2020
Approval of the thesis:

TRAINING OBJECT DETECTORS BY DIRECTLY OPTIMIZING LRP METRIC

submitted by BARIŞ CAN ÇAM in partial fulfillment of the requirements for the degree of Master of Science in Computer Engineering Department, Middle East Technical University by,

Prof. Dr. Halil Kalpçılar
Dean, Graduate School of Natural and Applied Sciences

Prof. Dr. Halit Oğuztüzün
Head of Department, Computer Engineering

Assist. Prof. Dr. Emre Akbaş
Supervisor, Computer Engineering, METU

Assoc. Prof. Dr. Sinan Kalkan
Co-supervisor, Computer Engineering, METU

Examiner Committee Members:

Assist. Prof. Dr. Ramazan Gökberk Cinbiş
Computer Engineering, METU

Assist. Prof. Dr. Emre Akbaş
Computer Engineering, METU

Assoc. Prof. Dr. Hacer Yalım Keles
Computer Engineering, Ankara University

Date: 28.09.2020
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: Barış Can Çam

Signature :
This thesis focuses on training deep object detection networks by directly optimizing the localisation-recall-precision (LRP) performance metric that can evaluate classification and localisation performance of an object detector in a unified manner (Oksuz et al., 2018). To achieve this goal, unlike the commonly used linear weighting approach, we aim to implicitly optimize the LRP metric first by using a bounded localisation loss from previous works and proposing a loss function that can bound the range of classification task loss. In addition to this range balancing approach, we aim to train an object detector with an LRP regressor trained with LRP values collected during the training stage. We show that the proposed regression architecture can estimate LRP values with low error rates. However, training an object detector by attaching the regressor architecture as a differentiable LRP error estimator did not yield satisfactory results. Finally, by adapting the perceptron learning algorithm based approach proposed by Chen et al. (2020), we show that we can embed the LRP metric as a loss function to train a deep object detector. In this thesis, this perceptron learning-based approach is examined, and its generalization to all IoU based
localisation loss functions is proposed.

Keywords: ranking based loss functions, training object detectors, evaluation of object detectors, deep learning, object detection
ÖZ

LRP METRİĞİNİN DOĞRUDAN EN İYİLENMESİ İLE NESNE TESPİTÇİLERİNİN EĞİTİLMESİ

Çam, Barış Can
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Emre Akbaş
Ortak Tez Yöneticisi: Doç. Dr. Sinan Kalkan

Eylül 2020, 78 sayfa

irdelenmiş ve bu yaklaşımın IoU tabanlı konumlandırma kayıp fonksiyonlarına genelleştirilmesi önerilmiştir.

Anahtar Kelimeler: sıralama tabanlı kayıp fonksiyonları, nesne tespitçilerinin eğitilmesi, nesne tespitçilerinin ölçümü, derin öğrenme, nesne tespiti
This thesis is dedicated to whoever might be interested in this work.
ACKNOWLEDGMENTS

I would like to express my deepest appreciation and gratitude to my supervisor Assist. Prof. Emre Akbaş and, co-supervisor Assoc. Prof. Sinan Kalkan for all their help, guidance, and supervision. This work would not have been possible without their continuous support and motivation.

I would like to thank the rest of my thesis committee members Assist. Prof. Dr. Ramazan Gökberk Cinbiş and Assoc. Prof. Dr. Hacer Yalım Keleş, for their insightful comments and questions.

I would like to thank Kemal Öksüz for being a "second supervisor" for me by his valuable guidance and contributions during our works as a research group.

I gratefully acknowledge Turkish Scientific and Technical Research Council (TÜBİTAK) for partially supporting this work under the Grant 117E054 and providing computational resources under TRUBA services.

I would like to express my gratitude to Erkan Oğur for himself and the existence of his music.

Finally, I would like to express my eternal gratitude to my parents, my brothers Alper Bilge and Serhan Gürmeriç, and my dear Rabiya Bıyıklı for their everlasting love and support. This thesis would not have been possible without them.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>v</td>
</tr>
<tr>
<td>ÖZ</td>
<td>vii</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>x</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>xi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>xiv</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>xvi</td>
</tr>
<tr>
<td>LIST OF ABBREVIATIONS</td>
<td>xxiii</td>
</tr>
</tbody>
</table>

## CHAPTERS

1. INTRODUCTION                     | 1    |
| 1.1 Motivation and Problem Definition| 3    |
| 1.2 Contributions and Novelties    | 8    |
| 1.3 The Outline of the Thesis      | 10   |

2. RELATED WORK AND BACKGROUND      | 11   |
| 2.1 The Notation for Deep Object Detection | 11   |
| 2.2 Related Work                   | 12   |
| 2.2.1 Most Commonly Used Loss Functions in Object Detection | 12 |
| 2.2.1.1 Classification Loss Functions | 13   |
| 2.2.1.2 Localisation Loss Functions | 15   |
4.3.4 LRP Estimation Network Architecture .......................... 52

4.4 Experiments ......................................................... 54

4.4.1 Association Experiments ..................................... 54

4.4.2 LRP Estimation Experiments ............................... 54

4.4.3 Object Detection Experiments ............................. 55

5 AVERAGE LRP LOSS .................................................. 59

5.1 Definition of aLRP Loss .......................................... 60

5.2 Deriving Gradients of aLRP Loss ............................ 61

5.3 Self Balance Mechanism for Localisation Component ....... 63

5.4 Extending aLRP Loss for Different IoU Variants ........... 63

5.5 Experiments .......................................................... 65

6 CONCLUSIONS ......................................................... 71
LIST OF TABLES

TABLES

Table 2.1 Representations for Input/Output Relations of Object Detectors used in this thesis. .......................... 12

Table 3.1 Exponential loss validation experiments results in CIFAR-10 Dataset. Each cell represents the accuracy of the trained model on CIFAR-10 test set. Linear Loss is used as a baseline method for both balanced and imbalanced cases. Balanced classification task experiments show that exponential loss is a good numerical range balanced candidate for replacing cross-entropy Loss. In imbalanced case, while exponential loss can not reflect the same behaviour of cross-entropy loss especially for hard examples (\( p_i < 0.5 \)), exponential loss with \( \gamma = 0.2 \) results show that using its scaled version can overcome this issue. ................. 40

Table 3.2 Exponential loss grid-search results for \( \gamma \) and number of samples sampled from foreground (fg) / background (bg) classes at second stage. The results proposed in each cell are AP measures. —— represents no results for the corresponding cell because regardless of the sampling ratio for the corresponding \( \gamma \) value detector cannot achieve a significant performance. The best performing combination is written in bold. ................. 41

Table 4.1 Confusion Matrix of the Associator ........................... 55

Table 5.1 Ablation analysis on COCO minival. For optimal LRP (oLRP), lower is better. .......................... 67
<table>
<thead>
<tr>
<th>S</th>
<th>×1.66</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>implies that the image is rescaled such that its longer side cannot exceed 1.66 × S where S is the size of the shorter side.</td>
</tr>
</tbody>
</table>

LIST OF FIGURES

FIGURES

Figure 1.1 (a) One-Stage Pipeline, (b) Two-Stage Pipeline

Figure 1.2 (a) Randomly sampled 32 positive RoIs using the pRoI Generator [1]. (b) Average classification and regression losses of these RoIs at the initialization of the object detector for MS COCO dataset [2] with 80 classes. We use cross-entropy for the classification task assuming that initially each class has the same confidence score, and smooth $L_1$ loss for regression task. Note that right after initialization, the classification loss has more effect on the total loss. (Figure source: [3])

Figure 1.3 A toy example suggesting that two individual tasks (classification and localisation) directly affects each other in object detection. Blue, green and red boxes represent a ground-truth, positive and negative bounding boxes respectively. The largest bounding box in (a) is shifted to the right in the spatial domain in (b) and (c). In (b) the IoU value of the aforementioned box decreases, however it is still a positive example due to the assignment heuristics. In (c) same box is shifted with an amount that will cause the box becomes a negative example. (Figure source: [3])
Figure 1.4 When two tasks are considered as correlated tasks, high-precision detections have high-IoUs, performance of the detector is significantly improved. (a) Classification and three possible localisation outputs for 10 anchors and the rankings of the positive anchors with respect to (wrt) the scores (for $C$) and IoUs (for $R_1$, $R_2$ and $R_3$). Since the regressor is only trained by positive anchors, “-” is assigned for negative anchors. (b) Performance of $R_1$, $R_2$ and $R_3$ when combined with $C$. When correlation between the rankings of classifier and regressor outputs decreases, performance degrades up to 17% AP. (Figure source: [4]).

Figure 1.5 Ranking operation is not differentiable wrt classification scores. (a) Ranking of each output example with respect to their confidence scores is shown. Calculating rank values of each output example come up with a piece-wise constant function [5]. (b) The most commonly used classification loss function [6,7], namely cross-entropy, is differentiable wrt to detection scores. Note that, in both cases detections with confidence score value less than 0.1 are ignored as in the toy example in Figure 1.4.

Figure 2.1 Comparison of focal loss with different $\gamma$ modulation parameters and cross-entropy loss. Focal loss reduces the contribution of well-classified (easy) examples ($p_i > 0.5$) to the resulting loss value, while boosts loss values for hard examples. (Figure source: [8]).

Figure 2.2 Differences between $L_1$ and $L_2$ loss functions for measuring spatial distance. Left: Spatial visualization of bounding boxes with a ground-truth bounding box in blue and three predicted bounding boxes. Right: IoU values, $L_1$ and $L_2$ loss values are calculated between each predicted bounding box and the ground-truth bounding box. While measuring $L_2$ distance for inlier bounding boxes (e.g. green bounding box) provides more stable loss values, $L_1$ distance prevent outliers (e.g. yellow bounding box) to dominate the overall loss in contrast to $L_2$ distance. (Figure source: [3]).
(a) The gradients of the smooth and balanced $L_1$ loss functions,

(b) The smooth and balanced $L_1$ loss functions (Figure source: [9]).

Figure 2.4 A toy example that presents the differences between IoU and its variants. Note that, all boxes have the same width and height values of 1.00. (a) A single ground-truth bounding box (red) and predicted bounding boxes (green) that can help differentiating behaviour of IoU and its variants. (b) Bounding boxes with its center points and for each predicted bounding box; IoU, GIoU and DIoU values with the GT bounding box are calculated. Note that, each box is a unit box with different center point locations. $B_0$ is the perfect matching box with the ground-truth box which has the same 1.00 IoU, GIoU and DIoU values. IoU can only provide discriminative information for $B_1$, for all other boxes IoU can not provide discriminative information since $I$ term in Equation (2.12) becomes 0.00. GIoU and DIoU can still measure the similarity between the ground-truth box and predicted bounding boxes $B_{vic}$, $B_{dist}$ and $B_{inf}$.

Figure 2.5 A toy example that presents the geometric interpretation of IoU variants (GIoU and DIoU). (a) A ground-truth bounding box and single predicted bounding box with their center points. (b) Minimum enclosing box $C$ that involve both ground-truth and the predicted bounding box. (c) An example that has the same IoU and GIoU values but, DIoU can still differentiate the quality measure between the ground-truth and predicted bounding boxes.

Figure 2.6 Detection outputs of three different detectors (D1,D2,D3) on the same image from [10] with a single class (car) are visualized in the first row (a,b,c) with their corresponding PR curves (d,e,f) in the second row. Red, blue and green bounding boxes represents the ground-truth boxes, TP and FP detections respectively. Red-cross represents the optimal point on PR curve with the minimum LRP value for a single confidence score $s$. (Figure source: [11]).
Figure 3.1  (a) Vanilla cross entropy loss and its shifted with modulating parameter $\gamma$. (b) Zoomed version of (a) in the region between $0.6 \leq p_i \leq 1$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

Figure 3.2  Comparison of original cross-entropy loss with exponential loss with different scaler parameters. While cross-entropy loss has an unbounded numerical range, exponential loss in bounded in the range of $[0, 1]$. Different scaler values ($\gamma$) implies modulation of exponential loss to weight hard examples (according to their $p_i$). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38

Figure 3.3  Loss weight distributions for (a) balanced and (b) imbalanced cases. Uniform distribution in (a) implies balanced among classes. In (b) $\chi^2$ distribution with $k = 1$ lets the class-imbalance can be simulated by weighting loss values for different classes. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39

Figure 3.4  Loss distribution for foreground and background classes in object detection experiments. Loss values are collected during the first epoch snapshot of the whole training process. (a) Cross-entropy loss fg/bg loss distribution. Cross-entropy loss can achieve balanced loss values for all bins when it is compared to the exponential loss. (b) Exponential loss fg/bg loss distribution. Exponential loss produces biased training for foreground examples. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42

Figure 4.1  Proposed neural network based LRP estimator. "Object Detector" module represents a generic object detection algorithm runs in inference mode and outputs prediction bounding box set $Y_s$. "Ground-truth annotations" module represents annotated bounding boxes with their class labels $X$. The "Similarity estimator" module is a neural network based architecture that outputs similarity scores between ground-truth and prediction sets. By providing these similarity scores into the "regressor" module, LRP values are estimated. Modules in green are trained in a supervised manner by using ground-truth LRP values provided by the LRP calculation module. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
Figure 4.2 Estimated and real cross-entropy loss functions. Ground-truth (G.T.) cross-entropy loss shows the real behaviour of cross-entropy loss (red). By collecting 80000 samples from cross-entropy loss with its input an MLP can successfully reconstruct cross-entropy loss function (green).

Figure 4.3 The DDML Architecture for Face Verification in work [12]. By applying image pairs \( (x_i, x_j) \) as an input to two similar networks that share parameters \( (W \) and \( b) \), distance (or similarity) between two image pairs are measured in their corresponding feature-space representations \( (h_1 \) and \( h_2) \). By comparing the thresholded distance value with corresponding labels they train the whole architecture to discriminate face image pairs. Blue arrows shows the forward pass, green arrows shows the gradient flow path for backpropagation.

Figure 4.4 Similarity estimator module inspired from [12]: The predicted bounding box and ground-truth bounding boxes are denoted by corner coordinates respectively as follows, \( [\hat{x}_1, \hat{y}_1, \hat{x}_2, \hat{y}_2] \) and \( [x_1, y_1, x_2, y_2] \) respectively. Corresponding \( p \) values are background and class confidence scores for this box. For the ground-truth \( p^c = 1 \) and \( p^0 = 0 \).

Figure 4.5 Learning based association architecture. Detection and ground-truth pairs are fed into the DDML module explained in Section 4.3.3. By means of the similarity estimator, similarity measures between ground-truth and detection pairs \( (\Delta_i) \) are fed into the associator. Overall architecture is trained in supervised manner as a classification problem (Pair associate or not).

Figure 4.6 Learning based LRP estimation architecture. There are two separate similarity estimation modules for extracting all components of LRP metric \( (N_{TP}, N_{FP}, N_{FN}) \). The remaining architecture is the same as the association network, except for this architecture is trained by considering the difference between estimated LRP values and real LRP values.
Figure 4.7 Training/Validation loss curves (left) and the performance over iterations/epochs for the associator (right). The loss curve shows that the associator converges after about 200th iterations; since it overfits, we used the model that is trained up to this point. Validation accuracy shows that the associator model can estimate TP, FP, FN, and TN predictions with acceptable accuracy.

Figure 4.8 (a) Effect of the pooling layer: The lines get thicker through 6 linear neurons model, 6 ReLU neurons model and 2 layer model. (b) Training and Validation Loss: The lines get thicker through 6 linear neurons model, 6 ReLU neurons model and 2 layer model. Loss values are averaged over all iterations in each epoch.

Figure 4.9 Visualization of both total and component-wise loss values during the training of Faster R-CNN. As we explained in the experiments section, we trained Faster R-CNN with original loss components (in black), RPN only losses (in red), and LRP estimation module with fixed thresholds (in blue), optimal thresholds (in green), and multiple thresholds (in magenta). In addition to this, we calculate LRP values at each step of the training to analyze each method’s training performance. As expected, the original training of Faster R-CNN converges (consider all loss components and LRP Loss). Considering the LRP Loss plot, we can see that RPN only model can not converge. We expected that our LRP estimation module performs in between the original training and RPN only model; however, it diverges after a few iterations.
aLRP Loss is aware of the correlation between object detection tasks. (a) Classification and three possible localisation outputs for 10 anchors and the rankings of the positive anchors with respect to (wrt) the scores (for $C$) and IoUs (for $R_1$, $R_2$ and $R_3$). Since the regressor is only trained by positive anchors, “–” is assigned for negative anchors.

(b) Performance of $R_1$, $R_2$ and $R_3$ when combined with $C$. When correlation between the rankings of classifier and regressor outputs decreases, performance degrades by 17% AP points. (c) Loss assignment under different loss functions for all classifier&regressor combinations. While, any linear combination of $L_c$ and $L_r$ cannot distinguish the correlation between two tasks. aLRP Loss directly penalizes the output anchors of object detector accordingly. (Figure source: [4])

aLRP Loss assigns gradients to each branch based on the outputs of both branches. Examples on the PR curve are in sorted order wrt scores ($s$). $L$ refers to $L^{aLRP}$. (a) A $p_i$’s gradient wrt its score considers (i) localisation errors of examples with larger $s$ (e.g. high $\mathcal{E}_{loc}(p_1)$ increases the gradient of $s_{p_2}$ to suppress $p_1$), (ii) number of negatives with larger $s$. (b) Gradients wrt $s$ of the negatives: The gradient of a $p_i$ is uniformly distributed over the negatives with larger $s$. Summed contributions from all positives determines the gradient of a negative. (c) Gradients of the box parameters: While $p_1$ (with highest $s$) is included in total localisation error on each positive, i.e.

$$L^{aLRP}_{loc}(i) = \frac{1}{\text{rank}(i)}(\mathcal{E}_{loc}(i) + \sum_{k \in P, k \neq i} \mathcal{E}_{loc}(k)H(x_{ik})),$$ with the largest $\text{rank}(p_i)$. (Figure source: [4])

aLRP Loss and its components. The localisation component is self-balanced. (Figure source: [4])
## LIST OF ABBREVIATIONS

### ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aLRP</td>
<td>Average LRP</td>
</tr>
<tr>
<td>AP</td>
<td>Average Precision</td>
</tr>
<tr>
<td>BG</td>
<td>Background</td>
</tr>
<tr>
<td>CE</td>
<td>Cross-entropy</td>
</tr>
<tr>
<td>ClIoU</td>
<td>Complete IoU</td>
</tr>
<tr>
<td>DDML</td>
<td>Deep Discriminative Metric Learning</td>
</tr>
<tr>
<td>DIoU</td>
<td>Distance IoU</td>
</tr>
<tr>
<td>exp</td>
<td>Exponential</td>
</tr>
<tr>
<td>FC</td>
<td>Fully Connected</td>
</tr>
<tr>
<td>FG</td>
<td>Foreground</td>
</tr>
<tr>
<td>FN</td>
<td>False Negative</td>
</tr>
<tr>
<td>FP</td>
<td>False Positive</td>
</tr>
<tr>
<td>GIoU</td>
<td>Generalized IoU</td>
</tr>
<tr>
<td>IoU</td>
<td>Intersection over Union</td>
</tr>
<tr>
<td>LRP</td>
<td>Localization Recall Precision</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long Short-Term Memory</td>
</tr>
<tr>
<td>mAP</td>
<td>Mean Average Precision</td>
</tr>
<tr>
<td>moLRP</td>
<td>Mean Optimal LRP</td>
</tr>
<tr>
<td>ND</td>
<td>N-Dimensional</td>
</tr>
<tr>
<td>NMS</td>
<td>Non-maximum Supression</td>
</tr>
<tr>
<td>oLRP</td>
<td>Optimal LRP</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>PR</td>
<td>Precision Recall</td>
</tr>
<tr>
<td>pRoI</td>
<td>Positive Region of Interest</td>
</tr>
<tr>
<td>ReLU</td>
<td>Rectified Linear Unit</td>
</tr>
<tr>
<td>RoI</td>
<td>Region of Interest</td>
</tr>
<tr>
<td>RPN</td>
<td>Region Proposal Network</td>
</tr>
<tr>
<td>SGD</td>
<td>Stochastic Gradient Descent</td>
</tr>
<tr>
<td>TN</td>
<td>True Negative</td>
</tr>
<tr>
<td>TP</td>
<td>True Positive</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

Object detection aims to determine a class label and a tight bounding box for each object in a scene. The methods based on deep learning architectures [6, 7, 8, 13, 14, 15] significantly outperform the methods from the pre-deep learning era [16]. As a result, major performance improvement in object detection ignited many successful applications in various domains such as pedestrian detection [17], object detection in aerial images [18], video object detection [19] and 3D object detection [20]. Considering its influence in different domains, object detection is a fundamental method in computer vision.

Deep object detection methods can be broadly analyzed in two main categories: one-stage (Figure 1.1(a)) and two-stage (Figure 1.1(b)). One-stage methods including SSD variants [7, 21, 22], YOLO variants [15, 23], and RetinaNet [8] aim to directly predict object classes and bounding box locations from anchors that are densely sampled from the feature space. Thus, one-stage detectors are faster, however they suffer from class imbalance due to dense anchor sampling strategy producing large amounts of negative anchors. The second set of object detectors are two-stage [6, 13, 14, 24]. The main idea of two-stage methods is to split the detection process into two cascaded stages: The first stage decreases the number of negative anchors so that the second stage is not affected by the positive-negative imbalance. For this purpose, two-stage methods use a region proposal network (RPN) as the first stage, which generates regions likely to have objects, called object proposals or Region of Interests (RoIs). After this stage, these proposals are passed through the detection network (the second stage) to predict object classes and their locations.

Most deep object detection architectures include four primary operations: feature ex-
traction, anchor generation/assignment/sampling, detection, and non-maximum sup-
pression (NMS). As a first step, a backbone network extracts the most discriminative
features from an input image. After this, by considering some pre-defined properties
(scale, ratio, stride) anchors are generated in feature space. In the anchor assignment
step, anchor boxes are separated as positive/negatives by measuring their localisation
quality with ground-truth boxes. After anchor & ground-truth pairs are obtained in
the assignment step, sampling is applied to anchor boxes by following some heuris-
tics (e.g., keeping a constant ratio between positive and negative anchors). In two-
stage detectors, candidate object regions (object proposals) are extracted by applying
a sliding-window search [14] or an extra network, namely RPN, that classify posi-

Figure 1.1: (a) One-Stage Pipeline, (b) Two-Stage Pipeline
tive/negative proposals to feed into the second stage [6]. Before applying NMS, in the detection step, positive anchors are fed to the localisation branch in order to optimize box tightness, while all anchor boxes are fed into the classification branch to predict object classes. Then, in the final step, NMS is applied to prevent duplicate detections.

By definition, object detection aims to localize (drawing tight bounding boxes around objects) and classify each instance of a set of object classes in a given image. To achieve this goal, object detectors are trained by minimizing two separate loss functions: one for classification task (e.g. cross-entropy) and one for localisation (e.g. smooth $L_1$ [6]). The most common approach for jointly training object detectors with two separate tasks is linear weighting [6, 13, 14] as follows:

$$L = L_c + w_r L_r,$$

where $L_c$ and $L_r$ are the loss functions for classification and localisation tasks, respectively; and $w_r$ is a constant scaling factor. To find the best performing value of weighting factor, $w_r$ is searched on a validation set. Despite being widely used, the linear weighting approach in Equation (1.1) has several deficiencies – see Section 1.1.

For performance evaluation of object detectors, the most commonly used evaluation criterion is average-precision (AP) [2, 25]. Despite its popularity, AP can not evaluate the performance of object detector tasks (classification, localisation) jointly. To address AP’s limitations, we proposed a new evaluation method (also a metric for object detection) that can jointly evaluate classification and localisation performances of object detectors quantitatively [11].

1.1 Motivation and Problem Definition

Loss function design is crucial for gradient-based learning methods because optimization algorithms (i.e., gradient descent and its variants [26]) minimize the loss function by calculating the gradient of loss function with respect to model parameters. As mentioned before, object detection is a multi-task learning problem which requires simultaneously minimizing loss functions for both classification (determining object categories) and regression (localising objects in the image).
In this thesis, our main aim is to propose solutions as an alternative to linear weighting strategy for better training of object detectors. One alternative solution for linear weighting strategy is directly using performance evaluation methods of each individual task (e.g. IoU for localisation and AP for classification). Since classification task requires ranking operation, it is non-differentiable and requires extra effort for integrating the end-to-end training pipeline of deep object detectors. By following these, we analyzed the effects of combining baseline loss functions (cross-entropy as classification loss, smooth $L_1$ as localisation loss) by linear weighting as in [6, 13, 14] and the challenges of integration of ranking operation into the training process. As a result, we observe the following deficiencies:

- **Uncontrolled contribution of each loss component.** Most of the deep object detection methods use cross-entropy loss as classification cost and smooth $L_1$ loss as localisation cost [6, 7, 8]. These two loss functions have very different numerical ranges. Cross-entropy has the numerical range of $(0, \infty)$, while smooth $L_1$ has range $(-\infty, \infty)$. Due to this range inconsistency between two individual loss functions, by considering input box properties (Figure 1.2 (a)), contributions of individual elements of the loss function to the resulting loss value may differ as shown in Figure 1.2(b). The most common solution for this problem is weighting tasks as shown in Equation (1.1) [6, 13, 14]. However, task weighting approach introduces an extra hyper-parameter ($w_r$ in Equation (1.1)). While many studies does not consider tuning the weighting parameter [7, 8], it is shown that tuning this weighting parameter $w_r$ may result in small changes in the object detector performance [27]. Tuning this hyper-parameter requires numerous validation set search operations in a large search space (two tasks to be balanced in one-stage detectors and two extra tasks to train RPN in two-stage). Because of this, determining the best performing $w_r$ is a costly process and does not guarantee an optimal solution. To sum up, to control the contribution of each element of the resulting loss function, there should be a systematical approach.

- **Classification and localisation tasks are associated.** Gradient signals which update the neural network’s parameters are calculated by taking the derivative of the resulting loss value calculated by the linear weighting of two individual
Figure 1.2: (a) Randomly sampled 32 positive RoIs using the pRoI Generator [1]. (b) Average classification and regression losses of these RoIs at the initialization of the object detector for MS COCO dataset [2] with 80 classes. We use cross-entropy for the classification task assuming that initially each class has the same confidence score, and smooth $L_1$ loss for regression task. Note that right after initialization, the classification loss has more effect on the total loss. (Figure source: [3])

As shown in Figure 1.3, a perturbation in the spatial domain (shifting in this example) makes the contribution of that example to the regression branch decrease, while its contribution to the classification branch would increase. In conclusion, classification and regression branches should be considered in conjunction with each other. This relationship between the two loss components is totally ignored in the weighted sum approach.

- **Positive correlation between two tasks improves the performance.** In order to further analyze the previously mentioned association property of two tasks; we create a toy example with detection outputs in Figure 1.4. There is one classifier output that is shared between three different localisation branch outputs. The first detector, which combines C&R1, has a positive correlation between two tasks (classification and localisation) has the highest AP value, as shown in Figure 1.4 (b). In the second detection scenario, C&R2, there is no correlation between the outputs of two tasks (uncorrelated), which degrades the detector
Figure 1.3: A toy example suggesting that two individual tasks (classification and localisation) directly affects each other in object detection. Blue, green and red boxes represent a ground-truth, positive and negative bounding boxes respectively. The largest bounding box in (a) is shifted to the right in the spatial domain in (b) and (c). In (b) the IoU value of the aforementioned box decreases, however it is still a positive example due to the assignment heuristics. In (c) same box is shifted with an amount that will cause the box becomes a negative example. (Figure source: [3])

performance with 8% AP. The last scenario, C&R3, draws a lower bound for the detector performance among all cases by considering the relationship between the detector branches as they are negatively correlated. If outputs of two individual tasks are negatively correlated, the performance degradation becomes 17% AP. Combining two loss functions with linear weighting strategy in Equation (1.1) can not distinguish previously mentioned performance differences between three cases. Assume we use cross-entropy loss as the classification loss ($L_c$ in Equation (1.1)) and, smooth $L_1$ loss as localisation loss ($L_r$ in Equation (1.1)), for all three cases $L_c$ has the value of 0.87 and $L_r$ has the value of 0.29. To enforce an object detector to generate high precision outputs with high IoUs, there should be a loss function that considers this relationship. While AP differentiates this correlation relationship in performance evaluation, at a single IoU threshold (e.g., AP@50) performances of all detectors are the same (Figure 1.4(b)). As mentioned before, LRP [11] can jointly measure the localisation and classification performance of an object detector without the need for multiple evaluations under different IoU thresholds as in AP. Due to the LRP metric’s distinctiveness, using LRP as a loss function enforces the de-
### Ranking

(b) Performance in AP = \( \frac{\text{AP}_{50} + \text{AP}_{65} + \text{AP}_{80} + \text{AP}_{95}}{4} \)

### Detector Output

<table>
<thead>
<tr>
<th>AP _50</th>
<th>AP _65</th>
<th>AP _80</th>
<th>AP _95</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C &amp; R_1)</td>
<td>0.51</td>
<td>0.43</td>
<td>0.33</td>
<td>0.20</td>
</tr>
<tr>
<td>(C &amp; R_2)</td>
<td>0.51</td>
<td>0.39</td>
<td>0.24</td>
<td>0.02</td>
</tr>
<tr>
<td>(C &amp; R_3)</td>
<td>0.51</td>
<td>0.19</td>
<td>0.08</td>
<td>0.02</td>
</tr>
</tbody>
</table>

### Figure 1.4: When two tasks are considered as correlated tasks, high-precision detections have high-IoUs, performance of the detector is significantly improved.

(a) Classification and three possible localisation outputs for 10 anchors and the rankings of the positive anchors with respect to (wrt) the scores (for \( C \)) and IoUs (for \( R\_1, R\_2 \) and \( R\_3 \)). Since the regressor is only trained by positive anchors, “–” is assigned for negative anchors. (b) Performance of \( R\_1, R\_2 \) and \( R\_3 \) when combined with \( C \). When correlation between the rankings of classifier and regressor outputs decreases, performance degrades up to 17% AP. (Figure source: [4])

- **Ranking operation is not differentiable.** By definition, LRP evaluates an object detector jointly by penalizing each positive example in the output set with its normalized localisation quality (i.e., IoU) and evaluates its classification performance by calculating the resulting precision-recall curve [11]. To calculate the precision-recall curve, the resulting set containing detection ground-truth pairs of an object detector is first thresholded with a confidence score threshold to eliminate detection boxes with low confidence scores. After this thresholding operation, the remaining examples are sorted with respect to their confidence scores. To obtain precision and recall values, number of false positives \( N_{FP} \) and number of false negatives \( N_{FN} \) should be calculated. While, IoU is a differentiable function with respect to (wrt) its input parameters (box-parameters).
Figure 1.5: **Ranking operation is not differentiable wrt classification scores.**

(a) Ranking of each output example with respect to their confidence scores is shown. Calculating rank values of each output example come up with a piece-wise constant function [5]. (b) The most commonly used classification loss function [6, 7], namely cross-entropy, is differentiable wrt to detection scores. Note that, in both cases detections with confidence score value less than 0.1 are ignored as in the toy example in Figure 1.4.

[28], calculating precision and recall values requires ranking operation which is not differentiable due to its piece-wise constant behaviour (Figure 1.5). Due to the non-differentiability of ranking operation, it is not trivial to directly use LRP or any other function that requires ranking as a loss function for training a deep object detector.

### 1.2 Contributions and Novelties

Motivated by the problems and deficiencies in Section 1.1 in this thesis, the main goal is to provide a deep learning architecture that can jointly optimize localisation and classification tasks in object detection in contrast to baseline approaches [6, 7, 8]. In order to reach this goal, our contributions are as follows:

- We provide an analysis of the uncontrolled contribution of loss components problem mentioned in Section 1.1 by balancing the numerical ranges of clas-
sification and localisation task losses. We also provide a comparative analysis between cross-entropy loss and bounded range classification loss (exponential loss) by providing results in both object detection and object classification problems.

- As a first attempt to train an object detector end-to-end, by extending our work LRP [11] as a loss function, we provide a neural network architecture that aims to regress LRP values of an image instance from the data collected during the training of an object detector. We show that the LRP measure can be successfully regressed using the box-level information collected in the training stage of an object detector. After the regression stage, due to the imbalanced distribution of box properties (spatial and categorical), this method can train an object detector neither from scratch nor by initializing from a pre-trained snapshot of a model.

- Finally, similar to how the work proposed in [29] extends precision for average precision loss, our work in [4] proposed averaging over the LRP values on the positive examples. aLRP Loss alleviates the non-differentiable nature of LRP by combining the error-driven update with the backpropagation. This is a joint work with Kemal Oksuz, and the scope of this thesis includes the generalisation of aLRP Loss for more robust forms of IoU based losses. In particular, we extend the IoU-based aLRP Loss for Generalised IoU (GIoU) [30], Distance IoU (DIoU) and Complete IoU (CIoU) [31].

This work has been conducted in close collaboration with Kemal Oksuz, a Ph.D. candidate at Dept. of Computer Engineering, as a group work for addressing limitations of object detectors. The work presented in this thesis is the outcome of the collaboration, where the author is the main contributor. Therefore, the current thesis complements the to-be-written thesis of Kemal Oksuz. Within the scope of this study, we published the following papers:


1.3 The Outline of the Thesis

In addition to the introduction, this thesis consists of five chapters. By following the taxonomy that we presented in [3], Chapter 2 categorizes the previous works related to objective imbalance problems and performance evaluation of object detectors and presents a literature review. In Chapter 3, we suggest a method for balancing the numerical ranges of two tasks in object detection. For regressing the LRP metric, a neural network-based regression model is proposed in Chapter 4. For training an object detector by directly optimizing the LRP metric, our work [4] aims to model LRP by a ranking based formulation. In Chapter 5, we propose the generalised form of aLRP Loss by integrating a more robust form of IoU variants. Finally, we conclude our results in Chapter 6. We propose three different methods for integrating LRP into the training process of deep object detectors; each proposed method is independent of each other. For the sake of completeness we propose our experimental results at the end of the Chapters 3, 4 and 5.
CHAPTER 2

RELATED WORK AND BACKGROUND

This chapter presents the related work and the relevant background for the loss functions for training object detectors. To present previously proposed solutions for loss functions of object detectors that are highly related to the scope of this thesis, first, the most commonly used loss functions for training object detectors are explained. One of the issues argued in this thesis is the association between tasks of object detectors, as mentioned in Chapter 1.1. Previous methods that are aimed to propose solutions to this issue are summarized. Embedding performance measure into the end-to-end training of deep object detectors requires solutions for the non-differentiable nature of the ranking operation, as mentioned in Section 1.1. Due to this, in the last part of the related work section, we talk about the ranking based loss functions used in object detection.

The methods that we propose in Chapters 4 and 5 aim to train object detectors by directly optimizing the LRP metric. To provide prior knowledge for a better understanding of these methods, we explain LRP in detail. After this, to provide background information for the average LRP Loss method proposed in Chapter 5, we first summarize the Average Precision Loss (AP-Loss) [29], which forms a basis for average LRP Loss. After this background, our work [4] is explained as background information for this thesis.

2.1 The Notation for Deep Object Detection

In this section, we are going to introduce the notations used for input/output relations of object detectors in this thesis. For (deep) object detection, we are provided an
Table 2.1: Representations for Input/Output Relations of Object Detectors used in this thesis.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_i )</td>
<td>Ground-truth class label of ( i )th example.</td>
</tr>
<tr>
<td>( \hat{p}_i )</td>
<td>Ground-truth class label distribution of ( i )th example.</td>
</tr>
<tr>
<td>( p_i )</td>
<td>Predicted class probability of ( i )th example.</td>
</tr>
<tr>
<td>( s_i )</td>
<td>Raw classification score of ( i )th example.</td>
</tr>
<tr>
<td>( \hat{B}_i )</td>
<td>Ground-truth bounding box of ( i )th example.</td>
</tr>
<tr>
<td>( B_i )</td>
<td>Predicted bounding box of ( i )th example.</td>
</tr>
</tbody>
</table>

image and the corresponding true class label for each example \( t_i \), and bounding box \( \hat{B}_i \). Along with the class label \( t_i \), true label distribution for \( i \)th example is represented by \( \hat{p}_i \). Each prediction obtained from an object detector with class probability and bounding box are represented by \( p_i \) and \( B_i \) respectively. Raw classification score of \( i \)th example is represented by \( s_i \). Bounding boxes (\( B_i \) or \( \hat{B}_i \)) are represented by their top-left (\( x_1, y_1 \)) and bottom-right (\( x_2, y_2 \)) coordinates in 2D spatial domain. Overall representations for input/output relations are shown in Table 2.1.

2.2 Related Work

2.2.1 Most Commonly Used Loss Functions in Object Detection

To train deep object detection networks, the most popular approach is to weight contributors of the overall loss function as in Equation (1.1). The overall loss function for object detection is generally defined as:

\[
L_{p,t,B,\hat{B}} = L_c(p, t) + w_rL_r(B, \hat{B}).
\]

(2.1)

In this chapter, we are going to delve into the most commonly used classification loss functions, namely, cross-entropy loss and its variant focal loss \[8\]. After this, for investigating previously proposed localisation (regression) loss functions we will start with smooth \( L_1 \) loss, which is a widely-used regression loss function before the introduction of IoU loss and its variants as regression loss function \[6, 7, 8, 14, 21\].
Then, IoU loss [28] and its variants GIoU loss [30], DIoU loss [31] and CIoU loss [31] are explained in detail. Moreover, since one of the motivations of this thesis is balancing numerical range of individual loss function components (Section 1.1), the numerical ranges of each loss function are analyzed.

2.2.1.1 Classification Loss Functions

**Cross-Entropy Loss.** Cross-entropy loss calculates an error value between the true and predicted probability distributions. Cross-entropy loss for binary classification can be defined as follows:

\[
L_{CEc}(p_i, t_i) = -t_i \log(p_i) - (1 - t_i) \log(1 - p_i),
\]

(2.2)

where \(p_i\) is the estimated probability of the \(i\)th example obtained by applying the sigmoid activation to the classification logits, and \(t_i = 1\) if \(i\)th prediction belongs to the ground-truth class and \(t_i = 0\) otherwise. For \(N\) examples, per-sample calculated loss value can be averaged to obtain the overall classification:

\[
L_{CEc} = \frac{1}{N} \sum_{i}^{N} L_{CEc}^{CE}(p_i).
\]

(2.3)

**Focal Loss.** In object detection, one of the problems that harm the training process is foreground-background imbalance [3]. Two-stage detectors can overcome this problem by using an extra Region Proposal Network (RPN) which provides a more balanced foreground-background samples to the classification and localisation heads [13]. In contrast to two-stage detectors, one-stage methods mainly suffer from the foreground-background imbalance problem [8]. Being motivated from this problem, focal loss addresses imbalance by using a balanced cross-entropy loss function. Consider previously mentioned cross-entropy loss function is reduced to the form:

\[
L_{CE}^{CE} = - \log(p'_i).
\]

(2.4)

where \(p'_i := p_i\) if \(t_i = 1\) and \(p'_i := 1 - p_i\) otherwise. To balance loss contribution of positive and negative classes vanilla cross-entropy loss is weighted by a factor \(\alpha_i\) which is set by inverse class-frequency:

\[
L_{CE}^{balancedCE} = -\alpha_i \log(p_i).
\]

(2.5)
Authors of focal loss state that, while the balanced cross-entropy loss can overcome the foreground-background imbalance problem, it has no effect on weighting easy/hard examples. To achieve this goal, they modified the shape of cross-entropy loss function. To do this, they introduce an additional modulating factor with a focusing parameter $\gamma$ as follows:

$$L_{c}^{focal} = -\alpha_i (1 - p_i)^\gamma \log(p_i), \quad (2.6)$$

where $\gamma$ aims to control the contribution of easy/hard examples to the overall loss function as shown in Figure 2.1.

**Numerical Ranges of Classification Loss Functions.** Two classification loss functions cross-entropy and its modified version, focal loss, have the same numerical range $(0, \infty)$ because of the entropy-based similarity measurement (using log) between the true and the predicted distributions. Because of their numerical range, these loss functions cannot guarantee a bounded numerical range for their contribution to the overall loss function for object detection (Equation (2.1)).
Figure 2.2: Differences between $L_1$ and $L_2$ loss functions for measuring spatial distance. **Left:** Spatial visualization of bounding boxes with a ground-truth bounding box in blue and three predicted bounding boxes. **Right:** IoU values, $L_1$ and $L_2$ loss values are calculated between each predicted bounding box and the ground-truth bounding box. While measuring $L_2$ distance for inlier bounding boxes (e.g. green bounding box) provides more stable loss values, $L_1$ distance prevent outliers (e.g. yellow bounding box) to dominate the overall loss in contrast to $L_2$ distance. (Figure source: [3])

### 2.2.1.2 Localisation Loss Functions

In this section, we are going to review localisation loss functions that are used for training the regression branch of deep object detection methods ($L_r$ term in Equation (2.1)). There are mainly two groups of loss functions: $L_p$ distance based (e.g. $L_1$, $L_2$) and IoU based loss functions.

We assume that bounding boxes $B$ and $\hat{B}$ in Equation (2.1) satisfy the condition $B, \hat{B} \subseteq S \in \mathbb{R}^4$ and each bounding box is represented with its parameters: top left coordinates ($x_1$ and $y_1$), and bottom right coordinates ($x_2$ and $y_2$).

**$L_p$-Norm Based Loss Functions.** Prior work on object detection [8, 14, 32] use $L_p$ distance based loss functions for training the regression branch of object detectors. For $L_p$-norm based loss functions can be defined as follows:

$$L_p = \sum_{j \in x_1, x_2, y_1, y_2} ||B_j - \hat{B}_j||_p.$$  \hspace{1cm} (2.7)

A widely-used form of $L_p$-norm based loss functions is $L_2$ loss [32]. According to the definition in Equation (2.7), by replacing $p$ with 2, $L_2$ loss simply takes the Euclidean
distance between predicted and ground-truth bounding box coordinates. By following the same definition, $L_1$ Loss evaluates the absolute value of the difference between two boxes’ parameters. As shown in Figure 2.2, using $L_2$ loss causes outlier boxes to dominate the overall loss value, but $L_2$ loss provides better information for converging inlier boxes to ground-truth box faster than the $L_1$ loss.

Smooth $L_1$ loss [6] combines $L_2$ and $L_1$ distances to prevent outlier boxes from dominating the overall loss value:

$$L_r = \sum_{j \in \{\text{cls}, \text{loc}\}} L_1^{\text{smooth}}(B_j - \hat{B}_j),$$

where $L_1^{\text{smooth}}(\cdot)$ is defined as:

$$L_1^{\text{smooth}}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1, \\ |x| - 0.5 & \text{otherwise.} \end{cases}$$

Because of its balanced behaviour, smooth $L_1$ loss is widely accepted by the community, and it is used as regression loss in many deep object detectors [6, 7, 8, 14, 21]. The smooth $L_1$ loss and its gradient are shown in Figure 2.3.

![Figure 2.3](image-url)

**Figure 2.3:** (a) The gradients of the smooth and balanced $L_1$ loss functions, (b) The smooth and balanced $L_1$ loss functions (Figure source: [9]).

In Libra RCNN [9], authors argued that smooth $L_1$ loss still suffers from the dominating loss values of outlier bounding boxes. In order to balance inlier/outlier box contribution to the overall regression loss, they aim to improve the gradient values of inlier boxes. To do this, they start with the gradient of the loss function which promotes gradients of inlier boxes. In order to control the contribution of inliers, they
add the parameter $\alpha$ which controls the trend of the gradient of the function for inlier bounding boxes ($L_r < 1.0$):

$$\frac{\partial L_1^{\text{balanced}}}{\partial x} = \begin{cases} \alpha \ln(b|x| + 1) & \text{if } x < 1, \\ \gamma & \text{otherwise.} \end{cases}$$

(2.10)

Here small $\alpha$ value increases the contribution of inlier bounding boxes (Figure 2.3(a)). $\gamma$ is used to balance the contribution of inliers/outliers and draws and upper bound for the loss function.

By integrating the derivative in Equation (2.10), balanced smooth $L_1$ loss is obtained as follows;

$$L_1^{\text{balanced}}(x) = \begin{cases} \frac{\alpha}{2} (b|x| + 1) \ln(b|x| + 1) - \alpha|x| & \text{if } x < 1, \\ \gamma|x| + C & \text{otherwise.} \end{cases}$$

(2.11)

where the $b$ term is used to ensure that the function is continuous around when the regression error is $1.00$.

**Numerical Ranges of $L_p$-Based Loss Functions.** As in most commonly used classification branch loss functions, $L_p$-based loss functions cannot guarantee a bounded numerical range. As shown in Figure 2.3 any $L_p$ based loss function has a range of $(0, \infty)$.

**IoU Based Loss Functions.** Intersection over Union (IoU) is commonly used to measure the similarity between two boxes for evaluating performance of object detectors [2, 11, 25]. Let $B$ and $\hat{B}$ be ground-truth and predicted bounding boxes with top-left corner $(x_1, y_1)$ and bottom right corner $(x_2, y_2)$ satisfying that $x_2 > x_1$ and $y_2 > y_1$. The area of the bounding box $B$ is simply defined as: $A(B) = (x_2 - x_1) \times (y_2 - y_1)$. The are of the intersection of two boxes $B$ and $\hat{B}$ can be defined as: $I(B, \hat{B}) = (\min(\hat{x}_2, x_2) - \max(\hat{x}_1, x_1)) \times (\min(\hat{y}_2, y_2) - \max(\hat{y}_1, y_1))$. By adopting these two definitions, IoU can be defined as follows;

$$\text{IoU}(B, \hat{B}) = \frac{I(B, \hat{B})}{A(B) + A(\hat{B}) - I(B, \hat{B})}.$$  

(2.12)

Since IoU is a differentiable function with respect to the input bounding box parameters, it is possible to use this quality measurement as a loss function for the regression
Figure 2.4: A toy example that presents the differences between IoU and its variants. Note that, all boxes have the same width and height values of 1.00. (a) A single ground-truth bounding box (red) and predicted bounding boxes (green) that can help differentiating behaviour of IoU and its variants. (b) Bounding boxes with its center points and for each predicted bounding box; IoU, GIoU and DIoU values with the GT bounding box are calculated. Note that, each box is a unit box with different center point locations. B₀ is the perfect matching box with the ground-truth box which has the same IoU, GIoU and DIoU values. IoU can only provide discriminative information for B₁, for all other boxes IoU can not provide discriminative information since $I$ term in Equation (2.12) becomes 0.00. GIoU and DIoU can still measure the similarity between the ground-truth box and predicted bounding boxes $B_{\text{vic}}$, $B_{\text{dist}}$ and $B_{\text{inf}}$.

branch by adopting its definition to form a loss function such as: $1 - \text{IoU}(B, \hat{B})$ [28].

**Generalized IoU (GIoU) Loss.** GIoU loss [30], by introducing a penalty term that considers the minimum enclosing box $C$ which includes predicted and the ground-truth box, can measure the similarity between two boxes even if there is no intersection between them:

$$
\text{GIoU} = \text{IoU} - \frac{|C \setminus (B \cup \hat{B})|}{|C|}.
$$

As shown in Figure 2.5(b) GIoU allows minimizing the total area between two boxes when intersection is zero. As a result of this, in Figure 2.4 for boxes $B_{\text{vic}}$, $B_{\text{dist}}$ and $B_{\text{inf}}$ GIoU can still provide non-zero similarity measure as opposed to IoU.
Figure 2.5: A toy example that presents the geometric interpretation of IoU variants (GIoU and DIoU). (a) A ground-truth bounding box and single predicted bounding-box with their center points. (b) Minimum enclosing box $C$ that involve both ground-truth and the predicted bounding box. (c) An example that has the same IoU and GIoU values but, DIoU can still differentiate the quality measure between the ground-truth and predicted bounding boxes.

**Distance IoU (DIOU) Loss.** Although GIoU loss can still measure the quality between boxes even they have zero intersection, it is reduced to the IoU measure if the intersection term has the value of 1.00 as shown in Figure 2.5(c). In addition to this, while the intersection term approaches to 1.00 during optimization, convergence of GIoU loss gets slower. To tackle with this problem, DIOU loss [31] offers to measure distances between two boxes when the intersection term is zero:

$$DIOU = IOU - \frac{\rho^2(b, \hat{b})}{c^2},$$

(2.14)

where $b$ and $\hat{b}$ are the center point locations of predicted and the ground-truth bounding boxes respectively; $c$ is the diagonal length of the minimum enclosing bounding box $C$. As shown in Figure 2.4, DIOU loss can provide faster convergence for the outlier boxes (0.00 intersection value) because it measures the Euclidean distance between center points of boxes which provides faster convergence.

**Complete IoU (CIOU) Loss.** Since any prior IoU-based loss function does not consider the aspect ratio similarity between the target and predicted bounding boxes. In the same work that proposes DIOU loss [31], authors also consider measuring the
aspect ratio similarity between two boxes as follows:

$$\text{CIoU} = \text{IoU} - \frac{\rho^2(b, \hat{b})}{c^2} + \alpha v,$$

$$v = \frac{4}{\pi^2} \left( \arctan \frac{w_{gt}}{h_{gt}} - \arctan \frac{w}{h} \right)^2,$$

where $w$ and $h$ represent the width and height of the corresponding bounding boxes, and $\alpha$ is the positive trade-off parameter. CIoU loss is able to penalize bounding boxes by considering the overlap area ($\text{IoU}$ term), the distance between the center coordinates ($\rho^2$ term) and the aspect-ratio similarity ($v$ term).

**Numerical Ranges of IoU Based Loss Functions.** By definition, IoU is the normalized overlap area between two boxes (Equation (2.12)). Since overlap area is normalized by the total area of two boxes it can get the maximum value of 1.00 (consider $B_0$ in Figure 2.4). For two boxes with 0.00 intersection value, simply IoU value becomes 0.00. However, this is not the case for other variants GIoU and DIoU. In order to determine the lower bound of these two functions, we provide GIoU and DIoU measures for a theoretical bounding box with the center point at the location ($\infty, \infty$) ($B_{inf}$ and the ground-truth bounding box with the center point location at $(0, 0)$ in Figure 2.4). By considering the definition of GIoU, IoU term becomes zero, and the difference between the area of the box $C$ and two bounding boxes converges to the area of the box $C$ since the GIoU value becomes $-1.00$. For DIoU, again, IoU value gets the value of 0.00, and the distance between center points of two boxes converges to the length of the box $C$, as in the GIoU case, DIoU value also becomes $-1.00$. All in all, the numerical ranges of both GIoU and DIoU become $(-1, 1)$, while the numerical range of IoU is $(0, 1)$. CIoU Loss introduces an additional term that measures the aspect ratio similarity between two boxes as in (2.15). Since this additional term’s numerical range is $(0, 1)$, the numerical range of overall CIoU becomes $(-1, 2)$.

### 2.2.1.3 Combining Loss Functions of Two Branches

In this subsection, we are going to summarize previously examined loss functions from the numerical range perspective. As mentioned before, deep object detectors are trained by combining loss functions of two branches with a constant scaling factor ($w_r$ in Equation (2.1)) [6, 7, 8, 14]. The main reason for using a scaling factor is to
balance numerical ranges of the two loss functions for classification and localisation. Because the most commonly used classification loss functions have the numerical range of $(0, \infty)$, all regression branch loss functions require such a scaling factor. The effect of the scaling factor is evaluated in [27]. While in the COCO dataset, the balanced $L_1$ loss performs best with $w_r = 1.0$, IoU and its variants require $w_r = 2.0$ for the best performing result under AP measure. Note that these weight values are dataset and architecture-dependent.

2.2.2 Loss Functions that Considers the Association of Branches

As mentioned in Chapter 1.1, classification, and localisation branches should be considered as associated tasks. While the most commonly used loss function presented in Equation (2.1) discards this property, some recent works point out this problem and propose solutions to associate two tasks in object detection by modifying the loss function [33, 34].

The Prime Sampling Attention (PISA) method [33] argues the association between classification and localisation tasks. By considering the correlation between the localisation and classification tasks mentioned in Chapter 1.1, they offer to suppress some object proposals that have larger localisation loss values in classification task by boosting their classification loss gradients. They modified the localisation loss ($L_r$) in Equation (2.1) as follows:

$$L_{carl} = \sum_{i=1}^{n} c_i L_{smooth}^{1}(B_i, \hat{B}_i),$$

(2.17)

where $c_i$ is a term estimated from the classification branch:

$$c_i = \frac{v_i}{\frac{1}{n} \sum_{i=1}^{n} v_i},$$

(2.18)

with $v_i$ being:

$$v_i = ((1 - b)p_i + b)^k.$$  

(2.19)

Note here that $p_i$ is the estimated classification probability for sample $i$. $v_i$ is calculated by transforming classification probabilities with a $k$th order exponential function. And to obtain final scaling factor $c_i$, $v_i$ values are rescaled according to their
average value calculated from all examples. By means of CARL, the regression loss provides gradient to the classification branch and finally the localisation quality of high precision examples are boosted.

Another model, LapNet [34], aims to make \( w_r \) parameter in Equation (2.1) learnable. By following the formulation that is proposed by Kendall et al., in addition to learn task weights, they also aimed to make object proposal weights learnable to overcome with the sample-wise uncertainty issue occurring during training of object detectors. Loss functions formulations for classification and the localisation tasks proposed in LapNet [34] are as follows:

\[
L_{cls} = \lambda_{cls} \frac{1}{N} \sum_c \sum_a \lambda_{cls}^{c,a} \sum_{i,j} L_{cls}(c, a, i, j), \quad (2.20)
\]

\[
L_{loc} = \lambda_{loc} \frac{1}{N+} \sum_c \sum_a \lambda_{loc}^{c,a} \sum_{i,j} L_{loc}(c, a, i, j), \quad (2.21)
\]

where \( \lambda_{cls} \) and \( \lambda_{loc} \) are learnable task weights and \( \lambda_{cls}^{c,a}, \lambda_{loc}^{c,a} \) represents the learnable object proposal wise weights for class \( c \) and object proposal (anchor) \( a \) to overcome with the homoscedastic uncertainty.

### 2.2.3 Optimization of Ranking-Based Loss Functions

Recent years have witnessed a trend to train deep object detectors by directly using the performance measure used to evaluate object detectors as a loss function instead of using the conventional loss function defined in Equation (2.1). Because the most commonly used evaluation measure is AP, all works aim to optimize AP while training an object detector. One needs to calculate the rank of each output instance of an object detector to calculate the AP score. As mentioned in Section 1.1, the ranking operation is not easy to integrate into the end-to-end training process of a deep object detector. To this end, all works that we will summarize in this section try to dissipate the challenges caused by the non-differentiable behaviour of ranking operation.

In work RaMBO [5], authors aim to extract informative gradient information from the discrete behaviour of a combinatorial solver to lead the optimization process of a deep neural network. To achieve this goal they first define a combinatorial solver as
follows:

\[ w \mapsto y(w) : y(w) = \arg\min_{y \in Y} c(w, y), \quad (2.22) \]

where \( w \) is the continuous input (e.g., classification probabilities) and \( y \) is a discrete output which minimizes the cost function \( c(\cdot) \). In order to make this formulation fit to the case in object detection. They define a cost function \( c(\cdot) \) that will give the ranking of output instances such as:

\[ \text{rk}(y) = \arg\min_{\pi \in \Pi_n} y \cdot \pi, \quad (2.23) \]

where \( y \) represents the vector of classification score outputs, and \( \pi \) outputs all permutations of rankings when it is applied to classification scores \( y \). Authors state that \( \text{rk}(y) \) is a function that is piece-wise constant. This means that the gradients of this function at any point with respect to its inputs may have the numerical value of zero and be unhelpful for optimization process. It is argued by the authors that, gradient signals can be calculated by using a zero-order approximation but for high-dimensional inputs it requires many function evaluations which increases the computational cost. Instead of using this, they prefer to construct \( \lambda \) interpolated functions of \( \text{rk}(y) \) function:

\[ f_{\lambda}(w) = f(y_{\lambda}(w)) - \frac{1}{\lambda} \left[ c(w, y(w)) - c(w, y_{\lambda}(w)) \right], \quad (2.24) \]

where \( \lambda \) is a positive constant that controls the trade-off between preserving the shape of the original function and informativeness of the gradient signal. The gradient of \( f_{\lambda} \) can be derived as:

\[ \nabla f_{\lambda}(w) = -\frac{1}{\lambda} \left[ \frac{dc}{dw}(w, y(w)) - \frac{dc}{dw}(w, y_{\lambda}(w)) \right] = -\frac{1}{\lambda} \left[ y(w) - y_{\lambda}(w) \right]. \quad (2.25) \]

All in all, by means of the \( \lambda \) interpolated functions, they can obtain meaningful gradient signals to train a deep learning based architecture for a loss function that is non-differentiable. By following this, they re-formulate the AP as a loss function for training deep object detectors. Despite the method is well-designed and relies on a strong theory, their experimental results show that they can improve the performance of Faster R-CNN [6] with only 1.0 AP in PASCAL-VOC dataset.

Another solution, Distributional Ranking (DR) loss [35], proposes to mitigate the foreground/background imbalance problem in one-stage detectors [3] by re-formulating
the classification loss by considering the ranking of foreground (positive) and back-
ground (negative) samples. Unlike the standard cross-entropy loss, under the as-
sumption that an ideal ranking between positives and negatives can rank the positive 
examples above negatives with a large margin, they formulated the classification loss 
that is able to rank the positive example with lowest score higher than the negative 
example with the highest score. They stated that, optimizing such a loss function 
by considering a single positive/negative pair may cause instability. In order to pre-
vent this, they formulated the overall loss function by considering the distributions of 
positive/negative examples:

\[ P_+ = \min_{q_+ \in \Delta} \sum_{j} q_j p_{j+}; P_- = \max_{q_- \in \Delta} \sum_{j} q_j p_{j-}, \]

(2.26)

where \( q_+ \) and \( q_- \) represent the uniform distributions for positive and negative ex-
amples, and \( P_+ \) and \( P_- \) denote the expectations for again positive and negative examples. 
\( \Delta \) is the simplex that holds the condition \( \Delta = q : \sum_j q_j = 1, \forall j, q_j \geq 0 \). By following 
the definition in Equation (2.26), overall loss function in DR loss is as follows:

\[ \min_{\theta} L_{DR}(\theta) = \sum_{i=1}^{N} l(P_{t_i} - P_{t_i} + \gamma), \]

(2.27)

where \( \theta \) is the network parameters, \( \gamma \) is the margin between positive and negative 
examples and \( l \) is the loss function. Authors stated that, they used hinge-loss for \( l \).

However, for better convergence of the loss function, they replaced hinge-loss with 
its smooth variants. Although this work does not aim to directly optimize a ranking 
based loss function, it provides an insight about how the convert the classification loss 
function into a ranking task.

In Average Precision (AP) loss [29], authors aim to train deep object detector by 
replacing the classification branch loss function with AP measure. To do this they 
first formulated the loss function by using \( AP_{50} \) which represents the \( AP \) measure at 
IoU value 0.50:

\[ L_c^{AP} = 1 - AP_{50} = 1 - \frac{1}{|P|} \sum_{i \in P} precision(i) = 1 - \frac{1}{|P|} \sum_{i \in P} \frac{\text{rank}^+_i}{\text{rank}_i}, \]

(2.28)

where \( P \) represents the positive examples, \( \text{rank}^+_i \) represents the rank of positive ex-
ample \( i \) and \( \text{rank}_i \) represents the rank of example \( i \) among all samples. The main
contribution of this method is modeling the non-differentiable \textit{rank} function to obtain gradients by using perceptron learning algorithm. Since the proposed method in Chapter 5 hardly relies on this method, details are left for the Section 2.3.2.

2.3 Background

This section will give background information to form a basis for the methods proposed in this thesis. This thesis aims to train object detectors using the LRP metric as a loss function; we will first summarize the LRP metric and compare it with AP. Then, the details of AP-Loss \cite{29} are provided in order to describe how a rank-based performance measure (AP) is embedded in the end-to-end training of deep object detectors. By following these, in the last part of this section, we are going to extend the error-driven update methodology in AP-Loss in order to explain how the LRP metric can be integrated into the end-to-end training pipeline of an object detector as a loss function.

Since this thesis’s main objective is embedding the LRP metric as a loss function for training object detectors, the architectural details of object detectors other than loss functions are excluded from the background section by giving an overview in Chapter 1.

2.3.1 Localisation Recall Precision (LRP)

To motivate LRP, we will first present and critique Average Precision (AP).

2.3.1.1 AP and its deficiencies

Most object detectors are evaluated using the AP measure \cite{6,7,8,14}, and AP is the well-accepted performance measure in object detection challenges \cite{2,25}. However, AP has some deficiencies for the complete evaluation of object detectors.

AP is calculated over the predicted boxes by an object detector and corresponding ground-truth annotations. First, to obtain detection set, detector outputs below a con-
stant confidence score threshold are eliminated. For each class in the dataset, each
detection box is evaluated by calculating its localisation quality by calculating the
IoU value with its corresponding ground-truth box. After this step, using a constant
localisation quality threshold value ($\tau$), true positives (TPs), and false positives (FPs)
are obtained to plot class-specific a precision-recall (PR) curve. Theoretically, AP is
the area under the class-specific PR curve. However, in practice, PASCAL [25] and
COCO [2] style AP calculation is performed by dividing the recall domain into 11
and 101 evenly spaced points, respectively. After this, the corresponding precision
values are averaged in order to obtain AP value. These class-specific AP values are
averaged for all classes in the dataset to obtain a single performance measure, namely
mean AP (mAP). Despite its wide-acceptance, in [11], we examined AP’s drawbacks
and introduced a performance metric, namely Localisation Recall Precision (LRP).
In contrast to AP, the LRP metric can evaluate object detectors by considering both
localisation and classification performance by unifying two aforementioned tasks of
an object detector.

Before giving the formal definition of the LRP metric, we will delve into details about
AP measure deficiencies by comparing it with the oLRP (optimal LRP) metric. By
definition, oLRP is the minimum LRP error that can be achieved by an object detector
with different classification score thresholds $s$. To do this, a toy example that consists
of three detectors with different detection characteristics is presented in Figure 2.6.
The first detector in (a) can output high precision detections, but it cannot retrieve
some of the ground-truths (low recall). The second detector in (b) can retrieve all
ground-truth boxes with low confidence scores (low precision). The last detector
in (c) can also retrieve all ground-truth boxes, but it outputs several low confidence
false positive detections, reflecting the behaviour of a conventional object detection
algorithm. By considering the toy example shown in Figure 2.6 deficiencies of AP
measure can be listed as follows:

- **AP can not distinguish very different PR curves.** All detectors in Figure
  2.6 has the same AP value of 0.50 which is the area under each PR curve (d),
  (e) and (f). However, LRP can investigate the best performing point of each
detector in PR domain. For example the optimal point of D1 in Figure 2.6(a) is
calculated for precision axis by $1 - \text{FPError} = 1 - 0 = 1$ and for recall axis
Figure 2.6: Detection outputs of three different detectors (D1,D2,D3) on the same image from [10] with a single class (car) are visualized in the first row (a,b,c) with their corresponding PR curves (d,e,f) in the second row. Red, blue and green bounding boxes represents the ground-truth boxes, TP and FP detections respectively. Red-cross represents the optimal point on PR curve with the minimum LRP value for a single confidence score \( s \). (Figure source: [11])

by \( 1 - \text{FNE}r\text{r} = 1 - 0.5 = 0.5 \). While AP can provide general information about the performance of an object detector, oLRP can point to the optimal performing point of an object detector.

- **AP can not measure the localisation quality explicitly.** An important measure that exhibits the differences between object detection algorithms is the localisation quality. AP achieves this by averaging performances of object detection algorithms with a set of localisation thresholds (\( \tau \)); however, this approach can not directly measure the localisation quality of an object detector. Despite this, by means of the oLRP_{IoU} component, LRP becomes able to take localisation quality into account while evaluating performances of object detectors. Since, D3 in Figure 2.6 (c) outputs tighter boxes than D1 and D2 it...
has a significant difference in oLRP\textsubscript{IoU} components as 0.40 and this also affect oLRP value while AP is the same for all detectors.

- **AP is not confidence score sensitive.** AP is calculated over the sorted list of detection, ground-truth matches. As long as the sorted list is preserved, AP gives the same result. However, to calculate LRP or oLRP, detections are evaluated for confidence scores in range [0, 1].

- **AP is calculated over the interpolated PR curve.** To calculate AP value, for each recall point, the actual PR curve is interpolated using the highest precision value presented at the right-hand side of the current recall point. After this interpolation, the area under the interpolated precision curve is calculated. However, oLRP value is obtained by iterating LRP calculation over each recall point and getting the minimum LRP value. Both LRP and oLRP values are achieved over the original PR curve.

By considering the above differences between AP and LRP, LRP is better at representing an object detector’s performance since it can jointly evaluate two sub-tasks of object detectors.

### 2.3.1.2 Definition of LRP

To define LRP, let $X$ represent the set of ground-truth boxes and $Y$ represent the detection boxes predicted by an object detector. The LRP($X, Y_s$) error is computed by considering detector outputs($Y_s$), thresholded by the confidence score threshold $0 \leq s \leq 1$ and IoU threshold $0 \leq \tau \leq 1$ and ground-truth box set $X$. First detector outputs are sorted wrt their confidence scores and assigned with corresponding ground-truth boxes as in AP calculation. After these processes, the following terms are computed: (i) $N_{TP}$, number of true positives; (ii) $N_{FP}$, number of false positives; (iii) $N_{FN}$, number of false negatives. By using these quantities, LRP can be calculated as follows:

$$LRP(X, Y_s) := \frac{1}{Z}(w_{IoU}LRP_{IoU}(X, Y_s) + w_{FP}LRP_{FP}(X, Y_s) + w_{FN}LRP_{FN}(X, Y_s)),$$

(2.29)
where \( Z = N_{TP} + N_{FP} + N_{FN} \) is the normalization constant and the weights are defined as follows: \( w_{IoU} = \frac{N_{TP}}{1 - \tau} \), \( w_{FP} = |Y_s| \) and \( w_{FN} = |X| \). These weights control the contribution of each term and, preserves the metricity property of LRP. LRP\(_{IoU}\) measures the localisation quality of each valid detection box under IoU. The definition of this term is as follows:

\[
LRP_{IoU} := \frac{1}{N_{TP}} \sum_{i=1}^{N_{TP}} \left( \frac{1 - \text{IoU}(x_i, y_i)}{1 - \tau} \right),
\]

which measures the average localisation error between ground-truth and detection boxes, different than AP by means of this term LRP can directly include the localisation quality into the calculation of detector performance evaluation. Here \( y_i \in Y_s \) is a valid detection box in thresholded detection set and \( x_i \in X \) is its matching ground-truth box. \( \tau \) is the localisation quality threshold as in AP.

The second and the last term \( LRP_{FP}, LRP_{FN} \) measure the number of false positives and false negatives respectively. As given below, these two terms represent the precision-recall relation for a given detection set \( Y_s \):

\[
LRP_{FP}(X, Y_s) := 1 - \text{Precision} = 1 - \frac{N_{TP}}{|Y_s|} = \frac{N_{FP}}{|Y_s|},
\]

\[
LRP_{FN}(X, Y_s) := 1 - \text{Recall} = 1 - \frac{N_{TP}}{|X|} = \frac{N_{FN}}{|X|}.
\]

Finally, as mentioned in the comparison between AP and LRP, \( oLRP \) is the minimum achievable LRP error in confidence score domain. To make \( oLRP \) parameter free, one can evaluate the object detector for a single \( \tau \) value. And in our original work [11] we use \( \tau = 0.5 \) for definition of \( oLRP \). The formal definition of \( oLRP \) is as follows:

\[
oLRP := \min_s LRP(X, Y_s).
\]
detector, oLRP values are average for each class over all classes \((C)\) in a dataset:

\[
\text{moLRP} := \frac{1}{|C|} \sum_{c \in C} \text{oLRP}_c.
\] (2.34)

### 2.3.2 Average Precision (AP) Loss

This section will summarize AP-Loss \([29]\) that aims to integrate AP performance measures as a classification branch loss function. This work’s main contribution is modeling ranking as a differentiable function to integrate AP measure into the training process of an object detector. The definition of the loss function proposed in AP-Loss is given in Equation (2.28) which shows that AP’s calculation relies on the ranking operation. Pogančić et al. \([5]\) stated that ranking operation is a piece-wise constant function. In addition to this claim, we visualized the ranking function’s behaviour with respect to classification confidence scores in Figure [1.5(a)]. In contrast to the cross-entropy loss, which measures the log similarity between two distributions (Figure [1.5(b)]), gradient signals that are derived from the basic ranking function cannot provide information to lead the optimization process of a deep object detector \([5]\).

Object detection pipeline up to the layer that outputs confidence scores and calculation of difference transforms are differentiable. After obtaining raw classification score of each example \((s_i)\), by means of a step function \(H(\cdot)\) these scores can be converted into rank value for each example. \(\text{rank}(i)\) can be expressed by partitioning the detection set into two parts as positive \(P\) and negative set \(N\) as follows:

\[
\text{rank}(i) = 1 + \sum_{j \in P, j \neq i} H(x_{ij}) + \sum_{j \in N} H(x_{ij}),
\] (2.35)

where \(x_{ij} = -(s_i - s_j)\). This score difference transformation return positive values if \(s_i < s_j\) which satisfies \(x_{ij}\) gets positive value if detections other than \(i\)th example receives higher confidence score values following that, \(H(x) = 1\) if \(x \geq 0\) and \(H(x) = 0\) otherwise. Together with Heaviside step function \(H(\cdot)\) the score transformation \(x_{ij}\) ensures calculating \(\text{rank}(i)\). The first term 1 stands for the \(i\)th example itself, the second term accumulates the rank values among other positive examples other than \(i\)th example and the third term repeats this operation among negative examples. By following the same notation in Equation (2.35) we can write \(\text{rank}^+(i)\) as
follows;
\[
\text{rank}^+(i) = 1 + \sum_{j \in P, j \neq i} H(x_{ij}).
\] (2.36)

Finally, Equation (2.28) can be re-written explicitly by using Equations (2.35) and (2.36):
\[
L_{cAP} = 1 - 1 - \sum_{i \in P} \frac{\text{rank}^+(i)}{|P|} \sum_{j \in P, j \neq i} H(x_{ij})
\] (2.37)
\[
= 1 - \frac{1}{|P|} \left( 1 + \sum_{j \in P, j \neq i} H(x_{ij}) \right)
\] (2.38)
\[
= \frac{1}{|P|} \sum_{i \in P} \sum_{j \in N} L_{ij} = \frac{1}{|P|} \sum_{i,j} L_{ij} \cdot y_{ij} = \frac{1}{|P|} \langle L(x), y \rangle,
\] (2.39)

where \( L(\cdot) \) is a vector valued activation function that takes vector of difference transform applied on confidence scores of detection pairs \( x \), \( y_{ij} \) is obtained by applying ranking label transformation which transforms true label \( t_i \) into true ranking as follows:
\[
\forall i, j, y_{ij} = 1_{t_i=1, t_j=0},
\] (2.40)
such that \( y_{ij} \) is equal to 1 if and only if the \( t_i = 1, t_j = 0 \) condition holds, otherwise 0. Finally, the loss function is reduced to the inner product between the vector form of primary term \( L_{ij} \) and true rank vector \( y \), with network parameters \( \theta \):
\[
\min_{\theta} = 1 - \text{AP}_{50}(\theta) = \frac{1}{|P|} \langle L(x(\theta)), y \rangle.
\] (2.41)
Since the resulting loss function in Equation (2.41) is not differentiable with respect to the network parameters, by inspiration from the perceptron learning algorithm \[36\] authors proposed that the error-driven update can be used with backpropogation in an end-to-end architecture. Error-driven update provides gradient information for the non-differentiable second part of the network. The input for the second part of the network is \( x_{ij} \) and after applying the activation function \( L(\cdot) \) the primary term \( L_{ij} \) is obtained (Equation (2.38)). The update for \( x_{ij} \) term is calculated by taking difference between desired (\( L_{ij}^* \)) and current (\( L_{ij} \)) value of the primary term:
\[
\Delta x_{ij} = L_{ij}^* - L_{ij}.
\] (2.42)
The minimum achievable value of desired primary term is 0.00 which is achieved when it satisfies the perfect ranking among output examples. The perfect ranking
condition holds if and only if $y_{ij} = 1$ as defined in Equation (2.40), otherwise $y_{ij} = 0$ which has no contribution to AP-Loss, since it is ignored during error-driven update. All in all, update rule for second-part of the network can be simplified as:

$$\Delta x_{ij} = L_{ij} \cdot y_{ij}. \quad (2.43)$$

By following this, the derivative of $L^c_{AP}$ wrt $x_{ij}$ becomes $-\Delta x_{ij}$. Therefore, the gradient of AP-Loss with respect to classification scores can be calculated as follows:

$$\frac{\partial L^{AP}}{\partial s_i} = \sum_{j,k} \frac{\partial L^{AP}}{\partial x_{jk}} \frac{\partial x_{jk}}{\partial s_i} = \sum_j \Delta x_{ij} - \sum_j \Delta x_{ji}. \quad (2.44)$$

### 2.3.3 Extending the Error-Driven Update

We summarized the formulation of the ranking-based loss function in AP-Loss in Section 2.3.2. To form a basis for Chapter 5, by following our work [4], we explain the method that can generalize the ranking-based formulation proposed in AP-Loss into a broader set of ranking-based functions. By following this, we primarily aim to formalize LRP [11] as a loss function. First, we will define targeted ranking-based functions; then, by introducing a formal definition for primary terms, we will explain the general error-driven approach proposed in [4].

Assuming we have a set of examples, $D$, the ranking task aims to promote ranking of samples in a subset $A \subset D$ (positives), while demoting the remaining set, $B = D - A$ (negatives). Here, we consider ranking functions as a function of the set $D$, $\mathcal{R}(D)$, that can be decomposed for a sample $i$ in the set $D$ on either it is a positive or negative ($i \in A$ or $i \in B$) as follows:

$$\mathcal{R}(D) = \frac{1}{Z} \sum_{i \in \mathcal{C}} \mathcal{E}^R(i), \quad (2.45)$$

where $\mathcal{E}^R(i)$ is the local error of a single sample $i \in \mathcal{C}$. Here, $\mathcal{C}$ represents the positive subset ($A$) or negative subset ($B$) and $Z$ is a normalization constant. For defining generalization of the error-driven update rule for $\mathcal{R}(D)$ we are going to set $\mathcal{C} = A$. First, we will identify a generalization of primary terms, and with this generalization, we will identify gradients.

**Step 1: Identifying Primary Terms.** Here, we provide a general definition of the primary terms of $\mathcal{R}(D)$. For $\mathcal{R}(D)$, the primary term between tuple of examples $(x_{ij})$
where \( i \in A \) and \( j \in B \) (otherwise the primary term is set to 0 according to Equation (2.40)), \( \Psi_{ij}^R \), is the reciprocal-error originating from \( i \) and distributed over \( j \) via the probability density function \( p^R(i, j) \) (i.e. \( \forall i \sum_{j \in B} p^R(i, j) = 1 \)). Formally,

\[
\Psi_{ij}^R = \mathcal{E}^R(i) \times p^R(i, j).
\] (2.46)

Then, we can compute the value of \( \mathcal{R}(D) \) by summing over this reciprocal errors:

\[
\mathcal{R}(D) = \frac{1}{Z} \sum_{i \in A} \mathcal{E}^R(i) = \frac{1}{Z} \sum_{i \in A} \mathcal{E}^R(i) \left( \sum_{j \in B} p^R(i, j) \right) = \frac{1}{Z} \sum_{i \in A} \sum_{j \in B} \Psi_{ij}^R. \quad (2.47)
\]

Defining primary terms in this fashion provides following advantages over the top-down approach: (1) \( \mathcal{E}^R(i) \) is trivial (e.g. \( \mathcal{E}^{AP}(i) = (1 - \text{precision}(i)) \)). (2) Determining a \( p^R(i, j) \) becomes more intuitive and adds more flexibility to the error distribution. To illustrate, while \( p^R(i, j) = \frac{H(x_{ij})}{\sum_{k \in N} H(x_{ik})} \) distributes the error uniformly over \( j \in B \) with \( s_j \geq s_i \), a more left-skewed one will promote harder examples (i.e. larger \( x_{ij} \)) more.

**Step 2: Identifying Gradients.** Here, we define the “error” of a difference transformation to use it for the update following the perceptron learning algorithm. In particular, \( x_{ij} \) is updated by its deviation from “the primary term when \( i \) is ranked properly” denoted by \( \Psi_{ij}^{R*} \), as follows:

\[
\Delta x_{ij} = \Psi_{ij}^{R*} - \Psi_{ij}^R.
\] (2.48)

Then, the gradients of the \( R \) wrt the input of the difference transform is determined as:

\[
\frac{\partial \mathcal{R}(D)}{\partial s_i} = -\sum_{j,k} \Delta x_{jk} \frac{\partial x_{jk}}{\partial s_i} = \sum_j \Delta x_{ij} - \sum_j \Delta x_{ji},
\] (2.49)

\[
= \sum_{j \in A} \left( \Psi_{ji}^R - \Psi_{ji}^{R*} \right) - \sum_{j \in B} \left( \Psi_{ij}^R - \Psi_{ij}^{R*} \right).
\] (2.50)

By following this, the magnitudes of the total gradients of the examples to be promoted (i.e. \( A \)) and demoted (i.e. \( B \)) are equal at every training iteration, which implies balanced training. Formally,

\[
\left| \sum_{i \in A} \frac{\partial \mathcal{R}(D)}{\partial s_i} \right| = \left| \sum_{i \in B} \frac{\partial \mathcal{R}(D)}{\partial s_i} \right|. \quad (2.51)
\]
CHAPTER 3

EQUALIZING THE RANGES OF THE LOSSES TO IMPLICITLY OPTIMIZE LRP

One of the issues in the conventional loss function (Equation (2.1)) used in training object detectors is the unbounded and unbalanced contribution of its components (classification and regression). Most object detectors use cross-entropy loss as the classification branch loss function and smooth $L_1$ Loss as the regression loss function [6, 7, 14]. Unlike these detectors, for the classification branch loss function, RetinaNet [8] modifies the cross-entropy loss to reduce the foreground/background imbalance problem and make modifications in the shape the of loss function to balance contributions of hard/easy examples. Neither of these classification branch loss functions has a bounded numerical range as examined in Section 2.2.1.1.

The same unbounded numerical range issue exists for the most commonly used regression branch loss function, smooth $L_1$. As a result, these loss functions contribute to the overall loss function during training, as visualized in Figure 1.2. To balance their contributions, the typical approach is weighting the localisation branch loss with a scalar, which requires validation experiments and hard to tune for obtaining optimal results.

3.1 Motivation

In order to propose a solution for these issues, motivated from the definition of LRP (Equation (2.29)), we aim to introduce loss functions with equal numerical ranges. Our expectation is that this approach allows us to indirectly optimize LRP.
Since LRP has a localisation component LRP\textsubscript{IoU}, we simply changed the localisation branch loss function with IoU Loss \cite{28} which has the numerical range of $[0, 1]$ as explained in Section 2.2.1.2. After this modification, our main idea is to propose a classification branch loss function with the same normalized range of IoU loss.

The main goal is to replace the conventional classification branch loss (e.g. cross-entropy) that compares predicted and true score distributions with an analytic function which aims to implicitly minimize LRP\textsubscript{FP} component of LRP. Because of the dense anchor sampling nature in deep object detectors \cite{6, 8} LRP\textsubscript{FN} becomes negligible (LRP\textsubscript{FN} = $\emptyset$). Following these, the loss function that approximates LRP error can be defined as follows;

$$L_{LRP}(x_i, y_i) \approx \frac{1}{N_{TP}} \sum_{i=1}^{N_{TP}} \left( \frac{1 - \text{IoU}(x_i, y_i)}{1 - \tau} \right) + N_{FP}, \quad (3.1)$$

where $y_i$ is the predicted detection output which matches with the ground-truth bounding box $x_i$. While $N_{FP}$ requires ranking operation, we can approximate this value by directly passing classification probabilities ($p_i$) to an analytical function $f(\cdot)$ that produces outputs in $[0, 1]$:

$$N_{FP} \approx f(t - p_i), \quad (3.2)$$

where $t - p_i$ term is used to measure how confidence score is distant from the expected value $t$. As a baseline method we directly use $t - p_i$ for representing $N_{FP}$ in loss function, namely linear loss. Since $p_i$ values are in the range of $[0, 1]$, this does not violate the numerical balance between loss components.

### 3.2 Shifted Cross-Entropy Loss

We start with the cross-entropy loss and its variant focal loss for $f(\cdot)$ function. The vanilla cross-entropy loss and focal loss has the range of $[0, \infty]$. To ensure that $f() \in [0, 1]$ we shifted the cross-entropy loss and focal loss as shown in Figure 3.1(a).

The emerging problem about the shifted cross-entropy loss is getting negative values for $p_i > 0.64$ as shown in Figure 3.1(a). In order to overcome this issue, we apply the modulation term used in focal loss \cite{8}. The final version of shifted cross-entropy loss
can be defined as follows:

\[ L_{CE}^\gamma(p_i) = -(1 - p_i)^\gamma \log(p_i + e^{-1}). \] (3.3)

The shifted cross-entropy loss with modulation term cause numerical inconsistency with respect to its inputs \( p_i \). As shown in Figure 3.1(b) (e.g. when \( \gamma = 1 \)), the shifted cross-entropy loss without modulation produces negative values while each shifted version follows the trend of first decreasing and then increasing, which produce inconsistent results for loss values; i.e. a prediction with lower confidence score may produce lower loss value than a prediction with higher confidence score value.

### 3.3 Exponential Loss

Because of the previously mentioned numerical inconsistency issues in shifted cross-entropy loss, we examined a different \( f(\cdot) \). By following Equations (3.1) and (3.2), we can define an approximated loss function based on the exponential function:

\[ L_{CE}^{exp}(p_i) \approx e^{-p_i/\gamma}, \] (3.4)

such as \( \gamma \) is used for modulating the loss function as in shifted cross-entropy loss. A comparison of cross-entropy and modulated exponential loss with different \( \gamma \) values is shown in Figure 3.2.
Figure 3.2: Comparison of original cross-entropy loss with exponential loss with different scaler parameters. While cross-entropy loss has an unbounded numerical range, exponential loss in bounded in the range of $[0, 1]$. Different scaler values ($\gamma$) implies modulation of exponential loss to weight hard examples (according to their $p_i$).

3.4 Validation of Exponential Loss

In order to validate exponential loss, we first evaluated its performance on a classification task. To this end, we trained the ResNet-18 [37] model on the CIFAR-10 dataset [38] by using cross-entropy loss, focal loss, exponential loss, and a simple loss function that maps the classification probabilities directly into loss values by applying $L_c = 1 - p_i$, where $p_i$ represents the classification probabilities.

In addition to experiments in the original CIFAR-10 dataset, to investigate the robustness of exponential loss to imbalanced data, we simulated an imbalanced class distribution case. Using the $\chi^2$ probability distribution function, we re-weighted each class’s loss values with a degree of freedom parameter $k = 1$. The original and imbalanced distributions of class loss weights are shown in Figure 3.3.

Implementation Details. We used the official PyTorch [39] implementation of the ResNet-18 model with a ten-dimensional fully connected output layer for the CIFAR-10 dataset. All proposed models are trained for 300 epochs by SGD with momentum.
The results of validation experiments are presented in Table 3.1. We use linear loss results as baseline for comparing balanced and imbalanced dataset results. We see that, while exponential loss without using modulation (γ = 1) cannot cope well with the imbalanced cases, by using modulation, exponential loss achieved similar performance with cross-entropy loss and focal loss in both balanced and imbalanced cases. As focal loss [8] show that modulating loss function can overcome foreground/foreground imbalance, classification experiments in imbalanced dataset imply the same situation. Because of this, modulating exponential loss can overcome this issue. In addition to this, the classification task experiments shows that exponential loss with modulation can be a good bounded candidate for replacing cross-entropy loss.

3.5 Exponential Loss in Object Detection

To evaluate exponential loss for object detection, we trained Faster R-CNN [6]. Our experimental results show that indirectly optimizing LRP loss function by the pro-
Table 3.1: Exponential loss validation experiments results in CIFAR-10 Dataset. Each cell represents the accuracy of the trained model on CIFAR-10 test set. Linear Loss is used as a baseline method for both balanced and imbalanced cases. Balanced classification task experiments show that exponential loss is a good numerical range balanced candidate for replacing cross-entropy Loss. In imbalanced case, while exponential loss can not reflect the same behaviour of cross-entropy loss especially for hard examples ($p_i < 0.5$), exponential loss with $\gamma = 0.2$ results show that using its scaled version can overcome this issue.

<table>
<thead>
<tr>
<th>Loss Function</th>
<th>Balanced</th>
<th>Imbalanced ($k = 2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-Entropy</td>
<td>92.37%</td>
<td>91.14%</td>
</tr>
<tr>
<td>Focal Loss ($\gamma = 2$)</td>
<td>91.24%</td>
<td>91.20%</td>
</tr>
<tr>
<td>Exponential Loss ($\gamma = 1$)</td>
<td>91.05%</td>
<td>61.03%</td>
</tr>
<tr>
<td>Exponential Loss ($\gamma = 0.2$)</td>
<td>92.30%</td>
<td>92.09%</td>
</tr>
<tr>
<td>Linear Loss</td>
<td>74.63%</td>
<td>36.2%</td>
</tr>
</tbody>
</table>

To test our approach, we train Faster R-CNN with a combination of cross-entropy and IoU loss functions for fair comparison. Then, by only replacing the classification branch loss function with the proposed exponential loss function we conducted experiments for different scalar $\gamma$ values and different sampling ratio values for the sample at the second stage of Faster R-CNN. In all experiments models are trained for 2 epochs in COCO [2] trainval35K dataset. The performance of the Faster R-CNN with different $\gamma$ values and sampler ratios are reported in COCO [2] minival split. Results are shown in the Table 3.2.

While the baseline Faster R-CNN model reaches 15.3 AP in two epochs, the best performing setting with exponential loss in Faster R-CNN can only achieve 6.3 AP. To investigate issues in exponential loss, we analyzed the Faster R-CNN model with best performing exponential loss and cross-entropy loss. In this analysis, we partition loss values at the second stage of Faster R-CNN as foreground (sum of all valid class losses) and background to investigate the robustness of loss functions to the foreground/background class imbalance [3]. As shown in Figure 3.4 while cross-
Table 3.2: Exponential loss grid-search results for $\gamma$ and number of samples sampled from foreground (fg) / background (bg) classes at second stage. The results proposed in each cell are AP measures. $\cdots$ represents no results for the corresponding cell because regardless of the sampling ratio for the corresponding $\gamma$ value detector cannot achieve a significant performance. The best performing combination is written in bold.

<table>
<thead>
<tr>
<th>Sampling Ratio (fg:bg)</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.05</td>
</tr>
<tr>
<td>128 : 384</td>
<td>0.8</td>
</tr>
<tr>
<td>128 : 128</td>
<td>3.8</td>
</tr>
<tr>
<td>128 : 64</td>
<td>4.3</td>
</tr>
<tr>
<td>96 : 32</td>
<td>4.8</td>
</tr>
<tr>
<td>128 : 32</td>
<td>4.7</td>
</tr>
</tbody>
</table>

entropy loss can achieve a balanced loss distribution between foreground and background classes, the exponential loss cannot achieve a balanced distribution. Because of the imbalanced loss distribution behaviour between foreground and background classes, the object detection model trained with exponential loss becomes biased to foreground classes, as shown in Figure 3.4 (b). A final comment on exponential loss can be that it requires extensive experiments to find the sampler fg/bg ratio parameter and its modulation factor $\gamma$. We showed exponential loss could overcome the fg/bg imbalance issue in classification experiments in Table 3.1. However, because of its sensitivity to sampler parameters, it cannot achieve even on-par performance with cross-entropy loss in detection experiments because of the fg/bg imbalance issue.
Figure 3.4: Loss distribution for foreground and background classes in object detection experiments. Loss values are collected during the first epoch snapshot of the whole training process. (a) Cross-entropy loss fg/bg loss distribution. Cross-entropy loss can achieve balanced loss values for all bins when it is compared to the exponential loss. (b) Exponential loss fg/bg loss distribution. Exponential loss produces biased training for foreground examples.
CHAPTER 4

TRAINING OBJECT DETECTORS BY REGRESSING LRP

Object detectors are trained by using loss functions that have no direct relationship with their performance evaluation criteria, neither AP nor LRP \[6, 7, 8\]. This apathy between training and evaluation methods causes an unconsidered performance gap for object detectors, as discussed in Section 1.1. Both AP and LRP are functions that require ranking operation. Due to the non-differentiable nature of ranking operation discussed in Section 1.1, deep object detection architectures cannot be trained directly using their performance evaluation criteria. Pogančić et al. \[5\] discussed that ranking-based functions are piece-wise constant. Due to this, they can only provide gradient information of zero at their constant regions, and due to the discontinuity between these constant regions, they cannot lead the optimization process of a deep neural network architecture to converge. They proposed a method that can interpolate these regions, and by the aid of this interpolated loss function they can get informative gradient information for training. By following the same idea, in this chapter, we propose a neural network-based regressor that aims to estimate LRP values in order to train an object detector with LRP.

4.1 Motivation

Due to their differentiable building blocks, neural networks are differentiable function estimators. Using discrete data as an input to a neural network, they can approximate parameters of a function that constructs the discrete data. We questioned if LRP values can be approximated by thinking out of the box using a neural network as a function estimator. After the successful training of a LRP estimator, we aim to attach
Figure 4.1: Proposed neural network based LRP estimator. "Object Detector" module represents a generic object detection algorithm runs in inference mode and outputs prediction bounding box set $Y_s$. "Ground-truth annotations" module represents annotated bounding boxes with their class labels $X$. The "Similarity estimator" module is a neural network based architecture that outputs similarity scores between ground-truth and prediction sets. By providing these similarity scores into the "regressor" module, LRP values are estimated. Modules in green are trained in a supervised manner by using ground-truth LRP values provided by the LRP calculation module.

this estimator at the end of an object detector to lead it in the training process using the gradient values provided by this estimator.

LRP as a metric for object detection aims to evaluate object detectors by measuring the similarity between a set of predicted bounding boxes with classification probabilities ($Y_s$) and corresponding ground-truth set ($X$) as mentioned in Section 2.3.1. By following this definition, our overall architecture for estimating LRP values is shown in Figure 4.1.

The proposed architecture aims to train the similarity estimator and regressor modules offline with the predicted bounding box outputs during object detector training and corresponding ground-truth bounding boxes. The designed architecture allows us supervised training of these modules by using true LRP values. The proposed regression method’s performance can be evaluated by partitioning the dataset collected during the object detector training.
4.2 A Validation Experiment for Regressing Loss Functions

To validate the idea of regressing the values of loss function, we conducted a simple experiment that aims to approximate loss values during training of a classifier with cross-entropy loss. By using a synthetic dataset that contains discretized sample points with their corresponding cross-entropy loss values, we train a simple MLP architecture. The results in Figure 4.2 suggest that even a simple MLP architecture can estimate loss values from its classification probabilities.

We benchmark the cross-entropy loss estimator by comparing its performance with analytic cross-entropy loss for classification problem in MNIST [40] dataset with a lightweight convolutional neural network architecture. While analytic cross-entropy loss achieves 98.45% test accuracy in 10000 test images, the same architecture with estimated cross-entropy loss can achieve 98.41% test accuracy with the same data.

This experiment supports our hypothesis that (i) a network can be trained to approximate a loss function and (ii) this estimator network can be used to train another
network.

4.3 LRP Estimation Module

In this section, we will delve into the details of the LRP estimation module that consists of two sub-modules; a similarity estimator and a regressor shown in Figure 4.1. This module’s primary purpose is to learn the association between two sets (prediction and ground-truth). Before estimating LRP as a bi-product, we first solve the association problem with the proposed architecture. After that, we extended this association network to solve the LRP estimation problem. Thus, in this chapter, we propose two architectures; the first one is for solving the association problem and, by extending this, the second architecture aims to solve the LRP estimation problem. Before explaining architectures’ details, we will explain our dataset generation strategies for both the association network and the LRP estimation module.

4.3.1 Dataset Generation

The LRP estimation module is trained in a supervised manner with the data collected during an object detector training. Since we aim to estimate LRP values, we structured our dataset by constructing \(<\text{prediction}, \text{ground-truth}>\) pairs \((X_i, Y_{X_i})\) that are inputs for LRP calculation for evaluating the performance of an object detector. A single bounding box output data structure of a prediction of object detector contains bounding box coordinates \([\hat{x}_1, \hat{y}_1, \hat{x}_2, \hat{y}_2]\) with classification probabilities of the estimated class \(\hat{p}^c\) and its corresponding background probability \(\hat{p}^0\). And its corresponding ground-truth bounding box follows the similar structure with bounding box coordinates \([x_1, y_1, x_2, y_2]\) and classification labels \(p^c = 1\) and \(p^0 = 0\).

We constructed the dataset with the data collected during the object detector’s training stage for association experiments. We manipulated the collected dataset to generate all possible pairings representing the matchings of predicted and ground-truth bounding boxes.

For the LRP estimation task, using an object detector’s outputs yields limited data.
Since the best-performing detector can only achieve moLRP value of 0.66 \[11\]. Because of this, we follow a stochastic box generation strategy for LRP estimation experiments which has the following properties:

- Running an object detector on an image and using the detection results with the optimal confidence score threshold \[11\]: In such a case, the regressor can learn the interval between around [0.5, 0.8].

- Running an object detector on an image and using the detection results with arbitrary confidence score thresholds sampled from $U(0, 1)$: In such a case the regressor can learn the confidence score interval between around [0.5, 1], but mostly concentrating on [0.8, 1].

- Randomly perturbing the ground-truth to generate samples in [0,0.5] interval. Since we will use the LRP as a loss function, the regressor must know the low results during the minimization process in training. For inference, one needs to plug the input into the network to have the regressed value.

By following the listed data generation strategies, we can obtain a dataset that has a uniform distribution between the confidence score interval of [0, 1].

4.3.2 Similarity Estimator

The first sub-module in the overall architecture shown in Figure 4.1 is the similarity estimator sub-module. Since our aim is to measure the similarity between prediction and ground-truth pairs we use a Discriminative Deep Metric Learning (DDML) architecture based on Siamese Network \[12\] – Figure 4.3. This work mainly aims on face-verification problem. As proposed in the original work, by accepting two image pairs as inputs by the aid of shared-weight similar deep neural networks they represent pair of face images in metric-space. By using this metric-space representation, the similarity between two spaces is used as a loss function to learn whether image pairs belong to the same face or not. Given a pair of images $x_i, x_j \in \mathbb{R}^d$ DDML aims to find a mapping $f : \mathbb{R}^d \mapsto \mathbb{R}^{p(M)}$ where $p^{(M)}$ represents the feature space representation of image pairs extracted by a neural network with cascaded $M$ layers.
Figure 4.3: The DDML Architecture for Face Verification in work [12]. By applying image pairs \((x_i \text{ and } x_j)\) as an input to two similar networks that share parameters \((W \text{ and } b)\), distance (or similarity) between two image pairs are measured in their corresponding feature-space representations \((h_1 \text{ and } h_2)\). By comparing the thresholded distance value with corresponding labels they train the whole architecture to discriminate face image pairs. Blue arrows shows the forward pass, green arrows shows the gradient flow path for backpropagation.

At the final layer they measure the distance between feature space representations as follows:

\[
d^2_d(x_i, x_j) = \|f(x_i) - f(x_j)\|^2_2.
\]  

(4.1)

After calculating the distance between feature space representations of image pairs, they apply a pre-defined thresholds \(\tau_1 \text{ and } \tau_2\) to discretize continuous distance values to construct a classification problem (whether two images belong to same image or not). For example, if \(d^2_d(x_i, x_j) < \tau_1\) two images corresponds to the same face, and if \(d^2_d(x_i, x_j) > \tau_2\) face image pairs belong to different faces.

By following this idea we construct a DDML architecture to measure similarity between the ground-truth set \(X\) and prediction set \(Y_s\). For these two sets by following
the previously explained data structure in Section 4.3.1, we proposed a similarity estimator network. The details of this network are shown in Figure 4.4.

\[ \Delta_i = ||\hat{\alpha} - \alpha|| \]

Figure 4.4: Similarity estimator module inspired from [12]: The predicted bounding box and ground-truth bounding boxes are denoted by corner coordinates respectively as follows, \([\hat{x}_1, \hat{y}_1, \hat{x}_2, \hat{y}_2]\) and \([x_1, y_1, x_2, y_2]\) respectively. Corresponding \(p\) values are background and class confidence scores for this box. For the ground-truth \(p_c = 1\) and \(p^0 = 0\).

The similarity estimator module consists of a senile layer with 3 ReLU neurons, as shown in Figure 4.4. This network aims to measure the similarity between two sets of inputs constructed by predicted and ground-truth bounding boxes. In the original work [12], this network is used to measure the distance \(\Delta_i\) between two feature-space representations \(\hat{\alpha}\) and \(\alpha\). The only difference from that work, rather than discretizing the distance measure outputs, by adding a regressor sub-module at the end of the similarity estimator sub-module as shown in Figure 4.1, we aim to predict whether two bounding boxes are associated or not for association experiments. For training object detectors, the regressor outputs the estimated LRP values.
4.3.3 Association Network

Basically, given two sets of examples, the aim of the association network is to determine the associated pairs and unassociated pairs. One example of the association problem’s application is the Bayesian multi-target tracking, in which the problem is to find the corresponding measurement among a set of gathered measurements for each target at every time interval. Thus, usually a hand-crafted association function is utilized. Outputs of this function construct an input set to the Hungarian Algorithm. There are also applications in object detection. In video object detection, at each time step, there is a set of detections that are to be associated with the ones from the previous time step. In [41], the authors devised a hand-crafted function for this aim again.

For computing the AP or LRP performance, the detection, and the ground-truth sets need to be associated. In this section, we propose a model to learn the association. To do this, we reduce the problem to classification. In which each example in one of the sets is labeled as either associated or unassociated.

![Learning based association architecture](image)

Figure 4.5: Learning based association architecture. Detection and ground-truth pairs are fed into the DDML module explained in Section 4.3.3. By means of the similarity estimator, similarity measures between ground-truth and detection pairs ($\Delta_i$) are fed into the associator. Overall architecture is trained in supervised manner as a classification problem (Pair associate or not).
Our association network is depicted in Figure 4.5. The input of the model is generated by manipulating the detections of an object detector and its related ground-truth set, as mentioned in Section 4.3.1. This manipulation involves generating all possible matching alternatives, i.e., pairings. A similarity score is then generated for each pairing by the discriminative deep metric learning (DDML) consisting of a single layer with 3 ReLU neurons. See Figure 4.4 for input data and DDML internal structure. We use the DDML [12] for computing the similarity between the \(i\)th ground-truth and the \(j\)th detection, denoted by \(\Delta_{ij}\).

For any performance metric for the object detection problem, an association between detection and ground-truth is necessary. DDML inherently achieves this by its parallel networks with shared weights applied to its both inputs: the detection and the ground-truth box, followed by a Euclidean distance. To discard the irrelevant data, a min-pooling step is applied for each detection, and in such a way, we have “the best” association score, denoted by \(\Delta^*_i\) in Figure 4.5, for each detection. Then these best scores are fed into the associator, consisting of a single fully connected layer with 6 ReLU neurons. Finally, the softmax is applied to the results. We use batch normalization before the ReLU layer. To sum up, the DDML part learns the similarity function, and the rest finds out the best association.

The strong points of an end-to-end, learning-based association network are as follows:

- **The Association network promises a general architecture for the bipartite graph matching problem.** The network architecture is appropriate to solve any association problem by the same strategy: Generate pairs and plug into the network. Furthermore, for the visual features, the DDML structure can easily be modified to have convolutional layers, as in the original DDML paper [12]. Following this, DDML architecture can be used in video object detection, multi-target tracking, and any matching problem.

- **The Association network is fast.** For the DDML, there is only one matrix multiplication, and in the associator part, only two matrix multiplication occurs. For this reason, the similarity function and the association can be estimated by using only three matrix multiplications.
• **The Association network is memory efficient.** DDML has the input size of 6 and 3 neurons for its single layer and single output neuron computing Euclidean distance. There are 18 weights and three bias terms, in total 21 learnable parameters. Similarly, the associator has 1D input with six neurons and two outputs. In total, the associator has 18 weights and eight biases. In total, there are 47 weights in the network. Thus, it is obviously memory efficient.

### 4.3.4 LRP Estimation Network Architecture

This section introduces a method to make LRP metric differentiable and make it available as a differentiable loss function. The proposed method is constructed on the association strategy presented in the previous section.

Our model is presented in Figure 4.6. Note that there are two association networks since it is not enough to calculate LRP error by estimating detection set only matchings. By considering the Equation (2.29), calculating LRP error requires not only number of false positives but also number of false negatives. In such a case, following Equations (2.31) and (2.32), we see that \( LRP_{FP} \) term is calculated by considering the detection set matchings and \( LRP_{FN} \) term is calculated by considering the ground-truth set matchings. For this reason, in such a setting, FP’s and FN’s can be incorporated by the top and bottom association networks, respectively. These similarity scores are fed into the regressor. If the summation of the number of detections and ground-truths is less than the regressor’s input size, then we pad the remaining portion of the input by zeros. Regressor is a multilayer perceptron with a sequence of three fully connected layers with 128, 64, and 8 neurons, respectively, each with ReLU activation. Overall, this architecture has two possible problems: Firstly, padding will affect the result, and secondly, the similarity scores of the TPs are provided into the network twice. Despite these issues, LRP estimation experiments have promising results.

Since a multilayer perceptron based regressor has a fixed size, we pad the remaining part of the input with 0s when the total number of pairings is less than the specified threshold value. However, padding harms the learning process since the former corresponds to a bad similarity score, while the latter is the best similarity score con-
Figure 4.6: Learning based LRP estimation architecture. There are two separate similarity estimation modules for extracting all components of LRP metric ($N_{TP}, N_{FP}, N_{FN}$). The remaining architecture is the same as the association network, except for this architecture is trained by considering the difference between estimated LRP values and real LRP values.

Considering the Siamese Network structure in the DDML. In order to remove padding, one approach is to replace the multilayer perceptron regressor with an LSTM. In addition to its adaptability of variable length inputs, LSTM may be a good candidate for revealing the recursive structure of LRP error. Assume $\text{LRP}(t) = \frac{\text{Numerator}(t)}{\text{Denominator}(t)+1}$ represents the LRP error between a single predicted bounding box and its corresponding ground-truth box. Two alternatives can update this error, whether updated due to a TP detection or updated due to a FP or FN. If it is a TP detection, LRP error is updated as follows; $\text{LRP}(t + 1) = \frac{\text{Numerator}(t)+\text{IoU}(TP)}{\text{Denominator}(t)+1}$. Else if it is FP or FN it becomes $\text{LRP}(t + 1) = \frac{\text{Numerator}(t)+1}{\text{Denominator}(t)+1}$. Thus, an LSTM can be more convenient for estimating LRP error without requiring padding operation. However, we did not explore LSTM based regressor architecture in this thesis.
4.4 Experiments

4.4.1 Association Experiments

Here we evaluate the association network explained in Section 4.3.3 on its ability to discriminate of TPs and FPs on the Pascal-VOC dataset [25] and the detections are generated by Faster R-CNN [6]. We train the network for eight epochs. Figure 4.7 represents the training and validation errors and the validation accuracy at each epoch. After the third epoch, we observed that overfitting begins. Because of this, we use the network, which is trained for three epochs. The resulting confusion matrix is presented in Table 4.1, which yields an overall accuracy of 96.7%. For an association problem, we believe that this is an acceptable value. Note that we do not propose a detailed parameter tuning methodology.

![Training/Validation loss curves and performance over iterations/epochs for the associator.](image)

Figure 4.7: Training/Validation loss curves (left) and the performance over iterations/epochs for the associator (right). The loss curve shows that the associator converges after about 200th iterations; since it overfits, we used the model that is trained up to this point. Validation accuracy shows that the associator model can estimate TP, FP, FN, and TN predictions with acceptable accuracy.

4.4.2 LRP Estimation Experiments

In this section, we are going to propose the LRP estimation module’s validation experiment results. After that, by plugging this module at the end of an object detector,
Table 4.1: Confusion Matrix of the Associator

<table>
<thead>
<tr>
<th>Ground-truth / Prediction</th>
<th>Assoc.</th>
<th>Not Assoc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assoc.</td>
<td>10579</td>
<td>251</td>
</tr>
<tr>
<td>Not Assoc.</td>
<td>300</td>
<td>5337</td>
</tr>
</tbody>
</table>

we propose object detection experiments with the LRP estimation module.

For the validation of the layers in the DDML, we have used three-layer configurations. The first one uses six linear neurons, the second one uses six ReLU neurons, and the third one uses six ReLU neurons in the first layer and six linear neurons in the second layer. Each configuration of the model is trained with and without pooling for eight epochs.

First, we evaluated the effect of the pooling layer after the DDML module. In Figure 4.8(a), the networks tested with and without pooling layers. We find that, discarding the redundant association scores contributes to the training significantly. This strategy also helps to decrease the number of parameters since without pooling; the regressor is provided with $M \times N$ pairs such that $M$ and $N$ are the detections and ground-truth number, while with pooling it is $M + N$. The best validation error that we have reached on the validation set is 0.043 mean average difference (not mean square error) with the model with 6 ReLU neurons.

Throughout the training, mean squared loss values are tracked and depicted in Figure 4.8(b). Again the model with 6 ReLU neurons has a good training trend. For this reason, we use min pooling and ReLU neurons in the DDML. We have not validated the combination of layers for regressor. However, still we have a satisfactory validation error around 0.04.

### 4.4.3 Object Detection Experiments

**Dataset and Methods.** We evaluate the training performance of Faster R-CNN[6] on PASCAL VOC [25] dataset in object detection experiments. Note that in the original training configuration of Faster R-CNN [6] has four loss components: RPN classifi-
Validation Accuracy of Different LRP Regression Models

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.04</td>
<td>0.12</td>
</tr>
<tr>
<td>0.06</td>
<td>0.14</td>
</tr>
<tr>
<td>0.08</td>
<td>0.16</td>
</tr>
<tr>
<td>0.1</td>
<td>0.18</td>
</tr>
<tr>
<td>0.12</td>
<td>0.20</td>
</tr>
<tr>
<td>0.14</td>
<td>0.22</td>
</tr>
<tr>
<td>0.16</td>
<td>0.24</td>
</tr>
</tbody>
</table>

(a) Effect of the pooling layer: The lines get thicker through 6 linear neurons model, 6 ReLU neurons model and 2 layer model.

(b) Training and Validation Loss: The lines get thicker through 6 linear neurons model, 6 ReLU neurons model and 2 layer model. Loss values are averaged over all iterations in each epoch.

Figure 4.8: (a) Effect of the pooling layer: The lines get thicker through 6 linear neurons model, 6 ReLU neurons model and 2 layer model. (b) Training and Validation Loss: The lines get thicker through 6 linear neurons model, 6 ReLU neurons model and 2 layer model. Loss values are averaged over all iterations in each epoch.

In order to identify the advantages and disadvantages of the employed LRP Loss, we build five models, all of which are further trained starting from a model trained for one epoch with the original training configuration of Faster R-CNN:

- **Original Model**: This is the original training with four loss components, as in Faster R-CNN.
- **RPN Only Model**: This model is trained only by RPN classification loss and RPN regression loss without using detector losses.
- **Fixed Threshold Model**: This model is trained by RPN components and the LRP Loss estimation module. The threshold is set to 0.5 for all classes. So, no threshold update occurs during training, and detections are thresholded with a single value; 0.5.
- **Optimal Threshold Model**: This model is trained by RPN components and the LRP Loss estimation module. Class-specific optimal thresholds are updated during training, and the detections are thresholded with class-specific optimal thresholds.
Multi Threshold Model: This model is trained by RPN components and the LRP Loss estimation module. During training, class-specific optimal thresholds are updated, and the detections are thresholded from three points in the RP curve: class-specific optimal thresholds: \( s, (1 - s)/2 \) and 0.03.

Even though we expected the results to be in between the original model and the RPN only model, replacing detector losses with the LRP estimation module worsened the training process. The trend of the loss function components is depicted in Figure 4.9 in which our models performed the worst. In these figures, one needs to concentrate on the below ones, since the RPN loss always exists during training. Even though it is not included in the loss function during optimal training, we observe that LRP loss decreases consistently. Thus, the loss function is regressed well, and it is linked with the original detector loss components. However, when we replace the detector losses (classification+regression) with the LRP estimation module, the detector losses increase in all three training methods. While the proposed LRP estimation module can regress LRP values with satisfactory results, it fails in training object detectors. The main reason behind this is that none of the data collection methods proposed for training the LRP estimation module can represent all cases required for successful object detector training. We left enhanced data collection methodologies as future work.
Figure 4.9: Visualization of both total and component-wise loss values during the training of Faster R-CNN. As we explained in the experiments section, we trained Faster R-CNN with original loss components (in black), RPN only losses (in red), and LRP estimation module with fixed thresholds (in blue), optimal thresholds (in green), and multiple thresholds (in magenta). In addition to this, we calculate LRP values at each step of the training to analyze each method’s training performance. As expected, the original training of Faster R-CNN converges (consider all loss components and LRP Loss). Considering the LRP Loss plot, we can see that RPN only model can not converge. We expected that our LRP estimation module performs in between the original training and RPN only model; however, it diverges after a few iterations.
In this chapter, we are going to explain our recent method aLRP Loss [4]. The main motivation of this method is to jointly optimize localisation and classification branches in a unified manner. To address this, we used our LRP metric [11], which can evaluate an object detector’s performance by including localisation performance. Further details about LRP is given in Section 2.3.1. aLRP Loss is a joint work, and the contribution of this thesis is the extension of aLRP Loss for robust forms of IoU variants (see Section 5.4).

Due to their definitions, AP and LRP consist of ranking operations. Using them as a loss function for training deep object detectors requires adapting non-differentiable ranking operation into the backpropagation based training process. To do this, AP-Loss [29] uses error-driven update to derive gradients that can be used together with backpropagation. We exploit the error-driven update proposed in AP-Loss [29] to use LRP error as a loss function in the training process.

Intuitively, aLRP Loss enforces an object detector to generate high precision outputs with high localisation quality. As shown in Figure 5.1, any combination of loss functions other than aLRP Loss does not take into account the ranking between the two tasks, and consequently, all have the same loss value for different classifier-regressor combinations ($C&.R_i$). In contrast, aLRP Loss assigns the highest loss value when classification and regression outputs are negatively-correlated, and the lowest loss value for the positively-correlated outputs. We first present the definition of aLRP Loss. After that, we explain aLRP Loss gradients, a generalised version of the gradient formulation proposed in AP-Loss [29]. Following this, we will generalize aLRP Loss for all IoU variants mentioned in Section 2.2.1.2. As a final section, we present
Figure 5.1: **aLRP Loss is aware of the correlation between object detection tasks.**

(a) Classification and three possible localisation outputs for 10 anchors and the rankings of the positive anchors with respect to (wrt) the scores (for \(C\)) and IoUs (for \(R_1\), \(R_2\) and \(R_3\)). Since the regressor is only trained by positive anchors, “–” is assigned for negative anchors. (b) Performance of \(R_1\), \(R_2\) and \(R_3\) when combined with \(C\). When correlation between the rankings of classifier and regressor outputs decreases, performance degrades by 17% AP points. (c) Loss assignment under different loss functions for all classifier&regressor combinations. While, any linear combination of \(L_c\) and \(L_r\) can not distinguish the correlation between two tasks. aLRP Loss directly penalize the output anchors of object detector accordingly. (Figure source: [4])

![Image](image.png)

our results comparing state of the art (SOTA) object detector performances.

### 5.1 Definition of aLRP Loss

AP-Loss is the average precision error calculated over positive detections [29]. By following this, aLRP Loss can be defined as the average LRP error over positive examples:

\[
L^{aLRP} := \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} E^{LRP}(i),
\]

where \(\mathcal{P}\) represents the positive examples in the output set. The design of object detectors comprise dense anchors which implies \(N_{FN}\) component in LRP definition.
Equation (2.29) becomes negligible (check Section 3.1 for formal definition of this assumption.). Also $N_{TP}$ and $N_{FP}$ are calculated over the set of positives $P$ and negatives $N$ respectively. We can rewrite LRP error from Equation (2.29) in a more compact form as follows:

$$\text{LRP}(s) = \frac{1}{N_{FP} + N_{FN} + N_{TP}} \left( N_{FP} + N_{FN} + \sum_{k \in TP} E_{loc}(k) \right),$$  \hspace{1cm} (5.2)

such that, $E_{loc} = \frac{1 - \text{IoU}(i)}{1 - \tau}$ is the normalized per bounding-box localisation error where $\tau$ is the localisation quality threshold for splitting output detection set into positives and negatives. By following the definition of AP-Loss in Section 2.3.2 box-level LRP error among positive examples $E_{LRP}^L(i)$ in Equation (5.1) is defined as:

$$E_{LRP}(i) = \frac{\sum_{k \in P, k \neq i} \frac{1 - \text{IoU}(k)}{1 - \tau} H(x_{ik}) + \frac{1 - \text{IoU}(i)}{1 - \tau} + N_{FP}(i)}{1 + \sum_{k \in P, k \neq i} H(x_{ik}) + \sum_{k \in N} H(x_{ik})}.$$  \hspace{1cm} (5.3)

Then, by embedding the above definition into Equation (5.1), we can define aLRP Loss as:

$$L_{aLRP} := \frac{1}{|P|} \sum_{i \in P} E_{LRP}(i) = \frac{1}{|P|} \sum_{i \in P} \frac{\sum_{k \in P, k \neq i} E_{loc}(k) H(x_{ik}) + E_{loc}(i) + N_{FP}(i)}{\text{rank}(i)},$$  \hspace{1cm} (5.4)

such that $\text{rank}(i) = 1 + \sum_{k \in P, k \neq i} H(x_{ik}) + \sum_{k \in N} H(x_{ik})$ represents the rank of $i$th example among all outputs.

### 5.2 Deriving Gradients of aLRP Loss

In this section, we derive the gradients of aLRP Loss for integrating error-driven update into the backpropagation based training of deep object detectors. In order to give more insight and make gradient derivation of aLRP Loss easier, we split its definition into classification ($L_{cls}$) and regression ($L_{loc}$) components satisfying that; $L_{aLRP} = L_{cls}^{aLRP} + L_{loc}^{aLRP}$ as follows:

$$L_{cls}^{aLRP} = \frac{1}{|P|} \sum_{i \in P} \frac{N_{FP}(i)}{\text{rank}(i)},$$  \hspace{1cm} (5.5)

$$L_{loc}^{aLRP} = \frac{1}{|P|} \sum_{i \in P} \frac{1}{\text{rank}(i)} \left( E_{loc}(i) + \sum_{k \in P, k \neq i} E_{loc}(k) H(x_{ik}) \right).$$  \hspace{1cm} (5.6)
This component-wise definition of aLRP Loss shows that localisation component of aLRP Loss ($L_{aLRP}^{\text{loc}}$) is differentiable with respect to the predicted box parameters $B$ because of IoU based localisation error term ($E_{\text{loc}}$) \cite{28,30,31}. However, neither of two components are differentiable with respect to classification scores because of the rank($i$) term. For the derivatives of $L_{aLRP}^{\text{cls}}$ and $L_{aLRP}^{\text{loc}}$ with respect to classification scores, we are going to follow the two-step approach proposed in Section 2.3.3.

First, by following the Equation (2.46) the local error of a positive example $i$ is simply LRP Error, hence we set ($E_R(i) = \text{LRP}(i)$). Using a uniform distribution for $p(\cdot|\cdot)$ leads to the following “primary terms” for aLRP:

$$
\Psi_{aLRP}^{ij} = E_{aLRP}(i) \times p_{aLRP}(i,j) = \text{LRP}(i) \times \frac{H(x_{ij})}{N_{FP}(i)}
$$

For defining gradients of aLRP Loss by following the Equation (2.48), next we derive target primary term ($\Psi^{R+}$). Perfect ranking of $i$th positive example leads to $N_{FP}(i) = 0$ and $\sum_{k \in P, k \neq i} E_{\text{loc}}(k) H(x_{ik}) = 0$. Following this target primary term of aLRP can be defined as follows:

$$
\Psi_{aLRP}^{ij+} = \frac{E_{\text{loc}}(i) H(x_{ij})}{\text{rank}(i) N_{FP}(i)},
$$

which shows that even if $i$th positive example is ranked accordingly, there is still a localisation error. As a result, target primary term can be non-zero if $s_i < s_j$.

Having determined $\Psi_{aLRP}^{ij+}$, $\Delta x_{ij}$ becomes:

$$
\Delta x_{ij} = \Psi_{aLRP}^{ij+} - \Psi_{aLRP}^{ij} = \frac{1}{\text{rank}(i)} \left( N_{FP}(i) + \sum_{k \in P, k \neq i} E_{\text{loc}}(k) H(x_{ik}) \right) \frac{H(x_{ij})}{N_{FP}(i)}.
$$

Finally, $\Delta x_{ij}$ can be embedded into Equation (2.44) to obtain gradient derivation of aLRP Loss.

**Comparison of aLRP Loss with AP-Loss.** Although AP-Loss and aLRP Loss follow error-driven update for integrating non-differentiable rank based functions into
backpropagation based training of deep object detectors, aLRP Loss takes into account the correlation between two tasks due to its gradients are calculated by considering the outputs of both branches. We showed this property with a toy example in Figure 5.2 by investigating its effect on branches. In Figure 5.2 (a), gradient of \( p_2 \) is increased by considering the localisation error of \( p_1 \) to suppress \( p_1 \) because it has a larger localisation error than \( p_2 \). This leads to examples with better localisation quality to be ranked higher, among other positive examples. Furthermore, aLRP Loss can demote the ranks of negative examples by uniformly distributing gradients of positive examples with lower classification scores than the corresponding negative examples.

In the last case, the localisation component of aLRP Loss for positive examples is examined. The localisation error of the lowest-ranked example \( p_3 \) (i.e. \( L_{loc}(p_3) \)) includes all of the localisation errors from other positive examples (i.e. each \( E_{loc}(p_i) \)), while \( E_{loc}(p_1) \) is calculated with only its \( L_{loc}(p_1) \). Because of this, \( E_{loc}(p_1) \), the localisation error of the positive with the largest score, contributes to all \( L_{loc}(i) \)s and aLRP Loss enforces localisation branch to put more focus on high-ranked examples.

### 5.3 Self Balance Mechanism for Localisation Component

Since LRP metric yields localisation error only for examples with high rank values (Section 2.3.1), aLRP Loss cause domination of the loss in favor of classification branch loss especially at early stages of training. This leads localisation branch to be trained poorly, as shown in Figure 5.3. To tackle this problem, we propose a Self-Balance Mechanism such that; \( \partial \mathcal{L} / \partial B \) is scaled with \( L^{a\text{LRP}} / L^{a\text{LRP}}_{loc} \) value that is updated over each training epoch.

### 5.4 Extending aLRP Loss for Different IoU Variants

By definition, the LRP metric penalizes each positive example with its localisation error, as mentioned in Section 2.3.1. While the definition of LRP includes localisation error measurement under IoU, recent works propose different forms of IoU-based losses [30][31]. The common motivation of IoU variants is to deal with IoU measure
Figure 5.2: **aLRP Loss assigns gradients to each branch based on the outputs of both branches.** Examples on the PR curve are in sorted order wrt scores ($s$). $L$ refers to $L_{aLRP}$. (a) A $p_i$’s gradient wrt its score considers (i) localisation errors of examples with larger $s$ (e.g. high $\mathcal{E}_{loc}(p_1)$ increases the gradient of $s_{p_2}$ to suppress $p_1$), (ii) number of negatives with larger $s$. (b) Gradients wrt $s$ of the negatives: The gradient of a $p_i$ is uniformly distributed over the negatives with larger $s$. Summed contributions from all positives determines the gradient of a negative. (c) Gradients of the box parameters: While $p_1$ (with highest $s$) is included in total localisation error on each positive, i.e. $L_{aLRP}^{loc}(i) = \frac{1}{\text{rank}(i)}(\mathcal{E}_{loc}(i) + \sum_{k \in P, k \neq i} \mathcal{E}_{loc}(k)H(x_{ik}))$, $p_3$ is included once with the largest rank($p_i$). (Figure source: [4])

deficiencies, especially for distant boxes. We discussed details of IoU variants in Section 2.2.1.2 In this section, we propose a formulation for generalizing aLRP Loss with different IoU variants.

The main issue about other IoU variants, GIoU and DIoU, they have different numerical ranges different from IoU. By following the definition of LRP localisation component in Equation (2.30), IoU error between a predicted box $y_i$ and its corresponding ground-truth box is normalized by the term $1 - \tau$ where, $\tau$ is the localisation quality threshold for splitting output detection set into positives and negatives. By considering the ranges of IoU variants discussed in Section 2.2.1.2 we modify LRP$_{loc}$ component as follows:

$$LRP_{loc} := \frac{1}{N_{TP}} \sum_{i=1}^{N_{TP}} \left( \frac{L_{IoU}(x_i, y_i)}{1 - \tau} \right),$$

such that $L_{IoU}$ should be an IoU based loss function with range $[0, 1]$. In order to
ensure this constraint, we incorporate an IoU-based loss function, $L_{\text{IoU}}^\prime$, with a range $[A, B]$ with the following formulation:

$$L_{\text{IoU}} = \frac{L_{\text{IoU}}^\prime - A}{B - A}. \quad (5.12)$$

Note that Equation (5.12) ensures $L_{\text{IoU}}$ to have a range of $[0,1]$. While, for the conventional IoU Loss (i.e. $L_{\text{IoU}}^\prime$ is the IoU Loss), Equation (5.12) reduces to $L_{\text{IoU}} = L_{\text{IoU}}^\prime$, for GIoU Loss and DIoU Loss, $L_{\text{IoU}} = L_{\text{IoU}}^\prime/2$ since these loss functions both have a range of $[0,2]$. Finally, for CIoU, $L_{\text{IoU}} = L_{\text{IoU}}^\prime/3$ since CIoU has a range of $[0,3]$ different than other IoU variants GIoU and DIoU. Hence, any IoU based bounded loss function can be incorporated into aLRP Loss by exploiting Equation (5.12).

### 5.5 Experiments

**Dataset.** In our experiments, we used COCO [2] trainval35K set for training. In our ablation study, we used minival set for testing, and in SOTA comparison we used test-dev set.

**Performance Measures.** In ablation analysis, we used both AP [2] and oLRP [11] performance measures. In addition to these, we measure the correlation between the
rankings of localisation and classification tasks. We use the Pearson correlation coefficient, which is represented by $\rho$.

**Implementation Details.** We used the official implementation of AP-Loss \cite{29} implementation as the base model for the implementation of aLRP Loss. AP-Loss use RetinaNet \cite{8} as the baseline model. We use 4 GPUs (Nvidia V100 or RTX2080Ti) for training. Batch size for aLRPLoss500 (with $512 \times 512$ images) is set to 32, and aLRPLoss800 (with 800 images) is trained with batch size of 16. Adopting the same setting in AP-Loss unless otherwise stated, we trained our models for 100 epoch with stochastic gradient descent with momentum factor of 0.9. Learning rate is set to 0.008 for aLRPLoss500 models and 0.004 for aLRPLoss800 models. We schedule the learning rate by scaling with the factor of 0.1 at epochs 60 and 80. Similar to AP-Loss, we use SSD style data augmentation \cite{7} for training. For ablation study we used aLRPLoss500 model with ResNet-50 backbone \cite{37}. We rescaled input images in test time such that the shorter side to be 500 and 800 for aLRPLoss500 and aLRPLoss800, respectively. Longer sides of images are set by ensuring it does not exceed $1.66 \times$ of the shorter side. We apply NMS to the output set to obtain 1000 top-scoring detections, and we set the IoU threshold to 0.50.

For the ablation study, we analyzed the contributions of using ranking for localisation branch, the effect of self-balancing mechanism, and using robust IoU variants.

**Effect of using ranking for localisation.** As shown in Table 5.1 using ranking based loss function for localisation branch improves AP significantly (from 35.5 to 36.9). Although $AP_{50}$ values of AP-Loss and aLRP Loss are similar, we observe that this affects the performance of high-localisation quality outputs (e.g., $AP_{90}$). This observation validates that aLRP Loss boosts the performance by prioritizing high-quality outputs in both branches.

**Effect of Self-Balancing (SB).** We have discussed the self-balancing mechanism in Section 5.3. By adaptively scaling the localisation component of aLRP Loss, we achieve 1.8 AP gain, as shown in Table 5.1.

**Using IoU Variants.** Using GIoU \cite{30} improves the performance of aLRP Loss
slightly as shown in Table 5.1. However, DIoU [31] can only produce on-par performance with aLRP Loss with IoU. Using CIoU [31] improves the performance of the model with DIoU slightly.

Table 5.1: Ablation analysis on COCO minival. For optimal LRP (oLRP), lower is better.

<table>
<thead>
<tr>
<th>Method</th>
<th>Rank-Based $\mathcal{L}_r$</th>
<th>Rank-Based $\mathcal{L}_f$</th>
<th>SB</th>
<th>AP 50</th>
<th>AP 75</th>
<th>AP 90</th>
<th>oLRP</th>
<th>$\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP-Loss [29]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>35.5</td>
<td>58.0</td>
<td>37.0</td>
<td>9.0</td>
<td>71.0</td>
</tr>
<tr>
<td>aLRP Loss</td>
<td>✓</td>
<td>✓ (w IoU)</td>
<td></td>
<td>36.9</td>
<td>57.7</td>
<td>38.4</td>
<td>13.9</td>
<td>69.9</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓ (w IoU)</td>
<td>✓</td>
<td>38.7</td>
<td>58.1</td>
<td>40.6</td>
<td>17.4</td>
<td>68.5</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓ (w GIoU)</td>
<td>✓</td>
<td>38.9</td>
<td>58.5</td>
<td>40.4</td>
<td>17.4</td>
<td>68.4</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓ (w DIoU)</td>
<td>✓</td>
<td>38.7</td>
<td>58.4</td>
<td>40.0</td>
<td>17.2</td>
<td>68.4</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓ (w CIoU)</td>
<td>✓</td>
<td>38.8</td>
<td>58.6</td>
<td>40.3</td>
<td>17.0</td>
<td>68.3</td>
</tr>
</tbody>
</table>

Comparison with SOTA. To compare aLRP Loss with SOTA object detectors, different than ablation study experiments aLRPLoss500 is trained for 160 epoch with learning rate decay at epochs 90, 110, and 140 epochs by following the training strategy in previous works [42, 43, 44]. Comparison of aLRP Loss with SOTA detectors is shown in Table 5.2.

Comparing with other ranking based loss functions (DR loss [35] and AP Loss [29]), aLRP Loss can achieve significant improvements. aLRP Loss provides 4.3 AP gain in scale 500, 3.3 AP gain in scale 800 over AP Loss. Similarly, if we compare DR loss and aLRP Loss in scale 800, aLRP Loss provides 2.8 AP improvement.

aLRP Loss performs better than other methods combining branches. Note that two methods that combine branches other than Retina+PISA [33] have different backbone networks than aLRP Loss. Because of that, a direct comparison with them is not fair. However, aLRP Loss performs better than Retina+PISA with more than 5.0 AP in scale 800.

Finally we compare aLRP Loss with one-stage methods in two training input image scales: 500 and 800. In scale 500, aLRP Loss with R-101 backbone outperforms all one-stage methods with the same backbone. Note that, although it uses a stronger backbone network (X-101) there is only a small gap 0.2 AP between HSD with X-
101 and aLRP Loss with R-101 backbone networks in scale 500. In scale 800, aLRP Loss performs better than its closest counterparts (FreeAnchor and CenterNet) with 1.0 AP with both R-101 and X-101 backbone networks.
Table 5.2: Comparison with the SOTA detectors on COCO test-dev. $S \times 1.66$ implies that the image is rescaled such that its longer side cannot exceed $1.66 \times S$ where $S$ is the size of the shorter side. R:ResNet, X:ResNeXt, H:HourglassNet, D:DarkNet, De:DeNet.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>Training Size</th>
<th>Test Size</th>
<th>AP</th>
<th>AP$_{50}$</th>
<th>AP$_{75}$</th>
<th>AP$_{S}$</th>
<th>AP$_{M}$</th>
<th>AP$_{L}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>One-Stage Methods</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RefineDet [45]†</td>
<td>R-101</td>
<td>512 $\times$ 512</td>
<td>512 $\times$ 512</td>
<td>36.4</td>
<td>57.5</td>
<td>39.5</td>
<td>16.6</td>
<td>39.9</td>
<td>51.4</td>
</tr>
<tr>
<td>EFGRNet [44]†</td>
<td>R-101</td>
<td>512 $\times$ 512</td>
<td>512 $\times$ 512</td>
<td>39.0</td>
<td>58.8</td>
<td>42.3</td>
<td>17.8</td>
<td>43.6</td>
<td>54.5</td>
</tr>
<tr>
<td>HSD [42]†</td>
<td>R-101</td>
<td>512 $\times$ 512</td>
<td>512 $\times$ 512</td>
<td>40.2</td>
<td>59.4</td>
<td>44.0</td>
<td>20.0</td>
<td>44.4</td>
<td>54.9</td>
</tr>
<tr>
<td>ExtremeNet [46]†</td>
<td>H-104</td>
<td>511 $\times$ 511</td>
<td>original</td>
<td>40.2</td>
<td>55.5</td>
<td>43.2</td>
<td>20.4</td>
<td>43.2</td>
<td>53.1</td>
</tr>
<tr>
<td>RetinaNet800 [8]</td>
<td>X-101</td>
<td>800, $\times$1.66</td>
<td>800, $\times$1.66</td>
<td>40.8</td>
<td>61.1</td>
<td>44.1</td>
<td>24.1</td>
<td>42.3</td>
<td>51.2</td>
</tr>
<tr>
<td>FCOS [47]†</td>
<td>R-101</td>
<td>(640, 800), $\times$1.66</td>
<td>800, $\times$1.66</td>
<td>41.5</td>
<td>60.7</td>
<td>45.0</td>
<td>24.4</td>
<td>44.8</td>
<td>51.6</td>
</tr>
<tr>
<td>HSD [42]‡</td>
<td>X-101</td>
<td>512 $\times$ 512</td>
<td>512 $\times$ 512</td>
<td>41.9</td>
<td>61.1</td>
<td>46.2</td>
<td>21.8</td>
<td>46.6</td>
<td>57.0</td>
</tr>
<tr>
<td>FreeAnchor [48]†</td>
<td>R-101</td>
<td>(640, 800), $\times$1.66</td>
<td>800, $\times$1.66</td>
<td>43.1</td>
<td>62.2</td>
<td>46.4</td>
<td>24.5</td>
<td>46.1</td>
<td>54.8</td>
</tr>
<tr>
<td>FCOS [47]†</td>
<td>X-101</td>
<td>(640, 800), $\times$1.66</td>
<td>800, $\times$1.66</td>
<td>44.7</td>
<td>64.1</td>
<td>48.4</td>
<td>27.6</td>
<td>47.5</td>
<td>55.6</td>
</tr>
<tr>
<td>CenterNet [43]*†</td>
<td>H-104</td>
<td>511 $\times$ 511</td>
<td>original</td>
<td>44.9</td>
<td>62.4</td>
<td>48.1</td>
<td>25.6</td>
<td>47.4</td>
<td>57.4</td>
</tr>
<tr>
<td>FreeAnchor [48]†</td>
<td>X-101</td>
<td>(640, 800), $\times$1.66</td>
<td>800, $\times$1.66</td>
<td>44.9</td>
<td>64.3</td>
<td>48.5</td>
<td>26.8</td>
<td>48.3</td>
<td>55.9</td>
</tr>
</tbody>
</table>

| Ranking Losses |               |               |           |    |           |           |          |          |          |
| AP Loss500 [29]† | R-101    | 512 $\times$ 512 | 500, $\times$1.66 | 37.4 | 58.6      | 40.5      | 17.3     | 40.8     | 51.9     |
| AP Loss800 [29]† | R-101    | 800 $\times$ 800 | 800, $\times$1.66 | 40.8 | 63.7      | 43.7      | 25.4     | 43.9     | 50.6     |
| DR Loss [35]‡   | R-101    | (640, 800), $\times$1.66 | 800, $\times$1.66 | 41.7 | 60.9      | 44.8      | 23.5     | 44.9     | 53.1     |
| DR Loss [35]‡   | X-101    | (640, 800), $\times$1.66 | 800, $\times$1.66 | 43.1 | 62.8      | 46.4      | 25.6     | 46.2     | 54.0     |

| Combining Branches |               |               |           |    |           |           |          |          |          |
| LapNet [34]      | D-53      | 512 $\times$ 512 | 512 $\times$ 512 | 37.6 | 55.5      | 40.4      | 17.6     | 40.5     | 49.9     |
| Fitness NMS [49] | De-101    | 512, $\times$1.66 | 768, $\times$1.66 | 39.5 | 58.0      | 42.6      | 18.9     | 43.5     | 54.1     |
| Retina+PISA [53] | R-101    | 800, $\times$1.66 | 800, $\times$1.66 | 40.8 | 60.5      | 44.2      | 23.0     | 44.2     | 51.4     |

| Ours                          |               |               |           |    |           |           |          |          |          |
| aLRP Loss500‡                 | R-101    | 512 $\times$ 512 | 500, $\times$1.66 | 41.7 | 61.5      | 44.1      | 21.2     | 45.2     | 56.8     |
| aLRP Loss500‡                 | X-101    | 512 $\times$ 512 | 500, $\times$1.66 | 43.4 | 63.4      | 46.0      | 23.0     | 47.3     | 58.2     |
| aLRP Loss800‡                 | R-101    | 800 $\times$ 800 | 800, $\times$1.66 | 44.1 | 64.8      | 47.0      | 26.4     | 47.3     | 56.2     |
| aLRP Loss800‡                 | X-101    | 800 $\times$ 800 | 800, $\times$1.66 | 45.9 | 66.8      | 48.8      | 28.8     | 49.3     | 57.4     |

1: multiscale training, †: SSD-like augmentation, *: Soft NMS [50] and flip augmentation at test time.
CHAPTER 6

CONCLUSIONS

In this study, we analyzed deficiencies of the conventional training strategy of deep object detection methods. As a starting point, the most commonly used evaluation strategy method, average precision (AP), is studied, and by following its deficiencies, a recent evaluation method localisation-recall-precision (LRP) is investigated in detail. These two evaluation strategies show that object detectors’ training and their evaluation strategies do not match (i.e., the correlation between two tasks is not considered in conventional training strategy).

Following the LRP metric definition, we propose a bounded classification loss function that aims to balance the contribution of two sub-tasks in object detection (classification and localisation). The broad definition of the proposed loss function implicitly optimizes the LRP metric. The results of this method show that object detectors should be trained not only by considering the numerical balance of two sub-tasks but also by the association between two sub-tasks. We show that while this method can mimic the evaluation methodology defined in LRP as a training loss function, it requires an extensive amount of hyper-parameter search.

Evaluation methods for object detection problem require ranking operation. Due to the non-differentiability of ranking based functions, it is not easy to directly use them in the training stage as a loss function. Due to this, we proposed an end-to-end learning-based estimation architecture for LRP. As a by-product, we propose a neural network-based association network that targets the assignment between predictions and ground-truths, included by LRP. This architecture helps us to validate our method’s estimation capability. While this method reaches satisfactory performance in both association problem and LRP estimation, it fails to train object detector be-
cause of the lack of representation capability of the collected dataset.

Our analyses on the relation of two-subtasks in object detection show that two tasks should be considered as associated tasks. Furthermore, these two-subtasks promote each other’s performance, showing that they should be considered positively correlated tasks. To address this, again following the LRP performance metric, we reformulate the LRP as a loss function by the aid of the error-driven update used in the perceptron learning algorithm proposed in AP-Loss. To do this, we exploit the property that error driven update can be used together with gradient descent algorithms. In addition to this, we generalise the localisation component of LRP as a loss function to be useful with all localisation quality measures. This approach outperformed SOTA object detectors.

All in all, the multi-task learning based training strategy in object detection is questioned, and a better solution for training object detectors from the perspective of loss functions is proposed, which considers the relationship between the sub-tasks of object detection.
REFERENCES


