UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN
"TECNOLOGIE DELL'INFORMAZIONE"

CICLO XXXIV

Object Detection and Instance Segmentation with Deep Learning Techniques

Coordinatore:
Chiar.mo Prof. Marco Locatelli

Tutore:
Chiar.mo Prof. Andrea Prati

Dottorando: Leonardo Rossi

Anni 2018/2021

Try not to become a person of success, but rather try to become a person of value.
Albert Einstein

Thanks to my parents, my friends, all the people who are part of me and who have gone through my life. They gave me love and time, the most precious things that exist.

"Semo forti!" cit.

Contents

List of Figures

List of Tables

Introduction

One day I will find the right words,
and they will be simple.

Jack Kerouac

Digital cameras are increasingly integrated into any modern system, whether they are surveillance cameras, cell phone cameras or artificial satellites. Visual information is present in most industrial environments as a powerful input for machinery control and preventive diagnosis. Nowadays, object detection and instance segmentation are two of the most studied topics in the computer vision community, because they reflect one of the key problems for many of the existing applications, where we have to deal with many heterogeneous objects inside an image. These tasks are fundamental for several applications, including medical diagnostics [8, 9], autonomous driving [10], alarm systems [11], agriculture optimization [12], visual product search [13], and many others. This because from a simple image we can discover e.g. the presence of a tumor, or the presence of traces of harmful insects for the crop, or the spread of a particular clothing among the users of the social network. Recognizing the objects in a image allows on one hand to be able to exclude everything that is not interesting for the task to be performed, and on the other hand it allows to build more complex reasoning and conceptual maps like for example in top part of Figure 1. Each of these applications introduce many open challenges for state-of-the-art algorithms. In order to work well, they need to take into account environments with an increasing variance of object shape, size, illumination, occlusion and so on.

For instance, the car system have to be aware that it could face heavy traffic situations, with many objects moving in any direction and partially occluded. It should work independently from space (e.g. in any part of the world) and time (e.g. in any weather condition). The car system not only must perform well with objects whose design is constantly evolving over the years, but also, as in bottom part of Figure 1, takes into account very particular situations where special means of transport, such as elephants, are used.

Figure 1: Top: Scene Graph Generation (SGG) task example from Visual Genome Dataset [3]. Down: Driving environment example from COCO Dataset [4].

Usually, the tasks of object detection and instance segmentation are strictly connected, offering in output the classification, localization and segmentation of a number of instances not defined a priori, each of them belonging to an object class in a predefined list. In Figure 2 some examples are shown taken from heterogeneous environments in common scenarios. Classical computer vision algorithms were still

comfortable tools that satisfied the required performances until a few years ago, because industrial environments are strictly controlled and bounded. Nowadays, the applications could involve more heterogeneous environments, with rather variable conditions. In this situation, advanced deep learning techniques have taken over any other approach, surpassing the state of the art basically in every field. Since our goal from the beginning has been the application of image processing to social network images, the search for innovative techniques related to this field of deep learning was evident from the beginning.

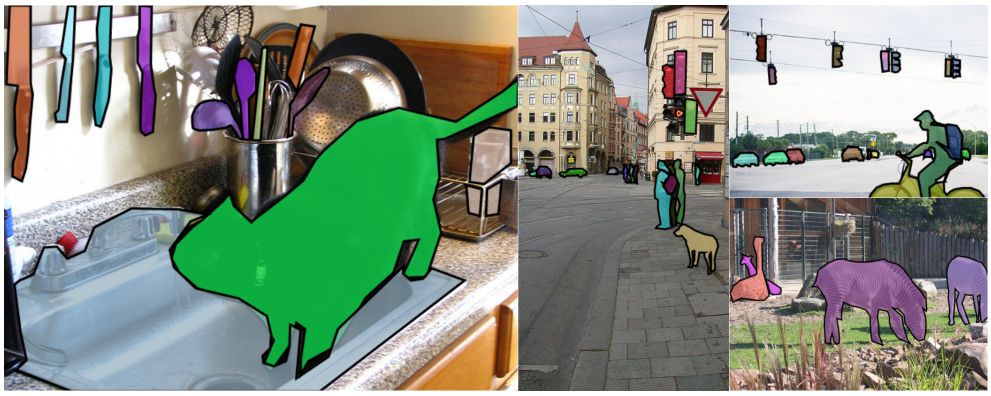


Figure 2: Object Detection and Instance Segmentation tasks examples from COCO Dataset [4].

Most of the recent deep learning models descend from the two-stage architecture called Mask R-CNN [14]. In this type of architecture, the first stage is devoted to the search of interesting regions independently from the class, while the second is used to perform classification, localization and segmentation on each of them. This divide-and-conquer approach was first introduced in the ancestor network called Region-based CNN (R-CNN) [15], which has evolved in several successive architectures. Although it achieved excellent results, several studies [16], [17], [18] have recently discovered some of its critical issues which can limit its potentiality. Most of these issues have not been solved yet and several blocks of these architectures are still under-explored and far from being optimized or well understood.

This thesis tackles some important aspects of these two tasks, Object Detection and Instance Segmentation, in multiple settings: Supervised Learning, Self-Supervised and Semi-Supervised Learning. We will go in details and tackle multiple intrinsic imbalance problems of current models, defining new tasks and new architectures to improve the general performance.

To summarize, this thesis has the following main contributions:

- A novel Region of Interest (RoI) extraction layer called GRoIE is proposed, with the aim of a more generic, configurable and interchangeable framework for RoI extraction to tackle the Feature Level Imbalance (FLI) problem.

- The redesign of the model heads (FCC) toward a fully convolutional approach, with an empirical analysis that supports some architectural preferences depending on the task.

- An extensive analysis of the IoU Distribution Imbalance (IDI) problem in the RPN generated proposals, which we treat with a single- and double-head loop architecture (R^3-CNN) between the detection head and the RoI extractor, and a brand-new internal loop for the segmentation head itself.

- The proposal of SBR-CNN, a new architecture composed by R^3-CNN, FCC and GRoIE, which maintains its qualities if plugged into major state-of-the-art models.

- The definition of two new datasets called Leaf Diseases Dataset (LDD) and ADIDAS Social Network Dataset (ASND).

- The definition of a new auxiliary task C^2SSL with the purpose of enhancing the instance segmentation training on vines diseases detection and segmentation.

- Multiple improvements on the Teacher-Student model on Semi-Supervised Learning setting for the Object Detection task (IL-net).

Resolution of Feature Level Imbalance with GRoIE

A particular attention has been recently given to Instance Segmentation, by exploiting the results achievable by two-stage networks (such as Mask R-CNN), derived from R-CNN. In these complex architectures, a crucial role is played by the Region of Interest (RoI) extraction layer, devoted to extracting a coherent subset of features from a single Feature Pyramid Network (FPN) layer attached on top of a backbone. Our research is motivated by the need to overcome the limitations of existing RoI extractors which select only one (the best) layer from FPN. Our intuition is that all the layers of FPN retain useful information. Furthermore, our GRoIE addresses the problem called Feature Level Imbalance (FLI) [16]. The hierarchical structure of FPN (originally designed to provide multi-scale features) does not provide a good integration between low- and high-level features among different layers. To address this problem, the classical approach is to balance the information before the FPN. On the contrary, our novel architecture puts forward a more effective solution, fusing information from all the FPN layers in a smart way.

Task-Head optimal design with FCC

In addition, we address the common problem of the explosion of the number of parameters, due to the introduction of new components or expansion of existing ones (e.g. [19]). The increased complexity leads to an increase in the search space for the optimization during the training, and, in turn, negatively impacts the generalization capability of the network. The research for a new and lighter head architecture gave an unexpected confirmation in support of an intuition made by [20] about the link between the task to learn and the latest layers. We extend their work toward a fully convolutional solution.

Resolution of IoU Distribution Imbalance with R^3-CNN

The problem, called IoU (Intersection over Union) Distribution Imbalance (IDI), arises when the positive Regions of Interest (RoIs) proposals provided by the RPN during the training of the detection and segmentation heads have an imbalanced dis-

tribution. Due to some intrinsic problems of the anchor system, the number of available RoIs decreases exponentially with the increase of the IoU threshold, which leads the network to easily overfit to low quality proposals. In this work, we first deeply investigate the origin of the IDI problem and, consequently, we propose a new way to balance positive samples by exploiting the re-sampling technique introduced by the cascade models [21].

Our proposed technique generates new proposals with a pre-selected IoU quality in order to equally cover all IoU values. Our new R^3-CNN [2] architecture and the corresponding training mechanism that we propose present a trade-off between the conflicting goals of making a more balanced training and maintaining the number of parameters as low as possible. We show that our model is able to obtain the same performance of complex networks such as HTC [22] with a network as light as Mask R-CNN [14].

SBR-CNN: a new Instance Segmentation model

To demonstrate that the previously defined network improvement could work together, we describe the experimental results obtained by our brand new architecture, called SBR-CNN, which merges the best configuration of R^3-CNN, GRoIE and FCC together.

New datasets: LDD and ASND

To proper meet well-defined prerequisites and also test the performance of our designed networks in different conditions, we defined two new datasets.

The first dataset, called Leaf Diseases Dataset (LDD), contains images of leaves and bunches of grapes, affected by one or more diseases. The purpose behind the creation of the dataset is the implementation of an automatic system for the analysis of vine diseases by an unexperienced user. It contains both annotations for object detection and instance segmentation in a COCO-like format, used in a Supervised Learning setting.

The second dataset, called ADIDAS Social Network Dataset (ASND), contains

images coming from Instagram social network. The purpose of this dataset is to train a network which is able to analyze images and find the required types of clothes in social network images. Because most of them come from the customers themselves, the images can have potentially any visual characteristic and could contain an undefined number of instances. The dataset contains annotations for object detection in a COCO-like format, used in multiple Semi-Supervised Learning settings, in addition the most common Supervised Learning.

Auxiliary self-learning task C^2SSL on leaf diseases segmentation

The performance of our previously introduced network R^3-CNN are evaluated by training it on our new LDD dataset, which contains diseased grape leaves and bunches. The unique feature of this dataset is that the disease instances are contained in total within instances of the leaf and bunches. Since the dataset contains both information about position of the objects and their segmentation, this allowed us to have very precise value of instances overlap. This leads us to introduce a new auxiliary self-learning task which, by adding it to the existing multi-task learning environment, pushes the network to distinguish between healthy and sick leaves or grapes and, on the other hand, to learn to detect those anomalous cases in which a disease is recognized outside a plant. These twofold aspect offers a way to increase both object detection and instance segmentation performance, with the advantage to keep unchanged the computationally cost of the evaluation phase.

Improving Localization with IL-net in a SSOD setting

Supervised learning usually requires a large amount of annotated training data which can be expensive and time-consuming to produce. Semi-Supervised Learning (SSL), on the other hand, addresses this issue by taking advantage of large unlabeled data accompanied by a small labeled dataset. We have two distinct datasets on which to train the network: for the first, we have the ground truth and, for the second, we have no additional information. In a similar context, the Semi-Supervised Object Detection (SSOD) is a SSL but with the specific task of Object Detection. We started from the

successful ideas applied to SSOD and collected in [7].

From our analysis it turned out that, since we have a Teacher who creates pseudo-labels for the Student on the dataset without ground-truth, we must keep in mind that about 50% of them are wrong. This means that we could jeopardize the Student training. Our priority is to preserve the latter as much as possible from the errors introduced by the former. In this Teacher-Student setting, the quality of these predictions is difficult to control by applying only a simple confidence thresholding. For this reason, there is a need for an additional way to strengthen the region proposals and reduce the number of erroneous predictions by the Teacher. In the proposed architecture, we introduce a new task used for the classification of the bounding boxes (bboxes), with the aim of distinguishing good quality ones from the others. This new score exploits the information complementary to the class score already used in these networks, allowing a different level of filtering.

In addition, we show how to take advantage of the regression tasks on the unsupervised learning part. Usually, they are excluded in the unsupervised training phase. The justification is that the classification score is not able to filter the potentially incorrect bboxes [7, 23]. In our hypothesis, the major problem is represented by how the loss contributions balance between them, a well-known problem in literature and usually called *Objective Imbalance* [16]. In order to obtain a positive effect from regression losses, an adequate balance of the contribution of these two losses (regression and category classification) is a possible solution to the abovementioned problem. In this way, we prevent the regression losses on unsupervised dataset from dominating the training, a phenomenon that greatly amplifies the error introduced by inaccurate Teacher predictions.

Thesis Structure

This thesis is organized in the following way:

- Part I will describe in details the state-of-the-art for the related topics of the dissertation.
- Part II will present the research findings. Firstly, in Section 2.1 we propose the

GRoIE architecture. In Section 2.2 the R^3-CNN architecture is defined. Then, in Section 2.3 the SBR-CNN architecture is proposed which contains multiple contributions: the review of the GRoIE architecture and the proposal of a new more performing one, the FCC heads which confirms and extends empirical results to define the rules to follow for an optimal architecture, towards a fully-convolutional approach, and the extension of the R^3-CNN model. In Section 2.4 the new IL-net architecture for Semi-Supervised Learning is shown. In Section 2.5 and 2.6 the datasets LDD and ASND are defined, respectively. Finally, in Section 2.7 the new auxiliary self-learning task C^2SSL is described.

- Part III will present conclusions and future works.

Chapter 1

Background

The Spartans do not ask how many the enemies are but where they are.

Plutarch

In this chapter we will focus on most important Object Detection and Instance Segmentation models, defining building blocks in details and exploring the most important learning settings.

1.1 Components

Backbone

The first building block of a Faster R-CNN, and all models that share the same structure, usually is a Convolutional Neural Network (CNN), previously trained to classify objects, without the classification head. A typical example is a ResNet [24], which has the particularity to introduce residual blocks at multiple levels.

The ResNet reformulates internally the layers as shown in Figure 1.1. Compared with previously defined CNN, it has the advantage to easier optimize the weights and gain accuracy with the increasing of the depth.

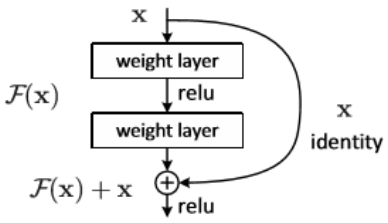


Figure 1.1: ResNet residual learning building block.

Region Proposal Network

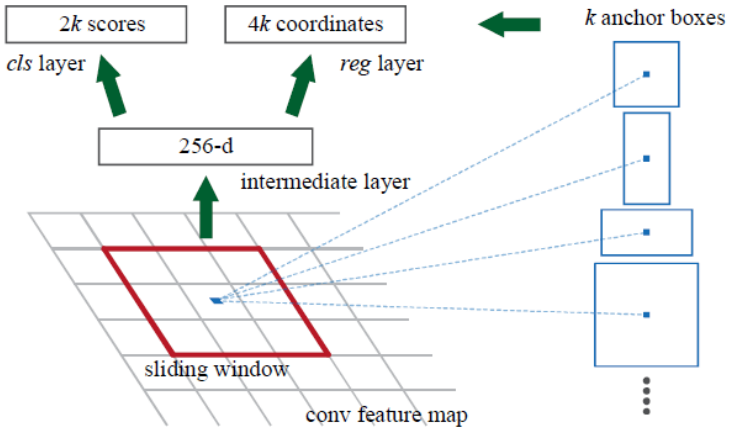


Figure 1.2: Region Proposal Network.

The Region Proposal Network (RPN) is the top head of a Faster R-CNN [25] for the first stage. It splits the image in input in regions with multiple shapes, where the center is called *anchor*. Authors chose three scales and three aspect-ratio for these regions, defining nine shapes. It is composed by a classification and a regression branches. The first to distinguish a generic object from the background, and the second to regress the coordinate of a bounding box respect to the fixed anchors. The classifier output is built as a 3D tensor, where the first two dimensions identify the coordinates of the Anchor respect to the image in input and the third dimension define

the region shape. To each output of this tensor p_i is assigned a label if the Intersection over Union (IoU) between the relative anchor and the ground truth bounding box is the highest. The regression output is built also as a 3D tensor, where the first two dimensions have the same meaning of the classification output. Each sub-tensor t_i, pointed by these two dimensions, contains four parameterized coordinates of the predicted bounding box.

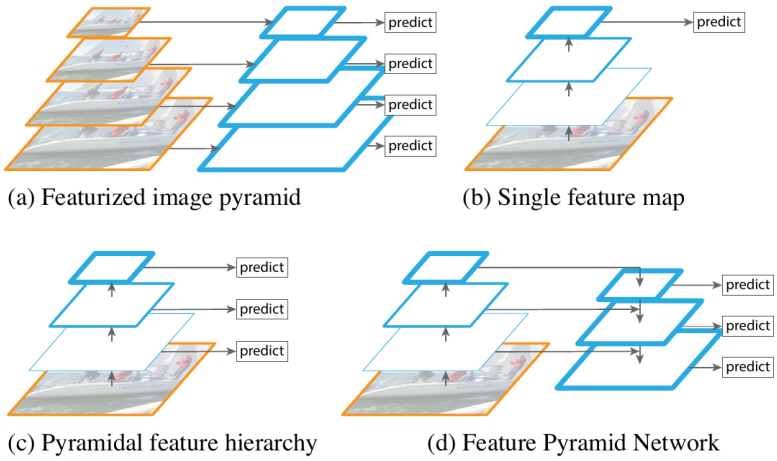
$$\begin{aligned} t_x &= \frac{x - x_a}{w_a}, t_y = \frac{y - y_a}{y_a} \\ t_w &= \log(\frac{w}{w_a}), t_h = \log(\frac{h}{h_a}) \\ t_x^* &= \frac{x^* - x_a}{w_a}, t_y^* = \frac{y^* - y_a}{y_a} \\ t_w^* &= \log(\frac{w^*}{w_a}), t_h^* = \log(\frac{h^*}{h_a}) \end{aligned} \tag{1.1}$$

Where the coordinates parametrization is shown, where x, y, w and h represents the region center and its width and height. Variable x, x_a and x^* are the predicted region, the anchor bounding box and the ground truth bounding box, respectively (likewise for y, w and h).

The final loss is composed as follow:

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{1.2}$$

Where, i is the index of an anchor, p_i is the predicted probability of anchor i to be an object; p_i^* is the ground-truth label, equals to 1 if the anchor is positive, 0 otherwise; t_i is the vector representing the 4 parametrized coordinates of the predicted bounding box; t_i^* is the ground truth bounding box associated with the positive anchor; L_{cls} is the classification loss over the two-classes (foreground and background), usually a binary cross-entropy, and L_{reg} is the regression loss, usually the smooth $l1$ loss; The two loss components are normalized by the hyper-parameters N_{cls} and N_{reg}.



(a) Featurized image pyramid (b) Single feature map

(c) Pyramidal feature hierarchy (d) Feature Pyramid Network

Figure 1.3: Feature Pyramids implementation: (a) using an image pyramid; (b) single scale feature map; (c) reuse of pyramidal feature hierarchy computed by the backbone; (d) FPN connected on top of a backbone.

Feature Pyramid Network

The Feature Pyramid Network (FPN) [26] is a building block connected on top of a backbone, to offer features optimized for different scales. Because elaborating the image at different scales is costly, the FPN reuses pyramidal hierarchy of backbone to construct a feature pyramids.

RoI Pool

A Region of Interest Pooling (RoI Pool) [27] is an operation performed to extract a small feature map for a RoI in input. In a Object Detection or Instance Segmentation network is used by the RoI Extractor to extract from the FPN layers the features corresponding to a RPN proposal, returning a fixed-size feature map independently from the RoI shape. Original model split the region proposal into equally sized sections and then max-pooling the values in each sub-window, making approximations about the size of the section with a quantization method.

To avoid quantization during the extraction, which introduces misalignments, a new RoI Pooling module called RoI Align has been proposed [14] and shown in Figure 1.4.

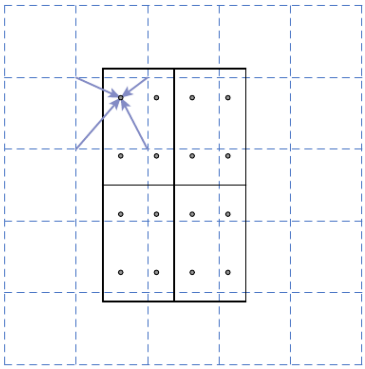


Figure 1.4: The dashed grid represents the input features, solid lines the RoI and the dots the four sampling points in each bin, computed with bilinear interpolation.

To properly align the extracted features with the input, it uses bilinear interpolation to compute the exact values for each location, aggregating the result with the max or average function.

RoI Extractor

As it can be seen in Figure 1.5, the layer (highlighted in red) connecting the two steps is usually represented by the RoI Extractor.

In a two-stage detection framework, a FPN layer is chosen heuristically as input for the RoI pooler. The original version [26] select with the following formula the FPN layer where apply the RoI pooling:

$$k = \left\lfloor k_0 + \log_2 \left(\sqrt{wh}/244 \right) \right\rfloor \tag{1.3}$$

where k_0 represents the highest level feature map, w and h are the width and height

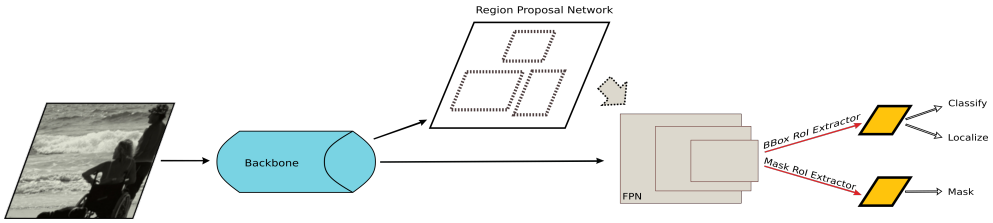


Figure 1.5: Typical components of a two-stage R-CNN-like architecture for instance segmentation. The RoI extractor is highlighted in red.

of the RoI, and 224 is the typical image size used to pre-train the backbone with ImageNet dataset.

Non-Local Attention

A particular type of attention module built with only convolutional layers is called Non-Local Attention [28]. It has the advantage to capture long-range dependency, shown in Figure 1.6, conversely to convolutional layers which take into account only a strict number of neighbors.

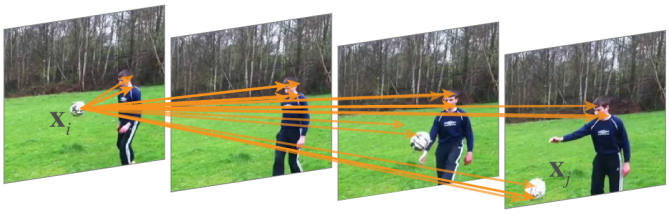


Figure 1.6: A spacetime non-local operation in a network trained for video classification.

In Figure 1.7 the non-local block is presented, where the feature maps has the following shape $T \times H \times W \times C$, where C is the number of channels, $\otimes$ is the matrix multiplication, $\bigoplus$ is the element-wise sum.

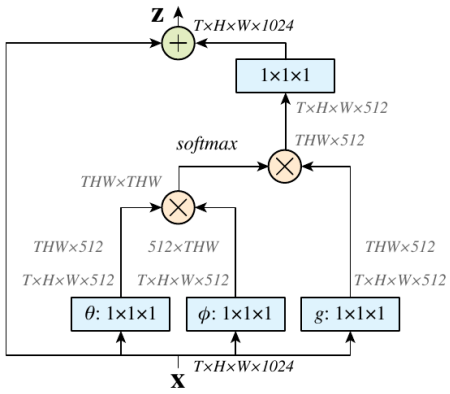


Figure 1.7: A spacetime non-local block.

The g function is a linear embedding in the form of $g(x) = Wx$, with W as a weight matrix to be learned. The f function could be one of the following:

- a Gaussian function $f(x_i, x_j) = e^{x_i^T x_j}$
- an embedded Gaussian function $f(x_i, x_j) = e^{\theta(x_i)^T \theta(x_j)}$
- the dot-product similarity $f(x_i, x_j) = \theta(x_i)^T \theta(x_j)$

The module is a generalization of the classical non-local mean operator [29].

1.2 Models

Object Detection and Instance Segmentation models followed an evolution strictly correlated. This is especially true for a type of architecture called two-stages, which formed the basis for most of the state-of-the-art networks. The first stage is devoted to the search of interesting regions independently from the class, while the second is used to perform classification, localization and segmentation on each of them. This strategy has proved to be very useful to keep under control the explosion of the number of parameters in case the number of classes is high. This divide-and-conquer

approach was first introduced in the ancestor network called Region-based CNN (R-CNN) [15], and in this section several successive and most important architectures are described.

Faster R-CNN

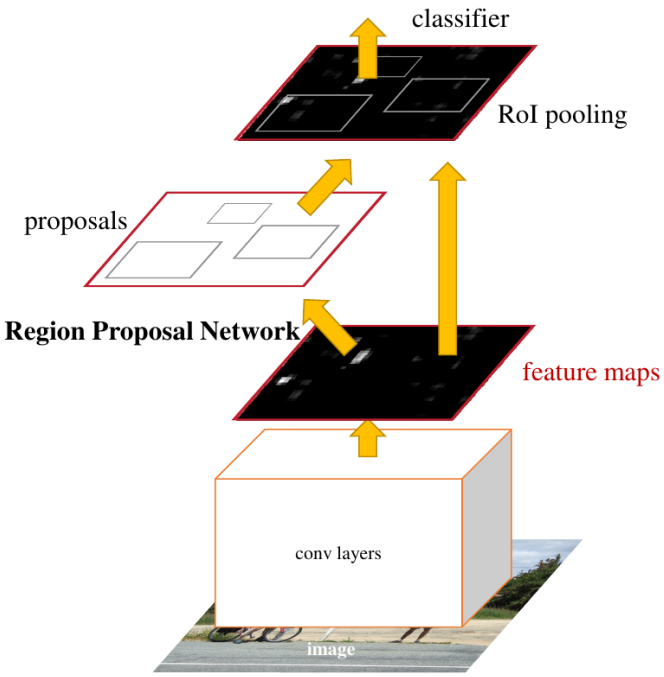


Figure 1.8: Faster R-CNN model as unified network for object detection.

The Faster R-CNN model contains the following building blocks:

- backbone (Section 1.1), a CNN network devoted to transform the input image information to a compressed and high-level representation.
- FPN (Section 1.1), also called *neck*, connected on top of a backbone, usually has the objective to offer a multiple-scale representation for the backbone features.

- RPN (Section 1.1), which complete the first stage and is composed by a classifier and a regression branches for bounding boxes.
- RoI Extractor (Section 1.1), useful to extract the corresponding features for each proposal from the RPN with the help of a RoI pool module (Section 1.1). It constitutes the link between the first and the second stage.
- RoI Head, which forms the second stage and it is devoted to classify each proposal from the RPN by the class and has another regression layer to better localize the object found.

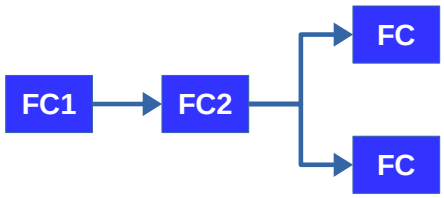


Figure 1.9: Faster R-CNN RoI head (or also Detection Head).

The RoI head (or also Detection Head) is shown in Figure 1.9, with two shared fully-connected layers first, followed by two branches, one for each task to solve, also in this case built as fully-connected layers. The classification branch gives in output $n+1$ softmax values, where n is the number of classes defined and the last class is the background. The regression branch gives in output 4 values, conditioned by the class, with the same meaning of the RPN output tensors.

The total loss for the RoI Head is the composition the weighted sum of the classification and regression losses as follow:

$$L_{RoI} = \alpha L_{cls}^{RoI} + \beta L_{regr}^{RoI} \tag{1.4}$$

Where, L_{cls}^{RoI} is a categorical cross-entropy loss and L_{regr}^{RoI} a smooth L1 loss, α and β are hyper-parameters to balance the contributions.

The total loss function for the Faster R-CNN is the composition of the two stages losses, RPN and RoI Head:

$$L = \alpha L^{RPN} + \beta L^{RoI} \tag{1.5}$$

Where, α and β are other hyper-parameters to balance the contributions.

Figure 1.10: Faster R-CNN model detection examples.

Some examples of final detections are shown in Figure 1.10.

Mask R-CNN

In addition to the object detection tasks, the Mask R-CNN [14] model includes also the segmentation task. It is composed by the same two-stage architecture of a Faster R-CNN, but includes also a branch to compute the segmentation, in parallel to RoI Head. In Figure 1.11 the RoI head architecture is shown, where the final binary mask in output is highlighted. It is formed by values trained by a binary cross-entropy to be 1 or 0 depending if the portion of image takes part of the object, or not.

Two more novelty have been introduced. The first is the replacement of the RoI pool module with a new one more effective, such as RoIAlign (see section 1.1) and an additional dedicated RoI Extractor for the segmentation branch.

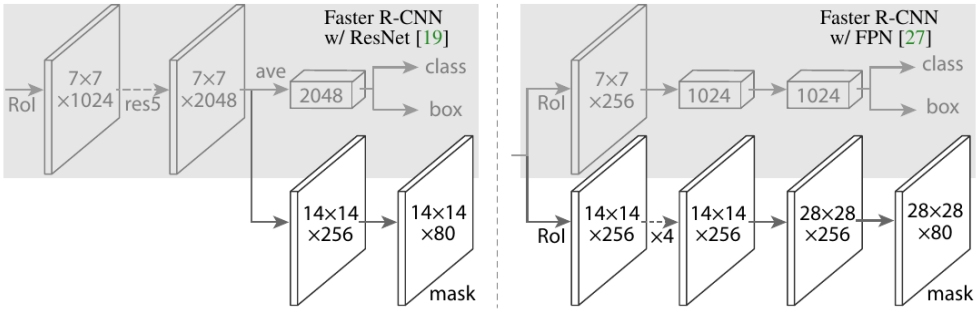


Figure 1.11: Mask R-CNN RoI head model for Instance Segmentation, with and w/out the FPN.

Some examples of final detections and segmentations are shown in Figure 1.12.

Cascade R-CNN

The Cascade R-CNN [19] is essentially a object detection model which improves the RoI head performance using a multi-stage detection sub-networks connected in cascade.

Authors identified the problem of performance degradation with the increase of the IoU threshold. They described one of the hidden problems called Exponential Vanishing Positive Samples (EVPS) problem, which causes the aforementioned effect, and they offered a new architecture to solve it.

The IoU threshold is used to determine if a detection could be considered positive or negative, comparing the overlap (IoU) between the bounding box and the ground truth. If the proposal has an IoU w.r.t the ground truth above the threshold u, then it is considered as an example for the class:

$$y = \begin{cases} g_y & \text{if } IoU\,(x,g) \geqslant u \\ 0 & \text{otherwise} \end{cases} \tag{1.6}$$

Where, g_y is the class label for the ground truth object g and x is a bounding box proposal.

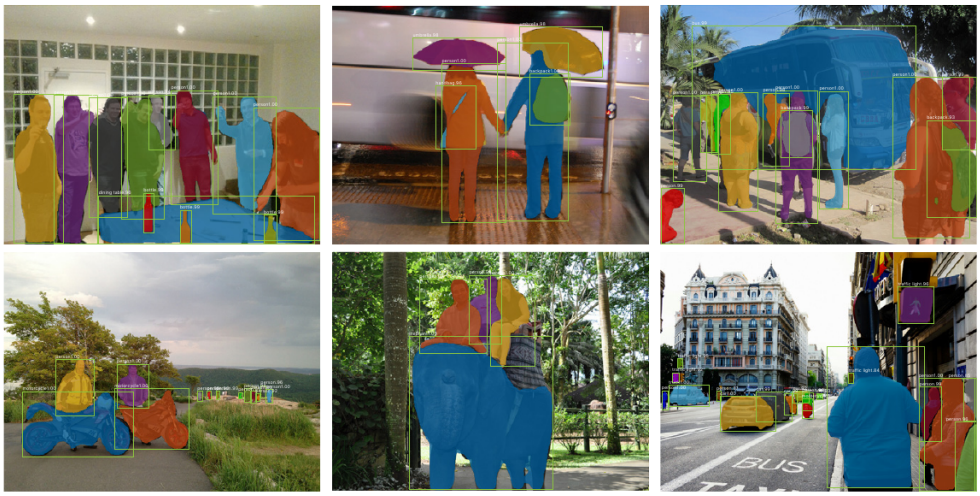
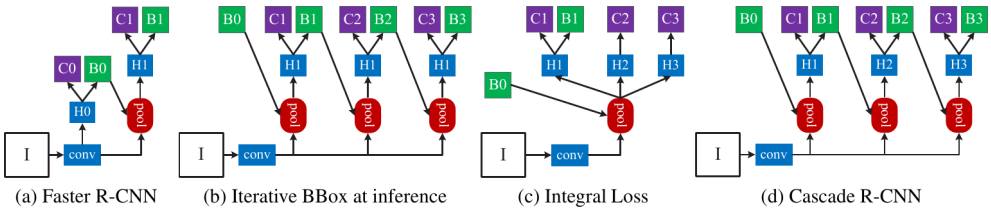


Figure 1.12: Mask R-CNN model detection and segmentation examples.

They follow the idea that a single detector is unlikely to be able to train uniformly on all quality levels of IoU. For this reason, they put multiple detectors in cascade, where each one trains with a different distribution of proposals. This specialization permits to train each one differently from the others and, at inference time, the collection and merge of all predictions increase the general quality of the results. They obtained this effect using the detection of the previous stage as new list of proposals for the following one. In addition, they use a IoU threshold u^t, increasing its value at each stage, where $u^t > u^{t-1}$.

HTC

In [22], authors proposed Hybrid Task Cascade architecture (HTC) to introduce the cascade also to instance segmentation. Instead of performing the object detection and instance segmentation tasks separately, they interweave them. In Figure 1.15, the connection between the detection and segmentation heads is shown. They connect multiple segmentation heads in cascade, exploiting the proposals at each stage to



(a) Faster R-CNN (b) Iterative BBox at inference (c) Integral Loss (d) Cascade R-CNN

Figure 1.13: Architecture frameworks: "I" is input image, "conv" backbone convolutions, "pool" region-wise feature extraction (RoI Extractor), "H" network head, "B" bounding box, and "C" classification. "B0" is proposals in all architectures coming from the RPN.

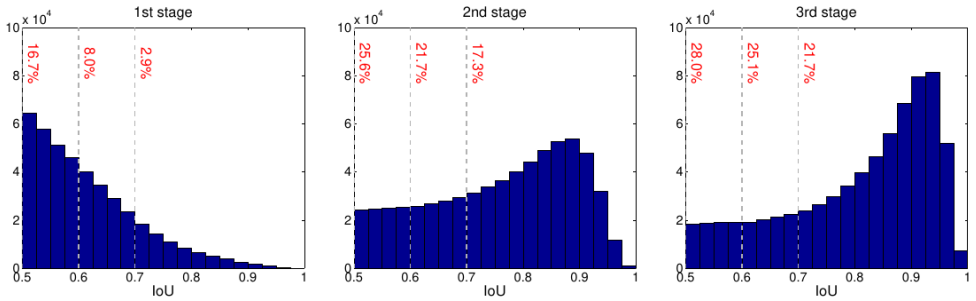


Figure 1.14: The IoU histogram of training samples. The distribution of the first stage is the output of the RPN, the others are the output of the following stages.

train and refine segmentation. The interleaved execution is described as follow:

$$\begin{aligned} x_t^{box} &= P(x, r_{t-1}),\ r_t = B_t(x_t^{box}), \\ x_t^{mask} &= P(x, r_t),\ m_t = M_t(x_t^{mask}), \end{aligned} \tag{1.7}$$

Where, x are the CNN features from backbone, x_t^{box} and x_t^{mask} the box and mask features extracted from x and corresponding to the RoI with the help of the RoI Extractor, $P(\cdot)$ is the pooling operator, B_t and M_t the detection and mask heads of the t-th stage, r_t and m_t the corresponding box and mask predictions.

Furthermore, a mask information flow is implemented to permit a further im-

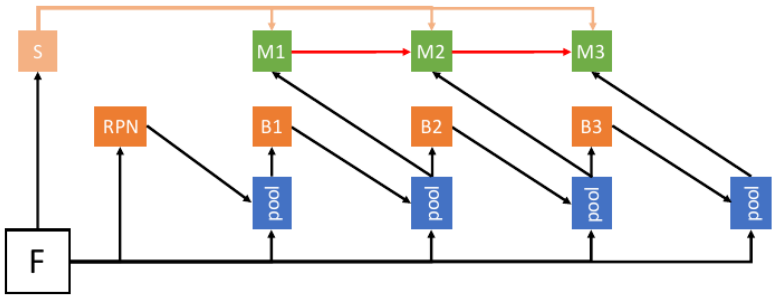


Figure 1.15: Hybrid Task Cascade architecture.

provement on mask prediction accuracy. The only difference respect to the previously defined mask prediction m_t is the follow:

$$m_t = M_t(F(x_t^{mask}, m_{t-1}^-)) \tag{1.8}$$

Where, m_{t-1}^- is the intermediate features of M_{t-1} and they use it as mask representation for the previous stage $t-1$. $F(\cdot)$ is defined as function which combine the output of the stage t with the preceding one. In this way, a progressive refinement is done. It is implemented as sum of two terms:

$$F(x_t^{mask}, m_{t-1}) = x_t^{mask} + G_t(m_{t-1}^-) \tag{1.9}$$

Where, m_{t-1}^- is the RoI features before the deconvolutional layer, with the size 14×14, G_t is a 1×1 convolutional layer. At stage t, all preceding mask heads forward with the RoI of the current stage to compute m_{t-1}^-.

$$\begin{aligned} m_1^- &= M_1^-(x_t^{mask}), \\ m_2^- &= M_2^-(F(x_t^{mask}, m_1^-)), \\ &\dots \\ m_{t-1}^- &= M_t^-(F(x_t^{mask}, m_{t-2}^-)), \end{aligned} \tag{1.10}$$

GC-net

In [30], authors made a empirical analysis of a network which is using Non-Local blocks (see Section 1.1), and found that global contexts is the same for different query positions. For this reason, they created a simplified version compared to NL-Net [28] and SENet [31], with the objective to maintain performance, but using less computation.

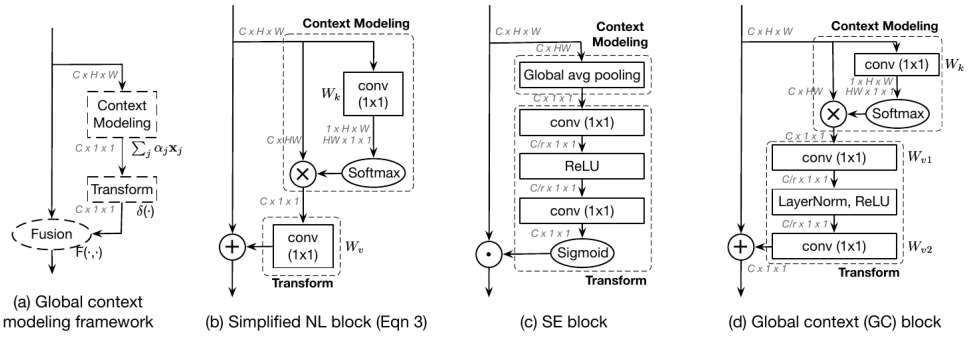


Figure 1.16: Architecture comparison between main attention blocks.

In Figure 1.16, the comparison between main attention blocks is done. The new Global Context (GC) block can be summarized with three sub-blocks:

1. global attention pooling, which uses a convolutional layer 1×1 and a softmax function to obtain the attention weights, and then performs the attention pooling.

2. feature transform with a convolutional layer 1×1.

3. feature aggregation, which uses the *add* function to aggregate the global context.

The GC block has been added to all layers in a ResNet (c3+c4+c5) to form the GC-net model.

Mask scoring R-CNN

In [32], authors studied the mask quality, defined as IoU between the instance mask and its ground truth.

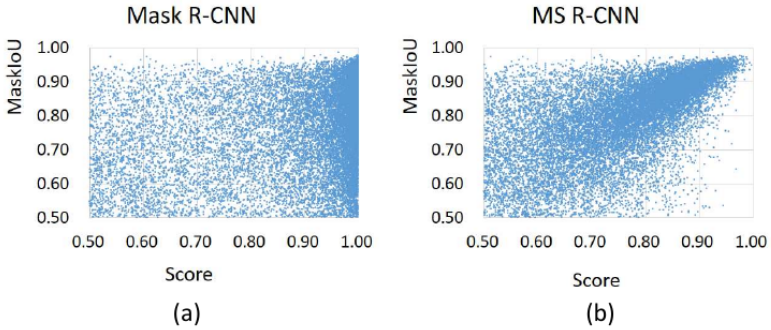


Figure 1.17: Comparison of Mask R-CNN and Mask Scoring R-CNN for the relationship between the classification score and the MaskIoU.

They taken inspiration from the AP definition for instance segmentation which uses pixel-level IoU between the predicted mask and its ground truth mask to define instance segmentation quality. In Figure 1.18, the architecture for the new branch is proposed and the final objective is to learn to predict the mask IoU for each segmentation.

The s_{mask} is defined as score for the predicted mask, as follow:

$$s_{mask} = s_{cls} \cdot s_{IoU} \tag{1.11}$$

Where, s_{cls} is the classification score done in the detection head and s_{IoU} is the regressed MaskIoU value, which represents the target to learn with a l_2 loss.

In Figure 1.17, the distribution of MaskIoU score over classification score is shown, comparing Mask R-CNN with the new Mask Scoring R-CNN model.

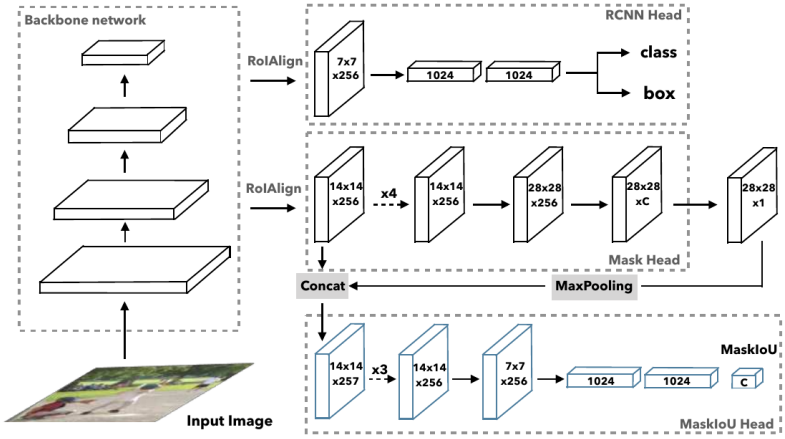


Figure 1.18: Mask Scoring R-CNN architecture.

1.3 Learning settings

Supervised Learning

A Supervised Learning strategy is a machine learning algorithm characterized by the use of two datasets, one for training the model to learn by examples, and one to validate how good the model has learned, comparing what the network predicts with what we expect.

$$\begin{aligned} \mathbf{D_s} &= \{x_i^s, y_i^s\}_{i=1}^{N_s} \\ \mathbf{D_v} &= \{x_i^v, y_i^v\}_{i=1}^{N_v} \end{aligned} \tag{1.12}$$

Where, D_s is the training dataset, composed by N_s pairs, each one formed by an input example $\mathbf{x^s}$ with the corresponding label $\mathbf{y^s}$ and D_v is the validation dataset, composed by N_v pairs, each one formed by an input example $\mathbf{x^v}$ with the corresponding label $\mathbf{y^v}$. The model $f(\theta)$, composed by the parameters θ, learns to map a input $\mathbf{x}$ into a output $\mathbf{y}$, also called ground-truth.

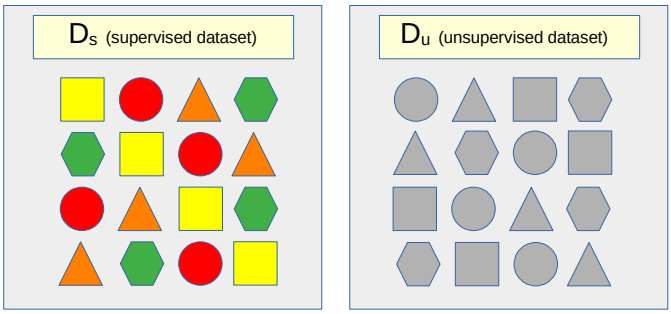


Figure 1.19: Supervised and unsupervised datasets.

Semi-Supervised Learning

Differently from the previous learning method, in case of Semi-Supervised Learning (SSL), in addition to the validation dataset, we have two datasets for training: D_u and D_s. In figure 1.19 both are showed.

$$\mathbf{D_u} = \{x_i^u\}_{i=1}^{N_u} \tag{1.13}$$

Where, $\mathbf{D_s}$ is a dataset provided with ground-truth and typically smaller, and the second dataset $\mathbf{D_u}$ contains N_u input examples $\mathbf{x^u}$ without any labels. The loss is composed by two distinct terms:

$$L = \alpha L_{sup} + \beta L_{unsup} \tag{1.14}$$

where L_{sup} is the loss coming from the training on the supervised dataset $\mathbf{D_s}$, L_{unsup} is the loss coming from the training on the unsupervised dataset $\mathbf{D_u}$, and α and β are constant values to balance the training.

This type of learning is used in case collecting examples in input $\mathbf{x}$ is an easy task, but producing the associated ground-truth is costly in terms of time and effort. This is usually true in the case we need to train a model for the tasks of Object Detection and Instance Segmentation. We refer to *Semi-Supervised Object Detection* (SSOD) when the final task is Object Detection.

Self-Supervised Learning

There is some special cases where the ground-truth can be generated automatically offline, before the training start, or online, when the training is on going. A typical example is jigsaw puzzles, where the image in input is divided in squares and rearranged in a random way, and the network learns to solve the puzzle. The creation of the supervised signal is entrusted to a function which generate a new input every training iteration.

Teacher-Student Learning

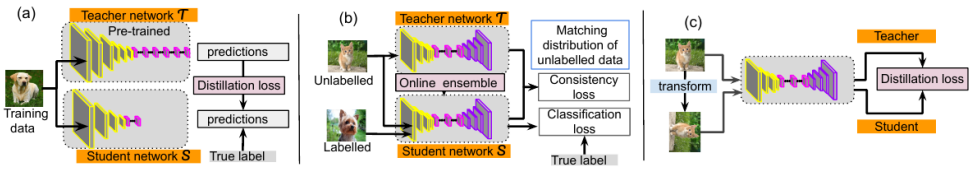


Figure 1.20: Illustrations of KD methods with S-T frameworks from [5]. (a) for model compression and for knowledge transfer, e.g., (b) semi-supervised learning and (c) self-supervised learning.

In a Teacher-Student learning setting [5], we have two independent models, where usually a transfer learning approach is applied. Broadly applied in model compression and knowledge transfer, where the Teacher is a model bigger than the Student, the paradigm has extended its borders also in other cases. Unlike from what one might expect, in [33] authors demonstrated that it can be useful also in case of Defective Teacher, using a poorly-trained teacher, or Reverse Knowledge Distillation, when the Teacher and the Student roles are inverted. In early implementations [34], the Teacher was trained first and the Student starts the train later. Nowadays, Teacher and Student training could happen at same time. In [7], the Teacher and the Student are the same model and the Student trains on D_s and D_u at same time, using the ground truth when available or using the pseudo-labels generated by the Teacher otherwise, and transfer Knowledge to the Teacher, alternatively in each iteration. In Figure 1.20,

some Teacher-Student methods are shown.

Model ensembling techniques

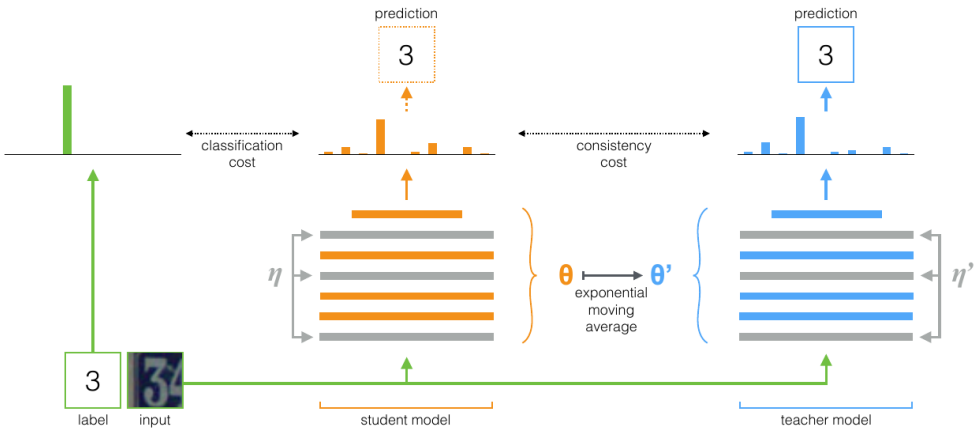


Figure 1.21: The Mean Teacher method from [6].

In literature, researchers have explored multiple ways to use predictions coming from multiple models. In the first type of ensembling, there are multiple predictions done to the same input and meshed together to form the final result. One of most recent example is the Temporal Ensembling [35] method, which extends the Π-model, by taking into account multiple predictions of same network, each one done in a different epoch of the training. The predictions are merged together with the exponential moving average of label predictions and included some penalization techniques for the ones inconsistent with the target.

Recently, authors in [6] proposed the Mean Teacher model, a new way alternative to Temporal Ensembling [35]. In this case, the method average together the weights of the network instead of the label predictions, always through the exponential moving average (EMA). In Figure 1.21, the training of a labeled example is shown. Both Teacher and Student evaluate the input applying a different level of noise. The softmax output of the models are compared using classification cost and a consistency

cost function is evaluated between Teacher and Student predictions. Then, the Student weights are updated with gradient descent and the Teacher weights are updated with exponential moving average of the student weights.

Later, in [7], authors used the Mean Teacher technique to knowledge transfer from the Student to the Teacher with the following EMA formula:

$$\theta_t \leftarrow \alpha\theta_t + (1-\alpha)\theta_s \tag{1.15}$$

Where θ_t are the Teacher weights, θ_s are the Student weights and α is a smoothing coefficient hyperparameter.

Pseudo-labeling

When we talk about Semi-Supervised Learning, usually we define a way to automatically generate the labels for the unsupervised dataset D_u. One example of this techniques is Pseudo-labeling [36], where we train a model to generate the best labels it can. This new dataset could be modeled as follow:

$$\begin{aligned} \tilde{\mathbf{D}}_\mathbf{u} &= \{x_i^u\, \tilde{y}_i^u\}_{i=1}^{N_u} \\ \tilde{y}_i^u &= y_i^u + n_i \end{aligned} \tag{1.16}$$

Where $\tilde{D}_u$ is the new unsupervised dataset, $\tilde{y}_i^u$ is the label predicted by the network for the i-th example, y_i^u is the real ground-truth that we do not know and n_i is a additional noise introduced by the network itself. If the model performance are high, then the noise part is small and $\tilde{y}_i^u$ will converge to y_i^u. Otherwise, the noise will predominate and the probability of an erroneous prediction will be high.

Unbiased Teacher

In [7], authors collected best performing ideas around Semi-Supervised Learning and built the Unbiased Teacher model. It consists of two stages: Burn-In Stage and Teacher-Student Mutual Learning Stage. The first consists of a number of iterations of Supervised Learning through the supervised dataset D_s, where the Student receives

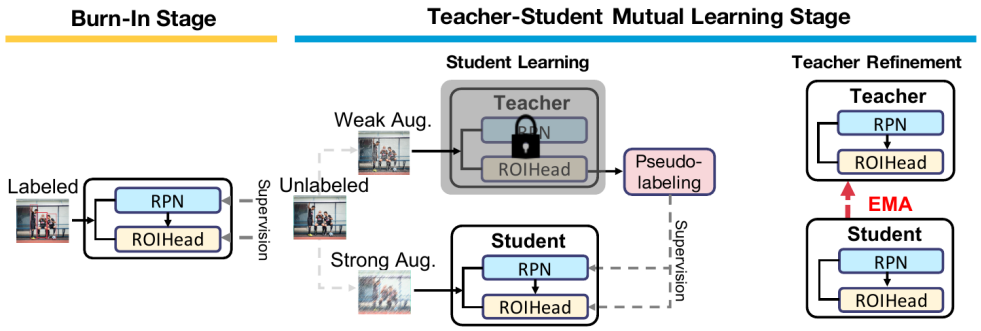


Figure 1.22: The Unbiased Teacher method from [7].

as input the images weak-augmented and the same input strongly-augmented. This stage is useful to warm-up the student model and usually consists of two thousand iterations. Because the Student model is a Faster R-CNN model, its loss for the burn-up stage is the following:

$$L_{sup} = \sum_i \left(L_{cls}^{rpn}(x_i^s, y_i^s) + L_{reg}^{rpn}(x_i^s, y_i^s) + L_{cls}^{RoI}(x_i^s, y_i^s) + L_{reg}^{RoI}(x_i^s, y_i^s) \right) \tag{1.17}$$

Where the L_{cls}^{rpn} and the L_{reg}^{rpn} are the classification and regression losses, respectively, for the Region Proposal Network (RPN) stage; the L_{cls}^{RoI} and L_{reg}^{RoI} are the classification and regression losses, respectively, for the RoI head model of Faster R-CNN.

Then, the Teacher model is initialized as clone of Student weights and the second stage starts and goes on until the end of the training. For each iteration, a batch of images from D_u are weakly-augmented and passed as input to the Teacher in evaluation mode, which generates the pseudo-labels filtered by a classification score threshold defined as hyperparameter. Now, the student is firstly trained on weak- and strongly-augmented images coming from D_s and its ground-truth. Then, it is trained with the strongly-augmented images from D_u with the pseudo-labels generated by the Teacher.

The Student weights are trained with back-propagation for the D_s as in Formula

1.17. For the dataset D_u, the losses applied are only the classification losses.

$$\theta_s \leftarrow \theta_s + \gamma \frac{\partial(L_{sup} + \lambda_u L_{unsup})}{\partial \theta_s}, L_{unsup} = \sum_i \left(L_{cls}^{rpn}(x_i^u, \hat{y_i^u}) + L_{cls}^{roi}(x_i^u, \hat{y_i^u}) \right) \quad (1.18)$$

The unsupervised losses from the bounding box are discarded since the naive confidence thresholding does not guarantee to filter out the wrong pseudo bounding boxes. Finally, the knowledge is partially passed back from the Student to the Teacher with EMA (see Formula 1.15).

Chapter 2

Research findings

All we have to decide is what to do with the time that is given us.

J.R.R. Tolkien

2.1 A Novel Region of Interest Extraction Layer for Instance Segmentation [1]

Normality is a paved road. It is easy to walk but no flowers grow on it.

Vincent Van Gogh

2.1.1 Introduction

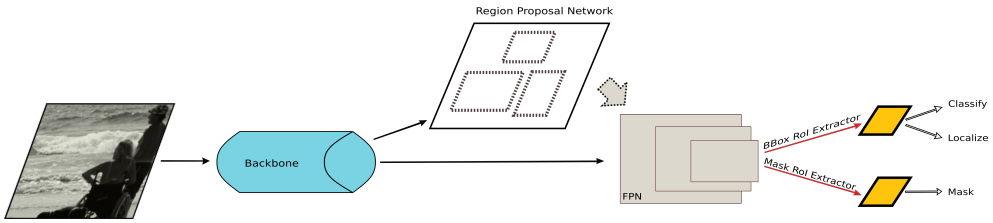


Figure 2.1: Typical components of a two-stage R-CNN-like architecture for instance segmentation.

Nowadays, instance segmentation is one of the most studied topics in the computer vision community. It differs from both object detection, where the final output is the set of rectangular bounding boxes which localize and classify any object instance, and semantic segmentation, where the goal is to classify any image pixel without considering if it is part of a specific instance. In instance segmentation, the final goal is to be able to cut the single instances of objects from the original image. Its characteristics make this task very useful for several advanced applications, such as object relationship detection, automatic image captioning, content-based image retrieval, and many others.

In the recent literature, many studies have addressed the instance segmentation problem. The proposed architectures can be grouped into two main categories: one-step and two-step architectures. The one-step architectures obtain the results with a

single pass, making a direct prediction from the input image. On the contrary, an architecture belonging to the second category (two-step) is usually composed of a Region Proposal Network (RPN) [15], which returns a list of Regions of Interest (RoI) that are likely to contain the searched object, followed by a more specialized network with the purpose of detecting or segmenting the object/instance within each of the bounding boxes found. These networks descend from their ancestor network called R-CNN [15].

The typical components of a two-step architecture are shown in Fig. 2.1. As it can be seen in the diagram, the layer (highlighted in red) connecting the two steps is usually represented by the RoI extractor, which is the main focus of this contribution. Since this layer plays a crucial role in terms of final results, it should be carefully designed to minimize the loss of information.

The main objective of this layer is to perform pooling in order to transform the input region, which can be of any size, to a fixed-size feature map. Several previous papers have tackled this problem using different RoI pooling algorithms such as RoI Align [14], RoI Warp [37] and Precise RoI Pooling [23]. Since instances of objects can appear in the image with different scales, the existing architectures (as shown in Fig. 2.1) exploit a Feature Pyramid Network (FPN) [26] combined with an RPN (e.g. Fast R-CNN [27], Faster R-CNN [25] and Mask R-CNN [14]), to generate multi-scale feature maps. An FPN is composed of a bottom-up pathway, where final convolutional layers from the backbone are often chosen, followed by a top-down pathway to reconstruct spatial resolution from the upper layers of the pyramid that have a higher semantic value. With the introduction of a FPN, the fundamental issue is the selection of a FPN layer to which the RoI pooler will be applied.

Traditional methods make the selection based on the RoI obtained by the RPN. They use the formula proposed by [26] to discover the best *k-th* layer to sample from, which is based on the width w and height h of the RoI as follows:

$$k = \left\lfloor k_0 + \log_2 \left(\sqrt{wh}/244 \right) \right\rfloor \tag{2.1}$$

where k_0 represents the highest level feature map and 224 is the typical image size used to pre-train the backbone with ImageNet dataset. This hard selection of a single

layer of FPN might limit the power of the network's description and our intuition (supported by previous works, such as [38]) is that if all scale-specific features are retained, better object detection and segmentation results can be achieved.

The main contributions of this section are the following:

1. A novel RoI extraction layer called GRoIE is proposed, with the aim of a more generic, configurable and interchangeable framework for RoI extraction in two-step architectures for instance segmentation.

2. Exhaustive ablation study on different components of the proposed layer is conducted in order to evaluate how the performance changes depending on the various choices.

3. GRoIE is introduced to the major state-of-the-art architectures to demonstrate its superior performance with respect to traditional RoI extraction layers.

The section is organized as follows. sub-section 2.1.2 describes the state of the art. In sub-section 2.1.3, the proposed architecture is described in detail. Section 2.1.4 describes the experimental methodology as well as our in-depth ablation study on component selection. Additionally, in this section, we show how the inclusion of GRoIE layer in state-of-the-art architectures can lead to significant improvements in the overall performance.

2.1.2 Related Work

As mentioned in the introduction, modern detectors employ a RoI extraction layer to select the features produced by the backbone network according to the candidate bounding boxes coming from a RPN. This layer was first introduced in R-CNN network. Since then, many architectures derived from R-CNN (e.g., Mask R-CNN, Grid R-CNN [39], Cascade R-CNN [21], HTC [22] and GC-net [30]) have used this layer as well. Usually, to be more invariant to object scale, the layer is not directly applied to the backbone features, but instead to an FPN attached on top of the backbone.

In [26], a RoI pooling action is applied to a single heuristically-selected FPN output layer. This approach suffers from a problem related to untapped information.

In [40], the authors propose to extract mask proposals from each scale separately, rescale them and include the resulting scales in a unique multi-scale ranked list. Eventually, only the best proposals are selected. In [41], the authors propose to fuse features belonging to different scales by max function, using an independent backbone for each image scale. In our work, on the contrary, we utilize a feature pyramid to simplify the network and avoid doubling the number of parameters for each scale. In SharpMask [42], the authors make a coarse mask prediction after which they fuse feature layer back in a top-down fashion until reaching the same size of the input image. In PANet [38], the authors highlight that the information is not strictly connected with a single layer of the FPN. They propagate low-level features, building another FPN-like structure coupled with the original FPN, where the RoI-pooled images are combined. Our proposed GRoIE layer is inspired by this approach with the difference that it is more lightweight because of not using any extra FPN-coupled stack and proposes a novel way to aggregate data from the RoI-pooled features. Auto-FPN [43] extends PANet model by applying the Neural Architecture Search (NAS) concept. Also, AugFPN [44] can be considered an extension of PANet model. The module we directly compare our module with is the Soft RoI Selector, which performs a RoI pooling on each FPN layer for concatenating the results. Subsequently, through the *Adaptive Spatial Fusion*, they are combined to create a weight map which passes through 1x1 and 3x3 convolutions sequentially. In our case, we first apply a distinct convolutional operation on each layer of the FPN output which very effectively helps the network to automatically focus on the best scales. Next, we apply a sum instead of concatenation because we have proven it has a greater learning potential for the network. Finally, an attention layer is applied that combines fully-connected layers and convolutions to further filter the multi-scale context.

In Multi-Scale Subnet [45], authors propose an alternative method to RoI Align which uses crop-resized branches to extract the RoI at different scales. They use convolution with 1x1 kernel to simply maintain the same number of outputs for each branch without the purpose of helping the network to process data. Then, before summing all branches, they apply an average pooling to reduce each branch to the same size. Finally, a convolutional layer with 3x3 kernel is used as post-processing stage.

In our ablation study, we demonstrate that these convolutional configurations for pre- and post-processing are not the best ones possible to achieve better performance.

IONet [46] proposes not to use any FPN network but concatenated, re-scaled and dimension-reduced features directly from the backbone before performing classification and bounding box regression. Finally, Hypercolumn [47] employs a hypercolumn representation to classify a pixel, using convolutions with 1x1 kernel and up-sampling the results to a common size to be able to sum them all. In this case, the absence of a optimized RoI pooling solution and an FPN can negatively affect the final performance. Moreover, simply processing columns of pixels taken from different stages of the backbone can be a limitation. In fact, in our ablation study we will demonstrate that adjacent pixels are important for optimally extracting information within the various features.

2.1.3 Generic RoI Extraction Layer

The FPN is an architecture commonly used to extract features from different image resolutions. It has been demonstrated to have an effective power to maintain spatial information avoiding the expensive computation caused by a separate elaboration of each scale. Inside a two-stage detection framework, one FPN output layer is heuristically selected as unique source of RoI Pooling action. Although the formula is well thought out, it is clear that the layer selection is the result of an arbitrary choice.

In order to demonstrate this statement, we have compared this heuristic (proposed by [26]) as baseline with a random selection of the FPN layer to sample from. Table 2.1 shows the average precision (AP) with different metrics (detailed in Section 2.1.4). Comparing the first two rows of the table, it is evident that the difference between the randomly-selected and the heuristic choice is not enormous. As a further proof, Fig. 2.2 shows the progress with training epochs and demonstrates that the progress is similar. This is understandable considering that each FPN layer is derived from the previous one. It means that information is existent in the FPN layers, but in a more or less tangled way to be classified by the following modules of the network.

These results highlight that the network is capable of extracting information with good enough quality to discriminate classes from any available scale. To corroborate

Method	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
baseline [26]	36.5	58.4	39.1	21.9	40.4	46.8
random	34.8	56.9	37.0	19.1	39.3	45.2
sum	**36.8**	**59.0**	**39.5**	**22.0**	**41.0**	**47.2**

Table 2.1: Comparison of different methods for selecting FPN layers. Training and testing are performed on COCO minival dataset with 12 training epochs. For explanation of the different evaluation metrics in the table columns, please refer to section 2.1.4

this finding, we have also tried to sum the FPN layers, obtaining an improvement of 0.3% in average precision (see Table 2.1 and Fig. 2.2). This enhancement suggests that if all the layers are aggregated appropriately, it is more likely to produce higher quality features.

Based on these preliminary ideas, we propose a novel RoI extraction layer called Generic RoI Extractor (*GRoIE*) whose architecture can be seen in Fig. 2.3.

GRoIE is composed of the following modules:

1. **RoI pooler module**: it is a module that performs a max pooling on non-uniform region of interest to obtain a fixed-size representation. Currently, many pooling techniques such as RoI Pooling [27] and RoI Align [14] are available. Among the existing RoI pooling techniques, we found RoI Align [14] as the most appropriate since it reduces a rectangular feature map region by dividing the original RoI in equal boxes and applying bilinear interpolation inside each of them. This helps to avoid pixel quantization.

2. **Pre-processing module**: its objective is to apply a preliminary elaboration to the pooled regions. This gives the network an additional degree of freedom which is specific for each image scale. This module is devoted to pre-processing the feature maps and it is usually obtained by means of a convolutional layer associated with each image scale. As will be shown in the ablation analysis reported in Section 2.1.4, the optimal configuration consists

of a single 5x5 convolutional layer per scale. Our experiments suggest that it is not convenient to process the features individually which can be explained by acknowledging that each feature is semantically connected with adjacent features. This is particularly true, remembering that the final objective is object detection/segmentation and, usually, objects are spread over a consistent region of the image.

3. **Aggregation module**: it defines how to aggregate the single RoIs coming from each branch. The most frequent operations are concatenation and summation. There are multiple ways of merging different branches. After our ablation anal-

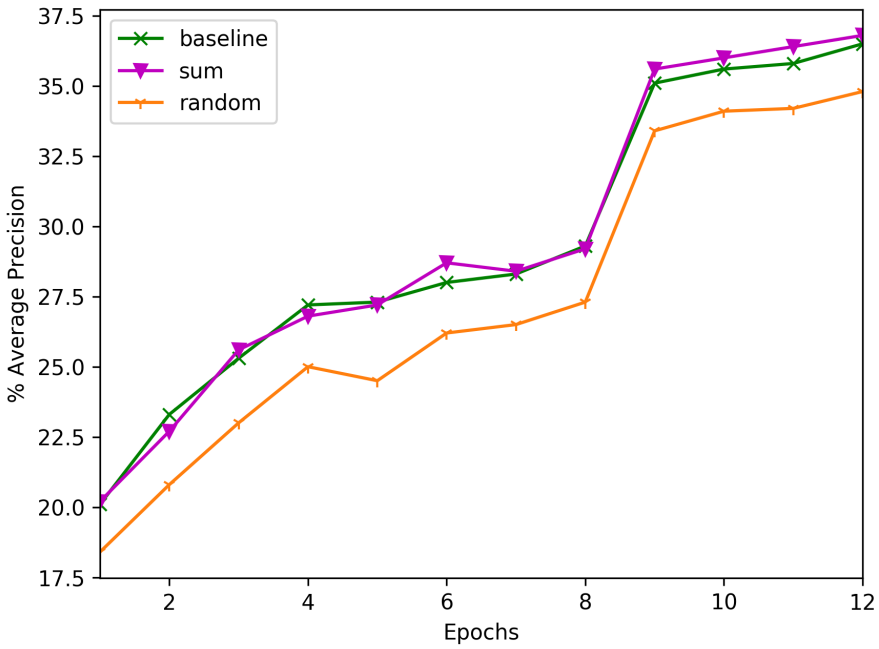


Figure 2.2: Average precision trend for different FPN layer selection strategies. Training and testing are performed on COCO minival dataset with 12 training epochs.

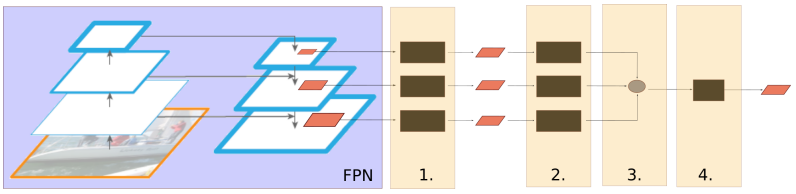


Figure 2.3: Generic RoI Extraction framework. (1) RoI Pooler. (2) Preprocessing phase. (3) Aggregation function. (4) Post-processing phase.

ysis, we found that the sum is able to minimize the number of features to be computed for the next layer, and this requires less effort from the network to converge to a stable training.

4. **Post-processing module**: it is an extra elaboration step applied to the merged features before eventually returning them. It permits the network to learn global features, jointly considering all the scales. To strengthen informative power of the final RoI, three module types have been considered for post-processing: a convolutional layer, a non-local layer [28] and an attention layer [48]. Although the attention module is more complex because it requires also a fully-connected layer, our ablation analysis demonstrates that it is the best performing choice. The reason is that unlike the pre-processing module, the main objective of this layer is to eliminate useless information. In particular, the “query content and relative position" configuration, called ε_2 in [48], attention factor is used. This is more sensitive to the query content and have the higher impact on image contents.

Summarizing, starting from a region produced by the RPN, for each scale, a fixed-size RoI is pooled from the region. The resulting n feature maps are, first, separately pre-processed and, then, merged into a single feature map. Finally, post-processing is applied to extract global information. This architecture grants an equal contribution of each scale and benefits from the information embodied in all FPN layers by overcoming the limitations inherent in the arbitrary choice of a single FPN layer. It

is worth noting that this procedure is valid for both object detection and instance segmentation.

2.1.4 Experiments

In this section two sets of experiments are reported. The first set is a module-wise ablation analysis of the proposed GRoIE layer with the aim of finding the best combination of choices for each of the modules described in the previous section. As was mentioned above, GRoIE can be plugged into architectures for both object detection (bounding box) and instance segmentation.

In the first set of experiments, we focus on object detection task only and employ the well-known Faster R-CNN as baseline. In the second set, we apply GRoIE, with the best configuration found, to different architectures with the aim of showing the improvement in average precision for both object detection and instance segmentation. This will allow us to show that the improvement produced by GRoIE is independent from both tasks as well as the utilized architecture.

Dataset and Evaluation Metrics

Datasets. In order to evaluate our proposal, we performed experiments on MS COCO dataset 2017 [4] which is the *de facto* standard dataset for large-scale object detection and instance segmentation tasks. It is composed of 80 object categories and contains more than 116 thousand images in its training set.

Evaluation Metrics. To extract the metrics, we used the official COCO python package. The validation dataset, referred to as *minival*, includes 5000 images.

The package calculates the Average Precision (AP) with different IoU (Intersection over the Union) thresholds for both bounding box and segmentation tasks. The primary metric, indicated simply as AP, is calculated with IoU thresholds from 0.5 to 0.95. Other metrics include AP_{50} with the IoU threshold of 0.5 and AP_{75} with 0.75. In addition, separate metrics are calculated for small (AP_s), medium (AP_m) and large (AP_l) objects.

Implementation details

All the results with which we compare ours are not taken from the original papers, but they were obtained by training on the same hardware, with the same configuration (apart the RoI extractor) and by using the original authors' code when available. These precautions are taken in order not to have the comparison affected by any small changes in either the configuration or the code. We used MMDetection [49] as base framework to develop our code.

The following base configuration was used for every experiment. Experiments were conducted on 6 GPUs (Nvidia Tesla P100 with 12 GB of memory) for 12 epochs with an initial learning rate of 0.015, with a weight decay of 0.0001 after 9 and 11 epochs, a batch size of 2 images per GPU, and a random seed always equals to the number zero. Since in most of the experiments reported in the literature, reference hardware is composed of 8 GPUs with batch size 2 and learning rate equal to 0.02, we followed the Linear Scaling Rule proposed in [50] to have a fair comparison. The long edge and short edge of the images were resized to 1333 and 800, but the aspect ratio was maintained. ResNet50 [24] was used as backbone and RoI Align was selected for the RoI Pooling module (no ablation analysis was conducted on this module).

Module-wise ablation analysis

In this section, we investigate how the choices of the GRoIE modules influence its final performance. We compare our RoI Extractor architectures with the baseline represented by the single-layer RoI extractor proposed as part of the Faster R-CNN on [25] paper.

Aggregation module analysis. We start from aggregation module because choosing how to merge the data technically has a significant importance on the architecture of the module itself. In order to evaluate the effects of different choices separately, neither pre-processing nor post-processing are applied in this experiment. Each FPN output layer is RoI pooled to create a 256 dimensional feature map and subsequently merged to form a single RoI.

Method	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
baseline	36.5	58.4	39.1	21.9	40.4	46.8
sum	**36.8**	**59.0**	**39.5**	**22.0**	**41.0**	**47.2**
sum+	36.0	57.9	38.3	21.6	39.7	46.1
concat	36.1	58.0	38.6	21.4	40.3	45.7

Table 2.2: Ablation analysis on aggregation module.

There are mainly two choices for aggregating different branches: concatenation and summation. In the first case, we need to reduce the feature maps from 1024 to 256 dimensions because we have 4 FPN layers, each one composed by 256 dimensions feature maps. This can be easily done using a convolutional layer with 1x1 kernel. A sum-based aggregation is simpler, but a fair comparison with concatenation is needed. Therefore, in addition to a naive *sum* operator, we included a variant of sum aggregation followed by a convolutional layer with 1x1 kernel as post-processing. We call this *sum+*.

Table 2.2 shows the comparison between the proposed choices and a single-layer RoI extractor module (indicated as "baseline"). To better justify our final choice, we show in Fig. 2.4 the trend in average precision when the training epochs progress. Looking at the results of *sum+* and concatenation, one might argue that the integration of different FPN layers, the basis of our work, is not always beneficial. This can be attributed to the added complexity which can be, in some cases, counterproductive and generate side effects. In the case of *sum*, while at the beginning the trend is very similar, later in the training this operator achieves better accuracy with a stable trend, suggesting that this gap could potentially increase with more training epochs. Therefore, we selected *sum* operator for the aggregation module of GRoIE.

Pre-processing module analysis. For this ablation analysis, as mentioned above and based on the findings of the previous module, we chose the *sum* operator for the aggregation module and did not apply any post-processing. With regard to pre-processing, we consider three possible choices: using a convolutional layer with different kernel sizes, using a non-local module or using an attention module which was

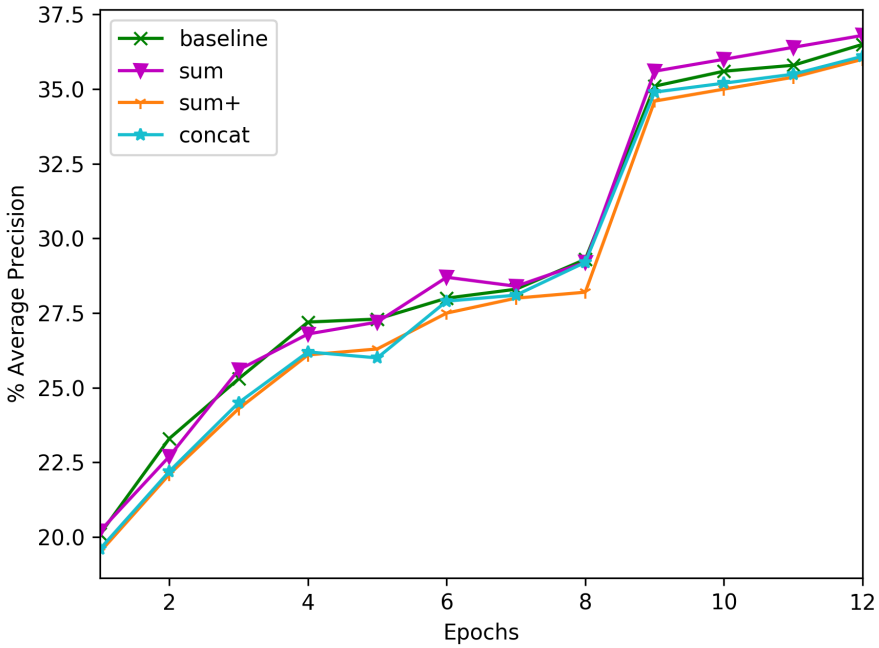


Figure 2.4: Aggregation module analysis of average precisions trend on training.

described in the previous section.

Table 2.3 shows the comparison of these choices with the baseline as in the case of the aggregation module. Regarding the convolutional layer, it can be noticed that by increasing the kernel size, the results are consistently improved. This confirms the close correlation between neighboring features. We should mention that the processed feature maps are only 7x7 in size. This stopped us from increasing the kernel furthermore.

Post-processing module analysis. Finally, we analyze the post-processing module, by keeping the *sum* operator as aggregation strategy and not applying pre-processing.

Comparing Tables 2.3 and 2.4 which contain results for pre- and post-processing

Method	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
baseline	36.5	58.4	39.1	21.9	40.4	46.8
conv 1x1	36.2	58.1	38.9	21.3	40.0	46.1
conv 3x3	37.0	58.6	40.1	**22.0**	40.9	47.0
conv 5x5	**37.2**	**59.0**	**40.4**	21.9	**41.1**	**48.3**
Non-local	36.5	58.5	39.0	21.9	40.5	46.7
Attention	36.4	58.3	39.1	21.8	40.4	46.7

Table 2.3: Ablation analysis on pre-processing module.

Method	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
baseline	36.5	58.4	39.1	**21.9**	40.4	46.8
conv 1x1	36.0	57.9	38.3	21.6	39.7	46.1
conv 3x3	36.6	58.3	39.3	21.3	40.5	46.6
conv 5x5	36.6	58.4	39.5	21.6	40.5	46.9
Non-local	36.7	**58.8**	38.9	21.8	**40.9**	46.8
Attention	**36.8**	**58.8**	**39.9**	**21.9**	40.4	**47.0**

Table 2.4: Ablation analysis on post-processing module.

Method	Backbone	Object detection						Instance segmentation					
		AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Faster R-CNN	r50-FPN	36.5	58.4	39.1	21.9	40.4	46.8	N/A	N/A	N/A	N/A	N/A	N/A
+GRoIE (ours)	r50-FPN	**37.5**	**59.2**	**40.6**	**22.3**	**41.5**	**47.8**	N/A	N/A	N/A	N/A	N/A	N/A
Grid R-CNN	r50-FPN	39.1	57.2	42.2	22.1	43.0	50.6	N/A	N/A	N/A	N/A	N/A	N/A
+GRoIE (ours)	r50-FPN	**39.8**	**58.1**	**42.9**	**23.6**	**43.9**	**51.5**	N/A	N/A	N/A	N/A	N/A	N/A
Mask R-CNN	r50-FPN	37.3	58.9	40.4	21.7	41.1	48.2	34.1	55.5	36.1	18.0	37.6	46.7
+GRoIE (ours)	r50-FPN	**38.4**	**59.9**	**41.7**	**22.9**	**42.1**	**49.7**	**35.8**	**57.1**	**38.0**	**19.1**	**39.0**	**48.7**
GC-net	r50-FPN	39.5	62.0	42.7	**24.6**	43.2	51.6	35.9	58.5	38.0	**20.4**	39.4	49.0
+GRoIE (ours)	r50-FPN	**40.3**	**62.4**	**44.0**	24.2	**44.4**	**52.5**	**37.2**	**59.3**	**39.8**	20.2	**41.0**	**51.2**

Table 2.5: Average precision w/ and w/o our GRoIE module. In the case of object detection networks, since they do not make image segmentation, an N/A has been inserted.

modules reveals a major difference. While in the former, convolutional layers with different kernel sizes improve the results but non-local/attention modules do not, in the latter table the outcomes are opposite; that is, improvement of convolutional layers is negligible, while non-local and attention methods bring about noticeable enhancement. This can be explained by the fact that while in pre-processing there is the need to extract spatial contributions of the different layers where convolution acts correctly, in the post-processing phase the layers have already been merged by the aggregation module. Therefore, convolution does not add significant information. On the contrary, in post-processing, non-local and attention methods are able to remove useless information by focusing only on the significant parts of the image with attention mechanism.

Application of GRoIE to different architectures

As stated at the beginning of this section, the second set of experiments starts from the choices made on GRoIE modules based on the ablation analysis and integrates our proposed layer within several state-of-the-art architectures, with the aim of evaluating its benefits for both object detection and instance segmentation. We have considered,

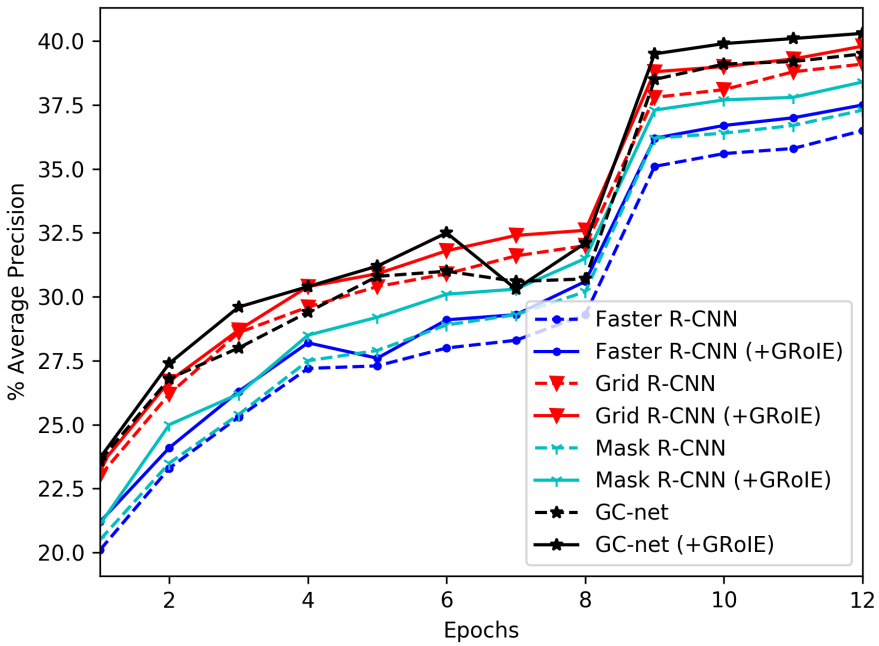


Figure 2.5: Object detection average precision on *minival* COCO dataset.

first of all, the networks that best represent the two-stage networks: Faster R-CNN and Mask R-CNN. Furthermore, we have taken into consideration the networks that have shown the best results in the recent years: Grid R-CNN [39] for object detection and GC-net [30] for instance segmentation too. For the latter network, there are two RoI extractors. The first one is used for the detection part to extract the RoIs provided by the RPN; the second one is used by the segmentation part to extract the RoIs provided by the detection.

For this experiment, we have thus replaced only the standard RoI extraction modules with GRoIE in its most performing configuration: *sum* as aggregation function, 5x5 convolution for pre-processing and attention module for post-processing.

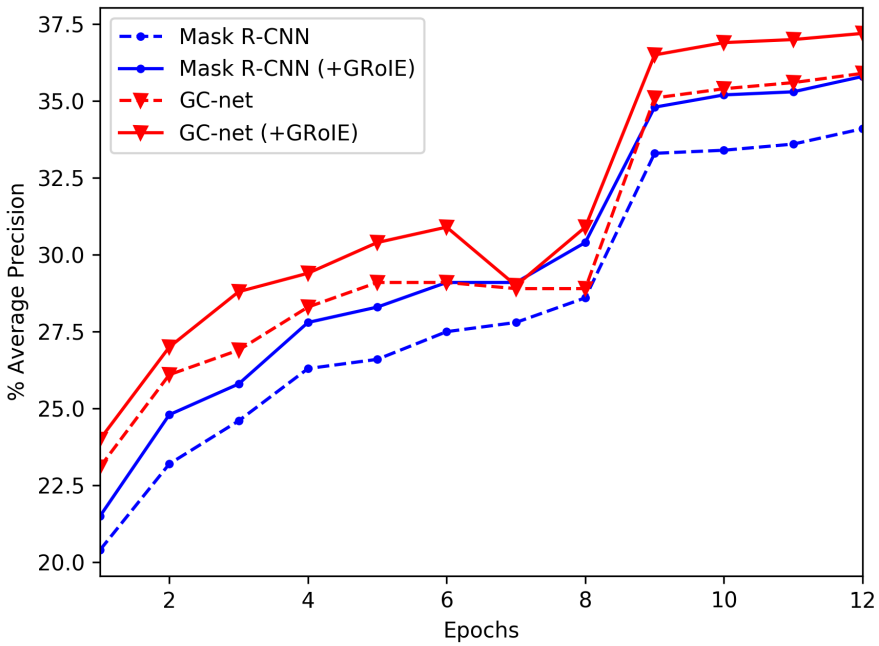


Figure 2.6: Instance segmentation average precision on *minival* COCO dataset.

Table 2.5 shows the achieved results for both object detection (bounding boxes) and instance segmentation. It is rather evident that the introduction of GRoIE as RoI extraction layer strongly contributes to an improvement in precision in all the tested architectures. As expected, the amount of this improvement is not always the same and varies from a minimum of 0.7% AP to a maximum of 1.1% AP for bounding boxes, and from a minimum of 1.3% AP to a maximum of 1.7% AP for instance segmentation. Looking at the other evaluation metrics, the gain is even more noticeable, with a maximum of 2.2% for AP_l in GC-net.

This improvement is even more evident from Figs. 2.5 and 2.6, where the average precision is illustrated with the progress of training epochs. In these graphs, it can be

seen that in later epochs the positive effect of GRoIE increases, suggesting that it can arguably be even higher with more training epochs.

2.1.5 Conclusion

In this thesis, we proposed a novel RoI extraction layer for two-step architectures designed for object detection and instance segmentation. The intuition underlying our proposal is that all the feature scales obtained by an FPN are potentially equally-useful for obtaining good final results. The proposed layer, called GRoIE (Generic RoI Extractor), builds upon this intuition by first pre-processing each single layer, then aggregating them together, and finally applying attentive mechanisms as post-processing in order to remove useless (global) information.

Experiments are conducted on COCO dataset and a comprehensive ablation study has been conducted in order to select the best configuration of modules. Furthermore, the addition of GRoIE to state-of-the-art two-step architectures for both object detection and instance segmentation has shown a consistent improvement in average precision in all the experiments.

While preliminary, the results reported in this theses are quite promising and seem to indicate the potentiality of GRoIE as novel extraction layer. As a consequence, our future works will concentrate on exploiting the modularity of GRoIE to further enhance the quality of the output features to improve the overall accuracy of different computer vision applications. In addition, neural networks are now increasingly heavy to perform. For this reason, an important field of exploration also for GRoIE regards precisely adopting every possible stratagem to lighten the workload while keeping performance unchanged.

2.2 Recursively Refined R-CNN: Instance Segmentation with Self-RoI Rebalancing [2]

Life is like an echo: if you do not like what you see, you need to change the message that you send.

James Joyce

2.2.1 Introduction

Computer vision is a field of continuous experimentation, where new and better performing algorithms are developed every day and are able to operate in environments with increasingly extreme conditions. In particular, object detection, and instance segmentation as its narrower extension, offers complex challenges which are utilized in various applications, including medical diagnostics [8], autonomous driving [10], visual product search [13], and many others. All these applications demand high-performing systems in terms of prediction quality, as well as low memory usage. Therefore, a desirable architecture is as light as possible regarding the parameter count since it reduces the search space and enhances generalization, while retaining high-quality detection and segmentation performance.

However, often these two goals are conflicting. The R^3-CNN architecture and the corresponding training mechanism that we propose present a trade-off between these two conflicting goals. We show that our model is able to obtain the same performance of complex networks (such as HTC [22]) with a network as light as Mask R-CNN [14].

The accuracy of instance segmentation systems is strongly based on the concept of intersection over union (IoU), which is used to identify the detection precision with respect to the ground truth. The higher this value is, the more accurate and the less noisy the predictions are. However, by increasing the IoU threshold, a problem called *exponentially vanishing positive samples* [51] (EVPS) is also introduced, meaning that it can give rise to the problem of good proposals scarcity compared to

low-quality ones. This usually leads to a training that is excessively biased towards low-quality predictions. In order to solve this issue, Cascade R-CNN [51] first, and its descendant HTC later, introduced a cascade mechanism where multiple object detectors are trained sequentially in order to take advantage of the previous one and to increase the prediction quality gradually. This means that each stage performs two tasks: first, the detector is training itself, and, then, it is also devoted to identifying the region proposals for subsequent stages. Unfortunately, this also translates into an increase in network complexity in terms of the number of parameters.

In this work, we propose a new way to balance positive samples by exploiting the re-sampling technique, introduced by the cascade models. Our proposed technique generates new proposals with a pre-selected IoU quality in order to equally cover all IoU values. We carry out an extensive ablation study and compare our results with the state of the art in order to demonstrate the advantages of the proposed solution and its applicability to different existing architectures.

The main contributions of this section are the following:

- An effective solution to deal with the EVPS problem with a single-detector model, rebalancing the proposals with respect to the IoU thresholds through a recursive re-sampling mechanism. This mechanism has the goal of eventually feeding the network with an equal distribution of samples.
- An exhaustive ablation study on all the components of our R^3-CNN architecture in order to evaluate how the performance is affected by each component.
- Our R^3-CNN is introduced into major state-of-the-art models to demonstrate that it boosts the performance independently from the baseline model used.

2.2.2 Related Works

Multi-stage Detection/Instance Segmentation. The early works on object detection and instance segmentation were based on the assumption that single-stage end-to-end networks are sufficient to recognize and segment the objects. For instance, YOLO network [52] optimizes localization and classification in one step. Starting with the

R-CNN network [15], the idea of a two-stage architecture was introduced, where, in the first stage, a network called RPN (Region Proposal Network) analyzes the whole image and identifies the regions where the probability of finding an object is high. In the second stage, another network performs a more refined analysis on each single region. After this seminal work, others have further refined this idea. The Cascade R-CNN architecture [51] uses multiple bounding-box heads connected sequentially, where each one refines the proposals produced by the previous one. The minimum IoU required for positive examples is increased at each stage, taking into account a different set of proposals. Other studies [53–55] introduced a similar cascade concept, but applied to the RPN network, where multiple RPNs are sequenced and the results from the previous stage are fed into the next stage. Our work is inspired by HTC network [22], which introduces a particular cascade operation also on the mask extraction modules. However, all these multi-stage networks are quite complex in terms of the number of parameters.

IoU distribution imbalance. Authors in [16] describe the problem as a skewed IoU distribution observed in bounding boxes used in training and evaluation. In [56], the authors highlight the significant imbalance between background and foreground RoI examples and present a hard example mining algorithm to easily select the most significant ones. While in their case the aim is balancing the background (negative) and the foreground (positive) RoIs, in our work the primary goal is to balance RoIs across the entire positive spectrum of the IoU. In [57], an IoU-balanced sampling technique is proposed to mine hard examples. However, the sampling always takes place on the results of the RPN which, as we will see, is not very optimized to provide high-quality RoIs. In our case, we apply re-sampling to the detector itself, which has, on average, a much higher probability of returning more significant RoIs. In [58], the sources of false positives are analyzed and an extra independent classification head is introduced to be plugged into the original architecture to reduce hard false positives. In [59], the authors introduce a new IoU-prediction branch which supports classification and localization. They propose to manually generate samples around ground truths instead of using RPN for localization and IoU prediction branches in training. In [22, 51], overfitting due to EVPS problem for large thresholds is addressed using

multiple detectors connected sequentially. They re-sample the ground truth in a sequential manner to progressively improve hypothesis quality. Unlike them, we tackle the problem with a single detector and a single segmentation head. In [60], they offer an interpretation similar to ours about the fact that IoU imbalance has an adverse effect on performance. However, while they use an algorithm to systematically generate the RoIs with the chosen quality, we rely only on the capabilities of the detector itself.

2.2.3 Recursively Refined R-CNN

In this section, first we briefly introduce the idea behind multi-stage processing. Then we describe our R^3-CNN architecture with its evolution from a sequential to a recursive pipeline, which offers a change of perspective on training.

As shown in Fig. 2.7 (a), the HTC (Hybrid Task Cascade) multi-stage architecture [22] mainly follows the idea that a single detector is unlikely to be able to train uniformly on all quality levels of IoU. The cascade architecture tries to solve the EVPS problem by training multiple regressors connected sequentially, each of which is specialized in a predetermined and growing IoU minimum threshold. Each regressor performs a conversion of its localization results into a new list of proposals for the following regressor. Although this type of architecture clearly improves the overall performance, it also introduces a considerable number of new parameters into the network. In fact, with respect to its predecessor Mask R-CNN, the number of detection and segmentation modules triples. To reduce the complexity of cascade networks and to address the EVPS problem, we design a lighter architecture with single detection and mask heads uniformly trained on all IoU levels. Authors in [51] underlined the cost-sensitive learning problem [61, 62], where the optimization of different IoU thresholds requires various loss functions. Inspired by this study, we address the problem using multiple selective training, which focuses on a specific IoU quality in each step and recursively feeds them into the detector. The intuition is that the detector training and its ability to return an adequate number of proposals of a certain quality level will happen at the same time.

In Fig. 2.7 (b), the new R^3-CNN architecture along with our training paradigm are

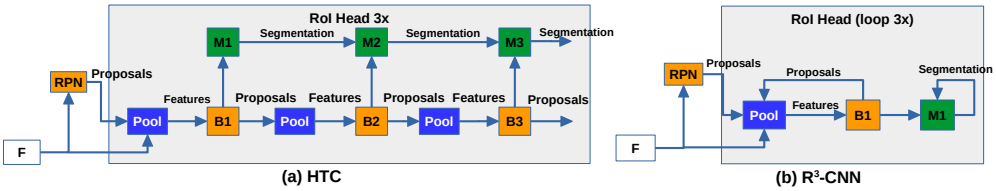


Figure 2.7: Network design. (a) HTC: a multi-stage network which trains each head in a cascade fashion. (b) R^3-CNN: our architecture which introduces a loop mechanism to self-train the heads.

shown. In this loop (recursive) architecture, the detector and the RoI pooling modules are connected in a cycle. As in HTC, the first set of RoI proposals is provided by the RPN. After that, the RoI pooler crops and converts them to fixed-size feature maps, which are used to train the B1 block. Then, with an appropriate IoU threshold, the ground truth re-sampling takes place by the B1 block to generate a new proposal set. The result is then used both in the segmentation module M1 and as the new input for the pooler which closes the loop. By the IoU threshold manipulation, the network can force the detection to extract those RoIs with IoU quality levels which are typically missed. The cycle continues three times (3x loop) to guarantee the rebalancing of RoI levels.

Fig. 2.8 (a) shows the generated RoI distribution for each IoU level in Mask R-CNN as well as the EVPS problem. The distribution of the rebalanced samples by our model, on the other hand, can be seen in Fig. 2.8 (b) and (c). For the latter, it is worth emphasizing some important details emerging from these graphs: (i) Considering only the first loop trend, R^3-CNN looks quite similar to Mask R-CNN; (ii) Conversely, considering the sum of the first two loops, our distribution looks much more balanced; (iii) The third loop significantly increases the number of high-quality RoIs.

Despite the fact that our architecture contains a single detector, its behavior shows a unique and well-defined trend in terms of RoI distribution within different loops. We believe this is the reason why R^3-CNN outperforms Mask R-CNN. It is able to

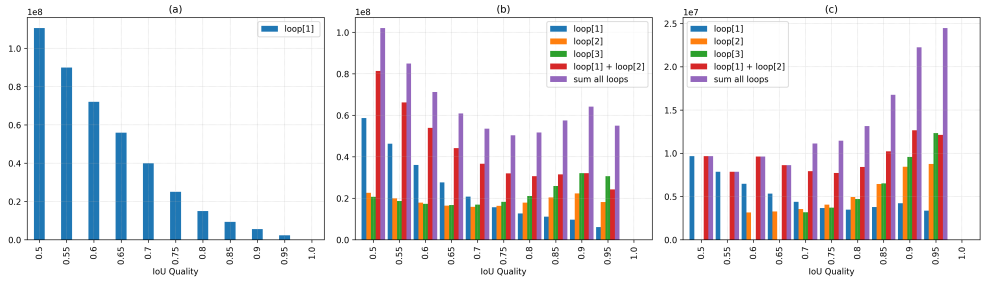


Figure 2.8: The IoU histogram of training samples for Mask R-CNN with a 3x schedule (36 epochs) (a), and R^3-CNN where each loop uses different IoU thresholds [0.5, 0.6, 0.7], decreasingly (b) and increasingly (c). Better seen in color.

mimic the Cascade R-CNN behavior in the RoI distribution (as also shown in [51]), achieved by HTC, but using only a single detector and significantly fewer parameters.

For a given loop t, let us define h as the sole classifier and f as the sole regressor which is trained for a selected IoU threshold u^t, with $u^t > u^{t-1}$, by minimizing the loss function of Cascade R-CNN [51]:

$$L(x^t, g) = L_{cls}\left(h\left(x^t\right), y^t\right) + \lambda\left[y^t \geqslant 1\right] L_{loc}\left(f\left(x^t, b^t\right), g\right) \tag{2.2}$$

where x^t represents the input features of the t-th loop, $b^t = f\left(x^{t-1}, b^{t-1}\right)$ is the new sampled set of proposals coming from the previous loop (with b^0 coming from the RPN), g is the ground truth, and λ is a positive coefficient. y^t represents the label of x^t given the IoU threshold u^t, the proposals b^t and the ground truth label g_y with the following equation:

$$y^t = \begin{cases} g_y & \text{if } IoU\left(b^t, g\right) \geqslant u^t \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

At inference time, the same loop procedure is applied and all the predictions are merged together by computing the mean of the classification values. As it will be shown in the experiments, using loops also at inference (or evaluation) time is not optional, meaning that the loop mechanism is intrinsic to the weights of the network.

2.2.4 Experiments

Dataset and Evaluation Metrics

Dataset. As the majority of recent literature on instance segmentation, we perform our tests on the MS COCO 2017 dataset [4]. The training dataset consists of more than 117,000 images and 80 different classes of objects.

Evaluation Metrics. We used the same evaluation functions offered by the python *pycocotools* software package. All the evaluation phases have been performed on the COCO minival 2017 validation dataset, which contains 5000 images. We report the Average Precision (AP) with different IoU thresholds for both bounding box and segmentation tasks. The main metric (AP) is computed with IoUs from 0.5 to 0.95. Others include AP_{50} and AP_{75} with 0.5 and 0.75 minimum IoU thresholds, and AP_s, AP_m and AP_l for small, medium and large objects, respectively.

Implementation details

To perform a fair comparison, we obtain all the reported results by training the networks with the same hardware and, when possible, the same software configuration. When available, the original code released by the authors or the corresponding implementation in MMDetection [49] framework were used. Our code is also developed within this framework. In the case of HTC, we did not consider the semantic segmentation branch.

We performed a distributed training on 2 servers, each equipped with 2x16 IBM POWER9 cores, 256 GB of memory and 4 x NVIDIA Volta V100 GPUs with Nvlink 2.0 and 16GB of memory. Each training consists of 12 epochs with Stochastic Gradient Descent (SGD) optimization algorithm, an initial learning rate of 0.02, a weight decay of 0.0001, and a momentum of 0.9. The learning rate decays at epochs 8 and 11. We used batch size of 2 for each GPU. We fixed the long edge and short edge of the images to 1333 and 800, maintaining the aspect ratio. ResNet 50 [24] was used as the backbone. If not specified differently, the number of loops in training and evaluation are the same.

Analysis of R^3-CNN

Description. In this part, we demonstrate the potentiality offered by a naive three-stage loop compared to Mask R-CNN and the original three-stage cascade HTC. To have a fair comparison, we select the optimal configuration for the HTC network as baseline and also apply it to training our R^3-CNN. In the *advanced* version, we replace fully-connected layers from detection head with lightweight convolutions with kernel 7×7 and a Non-Local block [28] with incremented kernel size of 7×7 to better exploit information. We also build a brand new branch using only convolutions and Non-Local blocks to include a new learning task to improve segmentation as described in [32]. Since our naive version has slightly fewer parameters than Mask R-CNN, it is also insightful to compare it with our model. Finally, we also want to demonstrate the following important claim: it does not matter the way or order with which the IoU thresholds are changed (either incrementally or decrementally), since in both cases a more balanced IoU distribution is achieved (see Figure 2.8).

Results. In Table 2.6, we report speed in evaluation and memory usage in training, distinguishing between *memory usage* of the entire training process and *model size* (proportional to the number of parameters). Comparing the naive version (row #4) with HTC (row #3), it can be seen that our model has significantly fewer parameters and is more memory efficient. While the segmentation precision (S_{AP}) is practically the same, there is a slight loss in B_{AP}. Also, the speed of naive is slightly better than HTC. Regarding the advanced version (row #6), it surpasses the HTC accuracy in both tasks, while saving a significant number of parameters and using the same amount of memory in training. The only disadvantage is the reduced speed due to Non-Local blocks.

Compared to Mask R-CNN (row #1), the naive R^3-CNN has the same complexity, but achieves a much higher precision in both tasks. To further investigate how well our recursive mechanism works, we also compare it with Mask R-CNN trained with triple number of epochs (row #2). While more training helps Mask R-CNN produce a higher precision, it is still outperformed by naive R^3-CNN. This demonstrates that our loop mechanism is not simply another way of training the network for more epochs, but that it represents a different and more effective training strategy. This can

#	Model	# Params	TS	L_t	H	B_{AP}	S_{AP}	Speed	Mem. usage	Model size
1	Mask (1x)	44,170 K	-	1	1	38.2	34.7	11.5	4.4 GB	339 MB
2	Mask (3x)	44,170 K	-	1	1	39.2	35.5	5.4	4.4 GB	339 MB
3	HTC	77,230 K	Inc	3	3	41.7	36.9	5.4	6.8 GB	591 MB
4	R^3-CNN (naive)	43,912 K	Inc	3	1	40.9	36.8	5.5	5.9 GB	337 MB
5	R^3-CNN (naive)	43,912 K	Dec	3	1	40.4	36.7	5.5	5.9 GB	337 MB
6	R^3-CNN (advanced)	50,072 K	Inc	3	1	42.0	38.2	1.0	6.8 GB	384 MB

Table 2.6: Comparing trainable parameters with the bounding box and segmentation average precision. K: thousand. Column L_t: number of stages. Speed is image per second. TS: Training strategies. Inc: progressively increasing and Dec: decreasing IoU quality through loops.

be explained by the fact that while in Mask R-CNN the RoI proposals are always provided by the RPN, in our case they are provided by the detection head which generates higher quality and more balanced RoIs (see Figure 2.8).

Finally, to show that the order of changes in IoU threshold is not crucial to performance, in rows #4 and #5, we report a comparison between increasing and decreasing IoU thresholds through loops. Although there is a slight degradation of precision using the decreasing training strategy, it is almost negligible due to a more balanced IoU distribution achieved in both cases, but skewed to high-quality RoIs in the first case and low-quality in the latter.

Ablation study on the evaluation phase

Description. In this subsection we focus on how the results are affected by the number of cycles in the evaluation phase. We consider the naive version mentioned above as pre-trained model, which consists of three loops in the training and evaluation phases.

Results. From Table 2.7, it is evident that the loop mechanism is of paramount importance for the evaluation phase too. In fact, when we train the network with the 3x loops and then evaluate it with one loop, it performs even worse than Mask R-CNN

(row #2). On the contrary, with two loops, the result is significantly better (row #3). It also underperforms the three-loop evaluation only slightly (row #4). Therefore, this version could be considered a good compromise between execution time and detection quality. From four loops onward, the performance tends to remain almost stable. This is consistent with our initial hypothesis of a link between evaluation and the loop mechanism, and confirms that in order to have higher performance with more than three loops in the evaluation, we also need to increase the number of loops in the training phase.

Ablation study on the training phase

Description. In this experiment, the network is trained with a number of loops varying from 1 to 4. The number of loops for the evaluation changes accordingly.
Results. The comparative results are reported in Table 2.8. The precision of the single loop (row #2) is comparable to Mask R-CNN, and not much different from the above-mentioned model with one-loop evaluation (row #2 of Table 2.7). This connection with the previous experiment suggests that the detector is strictly optimized on the corresponding sample distribution. The performance is improved by the training strategies with two and three loops, though significantly by the former and only slightly over that by the latter. Regarding more than 3 loops (row #5), the improvement is negligible.

Extensions on R^3-CNN

Description. Our final experiments show that R^3-CNN model can be plugged in seamlessly to several state-of-the-art architectures for instance segmentation, consistently improving their performance, which demonstrates its generalizability. In this experiment, we select our best-performing version previously called *advanced* (see Table 2.6 row #6) and renamed R^3-CNN-L model. The experiment tested different state-of-the-art models, namely GRoIE [1], GC-net [30], DCN [63], with and without the R^3-CNN-L version. When compatible, we also merged the original model with HTC as baseline. For example, *HTC+GC-Net* is composed of both HTC and

#	Model	L_t	H	L_e	B_{AP}	S_{AP}
1	Mask	1	1	1	38.2	34.7
2	R^3-CNN	3	1	1	37.9	35.4
3		3	1	2	40.5	36.6
4		3	1	3	40.9	36.8
5		3	1	4	40.9	36.7
6		3	1	5	40.9	36.7

Table 2.7: Impact of evaluation loops L_e in a 3-loop and one-head-per-type R^3-CNN model. Row #4 is the naive R^3-CNN in Table 2.6.

#	Model	L_t	H	B_{AP}	S_{AP}
1	Mask	1	1	38.2	34.7
2	R^3-CNN	1	1	37.7	34.7
3		2	1	40.4	36.4
4		3	1	40.9	36.8
5		4	1	40.9	36.8

Table 2.8: Impact of the number of training loops in a one-head-per-type R^3-CNN model. Row #4 is the naive R^3-CNN in Table 2.6.

GC-Net merged together.

Results. The results are summarized in Table 2.9, where best results for each comparison are reported in bold, while the second best is in red. From the table we can see that almost all the best and second-best scores belong to R^3-CNN architectures, both in object detection and instance segmentation. However, there are two cases in which this does not happen. The first one is the combination of GC-Net and HTC which outperforms R^3-CNN-L in AP_m by 0.6% (row #7), and the other one is the combination of DCN and HTC which outperforms R^3-CNN-L in AP_l by the same amount (row #10). Apart from these rare cases, these experiments confirm that the proposed R^3-CNN consistently brings benefits to existing object detection and instance segmentation models, in terms of both precision and reduced number of parameters.

2.2.5 Conclusions

In this section, we introduced the R^3-CNN architecture to address the issue of exponentially vanishing positive samples in training by rebalancing the training proposals with respect to the IoU thresholds, through a recursive re-sampling mechanism in a single detector architecture. We demonstrated that a good training needs to take into

#	Method	Backbone	Bounding Box (object detection)						Mask (instance segmentation)					
			AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
1	Mask	r50-FPN	37.3	58.9	40.4	21.7	41.1	48.2	34.1	55.5	36.1	18.0	37.6	46.7
2	HTC	r50-FPN	41.7	60.4	45.2	24.0	44.8	54.7	36.9	57.6	39.9	19.8	39.8	50.1
3	R^3-CNN-L	r50-FPN	**42.0**	**61.0**	**46.3**	**24.5**	**45.2**	**55.7**	**38.2**	**58.0**	**41.4**	**20.4**	**41.0**	**52.8**
4	GRoIE	r50-FPN	38.6	59.4	42.1	22.5	42.0	50.5	35.8	56.5	38.4	19.2	39.0	48.7
5	R^3-CNN-L+GRoIE	r50-FPN	**42.0**	**61.2**	**45.6**	**24.4**	**45.2**	**55.7**	**39.1**	**58.8**	**42.3**	**20.7**	**42.1**	**54.3**
6	GC-Net	r50-FPN	40.5	62.0	44.0	23.8	44.4	52.7	36.4	58.7	38.5	19.7	40.2	49.1
7	HTC+GC-Net	r50-FPN	43.9	63.1	47.7	26.2	**47.7**	57.6	38.7	60.4	41.7	21.6	42.2	52.5
8	R^3-CNN-L+GC-Net	r50-FPN	**44.3**	**64.1**	**48.4**	**27.0**	47.1	**58.9**	**40.2**	**61.1**	**43.5**	**22.6**	**42.8**	**56.0**
9	DCN	r50-FPN	41.9	62.9	45.9	24.2	45.5	55.5	37.6	60.0	40.0	20.2	40.8	51.6
10	HTC+DCN	r50-FPN	44.7	63.8	48.6	26.5	48.2	**60.2**	39.4	61.2	42.3	21.9	42.7	54.9
11	R^3-CNN-L+DCN	r50-FPN	**44.8**	**64.3**	**48.9**	**26.6**	**48.3**	59.6	**40.4**	**61.3**	**44.0**	**22.3**	**43.6**	**56.1**

Table 2.9: Performance of the state-of-the-art models with and without R^3-CNN model. Bold values are best results, red ones are second-best values.

account the diversity of IoU quality of the RoIs used to learn, more than aiming to have only high quality RoIs. Our extensive set of experiments and ablation studies provide a comprehensive understanding of the benefits and limitations of the proposed models. R^3-CNN offers a good flexibility to use intermediate versions between the naive version and HTC, permitting to play with the number of loops, depending if we privilege precision, number of parameters or speed. Overall, the proposed R^3-CNN architecture demonstrates its usefulness when used in conjunction with several state-of-the-art models, achieving considerable improvements over the existing models.

2.3 Self-Balanced R-CNN for Instance Segmentation

We do not free ourselves from something by avoiding it, but only by living though it.

Cesare Pavese

2.3.1 Introduction

Nowadays, instance segmentation is one of the most studied topics in the computer vision community, because it reflects one of the key problems for many of the existing applications where we have to deal with many heterogeneous objectives inside an image. It offers, as output, the localization and segmentation of a number of instances not defined a priori, each of them belonging to a list of classes. This task is important for several applications, including medical diagnostics [8], autonomous driving [10], alarm systems [11], agriculture optimization [12], visual product search [13], and many others.

Most of the recent models descend from the two-stage architecture called Mask R-CNN [14]. The first stage is devoted to the search of interesting regions independently from the class, while the second is used to perform classification, localization and segmentation on each of them. This divide-and-conquer approach was first introduced in the ancestor network called Region-based CNN (R-CNN) [15], which has evolved in several successive architectures. Although it achieved excellent results, several studies [16], [17], [18] have recently discovered some of its critical issues which can limit its potentiality. These issues have not been solved yet and several blocks of these architectures are still under-explored and far from optimized and well understood.

This section approaches mainly two of the imbalance problems mentioned in [16]. The first problem, called IoU Distribution Imbalance (IDI), arises when the positive Regions of Interest (RoIs) proposals provided by the RPN during the training of the detection and segmentation heads have an imbalanced distribution. Due to

some intrinsic problems of the anchor system, the number of available RoIs decreases exponentially with the increase of the IoU threshold, which leads the network to easily overfit to low quality proposals. Our work extends the analysis on R^3-CNN, first introduced in [2], to understand architectural limits and proposes advanced configurations in between and an architectural improvement for the segmentation head.

The second problem, called Feature Level Imbalance (FLI), arises when the features are selected from the Feature Pyramid Network (FPN) for their localization and segmentation. As highlighted in [16], the hierarchical structure of FPN (originally designed to provide multi-scale features) does not provide a good integration between low- and high-level features among different layers. To address this problem, the classical approach is to balance the information before the FPN. On the contrary, our work enhances the GRoIE [1] architecture and puts forward a more effective solution, fusing information from all the FPN layers.

In addition, we address the common problem of the explosion of the number of parameters, due to the introduction of new components or expansion of existing ones (e.g. [19]). The increased complexity leads to an increase in the search space for optimization during the training, and, in turn, negatively impacts the generalization capability of the network. Moreover, our empirical results support the intuition made by [20] about the connection between the task to solve and the utilized layers, extending their work toward a fully convolutional solution.

To summarize, this section has the following main contributions:

- An extensive analysis of the IDI problem in the RPN generated proposals, which we treat with a single- and double-head loop architecture (R^3-CNN) between the detection head and the RoI extractor, and a brand-new internal loop for the segmentation head itself.
- Redesign of the model heads (FCC) toward a fully convolutional approach, with empirical analysis that supports some architectural preferences depending on the task.
- A better performing GRoIE model is proposed for extraction of RoIs in a two-stage instance segmentation and object detection architecture.

- An exhaustive ablation study on all the components.
- The proposal of SBR-CNN, a new architecture composed of R^3-CNN, FCC and GRoIE, which maintains its qualities if plugged into major state-of-the-art models.

2.3.2 Related Works

Multi-stage Detection/Instance Segmentation. Single-stage and two-stage architectures for object detection have been researched for several years. For instance, YOLO network proposed in [52] optimizes localization and classification in one step and [64] proposes a single-shot network which uses bounding box regression. Since the single-stage architectures do not always provide acceptable performance and require a lot of memory in applications with thousands of classes, a region-based recognition method was proposed [15], where first part processes input images, while the second part processes bounding boxes found by the previous one. This approach has been used in the Mask R-CNN architecture [14], obtained by adding a segmentation branch to the Faster R-CNN [25]. This idea has been refined by several studies. For instance, [65] provides a composite backbone network in a cascade fashion. The Cascade R-CNN architecture [19] puts forward the utilization of multiple bounding box heads, which are sequentially connected, refining predictions at each stage. In [53–55], they introduced a similar cascade concept but applied to the RPN network. In addition, the Hybrid Task Cascade (HTC) network [22], by which this work is inspired, applies cascade operation on the mask head as well. Our work pushes in the same direction but changes the paradigm from cascade to loop, where the single neural network block is trained to perform more than one function by applying different conditioning in the input.

IoU distribution imbalance. A two-stage network uses the first stage to produce a set of bounding box proposals for the following stage, filtering positive ones through a threshold applied to the IoU between them and the ground truth. The IoU distribution imbalance problem is described as a skewed IoU distribution [16] that is seen in bounding boxes which are utilized in training and evaluation. In [56], the authors

propose a hard example mining algorithm to select the most significant RoIs to deal with background/foreground imbalance. Their work differs from ours because our primary goal is to balance the RoIs across the positive spectrum of the IoU. In [57], the authors propose an IoU-balanced sampling method which mines the hard examples. The proposed sampling is performed on the results of the RPN which is not very optimized in producing high-quality RoIs as we will see. On the other hand, we apply the resampling on the detector itself, which increases the probability of returning more significant RoIs.

After analyzing the sources of false positives and to reduce them, [58] introduces an extra independent classification head to be attached to the original architecture. In [66], the authors propose a new IoU prediction branch which supports classification and localization. Instead of utilizing RPN for localization and IoU prediction branches in the training phase, they propose manually generating samples around ground truth.

In [19,22,67], they address the exponentially vanishing positive samples problem, utilizing three sequentially connected detectors to improve the hypothesis quality progressively, by resampling the ground truth. It differs from our approach since we deal with the problem using a single detector and a segmentation head. Authors of [60] give an interpretation about the fact that IoU imbalance negatively impacts the performance which is similar to ours. However, differently from us, they designed an algorithm to systematically generate the RoIs with required quality, where we base our work on the capabilities of the detector itself.

Feature-level imbalance. A two-stage network deals with images containing objects of any size with the help of an FPN attached to the backbone. How the RoI extraction layer combines the information provided by the FPN is of paramount importance to embody the highest amount of useful information. This layer has been used by many derivative models such as Mask R-CNN, Grid [39], Cascade R-CNN [21], HTC [22] and GC-net [30]. In [26], the authors apply an RoI pooling to a single and heuristically chosen FPN output layer. However, as underlined by [16], this method is defective due to a problem related to untapped information. Authors of [40] propose to separately extract mask proposals from each scale and rescale them while includ-

ing the results in a unique and multi-scale ranked list, selecting only the best ones. In [41], the authors use a backbone for each image scale, merging them with a max function. On the contrary, we use an FPN which simplifies the network and avoids doubling the network parameters for each scale.

In SharpMask model [42], after making a coarse mask prediction, authors fuse feature layers back in a top-down fashion in order to reach the same size of the input image. Authors of PANet [38] point out that the information is not strictly connected with a single layer of the FPN. By propagating low-level features, they build another structure similar to FPN, coupled with it, combining the images pooled by the RoIs. While our proposed GRoIE layer is inspired by this approach, it differs from that in its size. We propose a novel way to aggregate data from the features pooled by RoIs making the network more lightweight without extra stacks coupled with FPN.

Auto-FPN [43] applies Neural Architecture Search (NAS) to the FPN. PANet has been extended by AugFPN [68]. The module with which we compare our module is called the Soft RoI Selector [68], which includes an RoI pooling layer on each FPN layer to concatenate the results. Then, they are combined using the *Adaptive Spatial Fusion* in order to build a weight map that is fed into 1x1 and 3x3 convolutions sequentially. In our work, we first carry out a distinct convolution operation on each output layer of the FPN network. After that, instead of concatenating, we sum the results since it is potentially more helpful for the network. In the end, we apply an attention layer whose job is to further filter the multi-scale context.

Authors of Multi-Scale Subnet [45] propose an alternative technique to RoI Align which employs cropped and resized branches for RoI extraction at different scales. In order to maintain the same number of outputs for each branch, they utilize convolutions with 1x1 kernel size, performing an average pooling to diminish them to the same size before summing them up. Finally, they use a convolutional layer with 3x3 kernel size as the post-processing stage. In our ablation study, we show that these convolutional configurations to carry out pre- and post-processing are not the optimal ones that can lead to better performance.

The IONet model [46] proposes doing away with any FPN network and, instead, using re-scaled, concatenated, and condensed (dimension-wise) features di-

rectly from the backbone before doing classification and regression. Finally, Hypercolumn [47] utilizes a hypercolumn representation to classify a pixel, with 1x1 convolutions and up-sampling the results to a common size so that they can be summed. Here, the absence of an optimized RoI pooling solution and an FPN layer and the simple processing of columns of pixels that have been taken from various stages of the backbone can be a limitation. In fact, we show in our ablation study that the adjacent pixels are necessary for optimal information extraction.

In [69] they avoid to select the FPN layer and then RoI crop the features, attaching a convolutional branch on top of the last FPN layer and conditioning on the instance. In our case, we avoid the risk to loose information in intermediate FPN layers, leaving to the network the job of conditionally merging them for each instance.

2.3.3 Recursively Refined R-CNN (R^3-CNN)

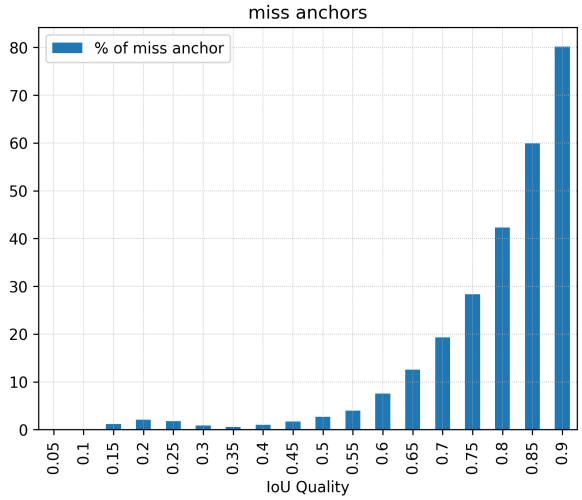


Figure 2.9: Percentage of times in which, during the RPN training, there does not exist an anchor with a certain value of IoU w.r.t. the ground-truth bounding boxes.

In a typical two-stage network for instance segmentation, to obtain a good train-

ing of the network, we need as good candidates as possible from the RPN. We could highlight at least two problems which are parts of so called IoU Distribution Imbalance (IDI) that afflict the training. The first one, shown in Fig. 2.9, is related to the anchor system. It is called Exponential Vanishing Ground-Truth (EVGT) problem, where the higher the IoU threshold to label positive anchors is, the exponentially higher the percentage of missed ground-truth bounding boxes (gt-bboxes) can be. For instance, more than 80% of the gt-bboxes do not have a corresponding anchor with an IoU (w.r.t. the gt-bbox) between 0.85 and 0.9. Since, for every image, the anchors' maximum IoU varies from one gt-bbox to another, if we choose a too high IoU threshold, some of the objects could be completely ignored during the training, reducing the number of truly used annotations. For example, if the gt-bbox is in an unfortunate place where the maximum IoU between that and all available anchors is 0.55 and we choose a minimum threshold of 0.6, then no anchors will be associated with that object and it will be seen as part of the background during the training. That is why we are usually obliged to use a very low threshold (typically 0.3 as a limit), since otherwise we could run into a case where a consistent part of the ground-truth is ignored. The second, called Exponentially Vanishing Positive Samples (EVPS) problem [19], is partially connected with the first one because training the RPN with a too low threshold will reflect the low quality issue on its proposals. Even in the best case, where each gt-bbox has the number of positive anchors greater than zero, the number of proposals from the RPN still diminishes exponentially with the increase of the required IoU threshold (see Fig. 2.11a).

Fig. 2.10(a) shows the Hybrid Task Cascade (HTC) model [22], greatly inspired by Cascade R-CNN network [19], which trains multiple regressors connected sequentially, each of which is specialized in a predetermined and growing IoU minimum threshold. This architecture offers a boost in performance at the cost of the duplicate heads, three times the ones used in Mask R-CNN.

In order to reduce the complexity, we designed a lighter architecture called Recursively Refined R-CNN (R^3-CNN) (see Fig. 2.10(b)) to address the IDI problem by having single detection and mask heads trained uniformly on all the IoU levels. In [19], it has been pointed out that the cost-sensitive learning problem [61, 62],

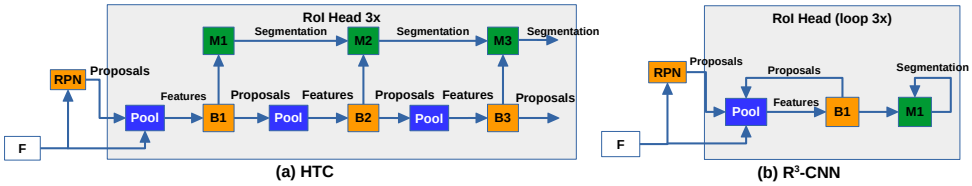


Figure 2.10: Network design. (a) HTC: a multi-stage network which trains each head in a cascade fashion. (b) R^3-CNN: our architecture which introduces two loop mechanisms to self-train the heads.

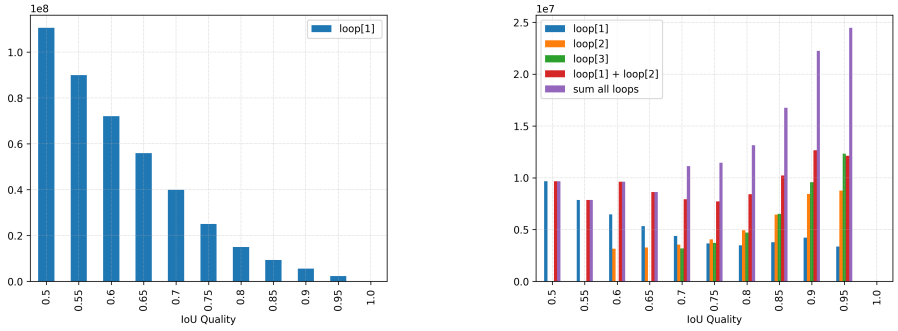
connected with the optimization of multiple IoU thresholds, needs multiple loss functions. This encouraged us to look for a multiple selective training to address the problem. Using a different and uniformly chosen IoU threshold in the range between 0.5 and 0.9 for each loop, we sample a proposal list to feed the detector itself each time with a different IoU quality distribution, relying on RPN only in the initial loop. Furthermore, this new list of proposals is used to feed the segmentation head M1, which incorporates an internal loop to refine the mask.

Fig. 2.11 (a) shows the generated RoI distribution for each IoU level in Mask R-CNN and it highlights the EVPS problem. In Fig. 2.11 (b) and (c), we can see how our model rebalances the sample distribution.

In order to show the rebalancing, during the training we collected the information of IoU of the proposals with the gt-bboxes. To make the comparison fairer, we trained the Mask R-CNN three times the number of epochs. In Fig. 2.11a, we can see that the distribution of IoUs in Mask R-CNN maintains its exponentially decreasing trend.

Shown in Fig. 2.11b, R^3-CNN presents a well-defined IoU distribution for each loop. With two loops, we already have a more balanced trend and, by summing the third, the slope starts to invert. The same trend can be observed in Cascade R-CNN, where the IoU histogram of the n^{th} stage of Cascade R-CNN (as shown in [19] - Fig. 4) can be compared with the n^{th} loop of R^3-CNN.

Let us now define how the detection head loss (see Fig. 2.10(b)) is composed, followed by the definition of the loss of the mask head. For a given loop t, let us



(a) Mask R-CNN trained 36 epochs.

(b) R^3-CNN

Figure 2.11: The IoU distribution of training samples for Mask R-CNN with a 3x schedule (36 epochs) (a), and R^3-CNN where at each loop it uses a different IoU threshold [0.5, 0.6, 0.7] (b). Better seen in color.

define B1 as the detection head, composed of h as the classifier and f as the regressor, which are trained for a selected IoU threshold u^t, with $u^t > u^{t-1}$. Let $\mathbf{x^t}$ represent the extracted features from the input features $\mathbf{x}$ using the proposals $\mathbf{b^t}$. In the first loop, the initial set of proposals ($\mathbf{b^0}$) comes from the RPN. For the rest of the loops, in loop t, we have a set of N_P proposals $\mathbf{b^t} = \left\{b_1^t, b_2^t, \ldots, b_{N_P}^t\right\}$ obtained by the regressor f using the extracted features $\mathbf{x^{t-1}}$ and the set of proposals $\mathbf{b^{t-1}}$ from the previous loop.

A given proposal $b_i^t \in \mathbf{b^t}$ is compared with all the N_{GT} gt-bboxes $\mathbf{g} = \{g_1, g_2, \ldots, g_{N_{GT}}\}$ by computing their overlap through the IoU. If none of these comparisons results in an IoU greater than the selected threshold u^t for the current loop, the label $y_i^t = 0$ corresponding to the class "background" is assigned to b_i^t. Otherwise, the label l_x corresponding to the class of the gt-bbox g_x with the maximum IoU is assigned to y_i^t:

$$l_x = \arg\max_{l_{\bar{x}}} IoU\left(b_i^t, g_{\bar{x}}\right) \quad \forall g_{\bar{x}} \in \mathbf{g} | IoU\left(b_i^t, g_{\bar{x}}\right) \geqslant u^t \tag{2.4}$$

where $l_{\bar{x}}$ is the label assigned to $g_{\bar{x}}$. The detection head loss for the loop t can be

computed similarly as in Cascade R-CNN [19]:

$$L^t_{bbox}\left(\mathbf{x^t},\mathbf{g}\right) = \begin{cases} L_{cls}\left(h\left(\mathbf{x^t}\right),\mathbf{y^t}\right) & \forall b^t_i \,|\, y^t_i = 0 \\ L_{cls}\left(h\left(\mathbf{x^t}\right),\mathbf{y^t}\right) + \lambda L_{loc}\left(f\left(\mathbf{x^t},\mathbf{b^t}\right),\mathbf{g}\right) & \text{otherwise} \end{cases} \tag{2.5}$$

where $\mathbf{y^t}$ is the set of labels assigned to the proposals $\mathbf{b^t}$ and λ is a positive coefficient. The classification loss L_{cls} is a multi-class cross entropy loss. If y^t_i is not zero, the localization loss L_{loc} is also used, which is computed with a smooth L1 loss.

Regarding the segmentation branch performed by the M1 mask head, a separate RoI extraction module is employed to obtain the features $\mathbf{x^t}$ for the proposals $\mathbf{b^t}$ provided by the B1 detection head. Similar to HTC, but with a single mask head, R^3-CNN uses an internal loop of j iterations, with $j = t$, meaning that in the first loop of R^3-CNN, a single iteration ($j = 1$) is performed, then two iterations in the second loop, and so forth. At each internal iteration, the mask head receives as input the features $\mathbf{x^t}$ summed with the result of a 1×1 convolution C_1 applied to the output of the previous internal iteration:

$$\begin{aligned} \mathbf{m^0} &= M1\left(\mathbf{x^t} + C_1(\mathbf{0})\right) \\ \mathbf{m^1} &= M1\left(\mathbf{x^t} + C_1(\mathbf{m^0})\right) \\ &\ldots \\ \mathbf{m^{j-1}} &= M1\left(\mathbf{x^t} + C_1(\mathbf{m^{j-2}})\right) \end{aligned} \tag{2.6}$$

where C_1 is applied to a null tensor $\mathbf{0}$ at the first loop, and to the output of the previous iteration for the subsequent. With this mechanism, the network iteratively refines its segmentation output.

The final output $\mathbf{m^{j-1}}$ of the internal loop is then upsampled with U to reshape its size from 14×14 to 28×28. Finally, another 1×1 convolution C_2 is applied in order to reduce the number of channels to the number of classes:

$$\mathbf{m^j} = C_2\left(U\left(\mathbf{m^{j-1}}\right)\right) \tag{2.7}$$

The loss function for the segmentation L^t_{mask} is computed over $\mathbf{m^j}$ as follows:

$$L^t_{mask} = BCE\left(\mathbf{m^j}, \hat{\mathbf{m}}\right) \tag{2.8}$$

where $\hat{\mathbf{m}}$ represents the segmentation of the ground-truth object and BCE is the binary cross entropy loss function.

In the end, the total loss for loop t is composed as the sum of previous losses:

$$L^t = \alpha_t \left(L^t_{bbox} + L^t_{mask}\right) \tag{2.9}$$

where α_t represents a hyper-parameter defined statically in order to weight the different contributions of each loop.

We maintain the loop mechanism also at inference time and, at the end, we merge all the predictions, computing the average of the classification predictions.

2.3.4 Fully Connected Channels (FCC)

In order to further reduce the network size, we propose to replace fully connected (FC) layers with convolutions. In R^3-CNN model, they are included in two modules: in the detection head and in the Mask IoU branch [32], which learns a quality score for each mask.

In the detection head, the first two FC layers are shared between the localization and the classification tasks, followed by one smaller FC layer for each branch (see Fig. 2.12(a)). Our goal is to replace the first two shared FC layers, which contain most of the weights, with convolutional layers, in order to obtain a lighter network (see Fig. 2.12(b)). With the term *L2C* we will refer, hereinafter, to these two convolutional layers together. The input feature map has the shape of $n \times channels \times 7 \times 7$ (n is the number of proposals), characterized by a very small width and height. A similar problem, addressed by [1], demonstrates how performance improves as the kernel size increases, covering almost the entire features shape. We chose a large kernel size of 7×7 with padding 3 in order to maintain the input shape, halving the number of channels in input. So, the first layer has 256 channels in input and 128 in output, while the second one reduces channels from 128 to 64.

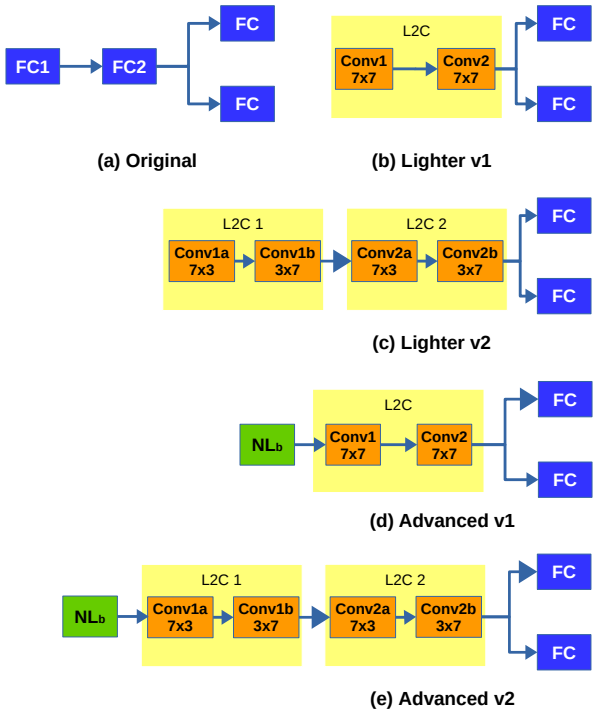


Figure 2.12: (a) Original HTC detector head. (b) Our lighter detector using convolutions with 7×7 kernels. (c) Evolution of (b) with rectangular convolutions. (d) Evolution of (b) with non-local pre-processing block. (e) Evolution of (c) with non-local pre-processing block.

Name	# Params	Description	
FC 1	12,846,080	256×7×7×1024 (W)	+ 1024 (b)
L2C (conv1)	1,605,760	256×7×7×128 (W)	+ 128 (b)
L2C (conv1a)	1,376,512	256×7×3×256 (W)	+ 256 (b)
L2C (conv1b)	688,256	256×3×7×128 (W)	+ 128 (b)
FC 2	1,049,600	1024×1024 (W)	+ 1024 (b)
L2C (conv2)	401,472	128×7×7×64 (W)	+ 64 (b)
L2C (conv2a)	344,192	128×7×3×128 (W)	+ 128 (b)
L2C (conv2b)	172,096	128×3×7×64 (W)	+ 64 (b)

Table 2.10: Parameter count for FC & L2C with 7×7 and rectangular kernels. W: weights; b: bias.

Fig. 2.12(c) shows an alternative version which substitutes each convolution with two of them but with a small kernel (7×3 with padding 3, and 3×7 with padding 1), with the aim of increasing the average precision and execution time. Table 2.10 shows the sharp reduction obtained by the introduction of both *L2C* versions.

A heavier version of FCC includes also one non-local layer before the convolutions (see Fig. 2.12(d) and (e)). Our non-local layer, differently from the original one [28], increases the kernels of internal convolutions from 1×1 to 7×7, in order to better exploit the information that is flowing inside the features in input. The disadvantage of increased execution time could be alleviated in future versions, for instance, by using depth-wise convolutions [70] or similar mechanisms.

In terms of number of parameters, FCC architectures reduces them from 14M to 2.2M, 2.8M, 8.6M, and 9.2M if we use versions *b*, *c*, *d*, and *e*, respectively.

These changes in the architecture have been considered also for the Mask IoU module, which is composed of four convolutional layers followed by three FC layers. Also in this case, the first two FC layers have been replaced, achieving the following weight reduction: from 16.3M to 4.6M, 5.1M, 10.6M and 11.1M with version *b*, *c*, *d* and *e*, respectively.

As previously noticed by [20], the architecture is influenced by the task that it tries to solve. In our case, we observed that convolutions can successfully substitute

FC layers in all cases. But, if the task involves a classification, a mechanism to preserve spatial sensitivity information is needed (with an enhanced non-local module). Conversely, when the network learns a regression task, as for the Mask IoU branch, an attention module is not needed.

2.3.5 Generic RoI Extraction Layer (GRoIE)

The FPN is a commonly used architecture for extracting features from different image resolutions without separately elaborating each scale. In a two-stage detection framework similar to one mentioned in this section, the output layer of an FPN network is chosen heuristically as a unique source of sequential RoI pooling. However, while the formula has been designed very well, it is obvious that the layer is selected arbitrarily. Furthermore, the mere combination of the layers that are provided by the backbone can result in a non-uniform distribution of low- and high-level information in the FPN layers [16]. This phenomenon necessitates finding a way to avoid losing information by selecting only one of them as well as correctly combining them in order to obtain a re-balanced distribution. The enhancement obtained from the GRoIE [1] (Fig. 2.13) suggests that if all the layers are aggregated appropriately through some extra convolutional filters, it is more likely to produce higher quality features. The goal is to solve the feature imbalance problem of FPN by considering all the layers, leaving the task of learning the best way of aggregating them to the network. To summarize, given a proposed region, a fixed-size RoI is pooled from each FPN layer. Then, the n resulting feature maps are pre-processed separately and summed together to form a single feature map. In the end, after a post-processing stage, global information is extracted. This architecture guarantees an equal contribution of all scales, benefiting from the embodied information in all FPN layers and overcoming the limitations of choosing an arbitrary FPN layer. This procedure can be applied to both object detection and instance segmentation. Our work focused on even improving the GRoIE model and evaluating new building blocks for the pre- and the post-processing stages. In particular, as we did for the FCC, we tested bigger and rectangular kernels for the convolutional layers, to better exploit the close correlation between neighboring features. The advantage to involve near features is even more evident when applied to a

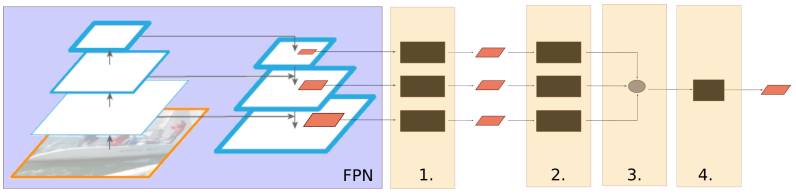


Figure 2.13: GRoIE framework. (1) RoI Pooler. (2) Pre-processing phase. (3) Aggregation function. (4) Post-processing phase.

more sophisticated non-local module, which includes an attention mechanism. However, as we will see in the ablation study, it is extremely important to do it in the right point of the chain.

2.3.6 Experiments

This section reports the extensive experiments carried out to demonstrate the effectiveness of the proposed architecture. After introducing the dataset, the evaluation metrics, the implementation details and the table legend, the following subsections report the results on the three main novelties of the architecture, namely the Recursively Refined R-CNN (R^3-CNN), the Fully Connected Channels (FCC) and the Generic RoI Extraction layer (GRoIE). Finally, the last subsection shows how all these novelties together can bring performance benefits to several state-of-the-art instance segmentation architectures.

Dataset and Evaluation Metrics

Dataset. As the majority of recent literature on instance segmentation, we perform our tests on the MS COCO 2017 dataset [4]. The training dataset consists of more than 117,000 images and 80 different classes of objects.

Evaluation Metrics. We used the same evaluation functions offered by the python *pycocotools* software package, performed on the COCO minival 2017 validation dataset, which contains 5000 images.

We report the mean Average Precision (AP) for both bounding box (B_{AP}) and segmentation (S_{AP}) tasks. The primary metric AP is computed as average over results with IoU thresholds from 0.5 to 0.95. Other metrics include AP_{50} and AP_{75} with 0.5 and 0.75 minimum IoU thresholds, respectively. Separate metrics are calculated for small (AP_s), medium (AP_m) and large (AP_l) objects.

Implementation Details

In order to perform a fair comparison, we use same the hardware and software configuration to carry out the experiments. When available, the original code released by the authors was used. Otherwise, we used the corresponding implementations in MMDetection [49] framework. In the case of HTC, we do not consider the semantic segmentation branch.

Unless mentioned otherwise, the following configuration has been used. We performed a distributed training on 2 servers, each one equipped with 2x16 IBM POWER9 cores and 256 GB of memory plus 4 x NVIDIA Volta V100 GPUs with Nvlink 2.0 and 16GB of memory. Each training consists of 12 epochs with Stochastic Gradient Descent (SGD) optimization algorithm, batch size 2 for each GPU, an initial learning rate of 0.02, a weight decay of 0.0001, and a momentum of 0.9. The steps to decay the learning rate was set at epochs 8 and 11. Regarding the images, we fixed the long edge and short edge of the images to 1333 and 800, maintaining the aspect ratio. ResNet50 [24] was used as the backbone.

Table Legend

To ease the understanding of the following tables, we shortly introduce the notation used. Since R^3-CNN has the loop both in training and evaluation phase, we denote the number of training and evaluation loops as L_t and L_e, respectively. Whenever only L_t is reported, L_e is intended to have the same value of L_t. In the case of HTC, L_t corresponds to the number of stages.

The column H (heads) specify how many pairs of detection (B) and mask (M) heads are included. In the case of multiple pairs ($H > 1$), the column Alt. (alternation)

#	Model	# Params	L_t	H	B_{AP}	S_{AP}	Mem	Model	Speed
1	Mask (1x)	44,170 K	1	1	38.2	34.7	4.4G	339M	11.6
2	Mask (3x)	44,170 K	1	1	39.2	35.5	4.4G	339M	11.6
3	HTC	77,230 K	3	3	41.7	36.9	6.8G	591M	3.3
4	R^3-CNN (naive)	43,912 K	3	1	40.9	37.2	6.7G	337M	3.4
5	R^3-CNN (deeper)	60,604 K	3	2	41.8	37.5	7.0G	464M	3.4

Table 2.11: Comparison between R^3-CNN, Mask R-CNN, and HTC. Column *Model* contains the number of parameters (millions). $3x$ means training with 36 epochs.

gives information about which one is used for each loop. For example, in row #3 of Table 2.14, the model is using three loops for training and evaluation, and two pairs of B and M. The column *Alt* reports "abb", meaning that B1 and M1 are used only in the first loop, while B2 and M2 are used for the second and third loops.

The columns M_{IoU}, $L2C$, NL_b, NL_a are flags indicating the presence of the Mask IoU branch with the associated loss, the substitution of the FC layers with convolutions ($L2C$), inside the detection head (in Table 2.15) or Mask IoU branch (in Table 2.16), and finally, the introduction of our non-local blocks with kernels 7×7 before (NL_b) and after (NL_a) the $L2C$ convolutions. The column Speed refers to the number of processed images per second on evaluation phase with batch size equal to one and one GPU.

Results for Recursively Refined R-CNN (R^3-CNN)

Preliminary analysis of R^3-CNN

Description. We compared Mask R-CNN and HTC with two R^3-CNN models: naive (one pair of bounding box B and mask M) and deeper (two pairs, with the alternation *aab*). To carry out a fair comparison, the Mask R-CNN was trained with 36 epochs instead of 12, and the optimal configuration for the HTC network was used.

Results. The *naive* version has the biggest reduction in terms of the number of pa-

rameters, loosing 0.8 in B_{AP} but gaining 0.3 in S_{AP} compared to HTC. Regarding the *deeper* version, it matches the HTC in B_{AP} and further increases the gap in S_{AP}, while still saving a considerable number of parameters. Both of them require the same amount of memory, as well as the inference time as HTC. This is due to the fact that the training procedure and the utilized components are very similar to those of HTC.

Compared to Mask R-CNN, our R^3-CNN (in both versions) outperforms it, even when Mask R-CNN is trained for a triple number of epochs (row #2). This can be explained by the very different way of training the network, helping to achieve a higher quality of the bounding boxes during the training. Moreover, the training phase for R^3-CNN is faster than Mask R-CNN (about 25 hours versus 35 hours), although R^3-CNN has the disadvantage of requiring the loop mechanism also in the evaluation phase.

Ablation study on the training phase

Description. In these experiments, the network is trained with a number of loops varying from 1 to 4. The number of loops for the evaluation changes accordingly. The basic architecture for all the tests in these experiments is the naive R^3-CNN with single pair of detection and mask heads. It means that all the R^3-CNN models have the same number of parameters but they are trained more if the number of loop increases.

Results. The results are reported in Table 2.12. Using a single loop (row #2) not only

#	Model	L_t	H	B_{AP}	S_{AP}
1	Mask	1	1	38.2	34.7
2	R^3-CNN	1	1	37.6	34.6
3		2	1	40.4	36.7
4		**3**	**1**	**40.9**	**37.2**
5		4	1	40.9	37.4

Table 2.12: Impact of the number of training loops in a R^3-CNN. Row #4 is the naive R^3-CNN in Table 2.11.

produces a similar IoU distribution to Mask R-CNN as mentioned in Section 2.3.3, but also leads to a similar performance. With two loops (row #3), we can reach almost the peak performance of R^3-CNN thanks to the rebalancing of IoU, surpassing the performance of Mask R-CNN. In the case of three loops, the network provides more high-quality proposals, reaching even better performance on both tasks. Adding four loops for training does not improve object detection task but still improves segmentation.

Ablation study on the evaluation phase

Description. In this experiment we focus on how the results are affected by the number of loops in the evaluation phase. We consider the *naive* architecture mentioned above as the pre-trained model and we vary the number of evaluation loops.

Results. From Table 2.13, we observe that we can not avoid to use the loop in the evaluation phase, because it plays the role to provide high quality RoIs to the network. Though, already with two loops the AP values are significantly better (row #3). From four loops onward, the performance tends to remain almost stable in both detection and segmentation tasks.

Ablation study on a two-heads-per-type model

#	Model	L_t	H	L_e	B_{AP}	S_{AP}
1	Mask	1	1	1	38.2	34.7
2		3	1	1	37.7	35.1
3		3	1	2	40.5	36.9
4	R^3-CNN	**3**	**1**	**3**	**40.9**	**37.2**
5		3	1	4	40.8	37.2
6		3	1	5	40.9	37.3

Table 2.13: Impact of evaluation loops L_e in a three-loop and one-head-per-type R^3-CNN model. Row #4 is the naive R^3-CNN in Table 2.11.

#	Model	L_t	H	Alt.	B_{AP}	S_{AP}
1	HTC	3	3	abc	41.7	36.9
2	R^3-CNN	2	2	ab	40.9	36.5
3		3	2	abb	41.8	37.2
4		3	2	aab	41.8	37.5
5		3	2	aba	41.5	37.2
6		**4**	**2**	**aabb**	**42.1**	**37.7**
7		4	2	abab	41.9	37.6
8		5	2	aabbb	41.8	37.5

Table 2.14: The impact of the number of training loops and pair alternation in two-heads-per-type (two pairs B/M) in the R^3-CNN.

Description. In this experiment, we evaluate the performance on changing the number of loops and the alternation between the pairs of heads in the architecture. It is worth emphasizing that increasing the number of loops does not change the number of weights.

Results. Table 2.14 reports the results. In case of two loops (row #2), the model shows good precision, but still worse than HTC. With three loops and *aab* alternation (row #4), R^3-CNN surpasses HTC in both task.

With four loops (rows #6 and #7), the performances are all higher than HTC, especially for *aabb* alternation (row #6). Finally, with five (row #8) loops the performance is not increasing anymore.

Results for Fully Connected Channels (FCC)

Ablation study on the Detection Head

Description. In this section, we evaluate the effect of the head redesign toward a fully convolutional approach. We tested both *L2C* versions (see Fig. 2.12 (b-d) and

Fig. 2.12(c-e) in orange) and the introduction of the non-local layer with larger kernels before (column NL_b) the *L2C* convolutions (see Figure 2.12(d) and (e)) and, to have a more complete ablation study, also after them (column NL_a). In order to provide a more comprehensive analysis, the case of two heads per type (column H) and four loops during training (column L_t) were also considered.

Results. Table 2.15 summarizes the results. As expected, the presence of only *L2C* (see Fig. 2.12(b)) has an impact on performance (see row #2 vs #3). Rectangular convolutions (row #4 and Fig. 2.12(c) and (e)) help to almost completely mitigate this loss, approaching the original performance (row #2), but with the advantage of lowering the number of parameters and speeding up the execution compared to row #3.

#	Model	L_t	H	*L2C*	NL_b	NL_a	B_{AP}	S_{AP}	Speed	# Params (M)
1	HTC	3	3				41.7	36.9	3.3	77.2
2	R^3-CNN	3	1				40.9	37.2	3.4	43.9
3		3	1	7×7			39.8	36.4	2.2	32.2
4		3	1	$7\times3 \rightarrow 3\times7$			40.6	36.8	2.8	32.7
5		3	1	7×7	✓		41.8	37.6	1.0	38.6
6		**3**	**1**	$7\times3 \rightarrow 3\times7$	✓		**41.7**	**37.6**	**1.2**	**37.9**
7		3	1	7×7	✓	✓	41.8	37.6	0.9	39.0
8	R^3-CNN	4	2				41.9	37.5	2.9	60.6
9		4	2	7×7			41.4	37.2	1.8	37.1
10		4	2	$7\times3 \rightarrow 3\times7$			41.0	37.1	2.4	38.3
11		**4**	**2**	7×7	✓		**42.9**	**38.1**	**0.8**	**50.0**
12		4	2	$7\times3 \rightarrow 3\times7$	✓		42.6	37.8	0.9	51.1

Table 2.15: Impact of FCC module configurations applied to R^3-CNN detector. Row #2 is the R^3-CNN in row #4 of Table 2.11.

The non-local block before *L2C* (row #5) boosts the performance, matching B_{AP} of HTC and surpassing its S_{AP} by a good margin. Conversely, its introduction after *L2C* does not bring any benefits.

In the case of two heads per type and four loops, *L2C* produces higher performance (see rows #9 and #8) compared to row #3. Rectangular convolutions (row

#10) worsen the performance compared to row #9, but have the advantage of a good increase in speed. As in the previous case, the introduction of our non-local module (row #11 and #12) produces a good performance boost with respect to the model without them (row #9 and #10).

To summarize, FCC with only *L2C* makes the network lighter, reducing the wight by 14 to 18 percent, while slightly worsening the performance compared to using FC layers. Moreover, a boost in performance is achieved by the non-local block inserted before *L2C*, surpassing the original performance with a good margin, albeit at the cost of a higher execution time.

Ablation study on Mask IoU module

Description. In order to increase performance even further, we borrowed the Mask IoU learning task from [32] and redesigned its branch to introduce as few weights as possible. After testing the original Mask IoU branch, as done previously on detection head, we conducted an ablation study. We considered two baselines: a lighter (row #2) and a better-performing (row #8) model in Table 2.16. They also correspond to rows #6 and #11 in Table 2.15, respectively.

#	Model	L_t	H	M_{IoU}	*L2C*	NL_b	NL_a	B_{AP}	S_{AP}	Speed	# Params (M)
1	HTC	3	3					41.7	36.9	3.3	77.2
2	R^3-CNN	3	1					41.7	37.6	1.2	37.9
3		3	1	✓				41.6	38.5	1.1	54.9
4		**3**	**1**	✓	7×7			**41.7**	**38.4**	**1.1**	**43.2**
5		3	1	✓	$7\times3\rightarrow3\times7$			41.8	38.4	1.1	44.3
6		3	1	✓	7×7	✓		41.4	38.3	1.1	49.6
7		3	1	✓	7×7	✓	✓	41.6	38.3	1.1	50.0
8	R^3-CNN	4	2					42.9	38.1	0.8	50.0
9		4	2	✓				42.7	38.6	0.9	66.3
10		**4**	**2**	✓	7×7			**42.7**	**38.7**	**0.8**	**54.6**
11		4	2	✓	7×7	✓		42.8	38.7	0.8	61.0
12		4	2	✓	7×7	✓	✓	42.7	38.6	0.8	61.4

Table 2.16: Impact of FCC to Mask IoU branch.

Results. Table 2.16 summarizes the results. As expected, original Mask IoU module (rows #3 and #9) improves the segmentation. Differently from the detection head, the redesigned Mask IoU branch with only *L2C* with 7×7 kernels (rows #4 and #10) is enough to maintain almost the same performance compared to the original branch (rows #3 and #9), but introduces few new parameters and almost does not affect the execution time. Contrary to the previous experiment, neither rectangular convolutions (row #5) nor our non-local blocks (rows #6 and #7) bring any noticeable improvement.

Results on Generic RoI Extractor (GRoIE)

For the following experiments, we chose the Faster R-CNN as the *baseline* to have a generic and lightweight model to compare with.

Pre-processing module analysis

Description. For this ablation analysis, we did not apply any post-processing. We tested two types of pre-processing: a convolutional layer with different kernel sizes and a non-local block.

Method	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
baseline	37.4	58.1	40.4	21.2	41.0	48.1
conv 3×3	38.1	58.7	41.5	22.2	41.7	49.0
conv 5×5	38.2	59.2	41.6	22.5	41.6	49.0
conv 7×7	**38.3**	**59.2**	**41.6**	**22.7**	**41.7**	**49.4**
conv $7 \times 3 \rightarrow 7 \times 7$	37.9	58.5	41.3	22.0	41.5	49.1
Non-local 1×1	37.7	58.9	40.7	22.0	41.4	48.5
Non-local 7×7	38.4	59.2	41.9	22.5	42.1	49.5

Table 2.17: Ablation analysis on pre-processing module.

Results. Table 2.17 shows the results. The increase in the kernel size improves the fi-

nal performance, confirming the close correlation between neighboring features. The use of a rectangular convolution did not help as it did in Section 2.3.6 for the detection head. In the case of the non-local module, the original one does not have the expected benefit. Our non-local module with a larger kernels gives a slight advantage over the others, but not enough to justify the introduced slowdown.

Post-processing module analysis

Description. In this experiment we analyze the post-processing module, by not applying any pre-processing.

Method	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
baseline	37.4	58.1	40.4	21.2	41.0	48.1
conv 3x3	37.3	58.3	40.4	21.2	41.0	48.5
conv 5x5	37.8	58.7	40.9	22.2	41.2	48.8
conv 7x7	37.9	59.0	41.2	21.5	41.8	48.6
conv 7x3 $\rightarrow$ 3x7	37.4	58.4	40.5	21.4	40.9	48.7
Non-local 1x1	37.8	59.1	40.5	22.0	41.7	48.3
Non-local 7x7	**38.7**	**59.7**	**42.3**	**22.7**	**42.4**	**49.7**

Table 2.18: Ablation analysis on post-processing module.

Results. Comparing Tables 2.17 and 2.18, we can notice that performance trend is the same. However, in the post-processing, the convolutional performance increment is less evident. Contrary to the original non-local, our version with 7×7 kernels obtained a considerably high improvement.

GRoIE module analysis

Description. Finally, we tested the GRoIE architecture with the best-performing pre- and post-processing modules: a 7×7 convolution as pre-processing and non-local

with 7×7 kernels as post-processing.

Method	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
baseline	37.4	58.1	40.4	21.2	41.0	48.1
GRoIE	**39.3**	**59.8**	**43.0**	**23.0**	**42.7**	**50.8**

Table 2.19: Best GRoIE configurations.

Results. From Table 2.19, we can observe a great improvement in the performance, surpassing the original AP by 1.9%.

Experiments on SBR-CNN

Description. In this experiment, we compare Mark-RCNN, CondInst [69] and HTC with our SBR-CNN (Self-Balanced R-CNN) model with the following configuration: the best-performing three-loop model with the rebuilt detection head and MaskIoU head (see row #4 of Table 2.16), with our GRoIE having its best configuration (see Table 2.19) in place of both Bounding Box and Mask RoI extractors. In addition, we take into account GC-Net [30] and Deformable Convolutional Networks (DCN) [63], investigating whether the performance benefit we bring is independent of the underlying architecture. To be as fair as possible, we compare also GC-Net and DCN joined with HTC. For example, HTC+GC-Net means that we considered the combination of both architectures.

Results. In Table 2.20 we see that, independently from the architecture, our SBR-CNN reaches the highest AP values in all metrics in both tasks, even if the counterpart is merged with HTC. More specifically, fusing other models with SBR-CNN not only maintains the performance increment but also increases the gap in favor of SBR-CNN.

In case of B_{AP}, for instance, looking at the B_{AP} in the standalone case (row #4), SBR-CNN outperforms HTC (row #3) by a 0.3% margin only. But, when combined

#	Method	Bounding Box						Mask					
		AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
1	Mask	37.3	58.9	40.4	21.7	41.1	48.2	34.1	55.5	36.1	18.0	37.6	46.7
2	CondInst	38.3	57.3	41.3	22.9	41.9	49.0	34.4	54.9	36.6	15.8	37.9	49.5
3	HTC	41.7	60.4	45.2	24.0	44.8	54.7	36.9	57.6	39.9	19.8	39.8	50.1
4	SBR-CNN	**42.0**	**61.1**	**46.2**	**24.2**	**45.3**	**55.3**	**39.2**	**58.7**	**42.4**	**20.6**	**42.6**	**54.2**
5	GC-Net	40.5	62.0	44.0	23.8	44.4	52.7	36.4	58.7	38.5	19.7	40.2	49.1
6	HTC+GC-Net	43.9	63.1	47.7	26.2	47.7	57.6	38.7	60.4	41.7	21.6	42.2	52.5
7	SBR-CNN+GC-Net	**44.8**	**64.6**	**49.0**	**27.2**	**48.0**	**58.8**	**41.3**	**62.1**	**44.7**	**23.1**	**44.6**	**56.4**
8	DCN	41.9	62.9	45.9	24.2	45.5	55.5	37.6	60.0	40.0	20.2	40.8	51.6
9	HTC+DCN	44.7	63.8	48.6	26.5	48.2	60.2	39.4	61.2	42.3	21.9	42.7	54.9
10	SBR-CNN+DCN	**45.3**	**64.6**	**49.7**	**27.2**	**48.8**	**60.6**	**41.5**	**62.2**	**45.0**	**22.9**	**45.1**	**58.0**

Table 2.20: Performance of the state-of-the-art models compared with SBR-CNN model. Bold and red values are respectively the best and second-best results.

with GC-Net and DCN, this improvement is even higher (0.9% in the case of GC-Net - row #7 vs #6 - and 0.6% in the case of DCN - row #10 vs #9). Considering all metrics, the improvement is up by 1.5% (see AP_{50} in row #7 vs #6).

In case of S_{AP}, it fluctuates from +2.1% up to +2.6%, when comparing SBR-CNN+GC-Net with HTC+GC-Net (row #7 vs #6). Considering all metrics, the highest improvement is +4.1% (see AP_l in row #10 vs #9).

2.3.7 Conclusions

We propose a new object detection and instance segmentation architecture called SBR-CNN, which addresses two of intrinsic imbalances which affect two-stage architectures descending from Mask R-CNN: the IoU Distribution Imbalance of positive input bounding boxes with the help of a new mechanism for refining RoIs through a loop between detection head and RoI extractor, and a loop for mask refinement inside the segmentation head. Furthermore, we address the Feature Imbalance that afflicts the FPN layers, proposing a better performing RoI Extractor which better integrates low- and high-level information. Finally, we investigate the effect of a redesign of the model head toward a lightweight fully convolutional solution. Our empirical studies confirmed that if the task involves classification, there is the necessity to main-

tain some spatial sensitivity information by the enhanced non-local block. Otherwise, when a regression task is involved, a convolutional head is enough. Our SBR-CNN proves to be successfully integrated into other state-of-the-art models, reaching a 45.3% AP for object detection and 41.5% AP for instance segmentation, using only a small backbone such as ResNet-50.

2.4 Improving Localization for Semi-Supervised Object Detection

We all have an unsuspected reserve of strength inside that emerges when life puts us to the test.

Isabel Allende

2.4.1 Introduction

Supervised learning usually requires a large amount of annotated training data which can be expensive and time-consuming to produce. Semi-Supervised Learning (SSL), on the other hand, addresses this issue by taking advantage of large unlabeled data accompanied by a small labeled dataset. Usually, in an SSL setting, we have two models that act as Teacher and Student. The latter is trained in a supervised way, drawing the needed ground truths from the dataset, if exist, and from the Teacher's predictions, otherwise.

This approach can be beneficial in many machine learning tasks including object detection, whose final result is a list of bounding boxes (bboxes) and the corresponding classes. In this task, the network is devoted to finding the position of the localized objects in an image, as well as identifying which class they belong to. Hereinafter, we will refer to this specific application as Semi-Supervised Object Detection (SSOD). In [7], authors have collected many interesting ideas applied to SSOD. Among them, the one that seems more promising is the Mean Teacher [71], which uses the Exponential Moving Average (EMA) as a knowledge transfer technique from the Student to the Teacher. In particular, while the Teacher produces pseudo-labels on unlabeled and weakly augmented images to obtain more reliable labels, the Student is trained using the pseudo-labels as ground-truth on the same images, but strongly augmented to differentiate the training. In weak augmentation, only a random horizontal flip is applied, while in strong augmentation, other transformations such as randomly adding color jittering, grayscale, Gaussian blur, and cutout patches are also used.

Although they obtain good performances with this pseudo-labeling technique, the Student model is still influenced by the Teacher's erroneous predictions. The quality of these predictions is difficult to control by applying only a simple confidence thresholding. For this reason, there is a need for an additional way to strengthen the region proposals and reduce the number of erroneous predictions by the Teacher. In the proposed architecture, we introduce a new task used for the classification of the bboxes, with the aim of distinguishing good quality ones from the others. This new score exploits the information complementary to the class score already used in these networks, allowing a different level of filtering. In addition, we show how to take advantage of the regression tasks on the unsupervised learning part. Usually, they are excluded in the unsupervised training phase. The justification is that the classification score is not able to filter the potentially incorrect bboxes [7, 23]. A well-known problem during training is the Objective Imbalance [16], which is characterized by the difficulty in balancing the different loss contributions. In our hypothesis, this is the case for the unsupervised training part. In order to obtain a positive effect from regression losses, an adequate balance of the contribution of these two losses (regression and category classification) is a possible solution to the above-mentioned problem. In this way, we prevent the regression losses on unsupervised dataset from dominating the training, a phenomenon that greatly amplifies the error introduced by inaccurate Teacher predictions.

The main contributions of this section are the following:

- a new bounding box IoU (Intersection over Union) classification task to filter out errors on pseudo-labels produced by the Teacher;
- the introduction of a regression task on the unlabeled dataset which can help the network to learn better;
- an exhaustive ablation study on all the components of the architecture.

2.4.2 Related Work

Semi-Supervised Learning for Object Detection. In [72], the authors proposed the Π-Model which is an ensemble of the predictions of the model at different epochs under multiple regularizations and input augmentation conditions. The predictions for the unlabeled images are merged together to form a better predictor. In [73], the authors proposed Consistency-based Semi-supervised Learning for Object Detection (CSD), which uses consistency constraints as a self-training task to obtain a better training of the network. In [74], an SSL framework with Self-Training (via pseudo label) and the Augmentation driven Consistency regularization (STAC) is introduced, exploiting weak data augmentation for model training and strong data augmentation for pseudo-labeling. Several works [7, 71, 75] proposed to use the Mean Teacher model, applying EMA to the weights instead of the predictions, facilitating knowledge transfer from Student to the Teacher model at each iteration. For one-stage object detection models, authors of [76] use an Expectation-Maximization approach, generating pseudo-labels in the expectation step and training the model on them in the maximization step, optimizing for classification in each iteration and for localization in each epoch. In [77], the authors propose a Soft Teacher mechanism where the classification loss of each unlabeled box is weighted by the Teacher classification score, also using a box jittering approach to select the most reliable pseudo-bboxes. In [34], authors utilize SelectiveNet to properly filter pseudo-bboxes trained after the Teacher. In [78], the authors propose two models, one of which generates a proposal list of bounding boxes and the second one refines these proposals, using the average class probability and the weighted average of the bounding box coordinates.

Bounding Box Intersection over Union (BBox IoU). In [79], the authors added a new branch on top of the Fast R-CNN model to estimate standard deviation of each bounding box and use it at Non-maximum Suppression (NMS) level to give more weight to the less uncertain ones. In [23, 66], the authors added a new branch on top of the Faster R-CNN to regress bounding box IoU and to multiply this value with the classification score to compute final score of the suppression criterion of the NMS. In the same direction, Fitness NMS was proposed in [80] to correct the detection score for better selecting bounding boxes which maximize their estimated IoU with

the ground-truth.

2.4.3 Teacher-Student Learning Architecture

In SSOD, we have two datasets. The first set $\mathbf{D_s} = \{x_i^s, y_i^s\}_{i=1}^{N_s}$, typically smaller, contains N_s images $\mathbf{x^s}$ with the corresponding labels $\mathbf{y^s}$, and the second set $\mathbf{D_u} = \{x_i^u\}_{i=1}^{N_u}$ contains N_u images $\mathbf{x^u}$ without labels. Similarly to [7], our architecture is composed of two identical models where one behaves as Teacher and the other as the Student. At each iteration of the semi-supervised training, the Student is trained with one batch of images coming from $\mathbf{D_s}$ (supervised training) and one from $\mathbf{D_u}$ (unsupervised training), using the pseudo-labels generated by the Teacher as ground-truth and filtered by a threshold. Then, the Student transfers the learned knowledge to the Teacher using the EMA applied to the weights.

In our model, the total loss L for the Student is composed of two terms, which come from the the supervised L_{sup} and unsupervised L_{unsup} training:

$$\begin{aligned} L &= L_{sup} + L_{unsup} \\ L_{sup} &= \sum_i L_{cls} + L_{reg} + L^{IoU} \\ L_{unsup} &= \sum_i \alpha L_{cls} + \beta L_{reg} + \gamma L^{IoU} \\ L_{cls} &= L_{cls}^{rpn}(x_i^s, y_i^s) + L_{cls}^{roi}(x_i^s, y_i^s) \\ L_{reg} &= L_{reg}^{rpn}(x_i^s, y_i^s) + L_{reg}^{roi}(x_i^s, y_i^s) \end{aligned} \tag{2.10}$$

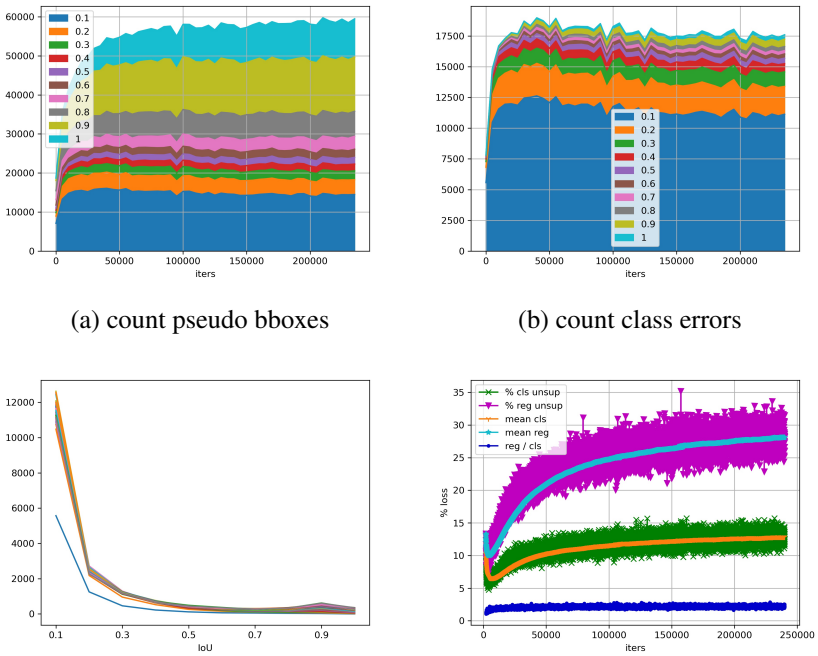
where L_{cls} contains the sum of Region Proposal Network (RPN) and Region of Interest (RoI) classification losses, while L_{reg} contains the sum of RPN and RoI regression losses. The L^{IoU} loss will be defined in Subsection 2.4.3. α, β and γ represent how much weight each component of the unsupervised training has. The new terms introduced in this section, w.r.t. the ones defined in [7], are shown in bold in the above equations and will be described in the following.

Bounding Box Regression on Unsupervised Training

The training consists of two phases, the *burn up* stage and *Teacher-Student Mutual Learning* stage. In the first stage, the Student is trained only on the labeled dataset, following the standard supervised training procedure. Then, for the second stage, at each iteration, the Student is trained on two batches at the same time, one coming from the labeled dataset and the other coming from the unlabeled dataset. For the latter, the baseline [7] trains only the RPN and RoI head classifier and disregards regression. The authors justify this choice noticing that the classification confidence thresholds are not able to filter the incorrect bounding box regression. Our hypothesis is that training also on pseudo bounding box regression could help the Student as long as the task is correctly weighted with respect to the others. Category classification and regression tasks are learned in parallel. Given that the classification performance improves during the training, we can also expect the bounding box regression to behave the same way. In other words, while the training proceeds, the average quality of the pseudo-labels increases and so does the classification confidence value.

During the Student training, although pseudo bounding boxes are filtered with a threshold on classification confidence score, we still have bounding boxes of any IoU quality. This problem is related to the uncertainty on prediction. Figure 2.14a visualizes the IoU distribution quality with respect to the ground-truth of the Teacher filtered predictions. We can notice that the number of pseudo bounding boxes with IoU less than the threshold (0.6 in our experiments) remains almost constant during the entire training, unlike the others which slowly increase. In Figure 2.14b, we show only the filtered pseudo bounding boxes that the Teacher has wrongly classified and split by their IoU. In Figure 2.14c, the same data are shown using a different graph. We can notice that the number of wrongly classified pseudo-bboxes decreases exponentially with the increase of the quality, and this trend remains the same during the training. The bounding boxes with low quality ($IoU < 0.6$) represent 45% of the total pseudo-bboxes and more than 90% of classification errors. This means that the low-quality IoU bboxes contain almost all the classification errors.

By looking at Figure 2.14d, we can see that the unsupervised regression loss (from RPN and RoI heads) represents between 20% and 30% of the total loss. Con-



(a) count pseudo bboxes

(b) count class errors

(c) count class errors per IoU

(d) Unsupervised L_{cls} and L_{reg}

Figure 2.14: The pseudo bounding boxes generated during training: (2.14a) IoU distribution of pseudo-bboxes, (2.14b) distribution of pseudo-bboxes when the predicted class is wrong. (2.14c) number of pseudo-bboxes per IoU collected every 5000 iterations. (2.14d) Unsupervised classification and regression losses comparison during the training. Better seen in color.

versely, the unsupervised classification loss (from RPN and RoI heads) accounts only for 15% (at most) of the total loss. This means that an error in pseudo-labels has almost three times (see blue line in Figure 2.14d) more weight on regression branch than on classification branch. To avoid the amplification of the errors, an appropriate value for β is chosen, by under-weighting the regression contribution and making it comparable with the classification one.

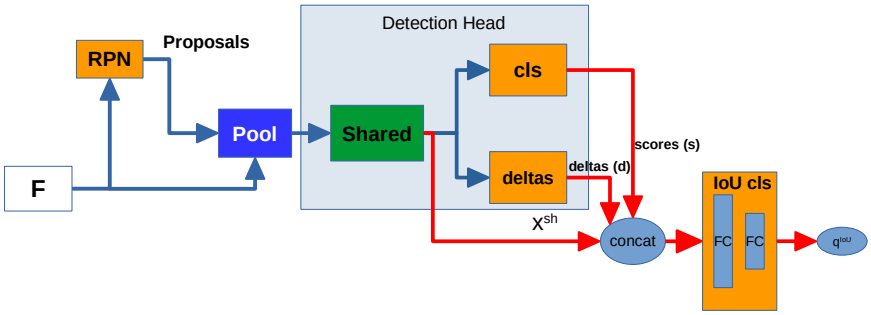


Figure 2.15: Faster R-CNN architecture with our branch in red.

Bounding Box IoU Classification Task

As noted by [23, 66], adding a new task related to the IoU prediction could help the network to train better. Our model learns to predict the bounding box quality, using this value to further filter pseudo-labels in conjunction with the classification score. Differently from them, we train the model to make a binary classification instead of a regression since it is sufficient (and easier to learn) for the purpose of filtering.

Fig. 2.15 illustrates the architecture of a two-stage object detection model such as Faster R-CNN [25] with our additional branch highlighted in red. For each positive bounding box (*i.e.*, the ones recognized as foreground), it concatenates the output of the shared layers (x_i^{sh}, with the size of 1024), all the classification scores (s_i, with the size of the number of classes plus the background) and all bounding box regression deltas (d_i, with the size of 4). All these features are passed to the *IoU cls* branch (also called $f(\cdot)$ in the following), which outputs a vector with the same size as s_i (*i.e.*, one for each class). This will return the i^{th} IoU classification, called *IoU score* q^{IoU}, corresponding to the predicted bbox. The $f(\cdot)$ branch consists of two fully-connected layers with an ELU [81] activation function between them and a *sigmoid* activation function at the end.

The loss L_i^{IoU} for the new branch of IoU classification, conditioned on the class,

is defined as:

$$\begin{aligned} q_i^{IoU} &= f(concat(x_i^{sh}, s_i, d_i)) \\ L_i^{IoU} &= FL(q_i^{IoU}, t_i) \end{aligned} \tag{2.11}$$

where the FL function is the Focal Loss [82] with its own γ value equal to 1.5 and t_i represents the binary target label. For the i^{th} bounding box, the branch output $q_i^{IoU} \in [0,1]$ is a single value which predicts if it is a high- or low-quality bounding box. The target t_i and its IoU are defined as follows:

$$\begin{aligned} IoU_i &= \max \text{IoU}\left(b_i^t, g\right) \quad \forall b_i \in \mathbf{B} | IoU\left(b_i, g\right) \geqslant u \\ t_i &= \begin{cases} 1 & IoU_i > \mu \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2.12}$$

where $\mathbf{B}$ is the list of proposals coming from the RPN, b_i are the i^{th} predicted bounding boxes, g is the ground-truth bounding box, u is the minimum IoU threshold to consider the bounding b_i as positive example (typically set to 0.5) and μ (set to 0.75) is the minimum IoU threshold to be classified as high-quality.

In the evaluation phase, the Teacher's filtering of the predicted bounding boxes with low confidence is preceded by the IoU classification filtering, which uses a new IoU inference threshold θ (set to 0.4).

2.4.4 Experiments

Dataset. We perform our tests on the MS COCO 2017 dataset [4]. The training dataset consists of more than 117,000 images and 80 different classes of objects, where 10% of the labeled images are used.

Evaluation Metrics. All the tests are done on COCO minival 2017 validation dataset, which contains 5000 images. We report mean Average Precision (mAP) and AP_{50} and AP_{75} with 0.5 and 0.75 minimum IoU thresholds, and AP_s, AP_m and AP_l for small, medium and large objects, respectively.

#	β	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
0	0.5	31.775	51.450	34.190	16.952	34.384	41.549
1	**1.0**	**31.947**	**51.530**	**34.270**	**16.949**	**34.900**	**41.306**
2	2.0	31.754	51.078	34.143	16.691	35.008	41.670
3	4.0	30.445	49.387	32.727	15.044	33.200	40.209

Table 2.21: Performance varying the weight loss β on unsupervised regression losses.

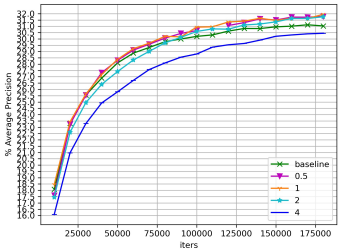
#	Method	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
1	UT	31.027	50.757	33.056	17.014	33.684	40.322
2	Ours (with filter)	31.604	51.181	33.962	16.816	34.283	40.809
3	Ours (w/out filter)	31.509	51.118	33.564	16.848	34.684	40.582

Table 2.22: Performance comparison with original Unbiased Teacher (UT) model: (2) Training with BBox IoU branch with and (3) w/out pseudo-labels filtering.

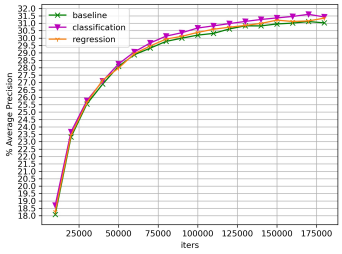
Implementation details. All the values are obtained running the training with the same hardware and hyper-parameters. When available, the original code released by the authors is used. Our code is developed on top of the Unbiased Teacher [7] source code. We perform the training on a single machine with 6 Tesla P100 GPUs with 12GB of memory. The train lasts 180,000 iterations with a batch size of 2 images per GPU for the supervised part and 2 images for the unsupervised part, with α set to 4. We use the Stochastic Gradient Descent (SGD) optimization algorithm with a learning rate of 0.0075, a weight decay of 0.0001, and a momentum of 0.9. The learning rate decays at iteration 179990 and 179995. We use the Faster R-CNN with FPN [26] and the ResNet 50 [24] backbone for the teacher and student models, initialized by pre-trained on ImageNet, and the same augmentation of Unbiased Teacher [7].

Ablation study

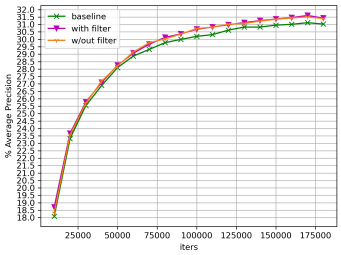
Unsupervised regression loss. In this experiment, we empirically show that we can also use regression losses on RPN and RoI head for the unsupervised part. We test different weights for the constant β in the loss formula (see eq. 2.10). In Figure 2.16a



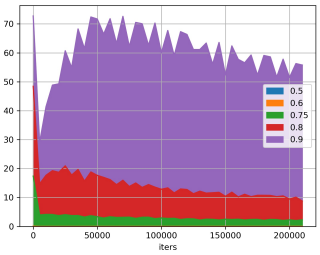
(a) Unsup regression loss weights.



(b) Classification vs Regression.



(c) Filtering with bbox IoU score.



(d) Count filtered pseudo-bboxes.

Figure 2.16: Ablation studies: (2.16a) weights for unsupervised regression loss on RPN and RoI. (2.16b) classification vs regression loss on bbox IoU branch. (2.16c) filtering bbox on inference with bbox IoU classification score. (2.16d) count bboxes filtered by inference threshold μ during training.

and Table 2.21, we can see that greatly amplifying the contribution can be deleterious, becoming counterproductive in the case of β equal to 4.

Bounding box IoU branch loss type. Our proposal involves a new IoU classification task, trained with a binary cross-entropy function. In this experiment, we test how performance changes in case our new branch learns a regression task instead of a classification, using a *smooth L1* loss. In this case, the ground-truth is represented by the real IoU value between the bbox and the ground-truth. In Figure 2.16b, we can see that classification branch is more stable and reaches a slightly higher perfor-

#	μ	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
1	0.5	31.199	51.009	33.047	16.187	34.000	40.180
2	0.6	31.128	50.785	33.268	17.102	33.805	39.932
3	0.7	31.461	51.319	33.637	16.714	34.217	40.100
4	**0.75**	**31.604**	**51.181**	**33.962**	**16.816**	**34.283**	**40.809**
5	0.8	31.336	50.601	33.707	16.327	34.180	40.476
6	0.9	27.125	43.515	28.800	12.815	29.486	36.034

Table 2.23: Performance using BBox IoU classification branch with inference threshold θ fixed to 0.5 and varying training threshold μ.

#	θ	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
0	0.3	31.404	51.205	33.792	16.273	34.542	40.851
1	**0.4**	**31.630**	**51.185**	**34.044**	**17.387**	**34.494**	**40.784**
2	0.5	31.604	51.181	33.962	16.816	34.283	40.809
3	0.6	31.158	50.227	33.418	16.203	33.687	40.323
4	0.7	30.649	49.216	33.138	16.532	33.409	39.542

Table 2.24: Performance using BBox IoU classification branch with training threshold μ fixed to 0.75 and varying inference threshold θ.

mance.

With and without filtering bboxes. The bbox IoU branch learns to recognize high quality bounding boxes and, as the default behavior, also to pre-filter Teacher's pseudo-bboxes depending on our new threshold score. In Figure 2.16c and in Table 2.22 (rows #2 and #3), we see that our new branch contributes to the increase of the general performance (+0.48% mAP). Then, another small improvement is given by the filtering phase, increasing the performance by +0.1% mAP.

Bounding box training threshold μ. In this experiment, we test the bbox IoU classification branch, setting inference threshold θ to 0.5 and varying the training threshold μ. From Table 2.23, it is clear that the choice of a correct threshold greatly influences the performance. On the one hand, if the threshold is too low, it does not help the

#	L_{reg}^{unsup}	x^{sh}	scores	deltas	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
1					31.027	50.757	33.056	17.014	33.684	40.322
2	✓				31.947	51.530	34.270	16.949	34.900	41.306
3	✓	✓			31.754	51.189	34.032	16.850	34.657	41.320
4	✓	✓	✓		**32.166**	**51.772**	**34.765**	**16.647**	**34.999**	**41.870**
5	✓	✓	✓	✓	31.923	51.464	34.070	16.202	35.197	41.368
6		✓	✓	✓	31.630	51.185	34.044	17.387	34.494	40.784

Table 2.25: Study on unsupervised regression losses and IoU classification loss.

network to learn more descriptive feature maps. On the other hand, if it is too high, the risk to wrongly filter out the bounding boxes will increase. As we can see in Fig. 2.16d, with the increase of IoU threshold μ, the number of teacher pseudo-bboxes filtered during the training increases exponentially. This is likely due to an imbalance in training, where the higher the threshold, the fewer high-quality examples are available. The best value is in the middle between the threshold u (0.5) and the IoU maximum value 1.0.

Bounding box inference filter threshold θ. In this experiment, we test our bbox IoU branch, setting the training threshold μ to 0.75 (best value previously found) and varying the inference threshold θ. In Table 2.24, we see that the best value for this threshold is in the middle as expected because the branch is trained to reply 1 if the bbox is good enough and 0 otherwise.

IL-net: Improving Localization net. Finally, we test the full architecture IL-net, composed of the unsupervised regression losses and the new IoU classification branch. Since, using both of them, the contribution of the new branch is absorbed by the loss of unsupervised regression (see rows 2 and 5 in Table 2.25), we performed an ablation study reducing the values in input to the new branch (see eq. 2.11). This analysis has allowed us to highlight that by removing the contribution of the deltas, we can increase the general performance. This behavior could be explained by the fact that the deltas are optimized from both losses (L_{usup}^{reg} and L^{IoU}), causing the conflict as a result.

2.4.5 Conclusions

In this section, we proposed two new architectural enhancements with respect to the network proposed in [7]: a new bounding box IoU classification task to filter out errors on pseudo-labels produced by the Teacher and the introduction of the unsupervised regression losses. For the former, we introduced a lightweight branch to predict the bounding box IoU quality. For the latter, we demonstrated how to successfully integrate it in the training, balancing the training tasks. Our new model called IL-net, which contains both, increases the general SSOD performance by a 1.14% AP on COCO dataset in a limited annotation regime.

2.5 A new dataset for Grape Diseases Instance Segmentation

Before you heal someone, ask him if he's willing to give up the things that make him sick.

Hippocrates

2.5.1 Introduction

Recognize plant diseases is as deeply felt problems in the agricultural sector. Early disease detection and diagnosis is a key aspect for a correct and sustainable disease control. Traditionally, the diagnosis of leaf and cluster diseases of grapes relies on expert judgment based on visual inspection of the disease symptoms and signs, which usually leads to high cost and a risk of error. With the rapid development of artificial intelligence, machine learning methods have been applied to plant disease detection to make it smarter. In recent years, deep learning approaches led to a huge improvement of image analysis with the tasks of object detection [83] and its evolution, instance segmentation [84]. A main problem related with these deep learning tasks is the need of a labeled dataset with a large amount of images in order to obtain an acceptable performance. The purpose behind the creation of this dataset is the implementation of an automatic system for the analysis of the leaves and bunches of vines through images, in order to identify the diseases that affect them. The project was born in collaboration with Horta s.r.l, a spin off company of the Università Cattolica del Sacro Cuore in Piacenza (Italy), that develops and provides web based services to agricultural and agro-industrial chains, with the aim of increasing competitiveness and sustainability of crop management and ensuring greater food safety, within the project "AGREED - Agriculture, Green & Digital" (ARS01_00254 on PON “Ricerca e Innovazione” 2014-2020 and FSC funds). The final purpose is to give the possibility to an inexperienced user to analyze the general status of the plants and identify any diseases through a photo taken from a mobile device. Our goal is to be able to iden-

tify each disease in a very precise way, and provide the user with information about the disease severity. The information provided could be integrated automatically in broader management tools, such as Decision support systems (DSSs), and increase the support they give to farmers for the sustainable management of vineyards. For this reason we have chosen the instance segmentation task because it allows to examine the leaves and grapes accurately. As far as we know, no similar methods have been developed before.

2.5.2 Related

At the state of the art there are several attempts to analyze plants and their diseases. Table 2.26 reports a summary with the salient characteristics for each of them. In [85], authors formed a dataset of grapevine leaves in order to detect the Esca disease through an image classification algorithm. In [86], the authors formed a dataset for three common grape leaf diseases (Black rot, Black measles, also called Esca, and Leaf blight) and for the Mites of grape; the dataset contains 4,449 original images of grape leaf diseases labeled with bounding boxes. In [87], authors formed a dataset of 300 images for five grape varieties: Chardonnay, Cabernet Franc, Cabernet Sauvignon, Sauvignon Blanc and Syrah; the dataset is public and contains label for grape detection and instance segmentation. In [88], authors formed a dataset for disease image recognition, which contains label to classify four diseases of wheat: stripe rust, leaf rust, grape downy mildew, grape powdery mildew. In [89], authors labeled with bounding boxes 5,000 images of tomato diseases. In [90], authors concentrated their work around the Esca diseases to be able to make object detection; for this scope, they labeled 6,000 images of Bordeaux vineyards.

2.5.3 Leaf Diseases Dataset

Our dataset contains 1,092 RGB images of grapes and 17,706 annotations in COCO-like [4] format for the tasks of Object Detection and Instance Segmentation. The images were collected from Horta's internal databases, the competition *Grapevine*

Name	Images (k)	Classes	Task
Alessandrini, M., et al. [85]	1,770	1 grape diseases	Image Classification
Xie, Xiaoyue, et al. [86]	4,449	4 grape diseases	Object Detection
Santos, Thiago T., et al. [87]	300	5 grapes	Object Detection and Instance Segmentation
Wang, Haiguang, et al. [88]	185	5 grape diseases	Image Classification
Fuentes, Alvaro, et al. [89]	5,000	9 tomato diseases	Object Detection
Rançon, Florian, et al. [90]	6,000	1 grape diseases	Object Detection

Table 2.26: Agriculture datasets.

(a) bunch of grapes affected by powdery mildew (oidio).

(b) leaves affected by downy mildew (peronospora).

Figure 2.17: Example of segmentation with the help of Label Studio.

Disease Images [1], and from search engine scraping; all segmentations were manually made by the authors. In Figure 2.17 there are some examples of labeled images. The annotations identifies 10 categories of objects: leaf, grape and eight diseases. Table 2.27 shows the following information: the ids, label key and name, if it is a diseases or not, and the total available annotations.

In Figure 2.18 the directory structure is highlighted. It is composed by: annotations directory for training and validation dataset in COCO-like format (*json* files) and the *images* directory, which contains the images in *jpg* and *png* formats.

Annotations are split in training (80%) and validation (20%) dataset, with the

[1] https://www.kaggle.com/piyushmishra1999/plantvillage-grape

	Key	Count	Label	Name	Type
1	BRF	5478	black_rot_foglia	Black Rot	leaf disease
2	BRG	1443	black_rot_grappolo	Black Rot	grape disease
3	BF	88	botrite_foglia	Botrite	leaf disease
4	BG	612	botrite_grappolo	Botrite	grape disease
6	FV	1647	foglia_vite	Vine leaf	leaf
7	GV	993	grappolo_vite	Bunch of grapevines	grape
9	OF	2086	oidio_foglia	Oidio	leaf disease
10	OG	1740	oidio_grappolo	Oidio	grape disease
12	PF	2565	peronospora_foglia	Peronospora	leaf disease
13	PG	1054	peronospora_grappolo	Peronospora	grape disease
		17706	Total		

Table 2.27: LDD annotations description.

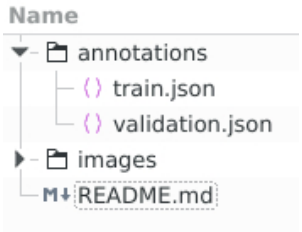


Figure 2.18: LDD directory composition.

help of the *cocosplit* [2] utility.

In Figure 2.23 some examples for each class are shown.

Annotation procedure.

To make the annotations *Label Studio v1.5* [3] was used with the following configuration:

[2] https://github.com/akarazniewicz/cocosplit

[3] https://labelstud.io/

Listing 2.1: "Label Studio Instance Segmentation configuration"

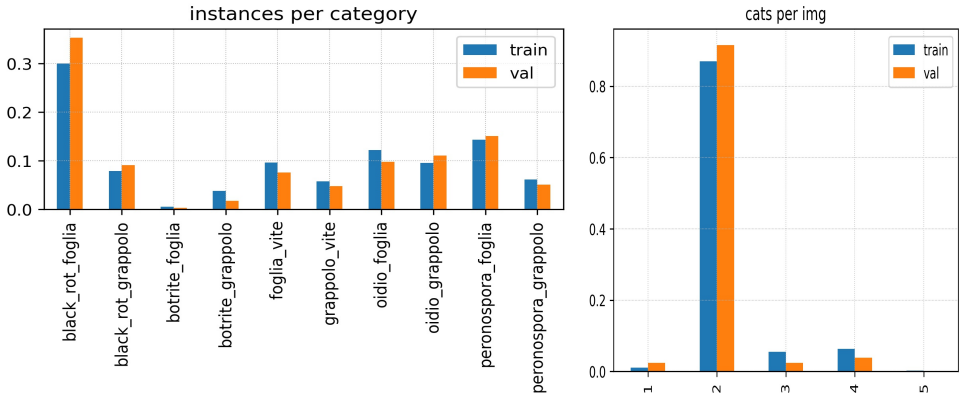
```
<View>
    <Image name="image" value="$image" zoom="true" zoomControl="true"/>
    <PolygonLabels name="label" toName="image" strokeWidth="3"
                pointSize="small" opacity="0.9">
        <Label value="black_rot_foglia" background="#D4380D"/>
        <Label value="black_rot_grappolo" background="#FFC069"/>
        <Label value="botrite_foglia" background="#AD8B00"/>
        <Label value="botrite_grappolo" background="#D3F261"/>
        <Label value="oidio_foglia" background="#096DD9"/>
        <Label value="oidio_grappolo" background="#ADC6FF"/>
        <Label value="peronospora_foglia" background="#F759AB"/>
        <Label value="peronospora_grappolo" background="#FFA39E"/>
        <Label value="foglia_vite" background="#AD8B00"/>
        <Label value="grappolo_vite" background="#D3F261"/>
    </PolygonLabels>
</View>
```

Figure 2.19: Label Studio web interface.

Example of web interface can be seen in Figure 2.19 where, for each image, the user can select the proper category and then the polygon which identify the instance.

First, the annotations has been created by a student. Then, to improve the quality, an agronomist expert on this field has reviewed and fixed them. At the end, the annotations have been exported in COCO-like format.

Statistics.



(a) Percentage of instances per category.

(b) Categories per image

Figure 2.20: Dataset LDD: (2.20a) Percentage of instances for each category. (2.20b) Percentage of annotated categories per image. Better seen in color.

In Figure 2.21, the percentage of instances for each category are represented. The dataset is strongly unbalanced, but the same distribution can be appreciated in both training and validation dataset. As shown in Figure 2.20b, the maximum number of unique categories for each image can vary from one to five, with more than 80% of the images containing two classes. This could be explained by the fact that, usually, the image contains a leaf or a grape affected by a particular disease. Contrary to the number of classes per image, the number of instances is much more varied, as shown in Figure 2.21, with more than one hundred different instances for complex images. Figures 2.22 offer a different point of view, showing that mean percentage

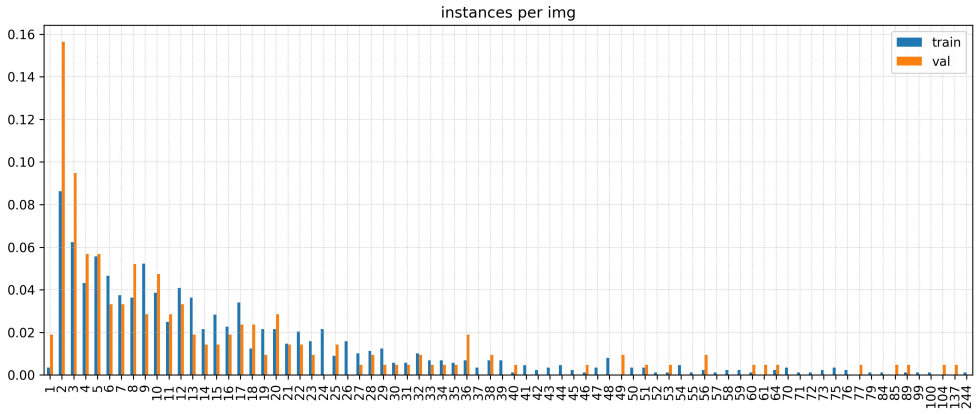


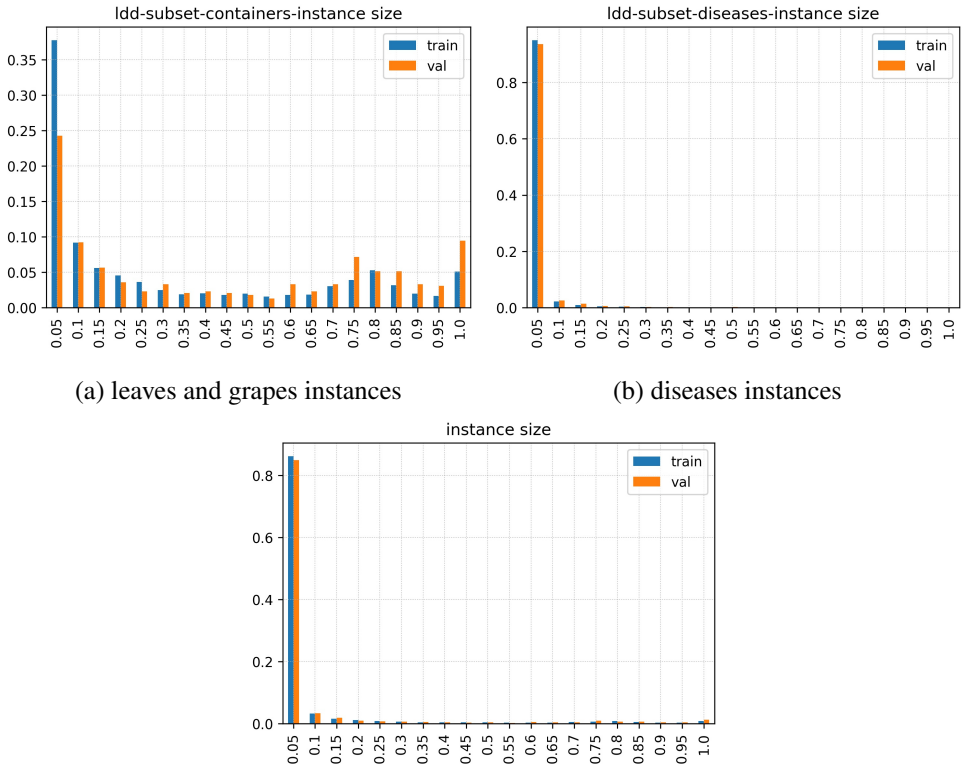
Figure 2.21: Annotations per image: percentage of images in LDD.

of the instance size respect to the image size is very low, with more than 80% of the instances being in the range $[0\%, 0.05\%]$. Figure 2.22a shows the distributions for leaves and grapes, which are much more diverse respect to diseases.

2.5.4 Experiments

Evaluation Metrics. All the tests are done on LDD validation dataset, which contains 212 images and 3'234 annotations. We report mean Average Precision (mAP) and AP_{50} and AP_{75} with 0.5 and 0.75 minimum IoU thresholds, and AP_s, AP_m and AP_l for small, medium and large objects, respectively.

Implementation details. All the values are obtained running the training with the same hardware and hyper-parameters. When available, the original code released by the authors is used. Our code is developed on top of the MMDetection [49] source code. We perform the training on a single machine with 1 NVIDIA GeForce RTX 2070 with 8GB of memory. The train lasts 12 epochs with a batch size of 2 images. We use the Stochastic Gradient Descent (SGD) optimization algorithm with a learning rate of 0.00125, a weight decay of 0.0001, and a momentum of 0.9. The learning rate decays at epoch 8 and 11. We use the ResNet 50 [24] backbone for the models,



(a) leaves and grapes instances

(b) diseases instances

(c) all instances

Figure 2.22: Dataset LDD instance sizes distribution: (2.22a) only leaves and Grapes. (2.22b) only diseases. (2.22) all dataset. Better seen in color.

initialized by pre-trained on ImageNet.

For the experiment, basic Mask R-CNN [14] and R^3-CNN [2] models were used, with the intention to test the models to the dataset and give a baseline.

Table 2.28 shows the training results for the task of Object Detection and Instance Segmentation. Best values are given by the R^3-CNN. More in detail, Tables 2.29 and 2.30 show AP values for each category for object detection and instance segmentation, respectively. Figure 2.24 shows some examples of R^3-CNN predictions compared with ground-truth. For black rot on leaves, one of most represented dis-

#	Method	Bounding Box						Mask					
		AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
1	Mask R-CNN	21.0	36.8	21.9	9.1	18.8	21.2	20.2	35.6	21.6	8.8	17.2	20.1
2	R^3-CNN	22.7	38.4	22.8	9.5	19.9	30.9	22.2	36.2	24.3	9.4	19.3	29.8

Table 2.28: Performance of Mask R-CNN and R^3-CNN models.

ease in our dataset, we reached promising results. The same for leaves, even though there were much fewer instances. Conversely, a grape of wine is more difficult to detect achieving a third of performances compared to leaves with two thirds of its instances. Despite a fair number of instances, most difficult disease to detect is *peronospora_grappolo* (Downy mildew on bunches). Figure 2.25 shows the confusion matrix for the R^3-CNN model.

#	Method	BRF	BRG	BF	BG	FV	GV	OF	OG	PF	PG	AVG
1	Mask R-CNN	45.8	14.3	41.7	6.0	51.3	14.0	4.8	16.9	14.2	0.9	21.0
2	R^3-CNN	49.1	15.2	40.4	7.0	55.3	16.7	6.0	18.2	17.4	1.4	22.7

Table 2.29: BBox AP per category. See Table 2.27 for class key.

#	Method	BRF	BRG	BF	BG	FV	GV	OF	OG	PF	PG	AVG
1	Mask R-CNN	48.6	15.0	40.4	5.4	50.3	10.3	4.4	14.3	12.1	0.7	20.2
2	R^3-CNN	52.0	16.2	44.6	4.6	54.8	11.6	5.0	15.8	15.7	1.3	22.2

Table 2.30: Segmentation AP per category. See Table 2.27 for class key.

2.5.5 Discussion

We introduced a new dataset for segmenting the most common leaf and grapes diseases in their natural environment. The main idea behind the realization of this dataset was to use images with few leaves or bunches affected by a few diseases in foreground in order to facilitate and speed-up the manual process of labeling. This choice explains the number of categories per image shown in Figure 2.20b and can possibly

led to a degradation of performances in case of the segmentation of images with an high numbers of leaves or grapes, that are very common into a natural environment.

Emphasis was placed on generating ground truth labels where the disease's polygon was entirely contained inside the leaf or cluster polygon, in order to encourage the strict connection between infected organ and disease, trying to avoid cases where an isolated disease is shown.

Another peculiarity of this dataset is related to the choice of decomposing big sized ground truth polygons into smaller ones in order to realize the most accurate labeling possible. In particular this approach was used for the labeling of cluster and their diseases in order to exclude the small gaps that are present between berries in a bunch. A consequence of this label partitioning is the predominance of small sized instances in the dataset, as show in Figure 2.22c, which can led to a more challenging segmentation task.

2.5.6 Conclusion

In this section, we have described our LDD dataset, which contains images coming mostly from the environment in which the vines are cultivated and catalogs the objects found inside the images through bounding boxes and segmentations in the COCO format, which is common in the state of the art.

For future developments, we need to increase the amount of images, in particular for those categories that are not enough represented like *Powdery (Oidio)* and *Downy (Peronospora)* mildews on both grapes and leaves. The final goal is to collect at least the amount of 500 images of each disease in order to become the first benchmark dataset for the most common instance segmentation architectures for this kind of application. We hope that our contribution will lead to new developments on the instance segmentation task in disease diagnosis on grape leaves and bunches.

Acknowledgments

We acknowledge Horta s.r.l for the collaboration and help to build the dataset.

Figure 2.23: Samples of annotated images in LDD.

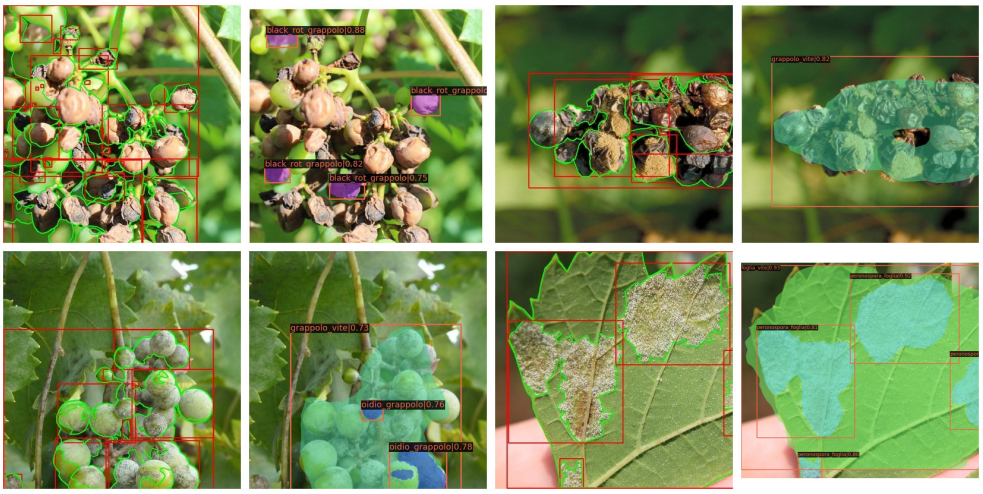


Figure 2.24: R^3-CNN prediction examples (right) compared with ground-truth (left).

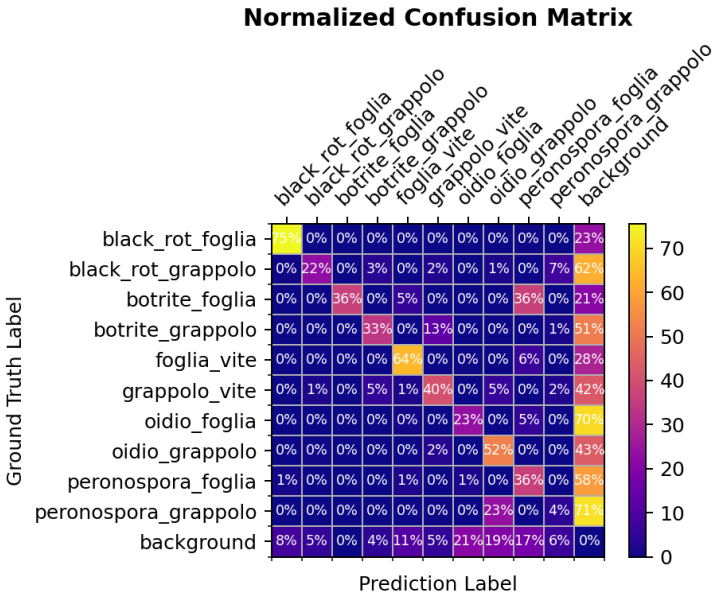


Figure 2.25: R^3-CNN confusion matrix.

2.6 A new dataset for Clothing Detection for Semi-Supervised Learning

The first step does not take you where you want, but it takes you away.

Alejandro Jodorowsky

2.6.1 Introduction

For companies that produce clothes it is very important to know what customers think about their products, to have an idea of what the distinctive characteristics of their brands are, and also to be able to plan future production. Social networks offer a huge amount of useful information that must be pre-processed and aggregated in a rational way. In particular, the social network Instagram offers the possibility to search based on hashtag to direct the search. These hashtags are additional information added directly by users of the social network itself. The information inside a post are many, but the most relevant for our scope are the image and the comments related to it. This image can have potentially any visual characteristic and could contain the most varied objects. First, an image processing phase should happen, consisting in understanding what it contains and, in a more advanced phase, extracting a conceptual map also of the correlations between the objects themselves. Then, the objects connection with the text would be analyzed.

The purpose of the following dataset is precisely cover the first step described before. That is, to be able to offer a set of images taken from the social network and catalog them with respect to a certain number of predefined classes. The defined classes are selected directly in collaboration with the Adidas company. In addition, the dataset has to cover the additional case of training with a Semi-Supervised Leaning setting. This is extremely important case study, because when an image cataloging project is in the early stage, the biggest problem is labeling new images, given that it is a time-consuming and costly process. On the contrary, getting new images is

very simple. This use case is usually covered by a particular type of training called Semi-Supervised Learning (SSL). For this reason, the dataset defines a set of configurations useful for testing models in a SSL settings. In the following pages, we will describe them more in details.

2.6.2 Related

Name	Images (k)	Classes	Source	Key features
ModaNet	55	13	chictopia.com	There is always one person wearing the objects.
DeepFashion	800	13	online shopping websites (Forever212, Mogujie)	Well-posed shop images to unconstrained consumer photos with low resolution (256×256).
DeepFashion2	491	13	DeepFashion, online shopping websites.	
Fashionpedia	48	46	Flickr	

Table 2.31: Fashion dataset comparison.

In Table 2.31 are reported some of the most recent fashion datasets which deal with the object detection topic. The dataset Modanet [91] collects images from the website *chictopia.com*, where usually there is a person wearing clothing belonging to the defined class. The Deepfashion [92] dataset which contains images from *Forever212* and *Mogujieonline* shopping websites. The DeepFashion2 [93], which extends previously cited dataset. Finally, Fashionpedia [94] which contains images taken from Flickr website. We could say that most of them already do a good job, but they are not enough for our particular needs. The first reason is because our research commitment points especially towards a particular domain, which is one of most used social network nowadays: Instagram. Because this social network is used especially by customers, we could deal with the so called "*into the wild*" case, where images could have any possible resolution, number of objects, light, etc. The second reason is because the classes of garments we are interested in not always are present in the datasets previously described. Because all these reasons, we decided to face up the creation of this new dataset, called *ASND*.

2.6.3 ADIDAS Social Network Dataset

The dataset contains 3,999 RGB images and 15,545 annotations in COCO-like [4] format for the tasks of Object Detection. The classes defined were selected directly in collaboration with the Adidas company. The images are coming from Instagram public pages and they have been collected with our scraper [4] written from scratch. It permits to search by hashtag and download the entire post automatically. Usually, the hashtag used to search images was *#adidas*. In Figure 2.31 we can see some images collected for each class.

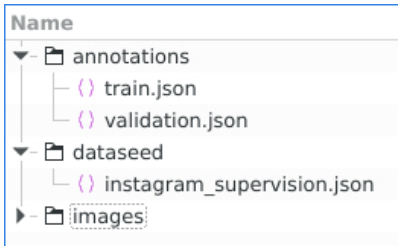


Figure 2.26: ASND directory composition.

In Figure 2.26 the directory structure is highlighted. It is composed by: annotations directory for training and validation dataset in COCO-like format (json files), the *images* directory, containing the images in *jpg* format, and the dataset seed containing the image ids for the semi-supervised learning settings (see section 2.6.3 for details). In Table 2.32 the ids, name and the total available annotations are shown.

With the help of the *cocosplit* [5] utility, we split the dataset in 80% for training and 20% for validation.

[4] https://github.com/hachreak/scraper
[5] https://github.com/akarazniewicz/cocosplit

	Key	Count	Label
0	B	361	Bras
1	J	1193	Jackets
2	P	2855	Pants
3	S	6615	Shoes
4	SH	964	Shorts
5	SD	1043	Skirt/Dresses
6	SL	1174	Sweatshirts(LONG SLEEVES)
7	T	1417	T-Shirt(SHORT SLEEVES)
8	TS	923	Tanks(SLEEVELES)
		15545	Total

Table 2.32: ASND annotations description.

Annotation procedure

To make the annotations, we made use of *Label Studio v1.5* [6] with the following configuration:

Listing 2.2: "Label Studio Object Detection configuration"

```
<View>
    <Image name="image" value="$image"/>
    <RectangleLabels name="label" toName="image">
        <Label value="Shoes" background="#FFA39E"/>
        <Label value="Shorts" background="#D4380D"/>
        <Label value="Pants" background="#FFC069"/>
        <Label value="Bras" background="#AD8B00"/>
        <Label value="Tanks(SLEEVELES)" background="#389E0D"/>
        <Label value="T−Shirt(SHORT SLEEVES)" background="#5CDBD3"/>
        <Label value="Sweatshirts(LONG SLEEVES)" background="#096DD9"/>
        <Label value="Skirt/Dresses" background="#ADC6FF"/>
        <Label value="Jackets" background="#9254DE"/>
    </RectangleLabels>
</View>
```

[6] https://labelstud.io/

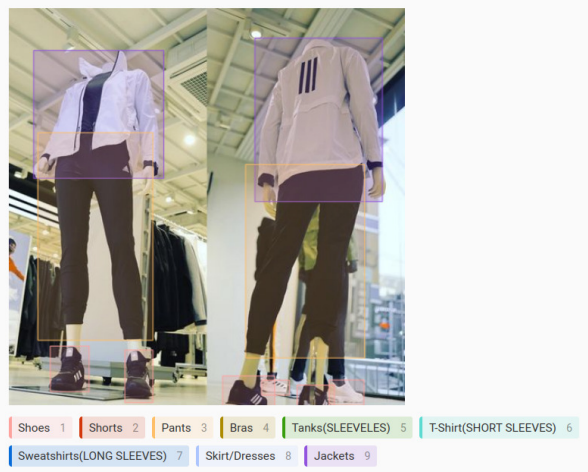


Figure 2.27: Label Studio web interface.

Example of web interface can be seen in Figure 2.27 where, for each image, the user has to select the proper category and, then, the bounding box which contains the instance.

First, the annotations has been created by a student. Then, to improve the quality, a review step has been done by another author. At the end, the annotations have been exported in COCO-like format.

Semi-supervised Learning settings

Usually, a Semi-supervised Learning setting implies to have two dataset for training: the first, which contains the images and also the ground-truth, and the second, which contains only the images, without any additional information. We used *cocosplit* again to split the training dataset in the two parts: supervised dataset (D_s) and unsupervised dataset (D_u). The following are the configurations taken into account to train a model on these Semi-Supervised Learning settings: 5%, 10%, 20% and 50%. E.g. for the last case we have 50% of all annotations in D_s and the rest in D_u.

In Table 2.33 are collected all useful information about the number of annotations

per class.

%	# files	# labels	B	J	P	S	SH	SD	SL	T	TS
5.0	159	612	13	36	80	228	50	39	54	82	30
10.0	319	1176	25	74	134	509	75	73	81	114	91
20.0	639	2633	65	164	264	1277	166	176	182	210	129
50.0	1599	6267	110	518	732	2803	336	375	477	544	372
100.0	3199	12361	277	980	1501	5292	718	821	925	1082	765

Table 2.33: ASND Semi-Supervised Learning settings for D_u.

Statistics

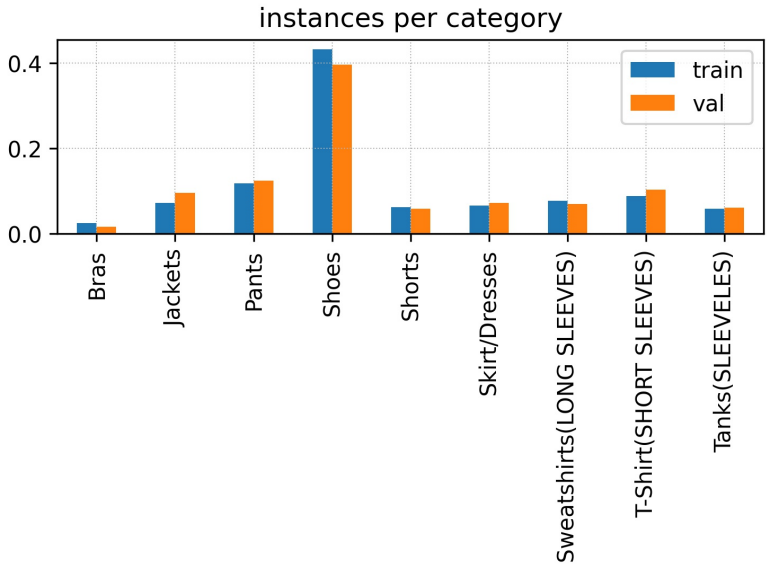


Figure 2.28: Annotations per image: percentage of images in ASND dataset.

In Figure 2.28, we show the percentage of instances for each category. Apart for shoes class, the others are almost balanced in number of annotations. As shown in Figure 2.30a, very few images contains all the categories. Half of the images contain

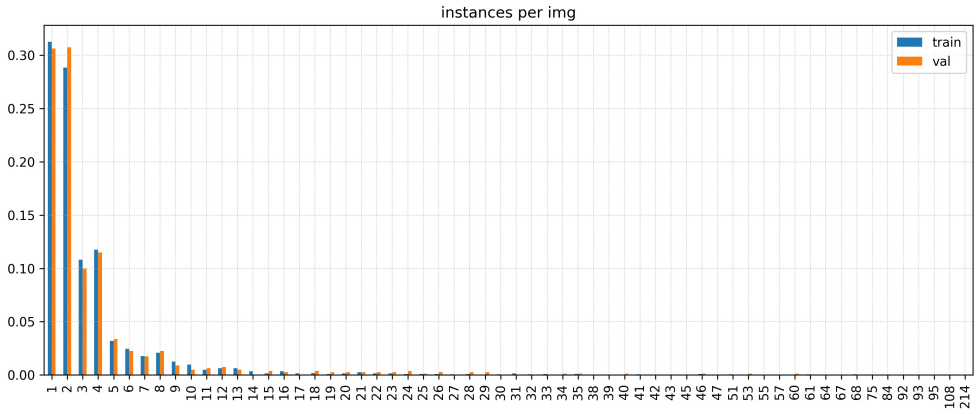


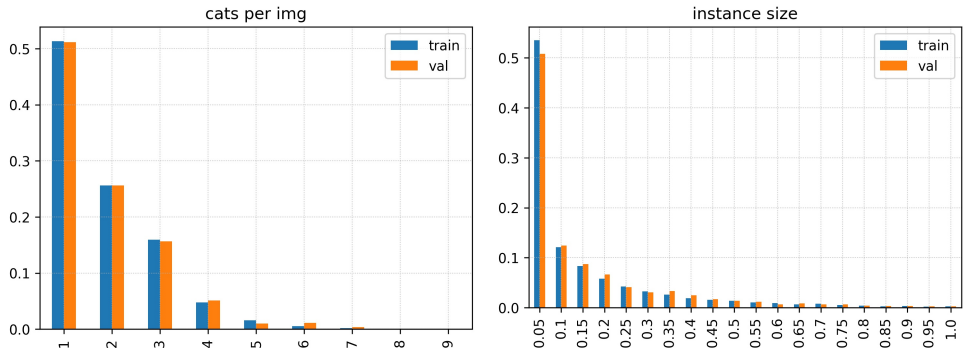
Figure 2.29: Percentage of annotated instances per image in ASND.

one single class, then less then 30% only two and so on, decreasing exponentially. A similar trend could be found in Figure 2.29, where we show the number of instances contained in a image. Very few images contain more than thirteen bounding boxes. Anyway, there are few of them that could have more than fifty annotations, up to 214 annotations. Finally, in Figures 2.30b, we can see how big the bounding boxes are respect to the image size. We see that 50% of them are quite small, covering less than 0.05% of the image size. The rest of annotations are distributed between all percentage values, decreasing with the increasing of the percentage.

2.6.4 Experiments

In the following experiments, we tested the dataset in multiple Semi-Supervised Learning settings. Firstly, it is interesting to observe what is the baseline for a state-of-the-art semi-supervised model [7], which use a Faster R-CNN network [25], a well-known network in the literature. This could give us a baseline for the performance and know what is the minimum quantity of images needed to complete a training without diverge.

In Figure 2.32a the results of the training in multiple Semi-Supervised Learn-



(a) Categories per image

(b) Instance size respect to image size

Figure 2.30: Dataset ASND: (2.30a) Percentage of annotated categories per image. (2.30b) The distribution of instance sizes. Better seen in color.

ing settings are shown and compared with a supervised training only. As expected, increasing the size of the supervised dataset (the dataset with the ground-truth) respect to the unsupervised dataset, greatly increase the performance. Below 5% the training fails due to lack of images. This could be understandable, because we are dealing with a dataset that could be considered small if compared with e.g. COCO dataset [4], which contains more then 117'000 images. What we did not expect is that already with only 50% of the images labeled we reached almost the same performance of a simple supervised training with all the ground-truth available. This could be a *shadow* effect of the EMA training, which permits a new degree of freedom respect to a training where we can modify only the learning-rate. EMA has a smooth effect on training and permits to obtain the best from the student despite the noise introduced by the teacher respect to the real ground-truth. In Figure 2.32b, the comparison of Supervised Learning and the 50% SSL settings is proposed in details.

2.6.5 Conclusions

In this section, we introduced a new dataset for the task of object detection for clothes, with the images coming from the users of the Instagram social network. They have

been all labeled respect to nine defined categories and made available in COCO-like format for Supervised Learning training and also for multiple Semi-Supervised Learning settings. We made available a baseline with state of the art model for Semi-Supervised Learning. For future works, we need to extend the dataset and annotate more images. In addition, we would like to include also some more advanced tasks, like Instance Segmentation. Finally, it would be interesting to investigate even more in detail the effect of EMA on training which, in our opinion, was decisive to achieve comparable performance with supervised training, but with only half of the annotations.

Acknowledgments

We acknowledge Adidas for the collaboration and help to build the dataset.

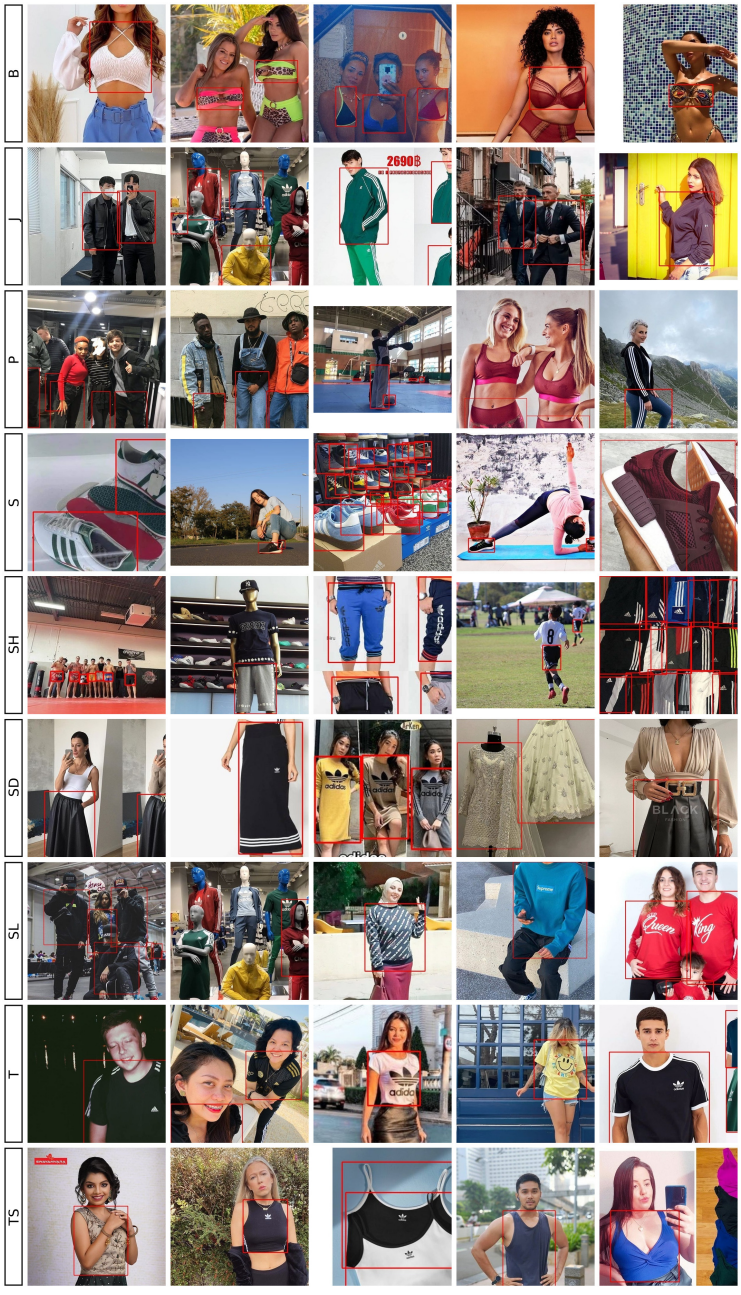
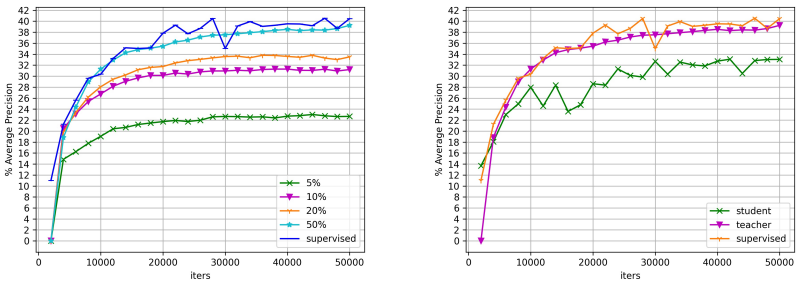


Figure 2.31: Samples of annotated images in ASND.



(a) SSL settings vs SL.

(b) 50% Teacher-Student vs SL.

Figure 2.32: Ablation studies: (2.32a) Faster R-CNN training on ASND dataset in Supervised and multiple Semi-Supervised Learning settings. (2.32b) Compare in details Supervised Learning with Teacher and Student on 50% Semi-Supervised Learning setting.

2.7 A Novel auxiliary self-learning task for Instance Segmentation

I don't need a friend who changes when I change and who nods when I nod; my shadow does that much better.

Plutarch

2.7.1 Introduction

The Instance Segmentation task, an extension of the well-known Object Detection task, is fundamental in many areas. One of these application fields that is gaining momentum is *precision agriculture*, where being able to automatically identify the fruits and the diseases associated with them, allows to effectively scale and automate the surveillance and their protection from pathogens and insects that could affect the plants growth. In this thesis we will focus on a particular plant, namely the vineyard. Nowadays enormous progress has been made in this field that guarantee its performance, especially in the context of deep learning, which has increasingly taken hold as a method capable of adapting to many challenging situations. Despite this, there is still a lot to do to ensure the robustness of these models and the scientific community works hard every day to improve them in many aspects such as precision, recall but also speed of execution and scalability.

The objective of this section of the thesis is to measure the effectiveness of models such as the Mask R-CNN [14] and R^3-CNN [2] into the *precision agriculture* territory, because they represent the, constantly evolving, state of the art. Starting from these models, we have tried to extend them to achieve even higher performance. This research led us to the definition of what we have called the "container and content" auxiliary self-training task, or simply C^2SSL. As the name suggests, a new type of self-training task has been defined, auxiliary respect to the main one, where the ground truth is automatically generated online during training. This new branch of

the network has been tested on the LDD dataset, defined in section 2.5, and takes advantage of a characteristic of the classes within it. Since diseases of leaves and grape bunches are always contained within the leaf or grape bunches themselves, we have used this additional information to define our new task.

The main contributions of these following sections are the following:

- a new auxiliary self-training classification task to identify, on one side, if leaves or grape bunches are healthy or sick and, on the other side, if the detected disease is correctly classified together with the leaf or grape bunch that contains it.
- an exhaustive ablation study on all the components of the architecture.

2.7.2 Related

Many different self-training task have been defined with the objective to improve the training without the need to manually define new labels [95]. One well-know technique involve the rotation prediction applied to the input image [96]. In [97], authors use an hole-filling self-training task to train an autoencoder. In [98], authors defined a Jigsaw Puzzles to pre-train the network and use the learned features in a different task. In [99], authors defined three new auxiliary self-training tasks recycling the bounding boxes: multi-object labeling, closeness labeling and foreground labeling. In our work, we have followed a similar idea but defining a different self-training task. A different approach has been used in [100], where authors included a new network to generate labels for the introduced auxiliary task. In a context of semantic segmentation, authors in [101] reused existing segmentation labels to create a new task which improve the main task.

2.7.3 Container and Content Self-Supervised Learning Task

The LDD dataset contains bbox and segmentation of leaves, bunch of grape and their diseases (see Section 2.5 for more details). The structure of the dataset provides us with one more relevant intrinsic information that can be exploited for the purpose of

general training improvement. We are talking about *container and content* information. If we consider the following two distinct sets "leaves and bunches of grapes" and "diseases", we realize that the segmentation of an instance of the second type always falls within the segmentation of an instance of the first type.

Hence, we can highlight two distinct situations. The first, called *container* situation, where an instance of the first type does contain or does not contain at least one instance of the second type. In other words, we could teach the network to recognize a healthy grape or leaf from a sick one. The second, called *content* situation, where an instance of the second type is contained or not inside an instance of the first type. In other words, we could teach the network to recognize those anomalous cases where the disease has been detected but not the corresponding leaf or grape.

Since the *container and content* information is not explicitly defined in the dataset, we have integrated an online system for computing the ground-truth during the training itself. For each bounding box in the proposal list coming from RPN, the system compute the Intersection over Foreground (IoF) in one of the following way:

1. if the object detected is a leaf or grape (*container* situation), then we compute the IoF between it and all ground-truth bounding boxes belonging to a leaf diseases or grape diseases, depending if its category is leaf or a grape.

2. if the object detected is a disease (*content* situation), then we compute the IoF between it and all ground-truth bounding boxes belonging to a leaf or grape, depending of the type of diseases.

This is expressed in the following formula:

$$IoF_i\left(b_i^t, g\right) = \begin{cases} \frac{Area_i \cup Area_i^{gt}}{Area_i} & \text{if } bbox_i \text{ class is container} \\ \frac{Area_i \cup Area_i^{gt}}{Area_i^{gt}} & otherwise \end{cases} \tag{2.13}$$

$$IoF_i = \max IoF\left(b_i^t, g\right) \quad \forall b_i \in \mathbf{B} | IoF\left(b_i, g\right) \geqslant u$$

After that, for each proposal, we select the max IoF_i value. Finally, we compute the ground-truth t_i comparing the IoF_i with the predefined threshold μ as described

in the following formula:

$$t_i = \begin{cases} 1 & IoF_i > \mu \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

In Figure 2.33 the proposed architecture for the new branch is shown. Inside the detection branch, we connected C^2SSL to the output of the shared fully-connected layer.

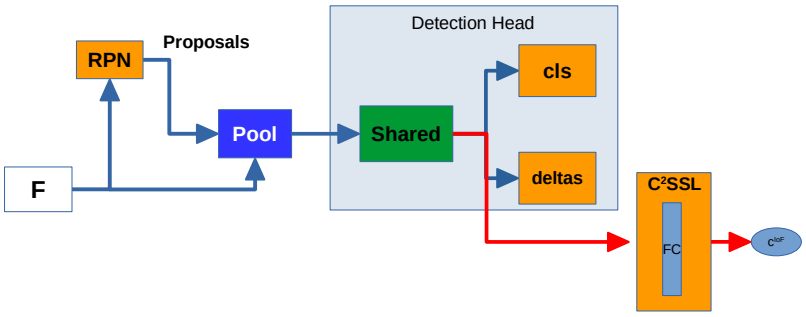


Figure 2.33: Faster R-CNN detection head architecture with our C^2SSL branch in red.

The loss computed between the output c_i^{IoF} and the ground-truth t_i is the weighted focal loss [82] (FL), where ω_i are the class weights previously computed by us to rebalance the loss respect to the classes.

$$L_{cc} = -\sum_{i=1}^{n} \omega_i * FL(x_i, t_i) \tag{2.15}$$

In this case, we defined four distinct classes: leaf or grape is healthy, leaf or grape is sick, disease is isolated (out of any leaves or grapes), disease is inside a leaf or grape.

To choose ω_i values, we monitored the quantities of examples per category we have during the training. In Figure 2.34 the mean value per epoch for each category

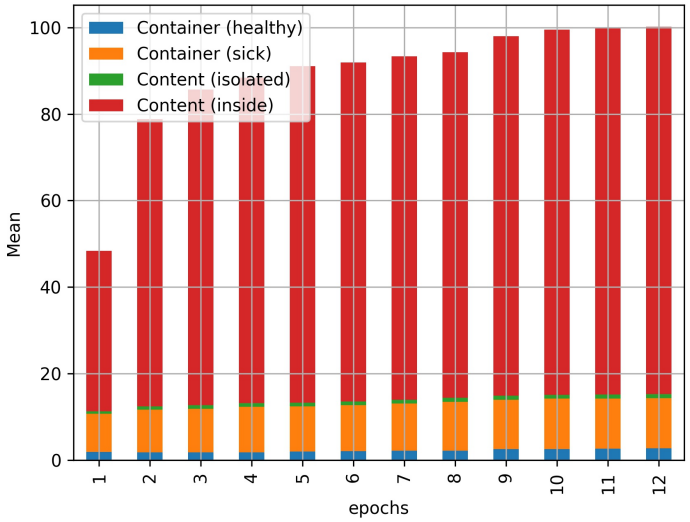


Figure 2.34: Average number of self-generated ground-truth per iteration.

is shown. We used the following formula to compute the final weight ω_i values:

$$\begin{aligned} \text{Rest}_i &= \frac{\sum_{j=1}^{n} \text{mean}_j - \text{mean}_i}{\sum_{j=1}^{n} \text{mean}_j} \\ \omega_i &= \frac{\text{Rest}_i}{\sum_{j=1}^{n} \text{Rest}_i} \end{aligned} \tag{2.16}$$

Where mean_i is the average number of ground-truth for the *i-th* category considering all twelve epochs, and n is the number of categories (four in our case). We found the following values: 0.33, 0.29, 0.33, 0.05.

2.7.4 Experiments

Dataset. We perform our tests on LDD, which contains 881 images, 14'721 annotations and 10 classes. See Section 2.5 for details.

Evaluation Metrics. All the tests are done on LDD validation dataset, which contains 212 images and 3'234 annotations. We report mean Average Precision (mAP) and AP_{50} and AP_{75} with 0.5 and 0.75 minimum IoU thresholds, and AP_s, AP_m and AP_l for small, medium and large objects, respectively.

Implementation details. All the values are obtained running the training with the same hardware and hyper-parameters. For each configuration, we run three times and compute the AP average value. When available, the original code released by the authors is used. Our code is developed on top of the MMDetection [49] source code. We perform the training on a single machine with 1 NVIDIA GeForce RTX 2070 with 8GB of memory. The train lasts 12 epochs with a batch size of 2 images. We use the Stochastic Gradient Descent (SGD) optimization algorithm with a learning rate of 0.00125, a weight decay of 0.0001, and a momentum of 0.9. The learning rate decays at epoch 8 and 11. We use the ResNet 50 [24] backbone for the models, initialized by pre-trained on ImageNet.

#	Method	Loss	Bounding Box						Mask					
			AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
1	Mask R-CNN		20.73	36.60	21.20	9.00	17.80	23.07	19.90	35.37	20.63	8.80	16.40	22.17
2	R^3-CNN		22.77	38.47	23.07	9.60	20.20	29.00	22.10	36.57	24.43	9.50	19.43	29.57
3	R^3-CNN + Container	FL	23.20	38.63	24.10	9.53	19.77	30.73	22.60	36.93	**25.93**	9.50	19.67	30.43
4	R^3-CNN + Content	FL	23.10	38.13	24.83	9.33	20.40	29.37	22.27	36.43	25.43	9.23	19.77	28.43
5	R^3-CNN + Both	WCE	23.47	38.67	24.50	**9.93**	19.93	30.17	22.63	37.03	24.70	**9.77**	19.10	30.23
6	R^3-CNN + Both	FL	**23.76**	**39.12**	**24.88**	9.86	**20.92**	**31.26**	**22.82**	**37.30**	25.28	9.70	**19.82**	**30.52**

Table 2.34: Performance of Mask R-CNN and R^3-CNN models with container/contained branch. WCE: Weighted Cross Entropy, FL: Focal Loss. Bold values are best results, red ones are second-best values.

Ablation study: losses comparison

In Table 2.34, the comparison of our model with Mask R-CNN and R^3-CNN is shown. Different loss modalities are investigated. In the first configuration (row #3) the loss is applied only in the case of *container* case, detecting healthy / sick plants with a focal loss [82]. In the next example (row #4), the loss applied only in the case

of *content* case, detecting if the diseases are in or out of the plant with the Focal Loss. Finally, we test the use of both self-supervised losses (row #5 and #6), with a weighted cross-entropy and the Focal Loss, respectively. We achieve the best results using both self-supervised losses, increasing performance by a +1.0% and 0.73% for object detection and instance segmentation, respectively.

Ablation study: threshold μ

#	μ	Bounding Box						Mask					
		AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
1	0.35	23.02	38.56	22.88	9.72	20.14	29.90	22.42	36.94	24.78	9.58	19.16	29.96
2	0.45	**23.76**	**39.12**	24.88	9.86	**20.92**	**31.26**	**22.82**	37.30	**25.28**	9.70	**19.82**	**30.52**
3	0.55	23.22	38.56	24.28	9.84	19.66	30.44	22.48	36.86	24.80	9.62	19.00	30.18
4	0.65	23.34	38.84	23.74	9.76	19.92	30.02	22.64	37.12	24.50	9.66	19.24	30.14
5	0.75	23.57	38.73	**25.25**	9.88	19.75	30.82	22.65	37.12	25.07	9.70	19.15	30.32
6	0.85	23.20	38.80	24.00	**9.97**	19.82	30.20	22.57	**37.45**	24.55	**9.88**	19.30	29.98
7	0.95	23.45	38.73	24.98	9.55	19.48	31.20	22.60	37.00	25.05	9.40	18.88	30.25

Table 2.35: Performance of R^3-CNN model with both container/content losses but varying the threshold μ value. Bold values are best results, red ones are second-best values.

In this experiment we test how much the performance are influenced by the threshold μ (see Formula 2.14). We use as baseline the previously winner model (see row #5 in Table 2.34). In Table 2.35 the results are shown. We can not define a unequivocal trend on results, but this could be also explained because the size of the dataset. We still can find a winner in row #2. Anyway, independently from the threshold, always the network gets benefits.

2.7.5 Conclusions

In this thesis, we proposed a new architecture for the Instance Segmentation task to exploit intrinsic characteristics of the LDD dataset, previously defined in Section 2.5. The dataset offers segmentation for leaves, grapes and height different diseases of vines. We found a way to build a new ground-truth information online during the

training, which gives us information about two new situations: the so called *container* situation, to classify the leaf or grape as healthy or sick, and *content* situation, to classify the diseases isolated or inside the vine. This new detection branch offer the opportunity to enrich the training with a new auxiliary self-training task, which proved to be useful to increase the performance, simply forcing an additional constraint on the search for the optimal values of the network weights. Indeed, our new architecture increases the AP value by a +1.0% and 0.72% for Object Detection and Instance Segmentation, respectively. It is worth notice that, the novel branch is used only in training, hence not computationally affects the evaluation phase.

Chapter 3

Conclusions

Success is not final, failure is not fatal: it is the courage to continue that counts.

Winston Churchill

In this thesis, the topics of Object Detection and Instance Segmentation have been addressed under different aspects. The most important subjects can be summarized in dataset creation, supervised, semi-supervised and self-supervised learning settings and architectural innovation. Firstly, in Section 2.1 a novel RoI extraction layer, called GRoIE, is proposed with the aim of a more generic, configurable and interchangeable framework for RoI extraction to tackle the Feature Level Imbalance (FLI) problem. In Section 2.2 the R^3-CNN model is defined by a loop architecture, to solve the IoU Distribution Imbalance (IDI) problem in the RPN generated proposals. Then, in Section 2.3 we deeply investigated about the origin of the IDI problem and defined the SBR-CNN architecture. This section contains the following contributions: (i) the review of the GRoIE architecture and the proposal of a new more performing one, (ii) the FCC heads which confirms and extends empirical results to define the rules to follow for an optimal architecture, towards a fully-convolutional approach, and (iii) the extension of the R^3-CNN model.

In Section 2.4 the new IL-net architecture for Semi-Supervised Learning is pro-

posed, which introduces the regression losses on the unsupervised learning part, and a new branch to classify the quality of the bounding boxes and, consequently, filter the pseudo ground-truth generated by the Teacher. In Section 2.5 and 2.6 the datasets Leaf Diseases Dataset (LDD) and ADIDAS Social Network Dataset (ASND) are described, respectively. Finally, in Section 2.7 the new auxiliary self-learning task C^2SSL is described with the purpose of enhancing the instance segmentation training on vines diseases detection and segmentation, building a new ground-truth online with the information of *container* and *content* classes.

To conclude, we can summarize the lessons learned as follows. Although neural networks remain a phenomenal tool, much work still needs to be done to make them less as black boxes. However, it is possible to identify the hidden problems through the identification of checkpoints that allow a deeper observability. Usually, hidden problems are always related to some kind of imbalance. One of the major problems is precisely to build these diagnostic tools and choose the relevant information to monitor in order to reconstruct the internal state of the system. The most problematic issues addressed are the Objective Imbalance in IL-net, the Feature Level Imbalance for GRoIE, the IoU Distribution Imbalance and Exponential Vanishing Ground-Truth problems for R^3-CNN. Of course, once a problem is identified, it is easier to identify the solution as well.

The other *take away* conclusion is connected with the Multi-Task Learning. Even if only in the training phase, the introduction of new related tasks has always had the power to improve the performance of the main task. We have seen this powerful tool in action during the description of IL-net and C^2SSL, where two new self-supervised tasks have been added, with the advantage of not requiring any additional labels. Obviously, *all that glitters is not gold* and requires that the definition of a new task makes sense for the network and adds new constraints to the loss function constructively, allowing to discard a-priori some local minima and allowing to move towards the optimal solution more quickly.

Chapter 4

Future works

As for the future, your task is not to foresee it, but to enable it.

Antoine de Saint Exupéry

A lot of works have been done and even more could be do to find new ways to improve models. Regarding GRoIE, new configurations can be explored, in terms of components, exploring how the architecture and layers could affect final result. For FCC, it is necessary to go deeper into the analysis and exploit the previously acquired empirical knowledge and look for a more generic rule for layer selection depending on the task. In case of R^3-CNN, it would be advisable to test its performance inside new models that contains vision transformers [102] to verify that it works as well as in the examined cases. The analysis of the weaknesses of the RPN highlighted the need to replace the anchor system with an anchor-free [18]. Future models will certainly do without it, so we will need to investigate new effective methods to improve training and find a way to move R^3-CNN into this research direction. Another important research path is represented by the C^2SSL model, which will need to be extended and generalized for a generic instance segmentation dataset, regardless from the a-priori knowledge of the interdependency between the classes. Semi-Supervised Learning could be considered a very important topic today, but still to be adequately

explored. In particular, IL-net needs to be extended and new techniques to catch errors in pseudo-labels need to be designed. For instance, the ambiguous pseudo-bbox concept could be reused from the Selective Net [103] model and integrated with our IoU quality score. Furthermore, the effect of the noise inside labels is a very intriguing and full of pitfalls topic that needs to be developed and deepened. Authors in [33] scratched the surface with an interesting dualism between knowledge distillation and label smoothing regularization, but more work is needed to extend these arguments in different contexts, as the label-noise context. Finally, there is the need to extend the datasets LDD and ASND to include more examples and new classes, as well as adding new tasks.

Chapter 5

Publications

Conference Papers:

1. **Leonardo Rossi**, Akbar Karimi, and Andrea Prati. A novel region of interest extraction layer for instance segmentation. In 2020 25th International Conference on Pattern Recognition (ICPR). IEEE, 2021.

2. **Leonardo Rossi**, Akbar Karimi, and Andrea Prati. Recursively Refined R-CNN: Instance Segmentation with Self-RoI Rebalancing. In Computer Analysis of Images and Patterns (CAIP), Springer International Publishing, 2021.

Conference Papers (Accepted):

3. **Leonardo Rossi**, Akbar Karimi, and Andrea Prati. Improving Localization for Semi-Supervised Object Detection. In 21st International Conference on Image Analysis and Processing, 2022.

4. **Leonardo Rossi**, Marco Valenti, Sara Legler, and Andrea Prati. LDD: A Dataset for Grape Diseases Object Detection and Instance Segmentation. In 21st International Conference on Image Analysis and Processing, 2022.

Journal Papers (Accepted):

1. **Leonardo Rossi**, Akbar Karimi, and Andrea Prati. Self-Balanced R-CNN for Instance Segmentation. In Journal of Visual Communication and Image Recognition, 2022.

Published Collaborations:

6. Akbar Karimi, **Leonardo Rossi**, and Andrea Prati. Adversarial Training for Aspect-Based Sentiment Analysis with BERT. In 2020 25th International Conference on Pattern Recognition (ICPR). IEEE, 2021.

7. Akbar Karimi, **Leonardo Rossi**, and Andrea Prati. Improving BERT Performance for Aspect-Based Sentiment Analysis. In Proceedings of The Fourth International Conference on Natural Language and Speech Processing (ICNLSP). Association for Computational Linguistics, 2021.

8. Akbar Karimi, **Leonardo Rossi**, and Andrea Prati. AEDA: An Easier Data Augmentation Technique for Text Classification. In Findings of the Association for Computational Linguistics: EMNLP 2021. 2021.

Workshop Paper:

9. Akbar Karimi, **Leonardo Rossi**, and Andrea Prati. UniParma at SemEval-2021 Task 5: Toxic Spans Detection Using CharacterBERT and Bag-of-Words Model. In Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), Association for Computational Linguistics, 2021.

Chapter 6

Acknowledgments

This research benefits from the HPC (High Performance Computing) facility of the University of Parma, Italy.
We acknowledge the CINECA award under the ISCRA initiative, for the availability of high performance computing resources and support.
We acknowledge Horta s.r.l for the collaboration and help to build the dataset LDD.
We acknowledge Adidas for the collaboration and help to build the dataset ASND.

Bibliography

[1] Leonardo Rossi, Akbar Karimi, and Andrea Prati. A novel region of interest extraction layer for instance segmentation. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2203–2209. IEEE, 2021.

[2] Leonardo Rossi, Akbar Karimi, and Andrea Prati. Recursively refined r-cnn: Instance segmentation with self-roi rebalancing. In Nicolas Tsapatsoulis, Andreas Panayides, Theo Theocharides, Andreas Lanitis, Constantinos Pattichis, and Mario Vento, editors, *Computer Analysis of Images and Patterns*, pages 476–486, Cham, 2021. Springer International Publishing.

[3] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

[4] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[5] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[6] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*, 2017.

[7] Yen-Cheng Liu, Chih-Yao Ma, Zijian He, Chia-Wen Kuo, Kan Chen, Peizhao Zhang, Bichen Wu, Zsolt Kira, and Peter Vajda. Unbiased teacher for semi-supervised object detection. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[8] Hao Chen, Xiaojuan Qi, Lequan Yu, Qi Dou, Jing Qin, and Pheng-Ann Heng. Dcan: Deep contour-aware networks for object instance segmentation from histology images. *Medical image analysis*, 36:135–146, 2017.

[9] Yun Liu, Krishna Gadepalli, Mohammad Norouzi, George E Dahl, Timo Kohlberger, Aleksey Boyko, Subhashini Venugopalan, Aleksei Timofeev, Philip Q Nelson, Greg S Corrado, et al. Detecting cancer metastases on gigapixel pathology images. *arXiv preprint arXiv:1703.02442*, 2017.

[10] Liqin Huang, Ting Zhe, Junyi Wu, Qiang Wu, Chenhao Pei, and Dan Chen. Robust inter-vehicle distance estimation method based on monocular vision. *IEEE Access*, 7:46059–46070, 2019.

[11] Maoqing Tian, Shuai Yi, Hongsheng Li, Shihua Li, Xuesen Zhang, Jianping Shi, Junjie Yan, and Xiaogang Wang. Eliminating background-bias for robust person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5794–5803, 2018.

[12] Yuanyue Ge, Ya Xiong, and Pal J From. Instance segmentation and localization of strawberries in farm conditions for automatic fruit harvesting. *IFAC-PapersOnLine*, 52(30):294–299, 2019.

[13] Si Liu, Xiaodan Liang, Luoqi Liu, Xiaohui Shen, Jianchao Yang, Changsheng Xu, Liang Lin, Xiaochun Cao, and Shuicheng Yan. Matching-cnn meets knn: Quasi-parametric human parsing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1419–1427, 2015.

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[15] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[16] Kemal Oksuz, Baris Can Cam, Sinan Kalkan, and Emre Akbas. Imbalance problems in object detection: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[17] Guanglu Song, Yu Liu, and Xiaogang Wang. Revisiting the sibling head in object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11563–11572, 2020.

[18] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9759–9768, 2020.

[19] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018.

[20] Yue Wu, Yinpeng Chen, Lu Yuan, Zicheng Liu, Lijuan Wang, Hongzhi Li, and Yun Fu. Rethinking classification and localization for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10186–10195, 2020.

[21] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[22] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4974–4983, 2019.

[23] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European conference on computer vision (ECCV)*, pages 784–799, 2018.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[27] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1440–1448, 2015.

[28] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[29] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005.

[30] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[31] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[32] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6409–6418, 2019.

[33] Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3903–3911, 2020.

[34] Yandong Li, Di Huang, Danfeng Qin, Liqiang Wang, and Boqing Gong. Improving object detection with selective self-supervised self-training. In *European Conference on Computer Vision*, pages 589–607. Springer, 2020.

[35] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[36] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[37] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016.

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

[39] Xin Lu, Buyu Li, Yuxin Yue, Quanquan Li, and Junjie Yan. Grid r-cnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7363–7372, 2019.

[40] Jordi Pont-Tuset, Pablo Arbelaez, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(1):128–140, 2016.

[41] Shaoqing Ren, Kaiming He, Ross Girshick, Xiangyu Zhang, and Jian Sun. Object detection networks on convolutional feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1476–1481, 2016.

[42] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollar. Learning to refine object segments. In *Proceedings of European Conference on Computer Vision*, pages 75–91. Springer, 2016.

[43] Hang Xu, Lewei Yao, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6649–6658, 2019.

[44] Chaoxu Guo, Bin Fan, Qian Zhang, Shiming Xiang, and Chunhong Pan. Augfpn: Improving multi-scale feature learning for object detection. *arXiv preprint arXiv:1912.05384*, 2019.

[45] Tran Duy Linh and Masayuki Arai. Multi-scale subnetwork for roi pooling for instance segmentation. *International Journal of Computer Theory and Engineering*, 10(6), 2018.

[46] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2874–2883, 2016.

[47] Bharath Hariharan, Pablo Arbelaez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 447–456, 2015.

[48] Xizhou Zhu, Dazhi Cheng, Zheng Zhang, Stephen Lin, and Jifeng Dai. An empirical study of spatial attention mechanisms in deep networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6688–6697, 2019.

[49] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[50] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[51] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[52] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[53] Thang Vu, Hyunjun Jang, Trung X Pham, and Chang Yoo. Cascade rpn: Delving into high-quality region proposal network with adaptive convolution. In *Advances in Neural Information Processing Systems*, pages 1432–1442, 2019.

[54] Jiaqi Wang, Kai Chen, Shuo Yang, Chen Change Loy, and Dahua Lin. Region proposal by guided anchoring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2965–2974, 2019.

[55] Qiaoyong Zhong, Chao Li, Yingying Zhang, Di Xie, Shicai Yang, and Shiliang Pu. Cascade region proposal and global context for deep object detection. *Neurocomputing*, 395:170–177, 2020.

[56] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769, 2016.

[57] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra r-cnn: Towards balanced learning for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 821–830, 2019.

[58] Bowen Cheng, Yunchao Wei, Honghui Shi, Rogerio Feris, Jinjun Xiong, and Thomas Huang. Revisiting rcnn: On awakening the classification power of faster rcnn. In *Proceedings of the European conference on computer vision (ECCV)*, pages 453–468, 2018.

[59] Li Zhu, Zihao Xie, Liman Liu, Bo Tao, and Wenbing Tao. Iou-uniform r-cnn: Breaking through the limitations of rpn. *arXiv preprint arXiv:1912.05190*, 2019.

[60] Kemal Oksuz, Baris Can Cam, Emre Akbas, and Sinan Kalkan. Generating positive bounding boxes for balanced training of object detectors. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 894–903, 2020.

[61] Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

[62] Hamed Masnadi-Shirazi and Nuno Vasconcelos. Cost-sensitive boosting. *IEEE Transactions on pattern analysis and machine intelligence*, 33(2):294–309, 2010.

[63] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019.

[64] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proceedings of European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[65] Yudong Liu, Yongtao Wang, Siwei Wang, TingTing Liang, Qijie Zhao, Zhi Tang, and Haibin Ling. Cbnet: A novel composite backbone network architecture for object detection. In *AAAI*, pages 11653–11660, 2020.

[66] Li Zhu, Zihao Xie, Liman Liu, Bo Tao, and Wenbing Tao. Iou-uniform r-cnn: Breaking through the limitations of rpn. *Pattern Recognition*, 112:107816, 2021.

[67] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. *arXiv preprint arXiv:2006.02334*, 2020.

[68] Chaoxu Guo, Bin Fan, Qian Zhang, Shiming Xiang, and Chunhong Pan. Augfpn: Improving multi-scale feature learning for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12595–12604, 2020.

[69] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 282–298. Springer, 2020.

[70] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[71] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NIPS*, 2017.

[72] Laine Samuli and Aila Timo. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations (ICLR)*, volume 4, page 6, 2017.

[73] Jisoo Jeong, Seungeui Lee, Jeesoo Kim, and Nojun Kwak. Consistency-based semi-supervised learning for object detection. *Advances in neural information processing systems*, 32:10759–10768, 2019.

[74] Kihyuk Sohn, Zizhao Zhang, Chun-Liang Li, Han Zhang, Chen-Yu Lee, and Tomas Pfister. A simple semi-supervised learning framework for object detection. *arXiv preprint arXiv:2005.04757*, 2020.

[75] Yihe Tang, Weifeng Chen, Yijun Luo, and Yuting Zhang. Humble teachers teach better students for semi-supervised object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3132–3141, 2021.

[76] Nhu-Van Nguyen, Christophe Rigaud, and Jean-Christophe Burie. Semi-supervised object detection with unlabeled data. In *VISIGRAPP (5: VISAPP)*, pages 289–296, 2019.

[77] Mengde Xu, Zheng Zhang, Han Hu, Jianfeng Wang, Lijuan Wang, Fangyun Wei, Xiang Bai, and Zicheng Liu. End-to-end semi-supervised object detection with soft teacher. *arXiv preprint arXiv:2106.09018*, 2021.

[78] Qiang Zhou, Chaohui Yu, Zhibin Wang, Qi Qian, and Hao Li. Instant-teaching: An end-to-end semi-supervised object detection framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4081–4090, 2021.

[79] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2888–2897, 2019.

[80] Lachlan Tychsen-Smith and Lars Petersson. Improving object localization with fitness nms and bounded iou loss. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6877–6885, 2018.

[81] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[82] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[83] Yali Amit, Pedro Felzenszwalb, and Ross Girshick. Object detection. *Computer Vision: A Reference Guide*, pages 1–9, 2020.

[84] Abdul Mueed Hafiz and Ghulam Mohiuddin Bhat. A survey on instance segmentation: state of the art. *International journal of multimedia information retrieval*, pages 1–19, 2020.

[85] M Alessandrini, R Calero Fuentes Rivera, L Falaschetti, D Pau, V Tomaselli, and C Turchetti. A grapevine leaves dataset for early detection and classifi-

cation of esca disease in vineyards through machine learning. *Data in Brief*, 35:106809, 2021.

[86] Xiaoyue Xie, Yuan Ma, Bin Liu, Jinrong He, Shuqin Li, and Hongyan Wang. A deep-learning-based real-time detector for grape leaf diseases using improved convolutional neural networks. *Frontiers in plant science*, 11:751, 2020.

[87] Thiago T Santos, Leonardo L de Souza, Andreza A dos Santos, and Sandra Avila. Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association. *Computers and Electronics in Agriculture*, 170:105247, 2020.

[88] Haiguang Wang, Guanlin Li, Zhanhong Ma, and Xiaolong Li. Application of neural networks to image recognition of plant diseases. In *2012 International Conference on Systems and Informatics (ICSAI2012)*, pages 2159–2164. IEEE, 2012.

[89] Alvaro Fuentes, Sook Yoon, Sang Cheol Kim, and Dong Sun Park. A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition. *Sensors*, 17(9):2022, 2017.

[90] Florian Rançon, Lionel Bombrun, Barna Keresztes, and Christian Germain. Comparison of sift encoded and deep learning features for the classification and detection of esca disease in bordeaux vineyards. *Remote Sensing*, 11(1):1, 2019.

[91] Shuai Zheng, Fan Yang, M Hadi Kiapour, and Robinson Piramuthu. Modanet: A large-scale street fashion dataset with polygon annotations. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1670–1678, 2018.

[92] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1096–1104, 2016.

[93] Yuying Ge, Ruimao Zhang, Xiaogang Wang, Xiaoou Tang, and Ping Luo. Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5337–5345, 2019.

[94] Menglin Jia, Mengyun Shi, Mikhail Sirotenko, Yin Cui, Bharath Hariharan, Claire Cardie, and Serge Belongie. The fashionpedia ontology and fashion segmentation dataset. *Cornell University*, 2019.

[95] Mahdi Pourmirzaei, Farzaneh Esmaili, and Gholam Ali Montazer. Using self-supervised auxiliary tasks to improve fine-grained facial representation. *ArXiv*, abs/2105.06421, 2021.

[96] Nikos Komodakis and Spyros Gidaris. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations (ICLR)*, 2018.

[97] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[98] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[99] Wonhee Lee, Joonil Na, and Gunhee Kim. Multi-task self-supervised object detection via recycling of bounding box annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4984–4993, 2019.

[100] Shikun Liu, Andrew Davison, and Edward Johns. Self-supervised generalisation with meta auxiliary learning. In H. Wallach, H. Larochelle, A. Beygelz-

imer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[101] Xiaohang Zhan, Ziwei Liu, Ping Luo, Xiaoou Tang, and Chen Loy. Mix-and-match tuning for self-supervised semantic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[102] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[103] Yandong Li, Di Huang, Danfeng Qin, Liqiang Wang, and Boqing Gong. Improving object detection with selective self-supervised self-training. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 589–607, Cham, 2020. Springer International Publishing.