UNIVERSIDADE D
COIMBRA

Diogo João Durães Antunes

THROUGH THE MAGNIFYING GLASS

IMPROVING REAL-TIME SMALL OBJECT DETECTION

Dissertation in the context of the Master in Informatics Engineering, specialization in Intelligent Systems, advised by Professor Luís Paquete and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September of 2022

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

DEPARTMENT OF INFORMATICS ENGINEERING

Diogo João Durães Antunes

Through the Magnifying Glass

Improving Real-Time Small Object Detection

Dissertation in the context of the Master in Informatics Engineering, specialization in Intelligent Systems, advised by Prof. Luís Paquete and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September 2022

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Diogo João Durães Antunes

Through the Magnifying Glass

Improving Real-Time Small Object Detection

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Sistemas Inteligentes, orientada pelo Professor Doutor Luís Paquete e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Setembro 2022

Acknowledgements

I would like to thank both of the advisors responsible for guiding and overseeing the progress of the project and report, Prof. Luís Paquete, as the advisor on behalf of the University of Coimbra, and Rui Lopes, as the internship mentor on behalf of Critical Software.

Abstract

Real-time object detection is widely applied in a variety of contexts to automate systems which require a detailed perception of the objects of interest present in an image. The accuracy and speed of detectors has continued to steadily rise since the implementation of deep learning models for object detection, however, the detection accuracy obtainable for objects of smaller size has always significantly trailed behind average, independently of the used architecture. With this in mind, we sought to improve the accuracy of real-time small object detection by restructuring and tuning the modern real-time object detection model YOLOR, incorporating features of higher resolution extracted by the model's backbone network into the detection component, and generating bounding box anchors tuned specifically for objects of small size. However, these approaches proved to be ineffective in improving the detection accuracy of the baseline models, providing information of limited value when compared to the collection of features previously used, and adding unnecessary complexity to the YOLOR architecture, detrimental to its computational performance.

Keywords

Object detection, real-Time object detection, small object detection, YOLOR.

Resumo

A deteção de objetos em tempo real é utilizada amplamente em vários contextos na automação de sistemas que requerem uma perceção detalhada dos objetos de interesse presentes numa dada imagem. A precisão e performance destes detetores tem vindo a melhorar continuamente desde a implementação de modelos *deep learning* para a deteção de objetos, contudo, a precisão da deteção de objetos pequenos sempre obteve resultados significativamente abaixo da média, independentemente da arquitetura utilizada. Tendo isto em conta, pretendemos melhorar a precisão da deteção de objetos pequenos em tempo real através da alteração do modelo moderno na deteção de objetos em tempo real YOLOR, incorporando *features* de maior resolução extraídas pela rede neuronal convolucional do modelo no componente de deteção, e gerando âncoras para *bounding boxes* adaptadas particularmente a objetos de pequenas dimensões. Contudo, estes métodos revelaram-se inefetivos no melhoramento da precisão dos modelos *baseline* escolhidos, adicionando informação de valor limitado quando comparada com o conjunto de *features* originalmente utilizadas, e resultando numa maior complexidade arquitetural, prejudicial para a performance computacional dos modelos implementados.

Palavras-Chave

Deteção de objetos, deteção de objetos em tempo real, deteção de objetos pequenos, YOLOR.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Object detection is one of the fundamental tasks in the field of computer vision, aiming to identify the objects of interest present in any given image, determining their location, dimensions, and category. Image annotation, automated transportation, surveillance, and mass production are some examples of the industries that stand to benefit from the evolution and successful implementation of object detection technology. Software capable of object detection has the potential to greatly enhance the perceptive abilities of automated systems, allowing them to find and distinguish a known set of relevant entities in their environment and act accordingly.

Object detectors, however, notoriously struggle to correctly detect small-sized objects consistently. Some object detection applications rely heavily on small object detection, as is the case with the analysis of satellite or aerial images, where any object occupies a tiny fraction of the whole image. Improving the detection performance of small objects would not only benefit these applications, but also all of the previously mentioned examples as well (where object size remains more varied), by making object detectors generally more robust to a wider range of object scales.

The core driving force behind this internship was to achieve an improvement in the detection of small common objects, in a real-time context. To this end, we proposed the design of an altered version of a state-of-the-art real-time object detection model, specialized exclusively in the task of detecting small objects.

This chapter will next further clarify the context, necessity and goals of the project in Sections 1.1 and 1.2, additionally presenting a short definition and discussion of the problem in Section 1.3.

1.1 Contextualization

This project was developed within the context of a curricular internship at Critical Software, as a part of the second year of the Master's Degree in Informatics Engineering at the University of Coimbra, with a specialization in Intelligent Sys-

tems.

The ultimate goal of the internship was to contribute to the advancement of real-time small object detection using a state-of-the-art object detection architecture as a basis. Similar projects developed in this branch of the company seek to research and realise innovative artificial intelligence solutions with the potential to be of valuable application in other ongoing projects or future endeavors. In the case of this project, the results obtained could potentially inform a previously active project focused on the detection of small personal objects in a public transport setting, such as small belongings left on tables or chairs. Within the scope of the internship, however, the context considered for the application of the developed models was much more generalized, consisting in the real-time detection of common objects inserted in various locations and scenarios,such as those included in Microsoft's COCO data set [1].

We intended to implement a new version of an existent real-time object detection method, enhanced for the detection results of small object instances, while striving to maintain its detection speed. This was planned so as to preserve the possible real-time applications of the solution, making it a more desirable choice for the majority of contexts.

1.2 Goals

The ultimate goal of the internship was to envision and produce alterations to modern object detection models with the purpose of increasing their small object detection performance. To achieve it, multiple tasks were distributed throughout the available time, split between the first and second semesters. The first half of the project had its focus placed on literature research, with the actual implementation and evaluation of the models being carried out during the second half.

In the first semester, the main objective was to write a review of the current literature on small object detection and define the proposed approach to the problem. In recent years, multiple in-depth surveys and reviews have been published on the topic [2–5]. This task was carried out with the sole purpose of gaining a better understanding of the best-performing methods in the field of object detection and the techniques employed to address the issue of small object detection, and with no intention of providing a more comprehensive or complete compilation of knowledge than previous works, due to scope, resource and time constraints of the project. The knowledge obtained in this stage served to better inform decisions regarding the method to be implemented and the context of its evaluation. The obtained result was a systematic review of the most recent contributions to the evolution of small object detection and object detection as a whole, a list of the most relevant data sets being currently used as detection benchmarks, and an initial proposal for the base architecture and techniques to be used in the new approach. This review is presented in Chapter 2.

It is important to note once more that, while the literature review touches on all aspects of the small object detection problem, the latter half of the internship

specifically aimed at producing models capable of real-time small object detection. The concept of real-time detection is addressed in the literature review, and further details pertaining to the desired context of application for the implemented models is given in Chapter 4. The second stage of the internship was dedicated to implementing and testing the proposed methods, evaluating them in comparison to selected state-of-the-art models. These models were used as the basis for the practical implementation of the planned changes, resulting in a final collection of two base models and multiple modified architectures. Various preliminary tests were performed as a means to evaluate the considered changes, with the final results being obtained towards the end of internship's schedule. The concluding experiments focused on extending the models' training time as much as possible within the remaining available time, so as to obtain a more clear picture of each approach's real potential when given time to develop.

1.3 Problem Definition

Object detection has experienced a steady rise in accuracy and speed over the past years stemming from the evolution of innovative and efficient methods reliant on deep learning. However, detection of small objects has obtained less satisfactory results compared to the detection of large-scale objects with most models [2]. It is important to attempt to narrow this gap in performance as it would not only be crucial to the evolution of certain applications but also lead to improvements in object detection as a whole.

Before attempting to tackle this issue, it is important to clearly establish what defines small objects in the task of object detection. Unfortunately, due to the variety of contexts to which small object detection can be applied, there is no widespread consensus regarding the size limit of small objects. There are, in any case, small object definitions with a higher degree of relevance within each object detection application, as is the case with the one presented in the evaluation metrics for Microsoft's COCO (Microsoft Common Objects in Context) object detection data set [1]. According to this metric, small objects must cover an area smaller than 32^2 pixels. Given the data set's fixed image resolution of 640×480 pixels, this represents roughly 0.32% of the total image area. This definition is of importance considering the popularity and usage of the COCO data set in object detection research, due to its large number of images containing a good variety of object classes in their natural context. Another set of criteria is presented in [6], where a data set tailored for small object detection is assembled. Included objects range in dimensions from 16×16 to 42×42 pixels, or 0.08% to 0.58% of the image area. Examples of the commonly considered size for small object instances are given in Figure 1.1, in the context of traffic sign recognition.

Usually, the measure used to define small objects focuses on either their resolution or the area percentage they occupy in the image that contains them, as these factors are closely related to the lower detection performance in small object detection. Criteria not tied to the image where an object is inserted, such as the object's physical dimensions, can't always be easily obtained and do not affect the

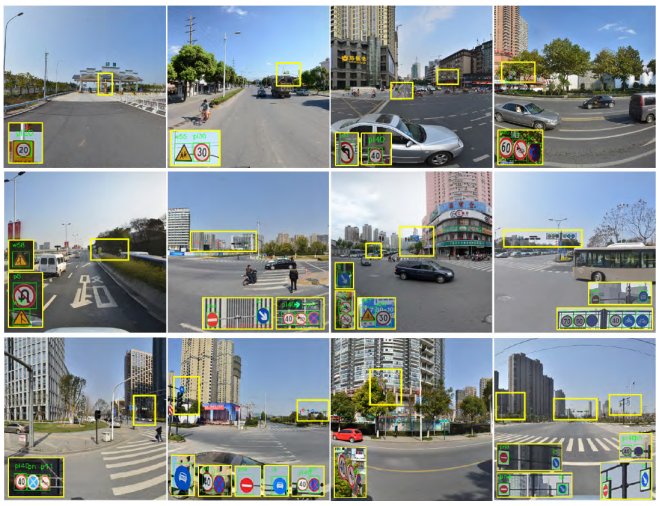


Figure 1.1: Examples of small object instances (traffic signs) in images, as presented in [7].

detection performance directly. The clarity and level detail with which any given object is represented heavily depends on its distance to the camera, and whether it is the focus of the picture in which it was captured, removing the significance of the object's real dimensions.

The accuracy difference that separates the detection of small objects from the rest can be attributed to four key factors [2–5]: the low resolution, the need for precise localization, the distinction from background noise, and insufficient training data.

Low resolution. The main problem with detecting small objects is the lack of information that can be extracted from low-resolution image regions. The finer details of the object are often not discernible at those scales, resulting in the extraction of lower quality features. As will be discussed later in Chapter 2, deep learning approaches to object detection sequentially create lower scale representations of objects in the process of extracting meaningful features, rich in semantic information. Given the already reduced size of small objects, these operations make their final feature maps less adequate for classification, as too much information is lost in the process.

Localization precision. The localization aspect of object detection, referring to the task of identifying an object's position, also offers a greater challenge when dealing with small objects. Traditionally, the metric used to evaluate the placement of bounding boxes during training is based on their area of overlap with the ground truth boxes defined on the training data set. The slight misplacement of a bounding box around a small object results in a more significant loss of information than it would for an object of larger size. The limited resolution of small object regions means that even the smallest of region placement errors can result in a larger part of the object being left out of the proposed area.

Separability from the background. The small dimensions of objects contained

in images can often be attributed to their long distance from the position where the image was captured, which makes small objects generally harder to distinguish from existing clutter or background noise. An object that is far enough away not only loses most of its intricate detail on the resulting image but also has a higher chance of being occluded by a larger object closer to the foreground. Distant objects also noticeably blend in more with the background, making them increasingly harder for an observer to discern their shape and differentiate them from background noise. This can happen partly due to atmospheric factors such as air pollution and humidity, making distant objects appear faded and lower in contrast, an effect easily visible when observing images obtained under foggy weather conditions.

Insufficient training data. Certain applications of object detection, as previously discussed, depend mainly on the detection of small objects. One example would be traffic sign detection, where the recognition of signs should ideally happen at the longest possible distance, to allow the appropriate response to be executed within a safe window of time. Most of the popular data sets for traffic sign recognition, such as Tsinghua-Tencent 100K [8], contain a wealth of smaller traffic sign examples due to this necessity, as well as the road setting depicted in the images obtained. This, however, is not the case with more generic object detection data sets. Data sets comprised of various common objects, like the COCO data set, include labelling to identify differently sized objects, useful when attempting to exclusively evaluate small object detection. Even so, the number of small objects offered by these data sets might not be sufficient for effectively training small object detection models. Additionally, using only certain portions of a larger data set can lead to heavier class imbalances, not present when using it in its entirety. While a data set comprised of various object scales might include a balanced distribution of its object categories, this balance is easily undone by only considering the small-sized objects within it.

Chapter 2

Literature Review

In this chapter, a comprehensive review of the most relevant methods in the recent evolution of small object detection is presented. As to provide further context to these methods, an overview of object detection and some related concepts is also presented, with a focus on deep learning methods. In this overview, an introduction is given to deep learning and convolutional neural networks, followed by a list of the most prevalent network architectures used in object detection models and the most representative methods themselves. Afterwards, the review centres on the problem of small object detection, explaining each of the solutions found to tackle the challenges presented by it. These contributions are ordered chronologically and grouped according to the nature of the approach used in the aforementioned solutions.

2.1 Object Detection

Object detection is one of the core applications of computer vision, having been the focus of a great deal of machine learning research for the last decade. By allowing the correct identification of objects of interest, detected in any position of an image or video, object detection opens up meaningful avenues for systems to more efficiently perceive their environment.

The object detection task is composed of two distinct aspects:

1. **Localization**, as the term suggests, is focused on determining the position of objects of interest. Each object detected is enclosed within a bounding box which encapsulates it, leaving ideally little to no space between its edges and the object.

2. **Classification** is often achieved with the information present within an object's bounding box, from which its category is predicted. The quality and informativeness of the features extracted from the image is paramount to the accuracy of this task.

Object detection also serves as the basis for other computer vision tasks such as object tracking, image segmentation, action recognition, pose estimation, among others.

2.2 Evaluating Object Detection Performance

For the evaluation of object detectors during training, metrics suited to its two main tasks, localization and classification, are required. Traditionally, to judge the overall performance of a detector, mean average precision (mAP) is used. To better understand this measure, it is first necessary to understand the precision and recall of a prediction model.

Given a set of predictions, *precision* represents the percentage of those that are correct. This metric determines the ratio of correct predictions made by the detector and is obtained by dividing the number of correct predictions (true positives) by the total amount of predictions (true positives and false positives). On the other hand, *recall* measures the percentage of positive examples that were correctly identified from the total amount available in the data set. In object detection, recall is obtained by dividing the number of objects correctly categorized by the total number of existent objects, encountered or not by the detector.

It is still necessary to assess the individual placement of bounding boxes in relation to the ground-truth examples during training. This is achieved with intersection over union (IoU), detailed in [9], which represents the overlap between two bounding boxes. It is obtained with the following formula:

$$IoU = \frac{|P \cap T|}{|P \cup T|}$$

, where P corresponds to the area of the predicted bounding box, and T corresponds to the area of its ground-truth counterpart. By applying this metric as a threshold, it is possible to distinguish between correct and incorrect predictions of object locations. As an example, bounding box predictions might be considered correct when the IoU value is above 0.5. This value directly impacts the precision and recall rates. If the IoU threshold is on the higher end, fewer predictions might pass as correct, resulting in a lower recall rate of positive instances, and potentially higher accuracy. Conversely, a relatively low IoU value would not limit the recall rate of the detector but could reduce the accuracy of its predictions, given the inclusion of less reliable bounding boxes.

Average precision (AP) is calculated with:

$$AP = \int_0^1 p(r)dr$$

, where p and r correspond to the precision and recall values of the model respectively. When plotting both of these values in a graph for a set of prediction outputs, sorted by their confidence level, a precision-recall curve is obtained,

commonly depicting precision decreasing as the recall nears the maximum value of 1, being that AP is meant to visually represent the area beneath this curve.

In light of the aforementioned effect the IoU threshold has on precision and recall, AP is usually computed separately for each of its values. IoU levels of 0.5, 0.75, and $0.5 : 0.95$ are commonly chosen for analysis, with the latter being calculated by averaging a set number of levels in between 0.5 and 0.95, obtained with a step of 0.05.

Finally, mAP can derived in the same way as AP, although it specifically refers to the overall result averaged from all of the existent classes.

2.3 Deep Learning for Object Detection

Deep learning is the currently preferred approach to the task of object detection, having undeniably surpassed the previously used methods. Deep convolutional networks represent the most widely used deep learning architecture in this regard, serving as the basis for the most relevant methods in the past decade, such as [10; 11].

In machine learning, deep learning is often described as a method that emulates the learning process of a human brain. In deep learning, networks with a high number of layers, deep neural networks, are employed to process data and extract complex patterns and relations. With the increase in layer depth, the information obtained becomes more abstract and harder to understand, yet potentially more valuable for distinction and classification. The result is often a black box solution to the extraction of meaningful features, where the network's 'logic' can only be partly understood through specific visualization techniques.

To better understand the role of deep learning in object detection architectures, the following sections will explore the types of network usually implemented at the core of object detectors and their variants.

2.3.1 Convolutional Neural Networks

For the task of image processing, Convolutional Neural Networks (CNN) have largely been the favoured deep learning approach. The core concept of this architecture is present in the convolutional layers, responsible for learning small filters that are moved across the image to obtain an output, a process referred to as *convolution*. These filters, also named *kernels*, take on the shape of small matrices with a set of values, and are used to calculate a value for each pixel of the image during the convolution. This value is the dot product between the kernel and the image patch of the same size around a given pixel. This type of localized information processing is ideal for images as it extracts the relations and patterns formed by closely positioned pixels, eliminating the unnecessary task of fully connecting each pixel to every output value.

Between the convolutional layers of a CNN, pooling layers are placed, with the goal of down-sampling the image to a lower resolution. This process is achieved with a variety of functions, the most widely used being the max-pooling method. With max-pooling, for each group of pixels, the one with the highest value is kept for the down-sampled image. The intent of pooling is the reduction of the image's scale for subsequent convolutions, so that higher-level features may be extracted. Instead of considering the correlation between adjacent pixels, useful in the detection of smaller features such as corners or borders, the focus becomes the placement of these features in relation to others. Higher-level features, or deeper features, are utilized to distinguish more complex forms and patterns, ideal for object detection.

Other components relevant in the CNN architecture are the ReLU layers (rectified linear activation unit), the fully connected layers, and the loss function. The ReLU non-linear activation function nullifies negative image values after convolutions. The fully connected layers' function is to predict the object categories given as the output, and the loss function is used to calculate the prediction error of each classification during training in order to adjust the network parameters accordingly.

2.3.2 Backbone Networks

The networks used to initially extract image features in object detectors are referred to as backbone networks. Usually, these networks are inserted as components in detector architectures, being pre-trained on a generic image classification task. Despite a large majority of these networks being CNNs, based on the concepts and components presented in the previous section, recent methods have opted for different architectures, as is the case with the Swin Transformer [12]. Some of the most relevant backbone networks include [12–16]. A brief overview of each of these networks is also given in this section, seeing as they play a major role as central components of any object detector.

VGG [13]. The VGG network is a simple deep CNN architecture, characterized by its depth and small layer receptive fields. It was introduced after the well-known AlexNet [17], and focused on increasing the depth of previous networks while reducing the size of the kernels used in convolution. With these reduced receptive fields, the minimum filter size being 3×3, the network is capable of capturing smaller image details in earlier layers and prolonging itself further horizontally (depth-wise). The VGG-16 architecture is one of the most utilized VGG network layouts, made up of five distinct convolution blocks separated by max-pooling layers, followed by a sequence of three fully-connected (FC) layers (Figure 2.1). The first two convolutional blocks include two convolutional layers each, whereas the three following blocks are composed of three layers each, all followed by a ReLU non-linear activation layer. The fully-connected layers are followed by a final activation layer, commonly a softmax layer.

GoogLeNet [14]. Also known as Inception v1, the first iteration of this architec-

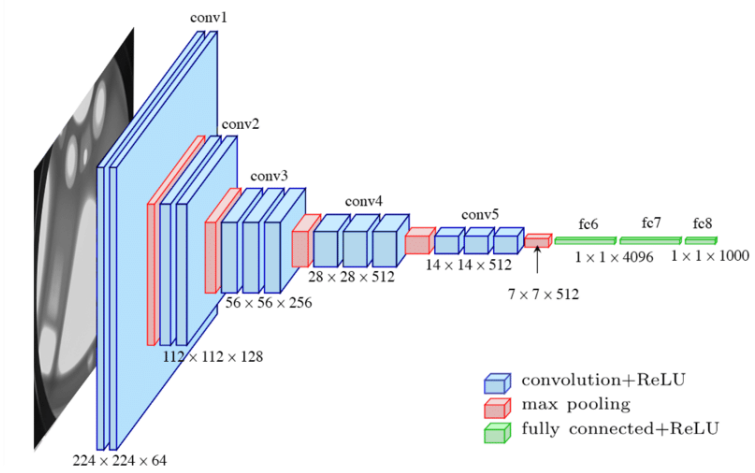


Figure 2.1: Standard architecture of the VGG-16 backbone network [18].

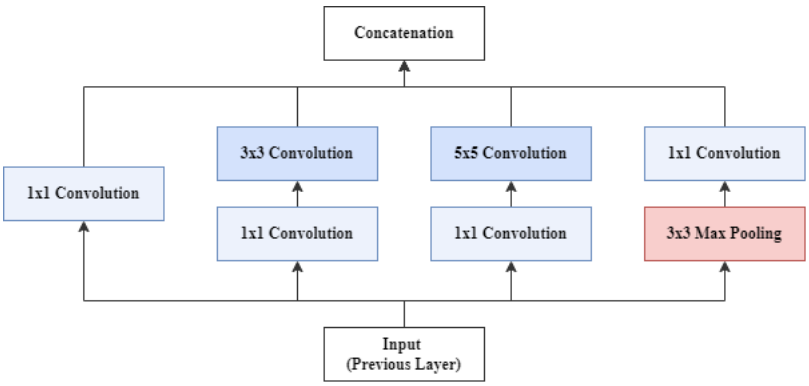


Figure 2.2: Architecture of the inception block with dimensionality reduction.

ture attempted to tackle the issue of objects of multiple scales by introducing parallel convolutions with filters of different sizes. These were inserted in what are referred to as Inception blocks, which initially included convolution operations with 1×1, 3×3, and 5×5 filters, additionally to max pooling. This block also employs 1×1 convolutions as a means to reduce the dimensionality of the input before computationally intensive operations such as 5×5 convolutions (Figure 2.2). Later versions of the Inception network worked to optimize the architecture further, in part by factorizing the aforementioned expensive convolution operations. As an example, it was found that a 3×1 convolution followed by a 1×3 convolution were computationally cheaper than a standard 3×3 convolution while obtaining similar results. Furthermore, 7×7 convolutions were also added with the third iteration of the network, partitioned into smaller convolutions, expanding the network in width instead of depth. The Inception-ResNet network architectures focused on combining the Inception blocks of previous architectures with residual connections, introducing the output of each block into its input.

ResNet [15]. Residual networks were developed with the intent of tackling the problems related to the learning gradient used in deep neural networks. When information is back-propagated to the earlier layers, a gradient is applied in the

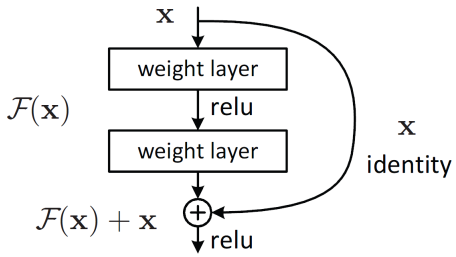


Figure 2.3: Residual block architecture with skip connection [15].

calculation responsible for the alteration of layer parameters. When a high number of layers is in place, this gradient may be exponentially decreased or increased depending on the activation functions in place, resulting in the vanishing and exploding gradient issues respectively, which greatly hinder the training process of the network. To this end, ResNet employs skip connections from shallower layers to deeper layers, which jump over one or more layers (Figure 2.3). With these connections, higher layers can receive information from previous ones, improving the viability of training deeper networks by allowing the gradient to easily propagate across the network without having to traverse it sequentially. This is also referred to as mapping the identity of the network. The model itself is divided into components named residual blocks, where the skip connections are inserted, which can be easily stacked to form increasingly deep networks. ResNet-34, ResNet-50, and ResNet-101 are a few examples of this architecture, differing mainly when it comes to their depth.

DenseNet [16]. Densely connected networks take the concept seen in ResNet further, by propagating any layer's information to all subsequent layers. Likewise, any layer in the network additionally takes as input all of the feature maps produced by previous layers. As a result of this setup, the feedback necessary to alter each layer's parameters reaches it directly from deeper layers. Given the progressive down-scaling of image representations required in computer vision tasks, the DenseNet architecture is made up of what are referred to as dense blocks, wherein layers are densely connected and separated by the typical pooling operations.

Swin Transformer [12]. Contrary to the great majority of backbone networks used in the last decade, the Swin Transformer does not follow a CNN architecture, aiming instead to apply the concept of transformer networks to computer vision tasks. It extracts image features on multiple scales by progressively merging small patches into larger ones throughout the network, as presented in diagram (a) of Figure 2.4. Self-attention is calculated for each patch, in Multi-Head Self-Attention layers [19]. In short, this measure determines not only the value of each patch regarding the presence of relevant features but also its correlation with all other patches. By further dividing the image into multiple windows, each containing numerous patches, it is possible to calculate self-attention values separately within each window, greatly reducing the consumption of resources otherwise required when executing this operation for the entire image at once. These windows are then shifted a certain number of patches, to avoid measuring self-attention repeatedly inside the same isolated image segments. The initial size

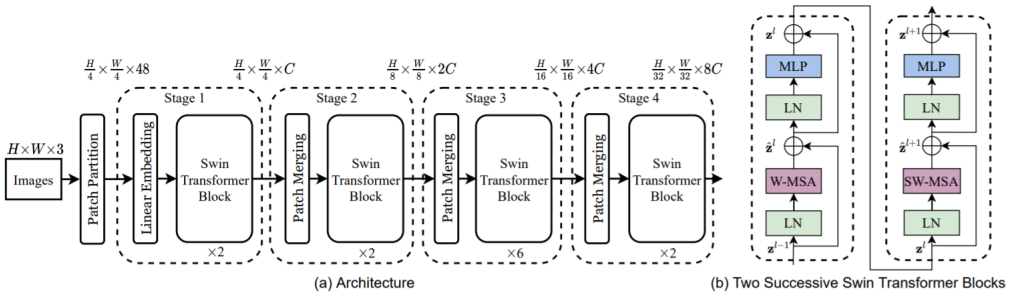


Figure 2.4: Architecture of the Swin Transformer alongside its building blocks [12].

of patches also guarantees that simpler features of lower scale are obtained first, and used to extract higher scale features deeper into the network, producing a result akin to that of down-sampling operations found in CNN architectures. As with other backbone architectures, the Swin Transformer is divided into blocks, represented in diagram (a) of Figure 2.4. The first block is responsible for calculating attention for the initial image patches, while the subsequent blocks start by merging adjacent patches before executing the same operation. Inside each block,as illustrated in diagram (b) of Figure 2.4, the output is obtained as a result of computing self-attention within a set of windows, followed by computing self-attention within the shifted windows. Recently, this backbone architecture has obtained some of the best results in important object detection benchmarks, such as [1], with increasingly more research being dedicated to iterating on the initial proposal.

2.3.3 Deep Learning Object Detectors

With the widespread adoption of deep learning in object detection, two separate types of methods took shape, depending on how the two tasks of object detection were tackled: the "two-stage" detectors, and the "one-stage" detectors. The following sections will discuss some of the most meaningful contributions of the past decade that introduced and iterated over these two distinct approaches, shown in Table 2.1.

2.3.4 Two-Stage Detectors

Two-stage object detectors perform the tasks of localization and classification with separate models. One of the model components is responsible for analysing regions of interest (RoI) within an image and outputting object proposals, depending on the chance of an object being present within each region, regardless of its category. The second component takes the proposals as input and attempts to determine the class of the object in question, improving the accuracy of the bounding box location in the process. Some representative two-stage detectors include [10; 20–23].

Architecture	Method	Year
Two-Stage	**R-CNN** [10]	2014
	SPP-Net [20]	2015
	Fast R-CNN [21]	2015
	Faster R-CNN [22]	2016
	FPN [23]	2017
One-Stage	**YOLO** [11]	2016
	SSD [24]	2016
	YOLO9000 [25]	2017
	RetinaNet [26]	2017
	YOLOv3 [27]	2018
	YOLOv4 [28]	2020

Table 2.1: Representative object detection methods described in Section 2.3.3, separated by architecture type.

R-CNN (Region-Based CNN) [10] pioneered research on this type of approach, introducing convolutional neural networks as a mean to extract meaningful features from region proposals, obtained with the selective search algorithm [29]. This architecture was significantly improved with its subsequent iterations, which aimed to increase the speed and optimize the usage of computation resources of the original model, by reducing the number of convolution operations and employing different training strategies. While R-CNN extracted features from each individual region proposal, **Fast R-CNN** [21] opted to first extract a feature map of the entire input image with a backbone convolutional neural network, obtaining the regions of interest from this representation through an RoI pooling layer. **Faster R-CNN** [22] proposed a Region proposal Network (RPN) to extract regions of interest from the image feature map with the use of predefined anchors. For any given region, the RPN determines the size and placement of object bounding boxes, but an "objectness" score for each of those regions, which refers to the likelihood of it containing any object from a finite set of classes versus the background. The anchors used to obtain these results, overlaid on the input image as small windows, are built according to different object scales and shapes, being of various sizes and aspect ratios (1:1, 1:2, and 2:1 are some examples of possible aspect ratios for these anchors). While managing to achieve state-of-the-art results at the time it was developed, the Faster R-CNN method also achieved a running frame rate of 5 FPS when obtaining these results, demonstrating the capability of the heavily modified R-CNN approach to perform detection in real-time.

SPP-Net [30] employed a spatial pyramid pooling strategy to create a fixed-length image representation regardless of the input image dimensions, addressing the need to resize images to a fixed size before feeding them to a CNN. This strategy works by pooling a given feature map in a fixed number of local spatial bins for each pyramid level. The size of these bins is determined proportionally to the image size, and the last level of the pyramid results in a representation of 1×1, obtained with global pooling of the image, where the single considered bin covers its entire area. By introducing an SPP layer before classification, posterior to

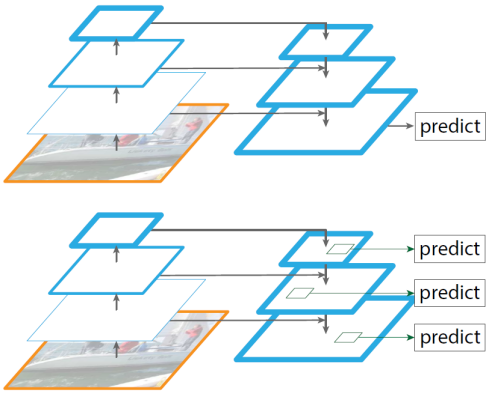


Figure 2.5: Top-down architecture with skip-connections and a single feature map for prediction (top), and the FPN architecture, with prediction at multiple scales (bottom) [23].

the extraction of features by the convolutional layers, one obtains a vector made up of the various results from the pooling operations, which maintain a fixed size and preserve the spatial information of the image while allowing the CNN to process images and output features of various dimensions.

FPN (Feature Pyramid Networks) [23] introduced multi-scale feature representations, widely applicable to other detection architectures. Instead of extracting and utilizing a single feature map from the backbone network, FPN takes advantage of a pyramid containing multiple scales of the same image, performing predictions on every level of the pyramid. This improves a model's robustness when dealing with different object scales, allowing more information to be available for the task of classification. This type of pyramid is easily obtainable using the already employed convolutional networks, as sequential down-sampling already occurs as part of their operations, and the output of each convolution can be stored as an individual feature map, as illustrated in Figure 2.5. When applying FPN to other methods, such as Faster R-CNN, the feature map selected for detection can be determined by the size of each RoI identified by the RPN, providing the model with some added adaptability to different object scales.

2.3.5 One-Stage Detectors

One-stage detectors, having been introduced as an evolution of the two-stage architecture, attempt to unite the two tasks, by skipping the class-agnostic object proposal stage. Instead, the network responsible for the classification of the objects directly analyses the processed image, determining whether there is a strong response to any of the object classes learned during training. These methods boast a significantly higher detection speed, though at the cost of classification accuracy in the past, which has been subsequently diminished. Among one-stage detectors, some of the most relevant contributions were [11; 24–28].

YOLO (You Only Look Once) [11] introduced single-stage detectors, utilizing a

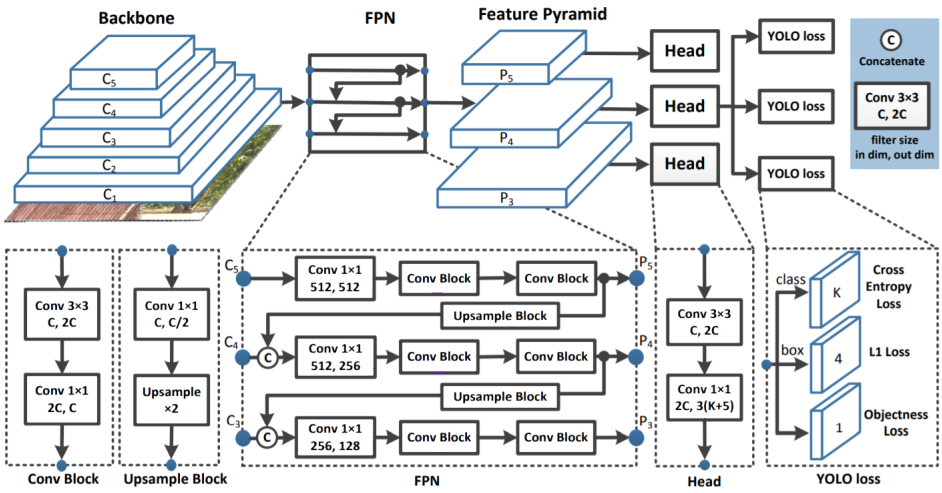


Figure 2.6: Representation of the YOLOv3 architecture, with the backbone, neck, and detection head. The model predicts category, bounding box and objectness values [31].

single network, similar in architecture to GoogLeNet, to predict the location and class of objects in an image. The model analyses the image by dividing it into a grid, computing a number of bounding boxes and confidence scores for each of its cells. The confidence score metric conveys the probability of an object being enclosed in the predicted bounding box, alongside the quality of the box prediction, by measuring IoU with the ground truth during the learning process. Each cell with an identified object and bounding box also outputs an independent number of category predictions, completing the object detection task. Numerous versions of YOLO have since been developed, starting with **YOLOv2** [25], also known as YOLO9000, which implemented batch normalization, increased the resolution of the model and employed predefined anchors in the prediction of bounding boxes, similarly to Faster R-CNN. Batch normalization, in short, refers to the strategy of normalizing inputs in each layer in small batches, or mini-batches. The **YOLOv3** [27] model built on the backbone established in the previous version, named Darknet-19, increasing its depth and proposing Darknet-53. From this network, the model extracts three distinct representation scales for classification using the FPN method, as shown in Figure 2.6, improving the detection accuracy of smaller objects. Finally, **YOLOv4** [28] offered multiple improvements in relation to previous iterations, mainly in the form of the newly chosen backbone network and various training strategies employed to increase the accuracy and efficiency of the model. Data augmentation, for example, is used to significantly improve the performance of the model without drastically decreasing its time efficiency during training. On the other hand, techniques that result in a more noticeable increase in inference time, such as the Mish activation function, are justified by their significant contribution to the model in terms of classification performance. The network ultimately chosen to serve as the backbone for the model was the CSPDarknet53, which makes use of the network partitioning concept proposed with CSPNet, where the initial feature representation is merged with the output of each network stage. The multi-scale feature aggregation component was also altered from the FPN method to the PAN module, which includes the sequential

down-scaling performed in FPN followed by the correspondent up-scaling of the features, referred to as a bottom-up sequence.

SSD (Single Shot Multibox Detector) [24], using the VGG-16 backbone network as a base, was introduced shortly after YOLO, sharing the concept of predicting object locations and categories with a single pass through the network. SSD initially uses bounding boxes from a predefined set to enclose objects, with fixed sizes and width to height ratios, in order to compute the probability of object categories being present within these boxes. Their placement and dimensions are then adjusted depending on the predicted object class through bounding box regression, based on a loss function that takes into account the location of the bounding box and the confidence score of the prediction. The model also takes into account representations of multiple scales, similar to many of the previous methods, to increase its robustness when dealing with objects of smaller size.

RetinaNet [26] attempted to tackle the problem of class imbalance in object detection, by proposing a new loss function, denominated Focal Loss (FL). One-stage detectors require a high density and quantity of image regions to be considered for prediction, as opposed to the finer selection performed in the localization stage of two-stage architectures prior to classification. This increased number of considered regions can lead to a higher percentage of easily classified instances, which end up representing a disproportionately high amount of the total loss incurred, due to their sheer quantity. One way to offset this problem is by considering the frequency of the encountered examples for each object class, using it to inform the significance of those predictions. Focal Loss, however, seeks to tackle the imbalance resulting from an overabundance of easy examples, by exponentially down-weighing their impact.

To define the Focal Loss formula, p_t must first be defined, being given by:

$$p_t = \begin{cases} \text{p} & \text{if } y = 1 \\ \text{1-p} & \text{otherwise.} \end{cases}$$

, where y represents the ground-truth class ($+1$ or -1), and p is the probability, computed by the model, of the instance belonging to the positive examples of the class. Focal Loss can then be obtained through the following formula:

$$FL(p_t) = (1 - p_t)^{\lambda} log(p_t)$$

, where λ is the factor by which the easily classified instances get down-weighed.

2.4 State-of-the-Art Object Detectors

Research on small object detection is closely related to that of object detection. A sizeable amount of the approaches suggested to deal with the small object prob-

lem use previously developed object detection models or architectures as a basis. This is the case with [6; 32; 33], among others.

The influence of object detection advancements on the problem of small object detection results in a gap between the currently best-performing object detection models and the ones used as a basis to improve small object detection. With this in mind, it is important to explore the most recent object detectors tested with objects of all sizes. These not only represent the latest evolution regarding the task as a whole but might already implement meaningful improvements that increase the accuracy of small object detection, since a good degree of robustness to varying object scales is a factor of high importance in many applications of the technology.

Currently, one of the most relevant benchmarks for object detection is the COCO Benchmark, where a model's performance is evaluated on multiple versions of the COCO data set. This benchmark is divided into multiple categories, namely Object Detection and Real-Time Object Detection, for which the leaderboards can be found in [34] and [35], respectively. This section looks into the best performing models for both of these evaluation categories, providing a brief description of their contributions.

2.4.1 Standard Object Detection

The standard object detection benchmarks based on COCO are divided into two separate challenges, evaluated on different variants of the data set. One uses the COCO minival data set ($50K$ images), while the other is evaluated on the COCO test-dev data set ($20K$ images). However, the best performing models and their accuracy values are similar for both of the data sets at the time of this writing.

The sole focus of the standard object detection task is the mAP achieved by a given model. The lack of regard for inference time in this category makes the best performing networks not suitable for systems that must run in real-time. Setting aside those applications, the top-scoring models for this challenge still offer the best possible accuracy obtainable in the object detection task and represent the best options when time and resource usage are not major concerns.

Recently, the common deep convolutional network architectures have been surpassed in accuracy by implementations based on the Transformer architecture. The Swin Transformer, an implementation of this architecture for computer vision tasks has largely served as the groundwork for these new approaches, which manage to obtain state-of-the-art results on the standard object detection benchmark. As of the writing of this review, the following models presented hold some of the best results on the COCO test-dev benchmark.

Swin Transformer V2 [12]. The intent behind Swin Transformer V2 was to scale up the original Swin Transformer model in capacity and resolution (addressed in Section 2.3.2). To this end, modifications to the layout of the original network were proposed such as the use of the scaled cosine attention instead of dot product attention and the placement of normalization layers after the residual com-

ponents of the network, both contributions leading to a reduced amplitude of attention values and a further stabilized training process. The increase in the input resolution and number of parameters also leads to an increase in GPU memory usage, dealt with by proposing a new sequential self-attention computation method. The SwinV2-G model version, applied to the task of object detection manages to obtain state-of-the-art average precision scores for the COCO test-dev data set.

Swin-L with Soft Teacher [36]. With the Swin-L network as its backbone, this approach was developed in an attempt to improve detection performance using pseudo-labels, applied to large amounts of unlabeled data. Its architecture is divided into two models: the teacher and the student model. The teacher is tasked with producing pseudo-labels for the unlabeled instances present in images of the data set, potentially increasing the amount of useful information for the training of the detector. The student model, on the other hand, is trained with both the labelled data and the pseudo-labelled examples given by the teacher model, pretrained to detect objects and provide a sizeable quantity of new instances. The result is a semi-supervised object detection framework capable of being applied to most existent models.

2.4.2 Real-Time Object Detection

The real-time object detection benchmark is based on the standard COCO data set. The aim of real-time detection models is to perform accurate detection with a short inference time, usually measured in milliseconds. Another metric utilized to evaluate the speed of a model is the frame rate, in frames per second (FPS), providing essentially a reverse measure of inference time.

Real-time object detection is the preferred standard for most implementations of the technology, where a system is required to detect objects in the environment and react accordingly within the shortest time span possible. Such is the case with self-driving vehicles, for traffic-sign or pedestrian recognition.

Some of the models with the best performance on the COCO benchmark for real-time object detection at the time of writing this review are presented below.

YOLOR [37]. You Only Learn One Representation (YOLOR) proposes a unified network capable of encoding explicit and implicit information simultaneously. In this context, explicit information refers to the information directly obtainable from the input data, by early layers of the network, whereas implicit information refers to the information present within deeper network layers, gained as a result of its training process. By integrating implicit information into the network's learning process, after training it with explicit information, its overall performance can be improved. Its usefulness can also be widened to related tasks, such as being able to classify the parts that constitute an object in addition to identifying the object itself. The most relevant achievement of this model was its high running speed, above that of any model with similar levels of precision

accuracy. In the real-time COCO benchmark, the fastest YOLOR model to be evaluated, YOLOR-W6, managed to run at 66 FPS, while the most accurate version, YOLOR-D6, managed to obtain state-of-the-art average precision results while running at 34 FPS.

Scaled-YOLOv4 [38]. With Scaled-YOLOv4, strategies were presented to scale the original YOLOv4 model both up and down. Beyond adjustments to the width and depth of the model, layout changes are also proposed to the CSPDarknet53 backbone network, focused on optimizing the processing of the image by replacing the early stages of the network with the standard Darknet design. While the scaled-down version, YOLOv4-Tiny, successfully adapted the architecture to a reduced network size and low-end GPU devices, the improvements naturally originated from the YOLOv4-Large version of the model, scaled up in depth and resolution. This version managed to surpass the results of the original YOLOv4 model on the COCO test-dev data set while maintaining its high running speed.

EfficientDet [39]. The EfficientDet model appeared as an attempt to attain state-of-the-art results by scaling up baseline models, in width, depth and resolution. Compound scaling is proposed with this model, which is achieved by scaling these three parameters equally, which produces the most efficient increase in performance. Alongside this method, a new Feature Pyramid Network (FPN) architecture is also proposed, BiFPN. With standard FPN, the multi-scale feature fusion is only implemented in one direction, from top to bottom, meaning the information from later layers gets augmented with the output of earlier layers. BiFPN not only includes the same operations repeated in the opposite direction (bottom-up) but also employs multiple skip connections from each layer's initial feature map to its fused counterpart, repeating this architecture with segmented blocks. Finally, it also replaces the standard feature fusion method, based on summing the different feature maps, by a weighed formula with the intent of controlling the impact of different scale representations.

2.5 Small Object Detection Approaches

A wealth of strategies has been explored to deal with each of the challenges that small object detection presents. These include techniques such as the fusion of feature maps obtained at different scales, the use of contextual information external to an object's bounding box, or the generation of super-resolved versions of small objects prior to classification. All of these techniques have been shown to improve the accuracy of small object detection to some extent when incorporated into otherwise traditional detection architectures.

This section will go over the most representative methods which employed these strategies with the intent of improving the detection of small objects, seen in Table 2.2, providing a description for each approach.

Strategy	Method
Multi-Scale Features	**Finding Tiny Faces** [40] **DSSD** [41] **Faster R-CNN for SOD** [32] **Feature-Fused SSD** [42] **MDSSD** [33] **FSSD** [43] **HRDNet** [44]
Tuned Region Anchors	**R-CNN for SOD** [6] **Modified Faster R-CNN** [45] **Small Face Anchors** [46] **SCRDet** [47]
Contextual Information	**ContextNet** [6] **Modified Faster R-CNN** [45] **Spatial Context Analysis** [43]
Super-Resolution	**Perceptual GAN** [48] **SOD-MTGAN** [49] **Feature Super-Resolution** [50] **JCS-Net** [51]
Data Augmentation	**Modified Faster R-CNN** [45] **Augmentation for SOD** [52]

Table 2.2: Small object detection methods described in section 2.5, organized by technique.

2.5.1 Multi-Scale Features

Object detection traditionally relies on feature representations produced by the latter layers of deep convolutional network architectures. The features obtained from these deeper layers, also referred to as high-level features, contain richer semantic information, useful in the object classification stage. On the other hand, lower-level feature maps retain a higher resolution, closer to that of the original image.

Contrary to objects of larger size, smaller objects do not benefit nearly as much from the usage of deeper feature maps. The more robust information obtainable in high-level features maps comes at the cost of their resolution, which is increasingly reduced after each convolutional layer. This makes the resulting representation obtained from small objects inadequate in size, due to their low original resolution.

It is now known that leveraging the lower-level feature representations of small objects in their classification significantly benefits its accuracy, as shown in Section 2.5. This inclusion has been achieved with a variety of feature fusion techniques, which combine the feature maps, corresponding to representations of different scales obtained along the network, to improve the performance in the detection of small objects.

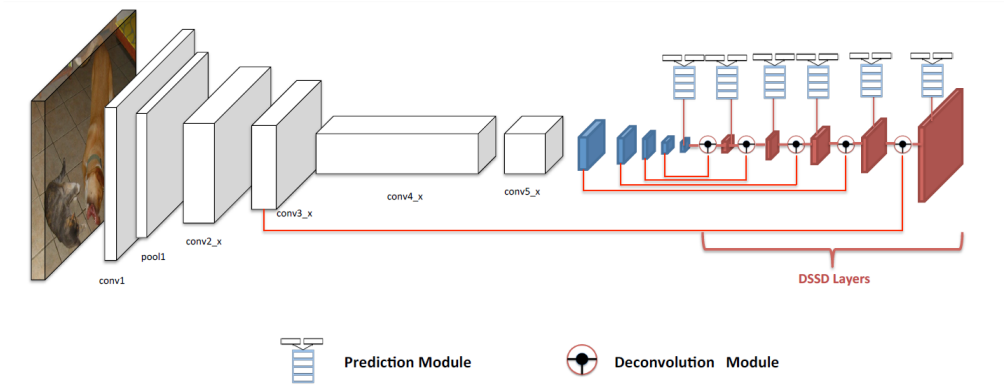


Figure 2.7: Representation of the DSSD architecture, with the SSD layers depicted in blue [41].

Finding tiny faces [40]. For this method, a multi-task model is developed for face detection as a means to simultaneously process the input image at multiple scales ($\times 1$, $\times 2$, and $\times 0.5$). This image pyramid is constructed first, followed by the processing of each image resolution by the network, naturally producing different feature responses at each different scale. The output for each of those resolutions is then compared and eliminated through non-maximum suppression, leaving the result correspondent to the scale that triggers the strongest response, which is then used in the prediction stage.

DSSD [41]. Deconvolutional SSD appeared as an attempt to introduce additional information into detection with the SSD architecture, by incorporating deconvolutional layers into the model, as is observable in Figure 2.7. It opted for Residual-101 as a backbone network and appended the new layers to the end of the original network's architecture, to augment detection with high-level context information. These layers provide an increase in resolution to the feature maps obtained from the standard SSD stage, partly reverting the down-sampling of the image by the convolutional network, resulting in a similar architecture to that of encoder-decoder networks. The deconvolutional stage of the model, however, is not as deep as the preceding convolutional network, with the intent of preserving the time efficiency achieved with the method. The final result of DSSD is also combined with the output of the SSD segment with a component designated deconvolutional module, which aims to integrate information from earlier layers of the network in the detection.

Faster R-CNN for small object detection [32]. Here, a technique based on FPN is applied to the Faster R-CNN model architecture, to improve the detection of small company logos. The effect of different feature scales is tested on the proposal and prediction stages, leading to the utilization of earlier convolutional layers in both stages. The performance of the RPN is measured by the use of the mean average best overlap (MABO) metric, which assesses if the network is capable of producing at least one object proposal which strongly overlaps with the ground truth. In the implementation of the model, the multiple feature scales, obtained from the 3^{rd}, 4^{th} and 5^{th} groups of convolutional layers, are fused into a single representation before the prediction stage. However, each iteration of this

fusion is separately fed into the RPN, which produces adequate object proposals for each scale.

Feature-Fused SSD [42]. Feature-Fused SSD modifies the SSD architecture by fetching features from earlier convolutional layers of the network and integrating them into the prediction stage of the model. The features chosen to be merged using the fusion block are outputted from the fourth (conv4_3) and fifth (conv5_3) convolutional blocks in the network. The output of the conv5_3 layer is first processed in a deconvolutional layer, to match the size of the first output. Two fusion modules are explored: the concatenation module and the element-sum module. The concatenation strategy stacks the feature maps together and applies a 1×1 convolution to combine the various levels, while the element-sum technique relies on the sum of the feature values from both representations. This method managed to obtain comparable accuracy with the state-of-the-art methods at the time while focusing on maintaining the fast detection speed characteristic of the SSD architecture.

MDSSD [33]. The Multi-Scale Deconvolutional Single Shot Detector was proposed with the intent of further adapting SSD for small object detection. It implements multiple fusion modules set to merge features from specific network layers at various depths, combining those with the standard outputs of hand-picked convolutional layers. The goal of MDSSD's architecture is to incorporate lower-level information in the detection process, which contains a higher degree of detail and significantly increases the accuracy in the prediction of small objects. Contrary to DSSD, which uses a set of deconvolutional layers to extract this information from the output of the convolutional network, this model uses deconvolutional layers in the feature map fusion process. Fusion Module 1 is an example of this, being responsible for combining the output of the conv3_3 layer and the comparatively small output of the conv8_2 layer (last layers of the third and eighth group of convolutional layers respectively), made possible by passing the output of the eighth block by a deconvolutional layer, which restores a higher resolution to the feature map. This method managed to improve small object detection accuracy on the Tsinghua-Tencent 100K data set [8], for traffic sign detection, when tested against Faster-RCNN and SSD, obtaining comparable results to those of DSSD. It was also tested on the small objects found in the COCO data set, overcoming SSD and DSSD.

FSSD [43]. Applied to the context of Unmanned Aerial Vehicles (UAV), this method utilizes feature representations of multiple scales to better detect small objects, employing similar strategies to FPN and Feature-Fused SSD. Using the SSD as a basis, FSSD extracts maps from multiple layers of the first convolutional network, fusing them into a single representation. This combined representation is then used as the input to extract a second pyramid of features, used in the prediction stage. Additionally, average pooling is performed in the original output of the convolutional network, the result of which is leveraged for prediction. Before the pooling operation, however, deconvolutional layers are introduced to increase the resolution of the feature map, followed by a single convolutional layer,

creating a separate branch of the network from the FSSD component, which manages to improve results for small objects further.

HRDNet [44]. The High-Resolution Detection Network was proposed to deal with small object detection by leveraging multiple scales of features. The model is built to process a pyramid of images with different resolutions using distinct convolutional networks of different depths: high-resolution images are processed by shallow networks and low-resolution images are fed to deeper networks. This component of the model is named Multi-Depth Image Pyramid Network (MD-IPN). It is followed by the Multi-Scale Feature Pyramid Network (MS-FPN), which combines the different feature representations, by first applying a technique similar to FPN to each of the networks included in the MD-IPN component, then combining the results obtained across the whole model.

2.5.2 Tuned Region Anchors

One of the aspects that most heavily influences the localization aspect of object detection are the pre-determined region anchors utilized to obtain the object bounding boxes. In the development of standard object detection models, these anchors often take the sizes and shapes that most benefit their overall application and get the best average response when overlaid with the objects most commonly present in the context of the desired application. When considering small objects, these parameters can also be appropriately adapted to achieve their goal more efficiently.

By reducing the smallest size of the region anchors used it is possible to obtain a stronger response for objects of smaller scales, which also obtain better locations within their ultimate bounding boxes, after regression. In this context, a better location would ideally have the detected instance centred within the bounding box and occupying the largest possible area percentage of it. Another possible change, somewhat related to the previous one, would be to lower the placement stride of the small anchors. By making their placement more frequent across the image, small objects have a higher chance of being adequately located and processed, although not without an increased computational cost.

R-CNN for Small Object Detection [6]. In this proposal, where the R-CNN is utilized as a basis to improve small object detection, one of the alterations suggested is the resizing of the region anchors employed in the RPN to obtain the initial object proposals. It was found that even considering the three different box scales initially proposed with the RPN, the smallest one proved to be too large to adequately detect objects of smaller dimensions. Anchor box dimensions were significantly reduced in this approach to meet that demand, though the various box ratios were maintained.

Modified Faster R-CNN for Optimal Remote Sensing [45]. The data set utilized in this proposal is that of small objects captured from an aerial perspective. One

(a) $S_A = 16$ (b) $S_A = 8$

Figure 2.8: In (a), an example of the adverse effect of a wide anchor stride, and in (b), the higher bounding box IoU score afforded by the increased frequency of anchor placements, where S_A is the value of the stride (relative to the feature map and not the image itself) [47].

of the simplest measures implemented to adapt the Faster R-CNN to this context was the reduction of the fixed anchor sizes. The new anchor scales better represented the average object size in display, resulting in a slight increase to the accuracy of the model. The minimum anchor size was, in this case, 16×16 pixels, as opposed to the original smallest scale of 128×128 pixels.

Anchors for small faces [46]. With this method, a new anchor placement strategy is studied which benefits the detection of small faces (under 16×6 pixels). This strategy consists in increasing the density of anchors overlapped on the image through various methods. First, the stride for the placement of anchors can be reduced by increasing the resolution of the feature map used, through artificial up-sampling, and the inclusion of information from previous convolutional layers. This increases the number of possible locations for anchors, making small objects more likely to be detected with a good overlap. Another method is to slightly displace the anchors or the feature map itself randomly throughout multiple iterations, increasing the odds of small objects landing closer to the centre of an anchor in one of the iterations. Both of these techniques were found to improve a model's ability to detect small faces, though at the cost of a heavier computational load, from the initial analysis of a significantly higher number of anchors.

SCRDet [47]. SCRDet tunes the RPN component to achieve a higher level of overlap with small objects in aerial images. Similarly to the strategy discussed in [46], SCRDet employs a reduced anchor stride, increasing the number of anchors placed across the image representation, improving the chances of small objects being strongly detected by an anchor and included in the object proposals. With low-resolution objects, a higher stride can likely lead to a lower overlap between the bounding box and the ground truth. This effect is easily visualized with the help of Figure 2.8, which shows the results of two different anchor strides. This method also combines features from multiple network layers in order to achieve the ideal balance between the high-level semantic information extracted by deep

convolutional layers and the more detailed features outputted by shallower layers. Even considering the good results obtained with the tuned anchor positioning, it is necessary to note the adverse impact this strategy has on the computational performance of the algorithm, as it must process the image at a significantly higher number of points.

2.5.3 Contextual Information

Given the scarcity of information offered by low-resolution objects, the addition of complementary data can become a larger asset in classification. With this in mind, contextual image information, obtained from areas of the image outside of the object's bounding box, can prove useful.

Additional context feature maps can be derived from the areas adjacent to that of the object, or the image as a whole, in order to aid in the prediction stage of an object detection algorithm. The context in which object categories appear can become valuable for a model's ability to identify them correctly. Certain object classes might frequently appear in close proximity to others, or with a similar background, making the inclusion of such information beneficial for those prediction cases. The scene recognition field of Computer Vision, focused on the identification of different scenes depicted in images, can also become relevant in this regard.

ContextNet [6]. This network architecture was proposed to increase the accuracy of small object detection by including information external to the object's region of interest. In this network, which used the R-CNN as a basis, a sizeable region surrounding the original object proposal area is processed by the convolutional layers and leveraged in the classification stage. While one branch of the network processes the region of interest as per usual, a separate one takes the context information as input, and the two distinct outputs resulting from both branches are concatenated and used to compute category scores.

Modified Faster R-CNN for Optimal Remote Sensing [45]. Similarly to the previous method, this altered Faster R-CNN architecture extracts two regions of different sizes from the object proposal stage. Additionally to the processing of the proposal region enclosing the object, a wider area is also considered, surrounding this bounding box. The architecture is temporarily split into two paths, performing separate ROI pooling operations for the original and the contextual regions. The two resulting feature maps obtained for a given object are concatenated and combined, as to include the contextual information in a single representation before the classification stage.

Spatial Context Analysis [43]. A particular interpretation of spatial contextual information is explored with this method, where different object instances near a given object are considered after its detection. If a detection confidence score indicates a less than reliable prediction, the model searches for objects of the same category reliably detected in the vicinity, increasing the confidence score of the

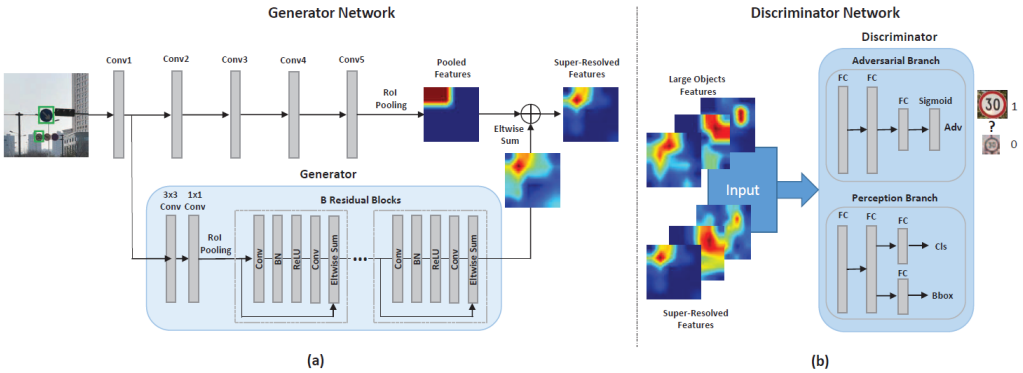


Figure 2.9: Representation of the Perceptual GAN architecture, with the generator and backbone network in (a), and the two discriminator modules in (b) [48].

original prediction in case those instances are present. This increase is dependent on the quantity of similar objects found and their distance to the original object. The category confidence score of the object may also be reduced when an increased number of similar objects are reliably detected at a greater distance from the original object, indicating a reduced probability of that type of object being placed where it is. The use of this strategy, named spatial context analysis (SCA), showed an improvement in the final mAP results the detector managed to achieve.

2.5.4 Super-Resolution

Super-resolution has been experimented within the context of small object detection, in an attempt to increase the detail found in small objects by studying their relation with their larger counterparts.

As discussed, the prediction accuracy of object detection models appears to decrease along with the size of the object it is attempting to recognize. By increasing the resolution of small object representations, bringing them closer in detail to larger objects, their category can be more consistently predicted, as the network's prediction model can be trained with objects of larger scale.

Perceptual Generative Adversarial Network [48]. In 2017, Perceptual GAN was the first to look to super-resolution with Generative Adversarial Networks (GAN) in object detection, with the intent of increasing the resolution and detail of small objects and achieve results comparable to those obtained in the detection of larger instances. As seen in detail in Figure 2.9, the GAN architecture is composed of two distinct adversary networks: a generator and a discriminator. In this context, the main goal of the proposed model is to train a generator network capable of producing realistic super-resolved versions of small objects. To this end, the generator has to introduce finer details into the feature representation of a small object, created using a residual network, while the discriminator is divided into two branches. One branch is responsible for learning to distinguish the artificially generated object representations from those of real large objects, while the

other predicts a bounding box and category scores for the object and is trained using real large object representations. Distinct loss functions are computed for these two tasks performed by the model, serving as feedback for the generator to create increasingly convincing super-resolved representations. The first is adversarial loss, computed from the result of the prediction made by the discriminators first branch (real or fake), and the second is perceptual loss, which includes in its computation the classification loss (category confidence scores), and the bounding box location loss.

SOD-MTGAN [49]. A second application of GAN in small object detection was proposed with SOD-MTGAN (Small Object Detection Multi-Task GAN), which likewise generated super-resolved versions of small objects to improve detection results. Its general architecture is in many ways similar to Perceptual GAN, with a discriminator outputting a real/fake score, category scores, and a bounding box, as well as back-propagating this information to the generator with multiple loss functions. Additionally to the adversarial and perceptual loss, a function referred to as pixel-wise loss is implemented, to minimize the difference between the generated super-resolved images and the images of real large objects.

Supervised Feature Super-Resolution [50]. This proposal aims to generate super-resolution versions of small feature maps, using a GAN-based architecture on top of a one-stage object detection model, by exploring the relation between high scale features and their low scale counterparts. These two distinct image scales are obtained for training by down-sampling the original high-resolution image, which serves as the super-resolution target. The model first obtains features maps for both of these image scales, feeding them into separate prediction modules: the features from the original image are given to the "large" predictor, and the features from the down-sampled version are given to the "small" predictor, after passing through the super-resolution (SR) generator. The SR generator also takes as input the features extracted from the original large image, feeding its output to the small prediction module and the GAN discriminator component. The discriminator's task is to distinguish between the generated super-resolved features and the features directly extracted from the original image, providing a binary answer according to the authenticity of the high-resolution features (original or generated). The generator receives feedback from the discrimination process, by implementing a function that attempts to minimize the difference between the targets and super-resolved proposals, while the discriminator is trained to differentiate them as much as possible, guiding the evolution of the generator. This approach was found to not only benefit the detection accuracy of small objects but also all of the object scales, when evaluated on the COCO data set.

JCS-Net [51]. JCS-Net attempts to use super-resolution to improve the detection of small-sized objects in pedestrian recognition, by studying the relationship between small scale and large scale pedestrians. This relation is exploited to boost the level of detail found in small pedestrian instances. Two network components are included in this model: the super-resolution network and the classification network. Firstly, a detection network is trained with large pedestrian instances,

which serves as the basis for the classification component of the model (initial parameters). The super-resolution component is trained to produce super-resolved versions of small scale pedestrians. The training of this network is achieved with a data set of large pedestrian instances, which are then down-sampled to simulate pedestrians of small-scale, with the original large-scale object serving as the target. The result is a super-resolution network capable of increasing the detection performance of small pedestrians, which is then combined with a standard CNN for large instances and hand-crafted features. Multi-scale features are also used in the final detection process, extracted from multiple layers of the super-resolution network, corresponding to multiple levels of detail added to the small pedestrian.

2.5.5 Data Augmentation

The goal of data augmentation is to artificially increase the amount of data available during training by reproducing and repurposing the existing samples. In the context of object detection, this might include copying certain objects and placing them in different areas of the image, transforming some of its aspects before doing so.

In small object detection, data augmentation can serve to mitigate a lack of examples of small objects in certain data sets. The concept was experimented in this context in the following presented proposals, having generally produced favourable performance results.

Modified Faster R-CNN for Optimal Remote Sensing [45]. In this proposal, one of the problems tackled was the insufficient training data for the intended application of the model. One of the strategies used to improve the learning process of the detector was data augmentation, by introducing Random Rotation (RR). Depending on a probability value, any image within a training mini-batch has the possibility to be picked for Random Rotation, resulting in an additional sample of it rotated by a random angle, with the ground-truth bounding boxes labelling objects in the image also being rotated according to the coordinates of their corners.

Augmentation for small object detection [52]. With this proposal, data augmentation techniques are studied as a means to better train models to detect small objects. It is shown that, by expanding the amount of small objects present in each image of a data set, therefore increasing their representation in the data set, their prediction results can be improved. To achieve this effect, strategies such as the oversampling of existent data and the copying of small object instances served as effective augmentation strategies. In consideration of the model's robustness and to avoid over-fitting, copied instances were also randomly rotated and transformed when reproduced.

2.6 Discussion

In this chapter, a review of the existing literature related to object detection and small object detection has been provided, going over the most representative and best performing object detectors, as well as distinct methods developed with the intent to tackle the small object detection problem.

Regarding standard object detection, the methods shown in Table 2.1 were first presented, in light of their impact and contributions. When analysing the state-of-the-art models evaluated on two distinct COCO benchmarks, it became apparent that models derived from the recent Swin Transformer architecture currently obtain the best performance in terms of accuracy (Swin Transformer V2), relatively to other approaches. With inference time taken into account as one of the evaluation metrics, as in the context of real-time detection, the YOLOv4 based approaches such as YOLOR and YOLOv4-Scaled produce the best mAP results balanced with a good computational performance, making them likely candidates for the base real-time detection model necessary for the project.

Several solutions focused on lessening the negative impact of small objects in detection accuracy were presented in Section 2.5, with the proposed methods connected to these solutions listed in Table 2.2.

Chapter 3

Model Implementation

This chapter will cover the implemented models and techniques, the foundations and evolution of which were discussed in the previous chapter. It will begin by covering the chosen base model architecture in detail and the main alterations explored throughout the second phase of the project, as well as the reasoning behind the most relevant choices in this regard.

3.1 Base Model Architecture

The model architecture chosen as the basis for implementation was the YOLOR architecture, as it holds the best performance in the COCO real-time object detection benchmark, with three distinct versions of the model placing at the top of its official leaderboard. Regarding this particular benchmark, not only does YOLOR obtain the highest mAP scores, but also fast running speeds, above the great majority of the competition. Another noticeable aspect in these results, shown in Table 3.1, is the trade-off between the prediction accuracy and speed, which offers a variety of possible choices depending on the intended application of the model and available resources.

The YOLOR architecture was introduced with the purpose of encoding implicit and explicit information simultaneously to benefit the performance of already existing models. In the context of the YOLOR method, implicit information represents vector information extracted from deeper layers of a trained network, which can then be leveraged in multiple stages of its architecture to improve results. The main contribution of YOLOR is the addition of this encoded implicit information to enhance various stages of the YOLOv4 architecture: feature alignment in the FPN, prediction refinement, and multi-task learning. Despite this significant addition, the developed YOLOR models can still be considered heavily based on the standard YOLO architecture, specifically that of the Scaled-YOLOv4 models proposed in [38].

The close relation to the previous iterations of YOLOv4 of the YOLOR method provides a solid foundation regarding the understanding and implementation of the model as a baseline. Although more unfamiliar and complex components are

Model	mAP	FPS
YOLOR-W6	**57.3**	34
YOLOR-E6	56.4	45
YOLOR-D6	55.5	**66**

Table 3.1: Three distinct YOLOR models evaluated on the COCO real-time object detection benchmark [35].

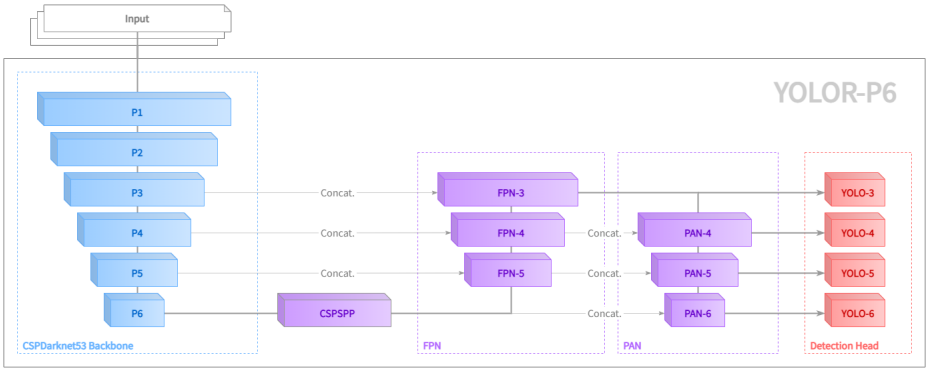


Figure 3.1: High-level diagram of the YOLOR-P6 base model architecture, divided into its main stages and components. The neck is composed of two distinct networks, the FPN and PANet [54], preceded by the CSPSPP component. The secondary dashed paths connecting intermediary levels of different components represent the concatenation of features.

introduced with the incorporation of implicit information, the core architecture can still be followed back to that of the first YOLO iterations. Considering the vast amount of research dedicated to the evolution of the YOLO architecture and its application to other computer vision tasks similar to object detection, YOLOR represents a contemporary, yet safe option for the purposes of this project.

3.1.1 YOLOR Model Variants

The YOLOR-P6 variant of the YOLOR model was initially chosen for the project, and implemented with the code provided by the authors in the official GitHub repository [53]. A diagram of this model's architecture in its original form can be found in Figure 3.1.

When compared with the YOLOR models shown in Table 3.1, which held high positions in the COCO real-time object detection benchmark leaderboards at the time, YOLOR-P6 failed to achieve similar high levels of accuracy, being instead designed to be the fasted and less resource-intensive alternative of the architectures presented by the authors at the time. As evident by observing the backbone used for YOLOR-P6, though it retained similarities in its structure, the quantity of layers present in each of its convolutional blocks, as well as their number of convolution filters, were inferior to those found in architectures such as YOLOR-D6. Its detailed backbone composition is discussed further in Section 3.1.2. To summarize these differences shortly, the backbone of YOLOR-P6 offered better

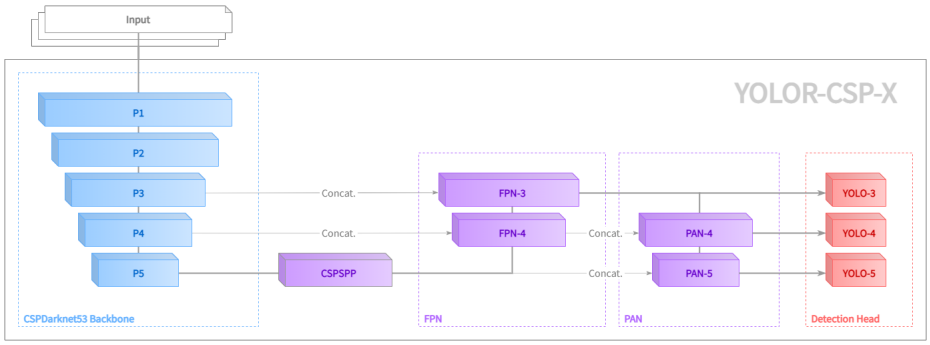


Figure 3.2: High-level diagram of the YOLOR-CSP-X model architecture, divided into its main stages and components. Five levels of features are extracted in the backbone, resulting in three detection scales. The model's input size is of 640^2 pixels.

computing performance due to its decreased depth and width.

Consequently, the main factor behind the choice of YOLOR-P6 were the resources available for the training and testing of the various planned changes within the time frame available for the conclusion of the project. It became apparent when exploring the implementation of any structural alteration in the early stages of the project, that the assessment of the impact of such changes required re-training the model weights from the ground up, a process only made feasible with the model versions that were ultimately implemented. Added to this, a significant period of time would also be required to obtain the final results, after the investigative phase of the project, as many of the early tests were performed with a reduced data set and limited number of epochs, as further explained in Section 4.2.3.

With these tasks in mind, it was determined early, through some quick experimentation with all the variants provided by the authors, that YOLOR-P6 would offer the most adequate foundation for the quick preliminary and final training and evaluation of the developed changes, affording some tolerance and malleability in the planning of the various implementation stages of the project.

Later in the project's development, experiments were also run with a different version of YOLOR, YOLOR-CSP-X, yet unpublished at the time of writing. This variant was developed with a reduction of the input size in mind, from the standard size of 1280^2 pixels, used in the YOLOR-P6 architecture and other previous versions, to 640^2 pixels. This difference was reflected in the depth of the networks used in the model, as seen in Figure 3.2, with the backbone network being divided into only five segments, as opposed to six, a change that also naturally affected the structure of the model's neck. As a consequence of reducing the input size, the sixth scale level of the standard models (P6) became futile in the acquisition of relevant features, as it possibly overly decreased the resolution of feature maps to the point where spatial relations and distinguishing features of objects in the image became unclear, therefore being of less use in the detection phase.

3.1.2 Detailed Model Structure

As detailed in Chapter 2.3.5, various single-stage models follow the same architecture at a high level, having three main components: the backbone, the neck, and the head. The backbone is responsible for the extraction of features from the input images, the neck assembles and enriches a multi-scale feature pyramid, and the head is responsible for predicting the detection results, being divided into a number of detectors equal to that of the number of feature maps produced by the neck.

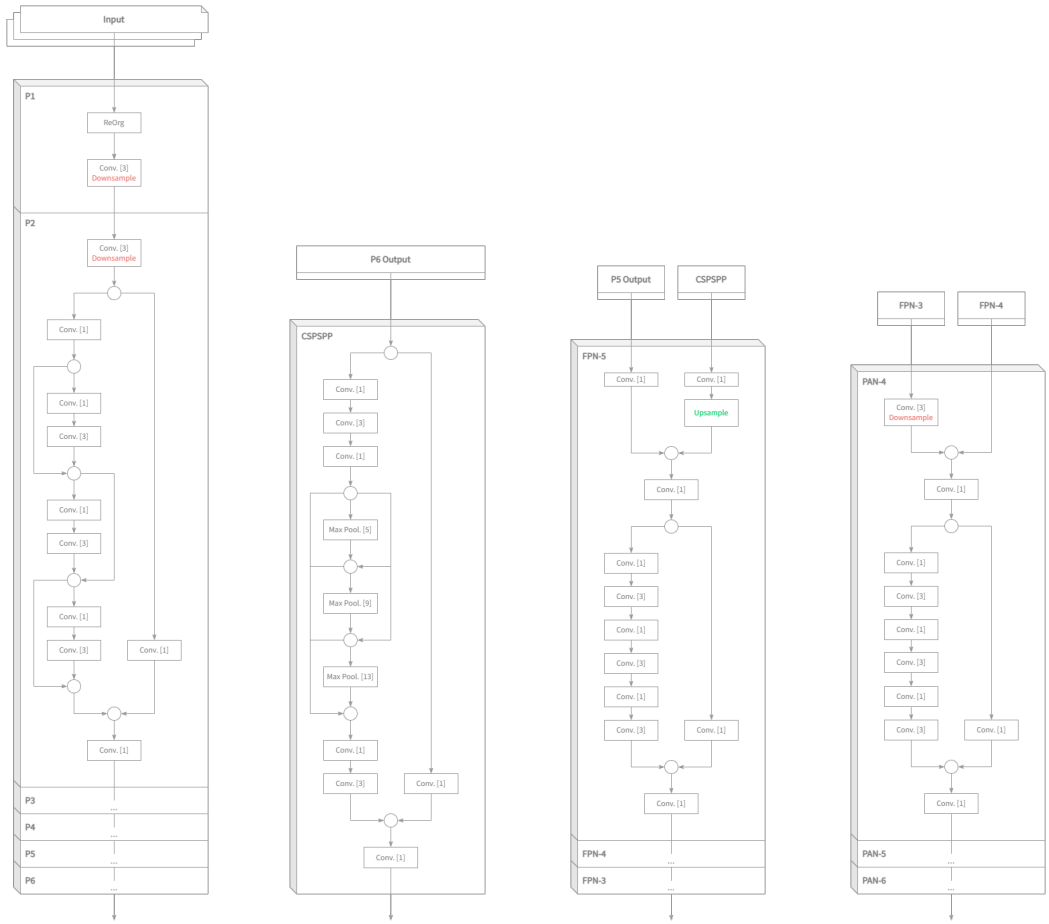
CSPDarknet53 backbone. The YOLOR-P6 model utilises a heavily modified version of the Darknet backbone, first proposed with YOLO [11], with a combination of FPN [23] and PANet [54] for its neck.

Across all YOLOR variants made available along with [37], the CSPDarknet53 backbone is used, first proposed with YOLOv4 [28]. Detailed in Figure 3.3a, as is implemented in the YOLOR-P6 model, the backbone is divided into six distinct segments, though this amount can vary between different variants. Each of these represents different scale of features extracted from the image. Each level starts by down-scaling the input with a convolutional layer before extracting meaningful information through the subsequent layers. A residual block is integrated into each segment as part of this process, as to facilitate the propagation of the gradient throughout the long sequence of convolutional layers utilized. Prior to this block, a split occurs that allows for the unprocessed input to be sent directly to the end of the segment, enriching the information ultimately analysed for detection, and once more easing the propagation of the gradient during training.

FPN and PANet (neck). A CSPSPP component is placed prior to the neck, taking as input the backbone's output. The goal of this structure, visually represented in image 3.3, is to allow the model to process inputs of various sizes, creating a uniform representation for the extracted features, regardless of the size of the original image. This process was also touched on in Section 2.3.4, in the overview of SPP-Net [30].

In the construction of the multi-scale feature pyramid, YOLOR-P6 employs two consecutive networks to augment the feature maps at each of the scale levels. The first structure is a Feature Pyramid Network (FPN), used to up-sample the final features obtained from the backbone and enrich them at each level with prior information, as seen in image 3.3c. In the original models, this process goes up to the scale of the features extracted in the P3 block of the backbone. It is then followed by a down-sampling process, executed by a Path Aggregation Network (PANet), represented in image 3.3d. This component acts similarly to a reversed FPN, once more bringing the features step by step to the scale of the backbone's P6 block, and enriching each level with prior information (this time, obtained from the FPN). Each feature map extracted by the PANet is later used independently for detection.

Detection head. The detection head of YOLOR is made up of different compo-



(a) CSPDarknet53. (b) CSPSPP. (c) FPN. (d) PANet.

Figure 3.3: Detailed architectures of YOLOR-P6's main components, including input sources, layout, and layer types and attributes. The convolutional and max pooling layers represented are accompanied by their kernel size.

nents used for classification, each corresponding to the different scales of features processed by the backbone and neck of the model. Bounding box anchors are placed throughout each of these feature maps, fulfilling the purpose of default bounding box templates that can be tuned to each detected object. These generally represent the most common dimensions and width to height ratios found in objects of the training data set.

Each feature scale uses three predefined box anchors that are simultaneously "placed" on every possible coordinate of the corresponding feature map. This results in 12 different anchors for the original YOLOR-P6 model architecture with four detection scales, and 9 different anchors for the YOLOR-CSP-X model with only three detection scales. Given the lower resolution of these maps when compared to the original image size, the coordinates of the boxes have to later be adjusted to the original scale of the image, through a set multiplier of the for the coordinates.

As mentioned, the output of each model is divided into distinct components in accordance with the established feature scales. The number of predictions for a

given scale is defined by the number of possible coordinates present in the corresponding feature map, multiplied by the three separate anchors placed concurrently on every coordinate. Each prediction contains values for the centre coordinates and dimensions of the object's final bounding box, as well as a score for each of the existent object categories. These values range from 0 to 1, with the bounding box parameters being set in relation to the total image size.

Automatic anchor generation. The generation of anchors is performed at the start of the model's training process. The goal of the algorithm is to provide a majority of the objects in the training data set with at least one of anchor of appropriate size. First, the object data set is partitioned into groups with K-means clustering, with each of the cluster centres corresponding to one of the required anchors. The result is an initial set of anchors, the dimensions of which are evolved with a genetic algorithm. To assess the fitness of a set of anchors, the size ratio between one object and the set of anchors is calculated with:

$$ratio_k = \min\left(w/w_k, w_k/w, h/h_k, h_k/h\right)$$

, where $ratio_k$ symbolises the size ratio measure between the object and an anchor k, w and h represent the object's width and height respectively, and w_k and h_k are the anchor's width and height respectively. Using this method, each ratio value corresponds to the dimension of the object farthest in size to that of the anchor, always staying in between the values of 0 and 1. The ratio value of the best matching anchor is then considered for each object, corresponding to the value closest to the value of 1 of all nine anchors, where 1 would represent the same exact size. The resulting structure is a list of size ratios with the length of the training data set, each corresponding to the most adequate anchor found for each object. To finalise the operation, values below a set threshold are converted to zero. For the base versions of YOLOR, this threshold is set to the value of 0.25, meaning an anchor's measurements cannot differ from the object's by a factor higher than four (be it four times larger or smaller). This evaluation process can be represented as follows:

$$best_{obj} = \max\left(ratio_{k1}, ..., ratio_{kn}\right)$$

$$metric_{obj} = \begin{cases} \text{best}_{obj} & \text{if } best_{obj} > thr \\ 0 & \text{otherwise.} \end{cases}$$

, where $best_{obj}$ symbolises the best anchor match for a given object obj, with $(ratio_{k1}, ..., ratio_{kn})$ representing all of the size ratios between that object and the set of anchors, $metric_{obj}$ represents the element of the final array corresponding to that same object, and thr stands in for the threshold value chosen, between the values of 0 and 1. The final fitness value for a set of anchors is given by the mean of the values included in the final array of metrics.

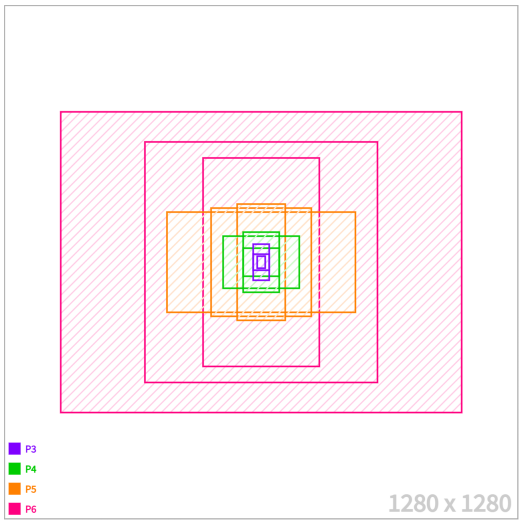


Figure 3.4: YOLOR-P6 bounding box anchors, generated by default using the full training data set from the COCO Object Detection Benchmark, separated and colour coded according to the feature scale for which they are used. Represented in relation to an input image of maximum size (1280 $\times$ 1280).

In summary, the fitness of a set of anchors can be increased by maximising the number of objects that have an anchor adequately similar in dimensions, as dictated by the defined threshold, and minimising their size difference as much as possible. When generated for the base YOLOR-P6 model (four feature scale levels) using the full training data set from the COCO Object Detection Benchmark, the following anchors are obtained, visually represented in Figure 3.4:

- P3: **19** $\times$ **28**, **42** $\times$ **40**, and **37** $\times$ **92** pixels;
- P4: **89** $\times$ **72**, **87** $\times$ **152**, and **189** $\times$ **127** pixels;
- P5: **121** $\times$ **288**, **246** $\times$ **271**, and **465** $\times$ **249** pixels;
- P6: **292** $\times$ **517**, **575** $\times$ **596**, and **996** $\times$ **745** pixels.

As evident, anchors used in lower resolution feature maps are of larger size, with the largest anchors belonging to the P6 detection scale. Deeper layers of the network extract higher level features from the image, which are more useful in the detection of larger objects that occupy a significant portion of it, hence the usage of more sizeable anchors. On the flip side, small anchors are used on higher resolution features, useful in the distinction of smaller objects and details.

3.2 Implemented Modifications

This section will address techniques implemented as alterations to the base detection model, with the goal of improving its performance relative to the problem of small object detection. The two major aspects explored during the project were

the use of higher resolution feature maps and the generation of anchors adapted to small objects. The model variants produced for each approach will be detailed, alongside the reasons for the changes made to the base architecture.

3.2.1 Addition of Higher Resolution Detection Component

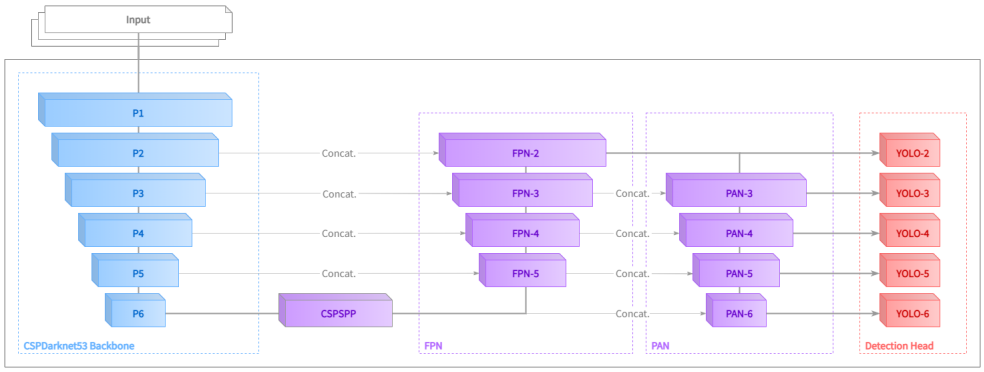
As discussed at length in Chapter 2, the main approach for tackling small object detection was the usage of feature maps of higher resolution, extracted from the early stages of the backbone network.

The majority of the modern real-time object detection architectures already employed multiple feature scales, making use of not only the final feature map produced by the backbone, but also of the result of multiple preceding convolution scales. Nevertheless, the general aim of these models was to balance a high detection accuracy across various object scales with an efficient usage of resources, meaning the feature maps used for detection were chosen with all object scales in consideration. By limiting the scope of our research to the detection of small objects, certain alterations could be made to the structure of these models with the sole objective of benefiting the precision of small object detection, disregarding objects of larger size.

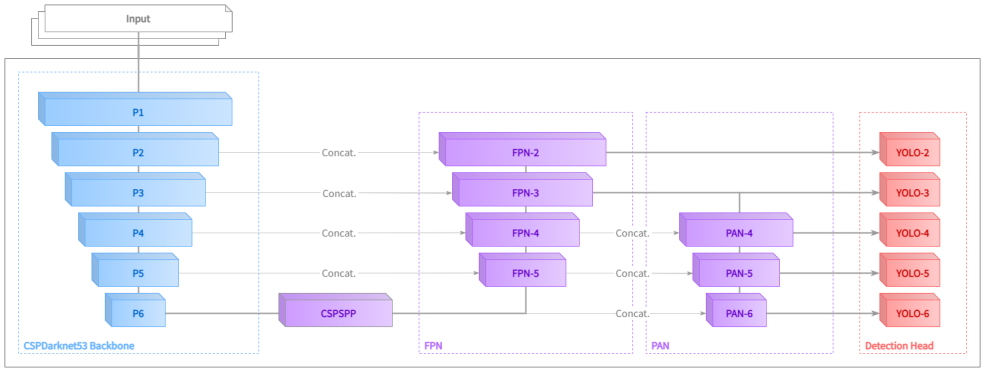
The principal strategy for improving small object detection explored in the project was the addition of the second convolutional feature map, the output of the P2 block, to the detection component of the model. This feature layer, although obtained as part of the backbone's processing of the image, was not leveraged in the neck or head of the base model architecture, possibly due to its low impact on the detection results of larger-scale objects relative to its impact on the running performance of the model.

One consequence of the insertion of the new detection feature map was the resulting five detection layers, in place of the original four, which raised, above all, concerns regarding the model's real-time performance, considering the added quantity of layers and connections posterior to the backbone. The choice of YOLOR-P6 helps to alleviate this impact however, simply by virtue of already vastly surpassing what could be seen in this context as the requirements for real-time detection. The results in Chapter 5 nonetheless conveyed the effect of this added complexity, which will be discussed accordingly.

Introduction of FPN-2 and PAN-3 (Modification A). The first implementation of this approach attempted to incorporate the introduced feature map into the neck in a similar manner to the already included layers, as seen in Figure 3.5a. Following the general structure of the network, the final layer of the backbone was successively up-sampled by the FPN, in this case up to the resolution of the P2 output. As mentioned before, the resulting features were combined throughout this process with the original features obtained by the backbone, before being once again down-scaled by the PANet [54]. In this modified architecture, this component first took the output of the last FPN block, FPN-2, necessitating for it to be down-sampled one additional time, by a PAN-3 block. The inclusion of the



(a) Added FPN-2 and PAN-3.



(b) Added FPN-2.

Figure 3.5: High-level diagrams of the altered model architectures.

P2 feature map in this manner added two more steps to the neck's process in the form of the additional FPN-2 and PAN-3 components, impacting all the detection layers in the second half of the neck. This increase of the model's complexity was also expected to result in a possibly significant loss of computational performance. To keep up with the additional feature scale transported to the detection stage of the model, a YOLO-2 block was added to the detection head, and the number of bounding box anchors was raised by three. The anchors themselves were newly generated and distributed in accordance with the detection head's complexity.

Introduction of FPN-2 without PAN-3 (Modification B). The second approach arose as a response to two of the aspects mentioned in the previous implementation: the impact on subsequent layers and lower of performance. In this version of the model, the additional scale of features was still incorporated into the neck and passed on to the detection component, but without altering the second half of the neck. As visualised in Figure 3.5b, the FPN-2 block was added in a similar way to the previous variant, resulting in the same number of up-scaling operations being present. It was however not used as the input for the subsequent PAN-4 block, which instead still received the output of the FPN-3 block as its input. This not only skipped the previously added PAN-3 segment, but completely detached the output of FPN-2 from the remaining connections in the model's neck. Once again, an additional YOLO-2 block for detection and three box anchors were incorporated to derive predictions from the newly carried over

feature scale.

Additional configurations (Modifications C and D). During the experimental stage of the project, two more architectural changes were implemented, once again in the form of alterations to the neck and detection head of the model. Using the YOLOR-P6 model with Modification A as a basis, both of these configurations focused on returning the number of detection components to the original amount (four rather than five). The first architecture produced saw the removal of the final detection block, corresponding to the P6 scale of features (Modification C), while for the second, we opted to eliminate the YOLOR-3 detection block and replace it with the newly added YOLO-2 block (Modification D). The objective of these alterations was to reduce some of the complexity added to the detection stage of the model with the previous modified versions, seeing as the added quantity of information appeared to be detrimental in the detection process.

3.2.2 Tuning of Bounding Box Anchors

The second avenue explored for improving the detection of small objects was the tuning of the size and frequency of the anchors used to detect objects. These changes did not require the modification of the model's architecture, focusing instead on the tweaking of parameters impacting the generation and placement of anchors.

Tweaking the density of anchors used to analyse each feature map, an approach discussed in Section 2.5 with SCRDet [47], proved to be an unfit option for the project's context and implemented base model. As explained in Section 3.1.2, the YOLOR architecture already used multiple predetermined box anchors for every possible position of the extracted feature maps, leaving no room to increase the proximity between the anchor centres with a "stride" parameter. In the context of the implemented YOLOR-P6 and YOLOR-CSP-X models, anchor stride is instead used to translate the centre position of anchors for each given feature scale to their pixel coordinates in the original image, included in the detection output.

The option of creating anchors with appropriate dimensions for the detection of small objects was also explored, with the intent of possibly improving small object detection at the cost of other object scales. For this, the anchors were set to be generated from a partition of the original data set, containing only small objects, so as to not consider larger object scales when assessing the fitness of the anchors during the process.

Small object anchors. The anchor generation algorithm follows the tendencies of the chosen training data set, which can be limited and modified to create more specialized templates. With a data set partition exclusively comprised of small objects, the generator yielded the following result:

- P3: **16** $\times$ **18**, **19** $\times$ **46**, and **37** $\times$ **31** pixels;
- P4: **36** $\times$ **74**, **68** $\times$ **43**, and **89** $\times$ **110** pixels;

- P5: **209** $\times$ **155**, **137** $\times$ **307**, and **249** $\times$ **433** pixels;
- P6: **511** $\times$ **269**, **476** $\times$ **622**, and **905** $\times$ **656** pixels.

The dimensions obtained were noticeably more compact when compared to the anchors generated by default, notably when analysing the anchors attributed to the first and second scale levels. This set of anchors offered a distinctly more varied array of small object dimension ratios as templates for bounding boxes, increasing the chances of subsequently detecting similar small objects with a more adequate initial outline.

Chapter 4

Experimentation Methodology and Environment

This chapter will serve the purpose of describing the working and testing environment for the second stage of the project, as well as the experimentation methodology used to obtain the results seen in Chapter 5. It includes a brief description of the tools used during development in Section 4.1, followed by Section 4.2, which details the reasoning behind the chosen evaluation metrics and benchmark, and clarifies the manner in which the preliminary and final experiments were carried out.

4.1 Resources and Tools

The development and testing of the potential enhancements discussed in Chapter 3 was mainly achieved using Python version 3.9.7 [55], making use of various machine learning libraries available online, chief among them PyTorch [56]. The training and testing of the implemented models was performed on the NVIDIA Geforce RTX 3090 high-end GPU, with approximately 25GB of VRAM, and compatibility with CUDA version 11.4. The base model implementation in PyTorch was obtained from the original authors' GitHub publication [53], and altered as necessary to fit the context small object detection, with the relevant Python scripts being edited in JupyterLab [57] and executed in a machine containing the aforementioned GPU.

4.2 Model Evaluation

During the experimentation stage of the project, each model was evaluated using a data set containing small objects, with the intent of obtaining results reflecting its real-time small object detection performance. To this end, appropriate evaluation metrics were chosen alongside a fitting data set.

4.2.1 Evaluation Metrics

The main purpose of the planned experiments was to determine the validity of the strategies implemented to improve small object detection. The evaluation was solely focused on the results obtained for small object instances, with the most relevant metric being the mAP score, previously detailed in Section 2.2. In short, average precision (AP) combines the recall and precision of the detector, reflecting its capacity to detect and classify objects respectively. An IoU threshold is also commonly used to determine the true predictions of a detector, which introduces an evaluation of the predicted bounding box into the final mAP score. Simultaneously, it was our intention to maintain the real-time detection capabilities of the base models, attempting to improve the accuracy on small objects while obtaining comparable speed results in relation to the baseline. This aspect required a performance metric to be monitored alongside the others, in this case, the running FPS of the model, derived from its inference time. These additional values were automatically obtained for each of the final experiments, as per originally programmed in the source code.

4.2.2 Evaluation Benchmark

Regarding the application of the evaluated detectors, the intent was to focus on small common objects found in various contextual scenarios, where every object was included within a realistic and potentially complex background where other objects may be present, likely belonging to different categories. This was a relevant aspect of the data, as it enabled the analysis of the information naturally surrounding the object, a recourse that is not available in data sets depicting isolated or cropped objects, such as CURE-OR [58] or CIFAR-10 [59].

The mentioned requirements were adequately fulfilled by the standard COCO data set, used as one of the most relevant real-time object detection benchmarks. The data set provided a clear standard to define small objects, as presented in Section 1.3, making it possible for the results to be analysed independently for the three main object scales: large, medium and small. During the analysis of the available object detection benchmarks, a list of the encountered note-worthy object detection data sets was compiled, given in Table 4.1, along with an assessment of their adequacy for the proposed task, focused on the abundance of small objects and the environment in which they are inserted.

COCO Object Detection Benchmark (2017). The selected benchmark data set was comprised of $118K$ images for training and $5K$ images for validation, used to evaluate the metrics mentioned in Section 4.2.1. Using this benchmark, the main AP metric averaged between 10 different IoU thresholds, ranging from 0.5 to 0.95 with a step of 0.05. Additional AP values were measured for the IoU thresholds of 0.5 and 0.75, as well as each of the three distinct object scales: small ($\leq 32^2$ pixels), medium ($\leq 96^2$ pixels), and large ($> 96^2$ pixels). To evaluate the computational performance of a model, the inference time of each image was measured in milliseconds, from which the FPS could be directly derived.

Data Set	Content Description	Small Objects	Common Objects	Context
COCO [1]	Common objects of various scales in context, with 80 object categories and approximately 330K images.	✓	✓	✓
PASCAL VOC [60]	Common objects of various scales in context, with 20 distinct object categories (VOC 2012).		✓	✓
SUN Database [61]	Scene recognition with labeled objects of various scales, with 5650 object categories and approximately 130K images.	✓	✓	✓
Objcets365 [62]	Common objects of various scales in context, with 365 object categories and 2M images.		✓	✓
Open Images [63]	Common objects of various scales in context, with 600 object classes and 1.9M images.	✓	✓	✓
CURE-OR [58]	Common objects of various scales captured in a controlled environment, with 100 object classes and 1M images.		✓	
CIFAR-10 [59]	Small images of 32×32 pixels containing cropped objects, with 10 object classes equally spread over 60K images.	✓	✓	
CIFAR-100 [59]	Small images of 32×32 pixels containing cropped objects, with 100 object classes equally spread over 60K images.	✓	✓	
FlickrLogos [64]	Company logos of various scales in context, with 32 distinct brand categories and approximately 8K images.	✓		✓
Tsinghua-Tencent 100K [8]	Traffic signs of various scales in context, with 128 traffic sign categories and 100K images.	✓		✓
DOTA [65]	Aerial Images containing objects of small scale, with 15 object classes and approximately 3K images.	✓		✓
KITTI [66]	Road traffic scenes containing pedestrians and objects of various scales, with approximately 7.5K labeled images.	✓		✓
WIDER FACE [67]	Faces of various scales in context, with approximately 32K images containing around 400K identified faces.	✓		✓
Caltech [68]	Pedestrians of various scales in context, with approximately 250K video frames containing around 2.3K unique pedestrians.	✓		✓
ImageMonkey [69]	Open-source data set containing objects of various scales in context, with 516 object categories and approximately 100K images to this date.		✓	✓

Table 4.1: List of relevant object detection data sets, considered for the evaluation of the proposed model.

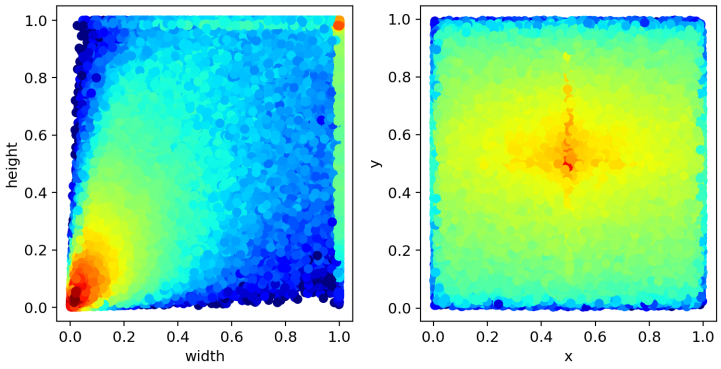


Figure 4.1: Size and position distribution of objects included in the COCO Object Detection benchmark data set. Both metrics are measured in the relation to the total image size.

Lastly, the data set boasted a good distribution of sizes and positions, as seen in Figure 4.1. Regarding the distribution of object centre coordinates, the highest frequency of positions was naturally observed at the centre of the image itself. However, the more clear benefit of using the COCO data set in this context was the object size distribution, which shows that a sizeable amount of the labeled examples was comprised of objects of small dimension in relation to the image size, occupying simultaneously a small portion of vertical and horizontal space.

4.2.3 Preliminary Assessment of Structural Alterations

In the second stage of the project, a significant period of time was dedicated to the exploration of possible enhancements to the base structure of YOLOR-P6, with the objective of improving its ability to detect small objects. This process was characterized by multiple experimental changes to the model of varied scale and impact, which were initially judged based on different sets of reduced tests. The intent was to decrease the amount of time between the implementation of these modifications and the detection performance results necessary for their evaluation, as fully training every version of the model produced would be unfeasible within the given time-frame. These preliminary tests mainly differed from subsequent evaluations when it came to the size of the training data set and the number of iterations afforded in the training process.

Given the large original size of the COCO object detection training data set, of $118K$ images, it only became feasible to superficially assess each configuration's performance on a heavily reduced portion of that set. The chosen size for most comparison tests was of $20K$ images, randomly selected from the original pool. This number allowed for much faster training times and temporary results, while not altering the distribution of classes and size categories relatively to the full data set.

For preliminary assessment, models were trained for a set number of iterations, with prediction results being compared afterwards. This limit varied from experiment to experiment, as to serve the necessities and time constrictions of each set

of alterations. While this approach did not allow for the analysis of the full potential of each structural change, as it occasionally hampered the evolution of models prematurely, it effectively allowed for the evaluation of their relative progression early-on.

For all models trained during this stage of experimentation, the learning parameters were set as follows:

- Momentum: 0.937.
- Weight Decay: 0.0005
- Initial Learning Rate (α): 0.01

The values used to define these parameters were taken from the original YOLOR experiments and left unchanged, with the initial intention of providing the closest possible results to the ones previously obtained with the baseline models. The learning rate value was also varied during training according to the schedule implemented by the original authors, where it was multiplied by a range of set values, alternating between the initial value and as low as 0.002.

4.2.4 Final Evaluation With Full Training Data Set

Nearing the end of the project, two of the developed architectures were evaluated using the full training data set from the COCO Object Detection Benchmark. Although these tests were prolonged as much as possible within the available time window, the number of learning training iterations set for each model was lower than had initially been planned due to the time required to train them using the available resources. In Figure 4.2, the YOLOR-P6 model's training process can be observed, showing the evolution of results and loss values over time. Although the evolution of the model predictably starts slowing down in a gradual manner after the first iterations, the reduced quantity of training iterations was not enough to reach a higher stability than the one displayed in the learning curve.

After training, the results were collected using the provided validation data set, containing $5K$ images, providing information related to the detection precision and computational performance of the models For these experiments, the training parameters were kept identical to those of the preliminary tests.

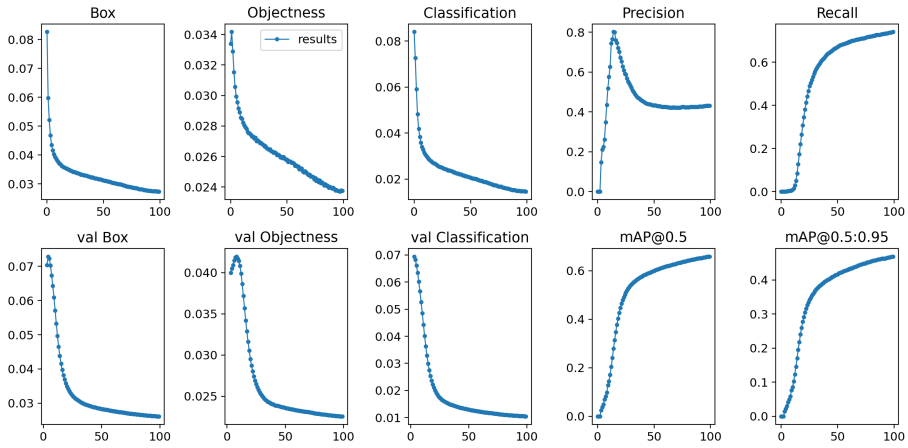


Figure 4.2: Learning curves of the YOLOR-P6 base model when trained for 100 epochs on the full COCO training data set. The six graphs on the left show the evolution of the loss values throughout training, while the four on the right represent the scores obtained.

Chapter 5

Results and Discussion

In this chapter, the results obtained as part of the final evaluation of the tested models will be presented, in Section 5.1, and discussed, in Section 5.2. The discussion will be comprised of an analysis of the potential causes behind the obtained results, and a reflection on the possible directions that the research concerning this problem can take in the future.

5.1 Evaluation Results

This section is dedicated to the presentation of the results obtained during the two experimentation phases of the project. The outcome of the preliminary testing phase is shown first, followed by the results obtained using the complete training data set.

5.1.1 Preliminary Test Results

Preliminary tests were carried out for all of the experimented configurations, focused on obtaining and analysing early results in a reasonably shorter time frame. For the results presented in this section, models were only trained for 50 epochs, using a partition of the originally chosen training data set, containing $20K$ randomly selected from the total pool of $118K$. The obtained AP values across different IoU thresholds and object scales are presented for the different architecture configurations in Table 5.1a and the tuned bounding box anchor generation in Table 5.1b.

Precision values obtained in this round of testing summarized the trend seen with the implemented configurations, with similar or lower small object detection AP results. The main architecture modifications (A and B) saw a slight decline in precision, while less significant changes such as the ones made with Modification D resulted in nearly identical AP values. The usage of box anchors generated for small objects were also shown to negatively impact the performance of the base YOLOR-P6 model, more so when dealing with objects of larger scale, as expected

Model	Size	AP	$AP_{0.5}$	$AP_{0.75}$	AP_S	AP_M	AP_L
YOLOR-P6	1280	28.7%	43.7%	30.9%	15.5%	31.5%	36.1%
YOLOR-P6 Mod. A	1280	25.5%	39.7%	27.3%	13.5%	27.5%	32.3%
YOLOR-P6 Mod. B	1280	27.4%	41.6%	29.7%	14.0%	30.0%	35.7%
YOLOR-P6 Mod. C	1280	22.2%	34.9%	23.9%	12.3%	26.1%	26.3%
YOLOR-P6 Mod. D	1280	28.8%	43.4%	31.4%	15.3%	32.0%	36.0%

(a) YOLOR-P6 base and modified models.

Model	Size	AP	$AP_{0.5}$	$AP_{0.75}$	AP_S	AP_M	AP_L
YOLOR-P6	1280	28.7%	43.7%	30.9%	15.5%	31.5%	36.1%
YOLOR-P6 Tuned Anchors	1280	26.8%	42.0%	28.9%	14.5%	29.4%	32.6%

(b) YOLOR-P6 base model with bounding box anchors generated by default and for small objects.

Table 5.1: Precision results obtained with 50 iterations of training on a segment of the COCO training data set (20K images), using the NVIDIA Geforce RTX 3090. AP_S, AP_M, and AP_L denote the AP values across the small, medium, and large object scales, respectively.

from the addition of lower-level feature maps.

5.1.2 Full Data Set Results

As addressed in the previous chapter, the architecturally modified versions of YOLOR-P6 and YOLOR-CSP-X were tested using the COCO training data set in its integrity, composed of 118K images. The number of training epochs afforded to the models during this phase of testing was somewhat limited, with versions of YOLOR-P6 being trained for 100 epochs, and versions of YOLOR-CSP-X being trained for 150 epochs. The performance of these models was evaluated under the conditions provided by the NVIDIA Geforce RTX 3090 GPU, with the batch size set to 1, so as to simulate the processing of individual images in a practical application. The obtained results are presented in Table 5.2a and Table 5.2b.

As can be observed in the obtained results, the modifications made to the YOLOR-P6 and YOLOR-CSP-X architectures had a clear impact in the both of the models' computational performance, while resulting in similar or lower AP values. Using the YOLOR-P6 model, the decrease in precision resulting from these modifications was seen across all of the IoU thresholds and object scales included, while the results produced by the YOLOR-CSP-X architectures were less distinct from one another. It is important to not compare the values obtained with the different architectures, as the number of training iterations afforded to each of them was unequal in this context. The precision-recall curve obtained for the YOLOR-P6 model variants can be observed in Figure 5.1, providing a visual representation of the slight difference seen in the AP values obtained. In the graph, we can see the curves obtained by the two modified architectures mostly overlap, along with the

Model	Size	FPS	AP	$AP_{0.5}$	$AP_{0.75}$	AP_S	AP_M	AP_L
YOLOR-P6	1280	**49**	**48.8**%	**66.6**%	**53.4**%	**32.9**%	**53.3**%	**60.9**%
YOLOR-P6 Mod. A	1280	41	46.8%	64.0%	51.5%	30.9%	51.6%	60.1%
YOLOR-P6 Mod. B	1280	45	47.1%	64.4%	51.5%	29.3%	52.2%	59.8%

(a) YOLOR-P6 base and modified models, trained for 100 iterations.

Model	Size	FPS	AP	$AP_{0.5}$	$AP_{0.75}$	AP_S	AP_M	AP_L
YOLOR-CSP-X	640	**44**	50.1%	**68.5**%	54.6%	**34.6**%	**55.3**%	63.8%
YOLOR-CSP-X Mod. A	640	38	**50.3**%	68.2%	**55.0**%	34.0%	55.2%	64.3%
YOLOR-CSP-X Mod. B	640	40	49.9%	68.2%	54.6%	33.3%	55.0%	**64.4**%

(b) YOLOR-CSP-X base and modified models, trained for 150 iterations.

Table 5.2: Performance and precision results obtained with the full COCO training data set, using the NVIDIA Geforce RTX 3090 and batch size set to 1. AP_S, AP_M, and AP_L denote the AP values across the small, medium, and large object scales respectively.

significant spread of the curves obtained separately for every single object class.

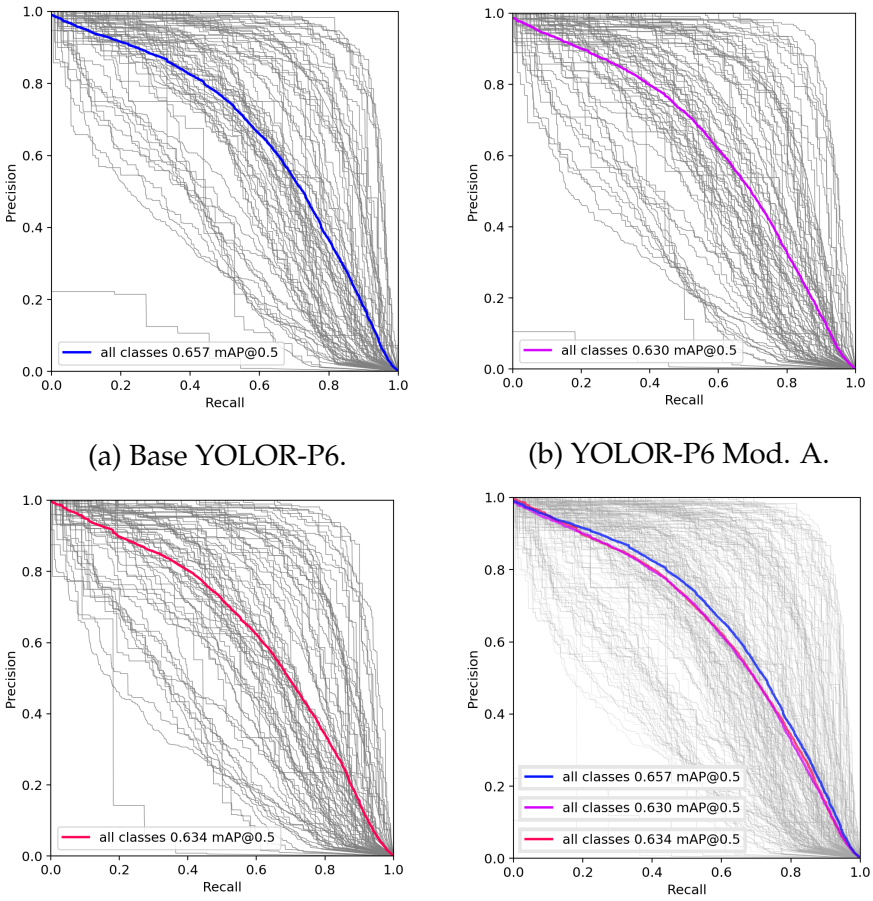
5.2 Discussion

This section will be dedicated to a discussion of the outcome of the presented experiments, as well as an assessment of the possible factors behind the obtained results. The two more significant techniques will be addressed: implementation of an additional detection scale, and bounding box anchor generation tuned for small objects. Moreover, other potential approaches to the problem will be discussed.

5.2.1 Addition of Higher Resolution Detection Component

The most important conclusion to be drawn from this set of experiments was the ineffectiveness of the addition of the features produced by the P2 block of the backbone network, using the proposed model architectures, when seeking to improve object detection results at any scale. This addition, previously thought to potentially benefit the detection of small objects, resulted instead in a loss both in small object detection accuracy and computational performance when applied to the YOLOR-P6 architecture, and no distinct improvements when implemented with YOLOR-CSP-X as the base model.

With the widespread usage of multiple feature maps of varying scale in standard and real-time object detection, models such as YOLOR are now optimized to take full advantage of the relevant information that can be extracted by their backbone networks. In this case, the chosen four levels of detection (from the P3 to the P6 network segments) appeared to be adequate in the detection of object of all the



(a) Base YOLOR-P6.

(b) YOLOR-P6 Mod. A.

(c) YOLOR-P6 Mod. B.

(d) Simultaneous comparison.

Figure 5.1: Three distinct precision-recall curves obtained in the full data set experiment with YOLOR-P6. The grey surrounding lines in each graph represent the curves of each individual object category.

previously defined scales, small, medium, and large. Though detection results for small objects remain inevitably lower due to the challenges mainly brought on by their lower resolution, the addition of lower-level features at this stage presented no value in the pursue of lessening this disparity.

5.2.2 Bounding Box Tuning for Small Objects

From the proposed approaches to the problem, the generation of bounding box anchors for small objects had the least potential to result in a significant improvement over the anchors used by default. The fact that each detection scale was already complete with three distinct anchors, making up a total number of 12 for the YOLOR-P6 model, conjoined with the adjustable nature of their dimensions to fit the detected objects, limited the impact that a more focused anchor generation might have on a the model's detection precision.

The initial intent behind this change was to improve the model's ability to accurately localise small objects, and by extension detect them, expectedly at the

cost of the detection precision of larger objects. However, by reducing the variety of object sizes and ratios that the anchors have to loosely be adapted to, the algorithm's task was greatly facilitated, to the point where the original quantity of anchors may have no longer been needed. Based on the results obtained, we were compelled to concede that the previous quantity and variety of original anchors was adequate to cover all three of the defined object scales.

5.3 Future Research Directions

On the topic of future research directions regarding the problem of small object detection, various possibilities can be found in the unexplored approaches presented in Section 2.5. The two stand out techniques that remained outside of the scope of this project were the usage of contextual information, and super-resolved object representations. Additionally, there are the explored methods related to the usage of lower-level features, and the tuning of anchors for small objects.

5.3.1 Explored Approaches

Due to the research presented in Chapter 2 coupled with the outcome of the chosen architectural changes for implementation during the project's second phase, many aspects of the explored approaches have no feasible continuity in modern object detection research. Though these avenues did produce the desired results in the context of this project, it is also important not to rule out their potential when incorporated through different means or into other model architectures.

Higher resolution feature maps. As previously discussed, the incorporation of higher resolution information into the model's detection process already is a widespread method used in most modern object detection architectures. YOLOR is no exception to this, with the inclusion of lower-level features extracted by earlier layers of its backbone, all the way up to the P3 feature scale. Models previously made sole use of the last feature map extracted by the backbone to detect any object, regardless of scale. However, what was once an innovative addition in the form of features extracted from the middle of the network is now a mainstay characteristic of model architectures focused on obtaining good detection performance across varied object scales. To further improve the detection of small objects, the best course of action might be to follow the current trend and go the opposite direction by scaling up models in depth and width [38; 39], serving the model's overall detection performance.

Anchor tuning for small objects. Similarly to the previous approach, the customisation of bounding box anchors to fit small objects is in some ways already achieved for modern models through their automatic generation, as seen in this project with YOLOR. One of the issues discussed in Section 2.5 was the space between the placed anchors, which could be minimised to create a more thorough analysis of the image and place box templates closer on average to the center of

objects. In the context of this project and YOLOR, anchors are already placed across all the possible positions of the used feature maps, resulting in a high density of outputs, leaving little room to meaningfully improve the process.

5.3.2 Unexplored Methods

Beyond the modifications included in the experiments, there were other options explored by the methods mentioned in Section 2.5. Approaches such as the generation of super-resolved representations of small objects offer potential future solutions to the problem at hand, by tackling it from a completely distinct perspective.

Super-resolution. Low resolution object representation is the defining challenge of small object detection, and what makes it as hard to circumvent as it is. One possible measure to address this problem is using machine learning to artificially increase this resolution, enhancing the image's detail [48; 49; 70], which has commonly been achieved with GAN and autoencoder architectures. Though it has been recently explored as an option to decrease the accuracy gap between the detection of small and larger objects, obtaining favourable results, it presents a constantly evolving solution to the issue, which should be considered for most applications relying on the frequent and accurate detection of small objects. A potential avenue for future research in this topic presents itself in developing the two technologies simultaneously, by implementing the image up-scaling and object detection into a single end-to-end model focused on the task of object detection [71]. It is important note that, while the original small object detection problem stemmed from the limited information available in low resolution representations, as is explored in this document, this approach shifts the main research concern to the improvement of the accuracy in the reconstruction of low resolution image features.

Chapter 6

High-Level Planning

This chapter will briefly detail the planning of the tasks performed throughout the project. It will disclose the initial high-level plan laid out for the second semester, the changes made to it over the course of the project, and the final distribution of work throughout this stage.

As discussed in Section 1.2, the project was divided into two stages focused on research and experimentation respectively. The first semester, chiefly allocated to the reading and compiling related work, did not require any documented prior planning, whereas the second semester's work was distributed along the available time before it was tackled. Initially, this progression was planned as follows:

1. **Research and design**: from February 7^{th} to March 6^{th}.
2. **Development environment**: from February 14^{th} to February 27^{th}.
3. **Data set preparation**: from February 21^{th} to March 13^{th}.
4. **Implementation of existent models**: from March 7^{th} to March 27^{th}.
5. **Evaluation (existent models)**: from March 28^{st} to April 17^{th}.
6. **Implementation of proposed changes**: from April 11^{th} to May 8^{th}.
7. **Evaluation (proposed models)**: from May 9^{th} to May 29^{th}.
8. **Analysis and report**: from May 16^{th} to June ^{26th}.

The chronological distribution of the planned experiments gradually shifted forward in time over the course of the semester do to the implementation of additional modified architectures, which required additional training and testing. A degree of underestimation of the time required to train the YOLOR models from the ground also greatly contributed to the gap between the original plan and the final progression of events. While certain aspects of the work were developed independently from experimentation, such as the implementation of models and writing, the long time periods required to obtain results delayed the assessment of the performed modifications, generally delaying the conclusion of the project

or any necessary adjustments to its scope and direction. The resulting distribution of tasks was the following:

1. **Research and design**: from February 14^{th} to April 17^{th}.
2. **Development environment**: from February 14^{th} to February 27^{th}.
3. **Data set preparation**: from February 21^{st} to March 13^{th}.
4. **Implementation of existent models**: from February 28^{th} to April 17^{th}.
   Base YOLOR-P6 and Scaled-YOLOv4: February 28^{th} - March 20^{th}.
   Base YOLOR-CSP-X: April 11^{th} - April 17^{th}.
5. **Implementation of proposed changes**: from March 14^{th} to June 19^{th}.
   YOLOR-P6 architecture modifications: March 14^{th} - May 17^{th}.
   YOLOR-P6 tuned anchors: May 9^{th} - June 12^{th}.
   YOLOR-CSP-X architecture modifications: June 6^{th} - June 19^{th}.
6. **Evaluation (base and modified models)**: from March 21^{st} to August 21^{st}.
   Preliminary YOLOR-P6 architecture experiments: March 21^{st} - May 29^{th}.
   Preliminary YOLOR-P6 tuned anchor experiments: May 16^{th} - July 3^{rd}.
   Final evaluation of modified YOLOR-CSP-X: July 4^{th} - July 24^{th}.
   Final evaluation of modified YOLOR-P6: July 25^{th} - August 21^{st}.
7. **Analysis and report**: from May 2^{nd} to September 4^{th}.
   Implementation and methodology: May 2^{nd} - July 31^{st}.
   Result analysis and discussion: August 15^{th} - September 4^{th}.

Additionally, a Gantt chart representing a more detailed evolution of the project can be consulted in Appendix A.

Chapter 7

Conclusion and Remarks

In this work, we explored previously proposed solutions for the problem of object detection, incorporating them into state-of-the-art real-time object detection models, namely YOLOR-P6 and YOLOR-CSP-X, variants of the YOLOR architecture [37].

The main considered approach was the addition of higher resolution features, extracted from lower level blocks of the backbone network, to the detection component of the chosen base models through their inclusion into the second stage of the pipeline, the model's neck. The second option explored was the generation of bounding box anchors exclusively for the dimensions of small objects, through their separation from the rest of the chosen data set.

Both of the approaches followed did not yield positive results, resulting in an expected loss of computational performance coupled with slight decrease in detection accuracy across the board, proving to be detrimental extensions in the context of modern object detection architectures under the used training and evaluation conditions. Further research focused on reducing the gap between the detection accuracy of small and large scale objects could instead be directed towards the improvement of image up-scaling through super-resolution, which has the potential to bring the level detail found in low resolution object representations in line with that of higher object scales.

References

[1] 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.

[2] Nhat-Duy Nguyen, Tien Do, Thanh Duc Ngo, and Duy-Dinh Le. An evaluation of deep learning methods for small object detection. *Journal of Electrical and Computer Engineering*, 2020, 2020.

[3] Kang Tong, Yiquan Wu, and Fei Zhou. Recent advances in small object detection based on deep learning: A review. *Image and Vision Computing*, 97:103910, 2020.

[4] Guang Chen, Haitao Wang, Kai Chen, Zhijun Li, Zida Song, Yinlong Liu, Wenkai Chen, and Alois Knoll. A survey of the four pillars for small object detection: Multiscale representation, contextual information, super-resolution, and region proposal. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[5] Yang Liu, Peng Sun, Nickolas Wergeles, and Yi Shang. A survey and performance evaluation of deep learning methods for small object detection. *Expert Systems with Applications*, page 114602, 2021.

[6] Chenyi Chen, Ming-Yu Liu, Oncel Tuzel, and Jianxiong Xiao. R-cnn for small object detection. In *Asian conference on computer vision*, pages 214–230. Springer, 2016.

[7] Zhigang Liu, Juan Du, Feng Tian, and Jiazheng Wen. Mr-cnn: A multi-scale region-based convolutional neural network for small traffic sign recognition. *IEEE Access*, 7:57120–57128, 2019.

[8] Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. Traffic-sign detection and classification in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2110–2118, 2016.

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

[10] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

[11] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[12] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[16] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[18] Max Ferguson, Ronay Ak, Yung-Tsun Tina Lee, and Kincho H Law. Automatic localization of casting defects with convolutional neural networks. In *2017 IEEE international conference on big data (big data)*, pages 1726–1735. IEEE, 2017.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[21] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[23] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017.

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[25] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[26] 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.

[27] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[28] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[29] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[30] Pulak Purkait, Cheng Zhao, and Christopher Zach. Spp-net: Deep absolute pose regression with synthetic views. *arXiv preprint arXiv:1712.03452*, 2017.

[31] Xiang Long, Kaipeng Deng, Guanzhong Wang, Yang Zhang, Qingqing Dang, Yuan Gao, Hui Shen, Jianguo Ren, Shumin Han, Errui Ding, et al. Pp-yolo: An effective and efficient implementation of object detector. *arXiv preprint arXiv:2007.12099*, 2020.

[32] Christian Eggert, Stephan Brehm, Anton Winschel, Dan Zecha, and Rainer Lienhart. A closer look: Small object detection in faster r-cnn. In *2017 IEEE international conference on multimedia and expo (ICME)*, pages 421–426. IEEE, 2017.

[33] Lisha Cui, Rui Ma, Pei Lv, Xiaoheng Jiang, Zhimin Gao, Bing Zhou, and Mingliang Xu. Mdssd: multi-scale deconvolutional single shot detector for small objects. *arXiv preprint arXiv:1805.07009*, 2018.

[34] Coco test-dev benchmark (object detection). `https://paperswithcode.com/sota/object-detection-on-coco`. Accessed: 11-2021.

[35] Coco benchmark (real-time object detection). `https://paperswithcode.com/sota/real-time-object-detection-on-coco`. Accessed: 11-2021.

[36] 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, 2021.

[37] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. You only learn one representation: Unified network for multiple tasks. *arXiv preprint arXiv:2105.04206*, 2021.

[38] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13029–13038, 2021.

[39] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[40] Peiyun Hu and Deva Ramanan. Finding tiny faces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 951–959, 2017.

[41] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[42] Guimei Cao, Xuemei Xie, Wenzhe Yang, Quan Liao, Guangming Shi, and Jinjian Wu. Feature-fused ssd: Fast detection for small objects. In *Ninth International Conference on Graphic and Image Processing (ICGIP 2017)*, volume 10615, page 106151E. International Society for Optics and Photonics, 2018.

[43] Xi Liang, Jing Zhang, Li Zhuo, Yuzhao Li, and Qi Tian. Small object detection in unmanned aerial vehicle images using feature fusion and scaling-based single shot detector with spatial context analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(6):1758–1770, 2019.

[44] Ziming Liu, Guangyu Gao, Lin Sun, and Zhiyuan Fang. Hrdnet: high-resolution detection network for small objects. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2021.

[45] Yun Ren, Changren Zhu, and Shunping Xiao. Small object detection in optical remote sensing images via modified faster r-cnn. *Applied Sciences*, 8(5):813, 2018.

[46] Chenchen Zhu, Ran Tao, Khoa Luu, and Marios Savvides. Seeing small faces from robust anchor's perspective. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5127–5136, 2018.

[47] Xue Yang, Jirui Yang, Junchi Yan, Yue Zhang, Tengfei Zhang, Zhi Guo, Xian Sun, and Kun Fu. Scrdet: Towards more robust detection for small, cluttered and rotated objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8232–8241, 2019.

[48] Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Perceptual generative adversarial networks for small object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1222–1230, 2017.

[49] Yancheng Bai, Yongqiang Zhang, Mingli Ding, and Bernard Ghanem. Sodmtgan: Small object detection via multi-task generative adversarial network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 206–221, 2018.

[50] Junhyug Noh, Wonho Bae, Wonhee Lee, Jinhwan Seo, and Gunhee Kim. Better to follow, follow to be better: Towards precise supervision of feature super-resolution for small object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9725–9734, 2019.

[51] Yanwei Pang, Jiale Cao, Jian Wang, and Jungong Han. Jcs-net: Joint classification and super-resolution network for small-scale pedestrian detection in surveillance images. *IEEE Transactions on Information Forensics and Security*, 14(12):3322–3331, 2019.

[52] Mate Kisantal, Zbigniew Wojna, Jakub Murawski, Jacek Naruniec, and Kyunghyun Cho. Augmentation for small object detection. *arXiv preprint arXiv:1902.07296*, 2019.

[53] Official yolor github repository. `https://github.com/WongKinYiu/yolor`. Accessed: 02-2022.

[54] 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.

[55] Official python web page. `https://www.python.org/`. Accessed: 02-2022.

[56] Official pytorch github repository. `https://github.com/pytorch/pytorch`. Accessed: 02-2022.

[57] Official jupyter webpage. `https://jupyter.org/`. Accessed: 02-2022.

[58] Dogancan Temel, Jinsol Lee, and Ghassan AlRegib. Cure-or: Challenging unreal and real environments for object recognition. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pages 137–144. IEEE, 2018.

[59] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[60] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[61] Jianxiong Xiao, Krista A Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, 119(1):3–22, 2016.

[62] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8430–8439, 2019.

[63] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018.

[64] Stefan Romberg, Lluis Garcia Pueyo, Rainer Lienhart, and Roelof Van Zwol. Scalable logo recognition in real-world images. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, pages 1–8, 2011.

[65] Jian Ding, Nan Xue, Yang Long, Gui-Song Xia, and Qikai Lu. Learning roi transformer for detecting oriented objects in aerial images. *arXiv preprint arXiv:1812.00155*, 2018.

[66] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[67] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5525–5533, 2016.

[68] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):743–761, 2011.

[69] Imagemonkey homepage. `https://imagemonkey.io/`. Accessed: 11-2021.

[70] Jacob Shermeyer and Adam Van Etten. The effects of super-resolution on object detection performance in satellite imagery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

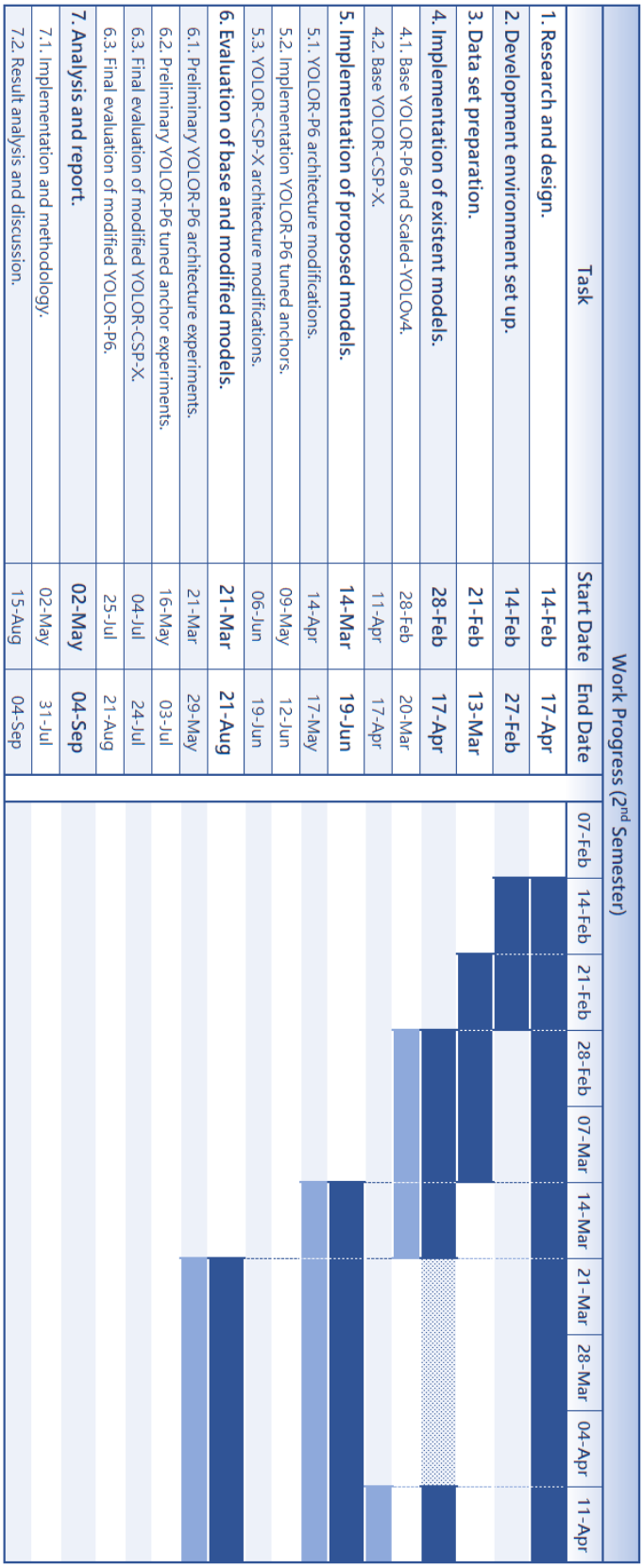
[71] Muhammad Haris, Greg Shakhnarovich, and Norimichi Ukita. Task-driven super resolution: Object detection in low-resolution images. In *International Conference on Neural Information Processing*, pages 387–395. Springer, 2021.

Appendices

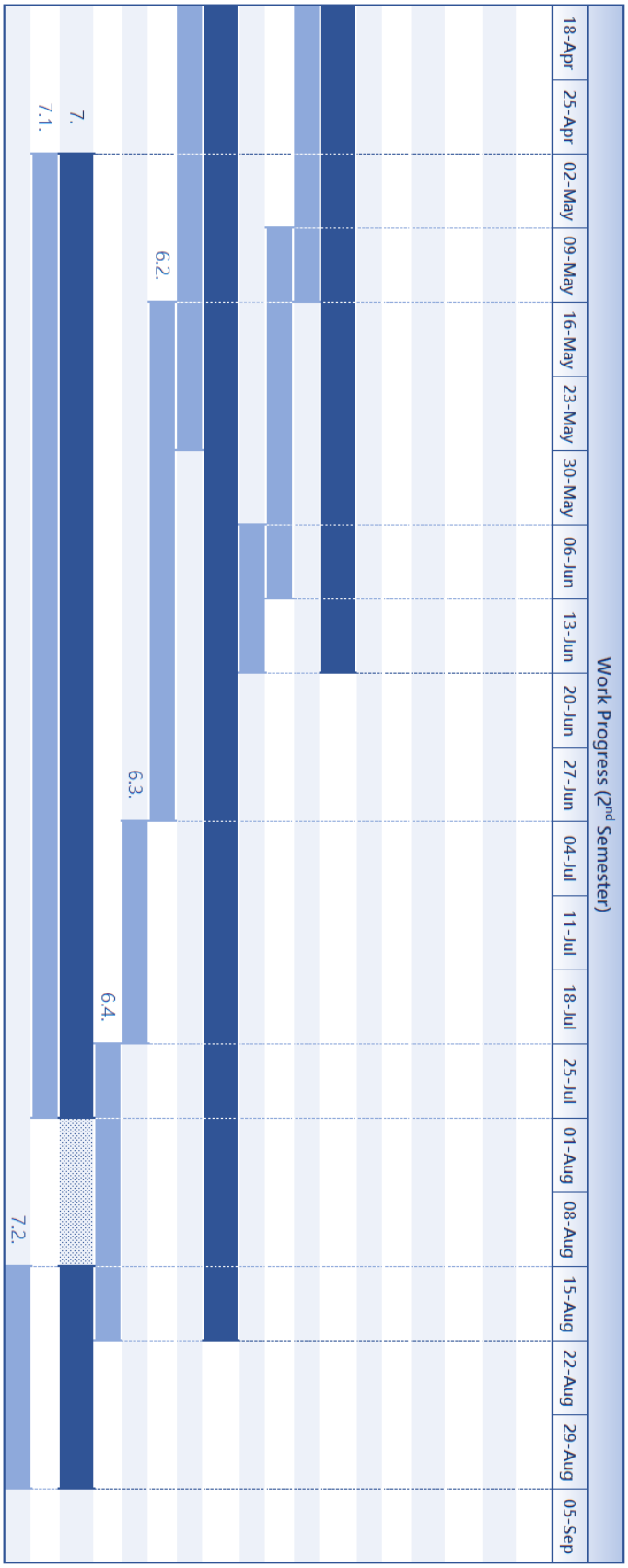
Appendix A

Progress Chart

In this appendix, a Gantt chart is included to help visualize the progression of the tasks performed during the second semester. The chart is divided into weeks (columns), each starting on Monday, and is split into two pages for visibility reasons. Some of the lengthier tasks included are broken up into sub-tasks, such as the implementation and evaluation of different base and modified models.



Task	Start Date	End Date
1. Research and design.	**14-Feb**	**17-Apr**
2. Development environment set up.	**14-Feb**	**27-Feb**
3. Data set preparation.	**21-Feb**	**13-Mar**
4. Implementation of existent models.	**28-Feb**	**17-Apr**
4.1. Base YOLOR-P6 and Scaled-YOLOv4.	28-Feb	20-Mar
4.2. Base YOLOR-CSP-X.	11-Apr	17-Apr
5. Implementation of proposed models.	**14-Mar**	**19-Jun**
5.1. YOLOR-P6 architecture modifications.	14-Apr	17-May
5.2. Implementation YOLOR-P6 tuned anchors.	09-May	12-Jun
5.3. YOLOR-CSP-X architecture modifications.	06-Jun	19-Jun
6. Evaluation of base and modified models.	**21-Mar**	**21-Aug**
6.1. Preliminary YOLOR-P6 architecture experiments.	21-Mar	29-May
6.2. Preliminary YOLOR-P6 tuned anchor experiments.	16-May	03-Jul
6.3. Final evaluation of modified YOLOR-CSP-X.	04-Jul	24-Jul
6.3. Final evaluation of modified YOLOR-P6.	25-Jul	21-Aug
7. Analysis and report.	**02-May**	**04-Sep**
7.1. Implementation and methodology.	02-May	31-Jul
7.2. Result analysis and discussion.	15-Aug	04-Sep


Work Progress (2nd Semester)
18-Apr
25-Apr
02-May
09-May
16-May
23-May
30-May
06-Jun
13-Jun
20-Jun
27-Jun
04-Jul
11-Jul
18-Jul
25-Jul
01-Aug
08-Aug
15-Aug
22-Aug
29-Aug
05-Sep
7.
7.1.
6.2.
6.3.
6.4.
7.2.