IMPROVEMENTS ON ONE-STAGE OBJECT DETECTION BY VISUAL REASONING

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

BY

TOLGA AKSOY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

MAY 2022

Approval of the thesis:

IMPROVEMENTS ON ONE-STAGE OBJECT DETECTION BY VISUAL REASONING

submitted by **TOLGA AKSOY** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** _____

Prof. Dr. Uğur Halıcı
Supervisor, **Electrical and Electronics Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering, METU _____

Prof. Dr. Uğur Halıcı
Electrical and Electronics Engineering, METU _____

Prof. Dr. İlkay Ulusoy
Electrical and Electronics Engineering, METU _____

Prof. Dr. Alptekin Temizel
Informatics Institute, METU _____

Assist. Prof. Dr. Mehmet Dikmen
Computer Engineering, BASKENT UNIVERSITY _____

Date: 09.05.2022

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

Name, Surname: Tolga Aksoy

Signature :

ABSTRACT

IMPROVEMENTS ON ONE-STAGE OBJECT DETECTION BY VISUAL REASONING

Aksoy, Tolga
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Uğur Halıcı

May 2022, 62 pages

Current state-of-the-art one-stage object detectors are limited by treating each image region separately without considering possible relations of the objects. This causes dependency solely on high-quality convolutional feature representations for detecting objects successfully. However, this may not be possible sometimes due to some challenging conditions. In this thesis, a new architecture is proposed for one-stage object detection that reasons the relations of the image regions by using self-attention. The proposed reasoning method considers semantic coherency between image regions and enhances features of these regions. Spatially and semantically enhanced features are fused with original features to improve performance. The proposed approach is applied to the current state-of-the-art real-time one-stage object detectors such as YOLOv3, YOLOv4 and YOLOR, then evaluated on COCO in terms of mAP.

Keywords: object detection, one-stage object detection, visual reasoning

ÖZ

GÖRSEL AKIL YÜRÜTME İLE TEK AŞAMALI NESNE TESPİTİNDE İYİLEŞTİRMELER

Aksoy, Tolga
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Uğur Halıcı

Mayıs 2022 , 62 sayfa

Mevcut son teknoloji tek aşamalı nesne tespiti algoritmaları, nesnelerin olası ilişkilerini dikkate almadan her bir görüntü bölgesini ayrı ayrı ele alarak sınırlandırılmıştır. Bu, nesneleri başarılı bir şekilde algılamak için yalnızca yüksek kaliteli evrişimli öznitelik temsillerine bağlılığa neden olmaktadır. Ancak bu yaklaşım, bazı zorlu koşullar nedeniyle her zaman mümkün olmayabilir. Bu tezde, öz-dikkat kullanarak görüntü bölgelerinin ilişkilerini oluşturan tek aşamalı nesne tespiti için yeni bir mimari önerilmiştir. Önerilen akıl yürütme yöntemi, görüntü bölgeleri arasındaki anlamsal tutarlılığı dikkate almakta ve bu bölgelerin özelliklerini geliştirmektedir. Uzamsal ve anlamsal olarak geliştirilmiş öznitelikler performansı iyileştirmek için orijinal öznitelikler ile birleştirilmektedir. Önerilen yaklaşım YOLOv3, YOLOv4 ve YOLOR gibi gerçek zamanlı olarak çalışan son teknoloji tek aşamalı nesne tespiti algoritmalarına uygulanmış, sonrasında COCO'da mAP ölçütüne göre değerlendirilmiştir.

Anahtar Kelimeler: nesne tespiti, tek aşamalı nesne tespiti, görsel akıl yürütme

To my dearest family...

ACKNOWLEDGMENTS

I want to thank my supervisor Prof. Dr. Uğur Halıcı, for her guidance, positive attitude, and support for this research.

The numerical calculations reported in this thesis were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources) through Neuroscience and Neurotechnology Center of Excellence (NÖROM).

Also, I wish to thank my friend Ferhat Can Ataman and my colleagues in ASELSAN Inc., Berk Kemal Özata, and Ufuk Umut Şentürk for their help and valuable feedback.

Finally, I am grateful to my family for their support and patience throughout my studies.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

IoU	Intersection over Uniun
AP	Average Precision
mAP	Mean Average Precision
TP	True Positive
FP	False Positive
FN	False Negative
FPS	Frames Per Second

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Object detection aims to classify and localize objects of interest in a given image. It has attracted the great attention of the community because of its close ties with other computer vision applications. Many traditional methods have been proposed to solve object detection problem before the major breakthrough in the deep learning area. These methods [5, 6, 7, 8, 9] were built on handcrafted feature representations. Inevitable dependency on handcrafted features limited the performance of traditional approaches.

The significant impact of the AlexNet [10], which is a deep neural network architecture, has put a new complexion on the object detection approaches, and then deep learning based methods have entirely dominated the literature. Deep learning based detectors can be divided into two categories: two-stage object detectors and one-stage object detectors. Two-stage detectors have low inference speeds due to the intermediate layer used to propose possible object regions. The region proposal layer extracts regions of objects in the first stage. In the second stage, these proposed regions are used for classification and bounding box regression. On the other hand, one-stage detectors could predict all the bounding boxes and class probabilities in a single pass with high inference speeds. This makes one-stage detectors more suitable for real-time applications.

Recent one-stage object detectors [3, 11, 1, 12, 13, 14] achieve good performance on datasets such as MS COCO [4] and PASCAL VOC [15]. However, they lack the ability to consider possible relations between image regions. The current one-stage

detectors treat each image region separately. They are unaware of distinct image regions due to small receptive fields when image size is considered. They depend solely on high-quality local convolutional features to detect objects successfully. However, this is not the way how human visual system works. Humans have the ability to reason to carry out visual tasks with the help of acquired knowledge. Many methods [16, 17, 18, 19] have been proposed to mimic human reasoning ability in object detection. On the other hand, these methods are mostly complicated and use two-stage detection architectures. Thus, they are not applicable to real-time applications.

In this thesis, a new architecture is proposed for one-stage object detection that can reason about the relationships between image regions. The proposed reasoner method takes into consideration semantic coherency between image regions to predict bounding boxes and class probabilities.

1.2 Contributions of the Thesis

Our contributions can be summarized as follows:

- One-stage object detection is improved by visual reasoning. A new architecture is proposed that can model and use semantic relations between image regions to predict bounding boxes and class probabilities.
- A transformer encoder-like [20] multi-head attention based reasoning layer is proposed to reason between image regions.
- The effect of using only the reasoning layer's features like DETR [19] on object detection performance is examined. A reasoner architecture is proposed that fuses both backbone output only-convolutional and reasoning features. It is shown that this model achieves the best performance improvement over the baseline network YOLOv3 while still running in real-time.
- The effect of utilizing reasoning on average precision improvement for each object category is analyzed.
- The proposed reasoner architecture is applied to the more recent state-of-the-art one-stage object detectors: YOLOv4-P6 [14], and YOLOR-P6 [21], then

evaluated in terms of mAP.

1.3 The Outline of the Thesis

In Chapter 2 presents the current state of the object detection problem. Firstly, a definition of object detection is given. Then, traditional object detectors are examined. It is explained how deep learning causes the evolvement of the proposed object detection methods. Finally, one-stage object detection and proposed reasoning approaches in object detection are investigated.

In Chapter 3, proposed method is explained in detail. Used baseline networks are presented. Then, the reasoning layer is described with all its sublayers. The role of a multi-head attention layer is given in modeling semantic relationships between image regions. Lastly, tried reasoner network configurations and differences between them are presented.

In Chapter 4, experimental results are shared. Firstly, the dataset, implementation details, and evaluation metrics are introduced. Then, early evaluation results of different reasoner configurations which use YOLOv3 as a baseline network are given. Performance improvements in different object categories are explained in detail. In the quantitative evaluation part, proposed reasoner networks YOLOv3-Reasoner2, YOLOv3-Reasoner4, YOLOv4-P6-Reasoner, and YOLOR-P6-Reasoner are compared with their baseline networks and other state-of-the-art object detection methods. Finally, qualitative comparisons between proposed reasoner networks and their baselines are given.

In Chapter 5, the work done in this thesis is summarized, and discussions are provided.

CHAPTER 2

RELATED WORK AND BACKGROUND

In this chapter, object detection methods and what type of reasoning approaches are applied to object detection problem is examined. Firstly, the object detection problem is defined. Then, traditional and deep learning based object detection methods are explained in detail. After that, the reasoning approaches applied to object detection problem in literature are presented.

2.1 Object Detection

Object detection is one of the challenging problems in computer vision. It has attracted significant attention in recent years since object detection is linked to many other computer vision tasks such as image captioning [22] and image segmentation [23]. Object detection helps us understand and analyze images by trying to answer a question: What objects are where [2]? It combines two subtasks: image classification and object localization. It tries to determine which class each object in an image belongs to and where these objects are located.

As shown in Figure 2.1, many milestone approaches have been proposed to solve object detection problem in the past two decades. Until the remarkable breakthrough of deep learning in object detection, traditional methods were built on handcrafted feature representations. Limited computing resources also led people to design complex speed-up ways.

Viola-Jones (VJ) detector, which is named with its developers Paul Viola and Michael Jones, was noticeably faster than other algorithms in its time while achieving compa-

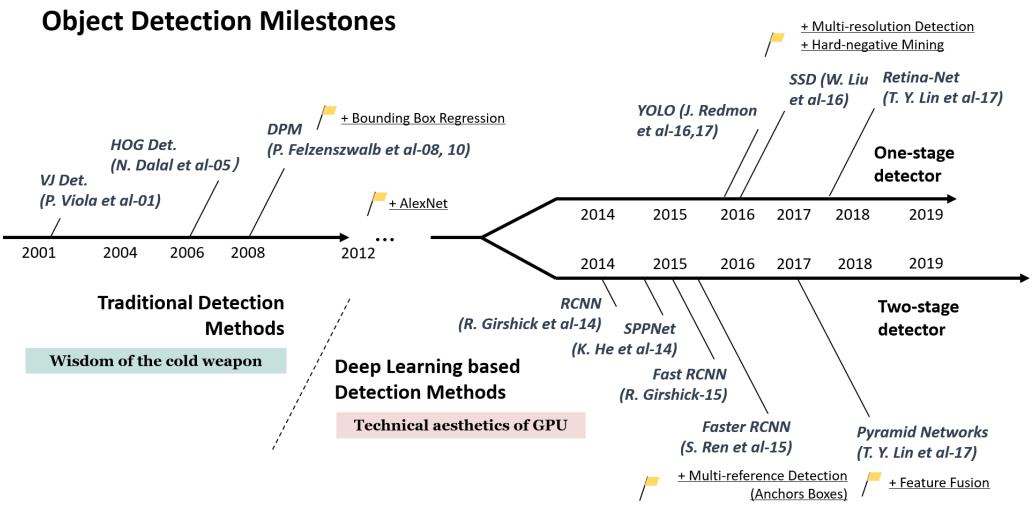


Figure 2.1: History of object detection in past two decades [2].

rable accuracy [5, 6]. Vialo-Jones detector is composed of four main parts: selecting Haar-like features, calculating an integral image, AdaBoost [24] training, and cascaded classifier. Rather than using raw intensities, this detector incorporates Haar-like features. In this way, useful information from an image is extracted. To reduce the time needed to extract Haar-like features, the authors introduce a new image representation called the integral image. The AdaBoost algorithm was used to select a critical subset of features among all available ones. In the final stage, the authors used a cascaded classifier to reduce computational overhead. The cascaded classifier quickly discards background regions and spends more time on image regions that most probably include an object.

N. Dalal and B. Triggs proposed the Histogram of Gradients (HOG) descriptor for human detection in 2005 [7]. HOG is based on calculating local histograms of magnitude and orientation of the gradients. This calculation is done in a localized manner. The image is divided into small cells, and a 1-D histogram of gradients is calculated for each of these small cells. The method also utilizes overlapping local contrast normalization on regions greater than previously defined cells for invariance to changes resultant from illumination conditions. The HOG descriptor analyzes the input image at multiple scales to improve performance in detecting objects on different scales.

Deformable Part-based Model (DPM) was proposed by Pedro Felzenszwalb [8] as

an improvement over the HOG detector. DPM was the best-performing approach among the traditional object detection methods. DPM consists of a coarse template covering the whole object, higher resolution part templates, and a spatial model for each part template. Allowable displacements for a part template are defined by the spatial model. Placement of the part template relative to a detection window directly affects the deformation cost. The overall score of the detection is the score of the coarse template on the window plus the score of the part templates minus deformation cost resulting from deviation of the part template from its ideal location [9].

P. Felzenszwalb and R. Girshick have also proposed some crucial concepts, such as hard negative mining and bounding box regression which have been an inspiration for following object detection approaches. However, proposed object detectors could not make a significant impact because of the limitations caused by hand-crafted features.

In 2012, Alex Krizhevsky, Ilya Sutskever, and Geoffery Hinton introduced a deep convolutional neural network called AlexNet [10] which achieved the best results on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [25]. The success of AlexNet has affected the whole computer vision community. Before AlexNet, neural networks were utilized in different problems. For example, a convolutional neural network architecture named LeNet [26] was proposed for character recognition in 1998. However, limitations of computing resources and lack of a large dataset were two main difficulties in the usage of deep neural networks. The utilization of GPUs and the introduction of ImageNet has shown that it is possible to use deep convolutional neural networks in different computer vision applications. This fact has also affected object detection and caused it to evolve. Since then, deep learning based approaches have completely dominated the literature.

Deep learning based object detectors can be grouped into two types: two-stage object detectors and one-stage object detectors. In the following parts, two-stage object detection and one-stage object detection will be explained.

2.1.1 Two-Stage Object Detection

Two-stage object detectors use an intermediate layer to propose regions that possibly contain an object. Then, bounding boxes and class probabilities are predicted for each of these proposed regions. The most crucial drawback of the two-stage detectors is that they run far from real-time.

2.1.2 One-Stage Object Detection

Unlike two-stage, one-stage object detection does not need an intermediate region proposal layer. This enables one-stage detectors to predict all the bounding boxes in a single pass. In this way, one-stage detectors run much faster in inference and are more suitable for real-time applications. Milestone one-stage object detectors are examined in the following subsections.

2.1.2.1 YOLO

You Look Only Once (YOLO) [3] was published by Joseph Redmon et al. in 2015. It was the first deep learning based one-stage object detector. YOLO predicts bounding box coordinates and class probabilities in a single pass by treating object detection as a regression problem. YOLO divides the input image into an SxS grid. Each of the cells of this grid is responsible for detecting an object whose center falls into these cells. As shown in Figure 2.2, each cell predicts B bounding boxes, which is composed of center, width, height and confidence, and C conditional class probabilities. YOLO predictions could be encoded as a $S \times S \times (B*5 + C)$ tensor for a given input image. However, all predicted bounding boxes do not belong to the objects in the image. Thus, predictions are need to be filtered to get rid of duplicate and weak boxes. Predicted bounding boxes whose confidence score is lower than the predefined threshold could be ignored. After this elimination, there could still be some high-confidence bounding boxes most probably belonging to the same object. To get rid of redundant bounding boxes, YOLO uses non-maximum suppression. Non-maximum suppression selects the bounding box with the highest confidence and suppresses all the remaining ones that have an overlapping ratio greater than the defined threshold.

Non-maximum suppression is done for each class, i.e., overlapping boxes that define different classes are not eliminated, and final detections are obtained.

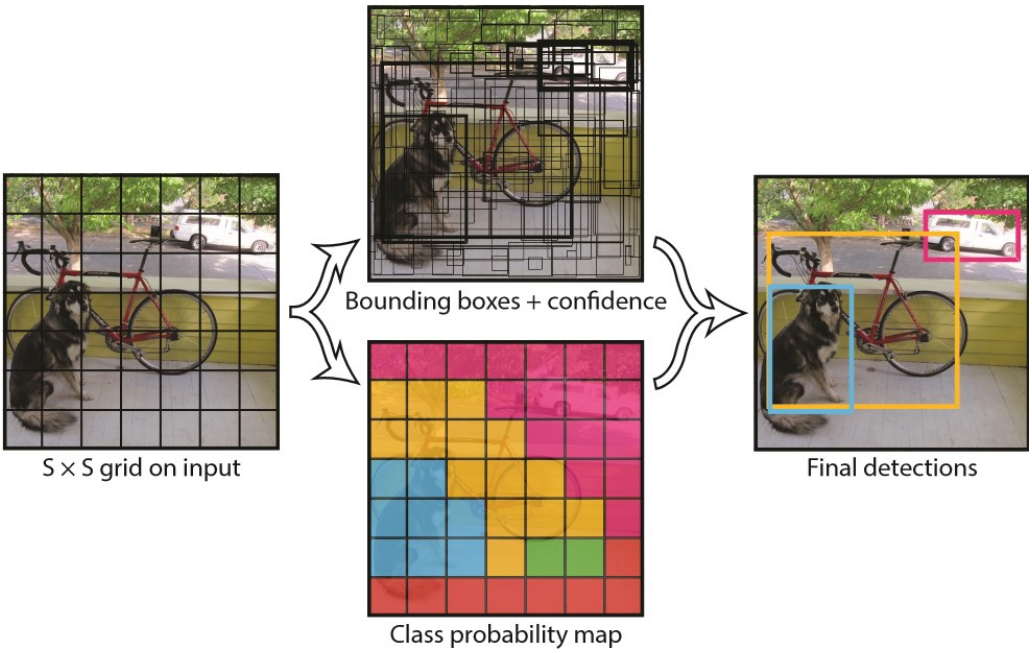


Figure 2.2: YOLO model [3].

There are some limitations that the first YOLO model faced. The localization error of YOLO is more when compared to the state-of-the-art of its time. Also, the YOLO model has the ability to predict only one class for each grid cell. Finally, YOLO struggles with small-sized objects. These limitations were tried to be solved in later versions of YOLO.

2.1.2.2 SSD

Shortly after the YOLO, SSD [11] was published by Wei Liu et al. as a second deep learning based one-stage detector. SSD was superior to YOLO in terms of both detection accuracy and speed. While YOLO achieves 63.4% mAP at 45 FPS on the VOC2007 test, SSD achieves 74.3% mAP at 59 FPS at Nvidia Titan X. SSD achieves this performance by producing predictions at different scales. YOLO performs detection from only one feature map, whereas SSD uses multiple feature maps from different layers. This strategy allows SSD to predict objects over a wider variety of scales compared to YOLO. Because of the usage of more feature maps, SSD outputs

a vast number of predictions. To reduce the effect of this imbalance on training side, SSD uses hard negative mining approach. Instead of using all the negative boxes, i.e., boxes that do not include an object, SSD picks the top ones using confidence loss and holds the ratio of at most 3:1 between the negative and positive boxes. SSD influenced following one-stage object detection methods to use different multi-scale approaches.

2.1.2.3 YOLOv3

In 2018, Joseph Redmon and Ali Farhadi announced a new version of YOLO [1]. They made some changes in design to improve accuracy and named this new model as YOLOv3. Being inspired by feature pyramid networks [27], YOLOv3 produces bounding box predictions at three different scales. YOLOv3 upsamples the feature map by 2x and concatenates upsampled features with a feature map from earlier layers in the network. In this way, YOLOv3 is more capable of getting meaningful information from upsampled high-level features and fine-grained low-level features. With this new multiscale strategy, YOLOv3 also performs much better on small-sized objects when compared with its prior YOLOv2 [28]. YOLOv3 uses a new feature extractor named DarkNet-53. This new backbone is larger than the network used in YOLOv2, Darknet-19. Darknet-53 uses 3x3 and 1x1 convolutional layers and ResNet-like [29] shortcut connections. As given in Table 2.1, Darknet-53 is more powerful than its prior and more efficient when compared with ResNet variants. Darknet-53 runs faster with less number of floating-point operations while still competing with ResNet-152. Furthermore, Darknet-53 better utilizes the GPU by achieving the highest floating-point operations per second.

Table 2.1: Comparision of backbones on ImageNet [1]

Backbone	**Top-1**	**Top-5**	**Bn Ops**	**BFLOPS/s**	**FPS**
Darknet-19	74.1	91.8	7.29	1246	**171**
ResNet-101	77.1	93.7	19.7	1039	53
ResNet-152	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

YOLOv3 uses multilabel classification to predict bounding box classes instead of softmax. Softmax requires the assumption that each bounding box contains exactly one class. However, this could not be the case for more complex datasets. One major drawback of YOLOv3 is that it can not perfectly align bounding boxes with the objects. Thus, a performance drop occurs as the intersection over union (IOU) threshold increases.

2.1.2.4 EfficientDet

EfficientDet was published by Mingxing Tan et al. in 2020. In the EfficientDet paper, the authors try to solve two main problems: multi-scale feature fusion and model scaling. In order to interpret different level features, many methods were proposed before EfficientDet. FPN [27] fuses multi-scale features in a top-down manner. PANet [30] improves FPN by adding an extra bottom-up pathway. NAS-FPN [31] tries to find better feature network topology by using neural architecture search. However, it is not very easy to interpret the found network. EfficientDet introduces a bi-directional feature pyramid network (BiFPN). BiFPN fuses feature maps coming from different layers by using learnable weights in a more principled and efficient way. Instead of directly fusing multi-scale features, BiFPN uses weights to learn the importance of features coming from different layers. The other problem that EfficientDet tries to solve is model-scaling. In order to improve accuracy, previous methods scale up the backbone network [32] or use larger input images. However, these approaches try to improve performance by focusing on limited scaling dimensions. EfficientDet proposes a compound scaling strategy that jointly scales up the width, depth, and resolution for backbone, feature-fusion, and prediction networks. Thanks to these improvements, EfficientDet achieved state-of-the-art in its time using fewer resources compared with other approaches.

2.1.2.5 YOLOv4

YOLOv4 [13] was developed by a different group from its original authors. The authors analyzed different features that could improve one-stage object detection.

State-of-the-art Bag-of-Freebies and Bag-of-Specials are examined in detail. Methods that change only training of the model without affecting inference cost are named as bag of freebies. The most common bag of freebies method is data augmentation. YOLOv4 uses CutMix [33] and Mosaic data augmentation. DropBLock is chosen as a regularization technique. CIoU [34] is also chosen as a loss function for better convergence speed and accuracy. Bag of specials can be named as plugin modules and post-processing methods that improve accuracy with little cost on the inference side. As one of the bag of specials method, Mish [35] is chosen as an activation function. YOLOv4 also utilizes Cross-stage partial connections (CSP) and Multi-input weighted residual connections (MiWRC). SPP [36] block is added over the backbone and modified version of PAN path-aggregation block [30] is used for future integration. To summarize, the authors analyzed the effect of a large number of the bag of specials and bag of freebies methods on accuracy and proposed a new object detection model named YOLOv4, which uses CSPDarknet53 as a backbone, SPP and PANet on the neck side, and anchor-based YOLOv3 detection head.

2.1.2.6 YOLOR

In 2021, YOLOR [21] was proposed by Chien-Yao Wang et al. as a new method for one-stage object detection. YOLOR is a unified network that integrates implicit and explicit knowledge together. Explicit knowledge is defined as the features obtained from shallow layers of the network. The authors also defined it as observation. On the other hand, implicit knowledge is defined as knowledge that does not correspond to the observation but is learned subconsciously in the human brain. Different ways are examined to model implicit knowledge, such as vector, neural network, and matrix factorization. Then, the authors applied implicit knowledge to three different aspects: feature alignment for feature pyramid network, prediction refinement, and multi-task learning. In the experiments, they verified the positive contribution of implicit learning on accuracy. Furthermore, YOLOR achieves comparable results with state-of-the-art methods with faster inference speed.

2.1.2.7 YOLOX

YOLOX [37] was announced by Zheng Ge et al., which brings a series of improvements to the YOLO detector. They choose YOLOv3 with the Darknet53 backbone as a baseline by considering YOLOv4 and YOLOv5 over-optimized. As a first improvement, YOLOX replaces the detection head with decoupled one to independently handle classification and regression tasks. This strategy solves the conflict between these two tasks [38, 39]. In this way, YOLOX improves precision and convergence speed. The second improvement YOLOX introduces is that an anchor-free design is utilized. The authors aim to eliminate some problems by using an anchor-free mechanism. Anchor-based detection requires the selection of an optimal set of anchors before training for each application domain. By using the anchor-free mechanism, YOLOX eliminates this step. Moreover, a potential bottleneck is eliminated in some systems since the anchor mechanism requires moving a large number of predictions between devices. Anchor-free mechanism simplifies the overall design and reduces the number of parameters. The third improvement authors introduce is that YOLOX uses multi positives instead of only one, which provides usage of high-quality predictions. Authors also state that while strong augmentation such as mosaic and mix-up is helpful for training large models, the same is not true for small models.

2.2 Reasoning Approaches in Object Detection

Two-stage and one-stage milestone object detection methods try to classify and localize objects separately. However, this is not the way how human visual system works. Humans have an ability to acquire knowledge from the visual world, and reason with the help of this knowledge to carry out visual tasks [40]. Some important methods have been published in the literature to mimic human reasoning ability. In the following subsections, these methods are examined.

2.2.1 Spatial Memory for Context Reasoning in Object Detection

In 2017, Spatial Memory Network (SMN) [16] was published by Xinlei Chen and Abhinav Gupta. SMN models spatial and semantic relationships between objects using a spatial reasoning approach. SMN extracts object instances and stores them in memory, i.e., a pseudo "image" representation. Then, this spatial memory is fed to another convolutional network for context reasoning. Spatial Memory Network uses Faster-RCNN [41] as a backbone. Features extracted from the conv5_3 layer of VGG16 [42] and fc8 Softmax scores are used as input features. Then the corresponding cells in the memory are updated by using a Gated Recurrent Unit (GRU) [43]. To extract spatial patterns from memory, SMN uses a 5-layer convolutional neural network composed of filters with a spatial size of 3x3 and a channel size of 256. Memory scores are combined with Faster-RCNN scores for region classification at the final stage.

2.2.2 Relation Networks for Object Detection

The method named Relation Networks for Object Detection [17] was inspired by the success of attention modules in natural language processing [20] and proposed in 2017. This method uses Faster-RCNN [41] as a backbone. It was applied to two different parts of the backbone. First, the relation module was applied after the fully connected layer in order to improve instance recognition performance. Then, the relation module was applied to remove duplicate bounding boxes. It eliminates the need for NMS.

2.2.3 Iterative Visual Reasoning Beyond Convolutions

Xinlie Chen et al. proposed the work named Iterative Visual Reasoning Beyond Convolutions [18] in 2018. Their framework is composed of two modules. The first module incorporates spatial reasoning by using the previously mentioned Spatial Memory [16]. The second module incorporates semantic reasoning by using a knowledge graph.

2.2.4 End-to-End Object Detection with Transformers

In 2020, a group of people from Facebook AI announced DEtection TRansformer (DETR) [19]. DETR treats object detection as a direct set prediction problem. DETR incorporates a CNN backbone to extract feature embeddings. Using transformer architecture [20] over these feature embeddings, DETR uses the whole image and learns global relationships between objects. DETR also uses a different loss function to remove hand-designed components like a non-maximum suppression or anchor generation, which directly affects the final performance of the systems. A set of N predictions are computed in a single pass. N is chosen in a way that it is larger than the possible number of objects in an image. By the usage of bipartite matching loss, DETR uniquely assigns predictions to ground truths which eliminates the duplicate removal stage.

2.2.5 Swin Transformer

Ze Liu et al. presented the work named Swin Transformer [44] in 2021. Swin Transformer is a general-purpose vision Transformer backbone that can be adapted to many applications of computer vision such as image classification and object detection. Swin Transformer serves some improvements over previous vision transformer [45]. Swin Transformer achieves linear complexity with increasing image size while the complexity of the original Transformer increases quadratically. Self-attention is computed in local and non-overlapping windows, which contain a fixed number of patches. Since the window-based approach limits the modeling power of self-attention, the authors suggest shifted window partitioning in consecutive blocks. This **S**hifted **win**dowing strategy provides connections between non-overlapping windows which significantly enhances the modeling capability of the Swin Transformer.

2.2.6 ViT-YOLO

Zixiao Zhang et al. proposed the work named ViT-YOLO: Transformer-Based YOLO for Object Detection [46] in ICCV Workshop in 2021. They introduced some improvements over the baseline YOLOv4-P7 to outperform the existing state-of-the-art

detectors on the VisDrone2019 test-challenge dataset [47]. Firstly, they integrated a multi-head attention block into the original CSP-Darknet backbone. They aimed to improve performance in challenging conditions of drone-captured images such as scale variance, viewpoint variation, and complicated background with the help of the transformer's capability of learning relationships across image feature patches. They replaced n CSP Bottlenecks of CSPDark Block in the P7 stack with n Multi-Head Self-attention layers. Secondly, they used a weighted bi-directional feature pyramid network (BiFPN) to aggregate multi-scale features. Unlike PANet, which fuses multi-scale features with a simple summation, BiFPN fuses multi-scale features in a smarter way by using learnable weights. Thirdly, they utilized Test-Time Augmentation (TTA) and Weighted Boxes Fusion (WBF) strategies to improve performance further.

CHAPTER 3

THE PROPOSED METHOD

In this chapter, the proposed method is described in detail. First, the general structure of the proposed method is given. Later, the baseline networks are explained. Then, the reasoning layer is described with its all sub-layers. The role of the multi-head attention layer to model visual reasoning between image regions is given clearly. Later, different reasoner network configurations that we attempted are presented. The differences between these configurations are explained in detail.

3.1 The General Structure of the Proposed Method

The general structure of the proposed method is shown in Figure 3.1. Firstly, convolutional features are extracted by the backbone, and low-level features are integrated into the high-level features by the neck. Then, these features are enhanced by the reasoning layer. At the final stage, the head predicts class probabilities and bounding boxes.

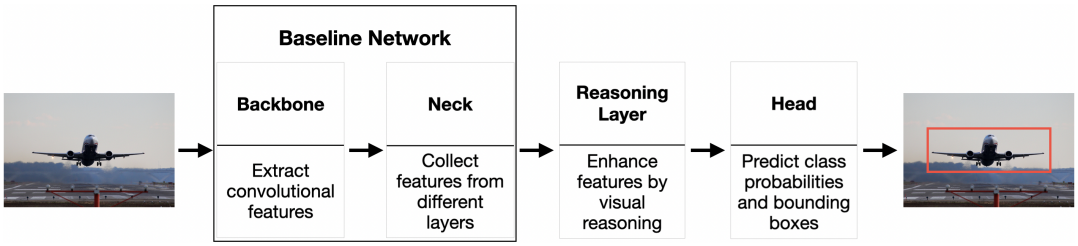


Figure 3.1: The general structure of the proposed method.

3.2 Baseline Networks

We use three different real-time one-stage detectors as a baseline: YOLOv3, YOLOv4-P6, and YOLOR-P6. All of them are explained in detail in the following subsections.

3.2.1 YOLOv3

The first baseline network is YOLOv3 [1]. YOLOv3 uses Darknet-53 as a backbone network. YOLOv3 architecture is preserved except last few layers. Like YOLOv3, the proposed method produces bounding box predictions at three different scales. Feature pyramid network [27] is used as a neck (terminology from [13]). This FPN is responsible for collecting feature maps from different layers of the backbone. These feature maps are concatenated after the necessary up-sampling operations. Then, the more meaningful, fine-grained, and enhanced feature maps are fed to the detection head. The detection head of the YOLOv3 is used as it is. Three different bounding boxes are predicted at each scale. Prediction of each grid region is in the size of [3x(1+4+80)] where 1 for objectness score, 4 for bounding box offsets, and 80 for class probabilities. Thus, at each scale, the network predicts a tensor in the size of NxNx[3x(1+4+80)]. For an input with a size of 416x416x3, the outputs of the different scales will be 13x13x255, 26x26x255, and 52x52x255 sized tensors, which are responsible for the detection of large, medium, and small objects, respectively.

3.2.2 YOLOv4-P6

The second baseline network is YOLOv4-P6 which is a scaled version of the fully CSP-ized YOLOv4-P5 [14]. The CSP-ization reduces the number of parameters and computation while preserving or improving the accuracy [48]. Authors re-design the YOLOv4 to improve the speed/accuracy trade-off and propose a compound scaling strategy for different sized networks in the Scaled-YOLOv4 paper. In our work, the YOLOv4-P6 model is preserved except last few layers. 1280x1280 is used as an input resolution. CSP-P6 backbone is used for feature extraction composed of CSPDark layers with depth scales [1, 3, 15, 15, 7, 7], respectively. CSP-ized PAN architecture,

which is composed of up and down layers, is used as a neck to integrate different level features. Bounding boxes and class probabilities are predicted at four different scales. Predictions are inferred over feature maps whose sizes are scaled-down of the input image with strides {8, 16, 32, 64} respectively.

3.2.3 YOLOR-P6

The third baseline network is YOLOR-P6 [21]. YOLOR-P6 follows a similar architecture to YOLOv4-P6-light. YOLOR uses YOLOv4-P6-light as a baseline which is a light version of the YOLOv4-P6 model [14]. It uses a re-organization layer as a down-sampling module in the first layer of the backbone. Base channels and repeat number of the backbone layers are set to {128, 256, 384, 512, 640} and {3, 7, 7, 3, 3} respectively. All Mish activations are also replaced with SiLU activation in YOLOR-P6. Like YOLOv4-P6-light, 1280x1280 is used as an input resolution. Bounding boxes and class probabilities are predicted at four different scales.

3.3 Reasoning Layer

As the reasoning layer, a transformer encoder-like [20] model is used. The architecture of the reasoning layer is shown in Figure 3.2. It is composed of Flatten, Multi-head Attention, Norm, MLP, and Rearrange layers. Furthermore, two residual skip connections and positional encoding are also utilized. All of these sub-layers of the reasoning layer are explained in detail in the following parts.

3.3.1 Flatten

The multi-head attention layer expects a sequence as an input. To handle this, Flatten layer takes a three-dimensional (HxWxC) feature map (batch dimension is ignored) and converts it to a two-dimensional (HWxC) form. In this way, grid regions are converted to a sequence and fed to a multi-head attention layer. Flattening operation is visualized in Figure 3.3.

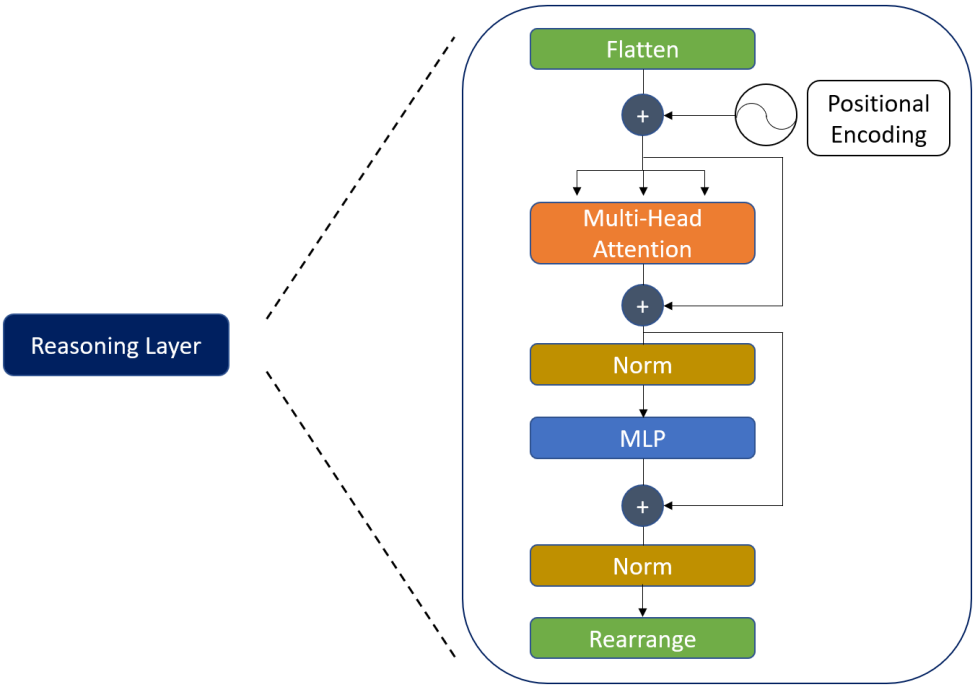


Figure 3.2: Reasoning Layer.

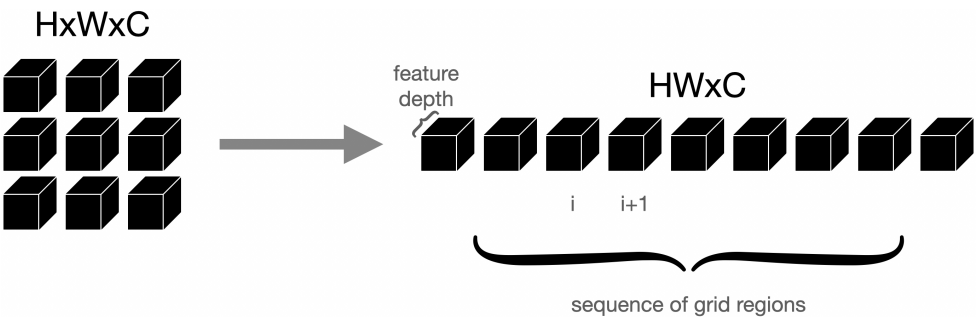


Figure 3.3: Visualization of flattening.

3.3.2 Positional Encoding

The multi-head attention layer is unaware of order in the input sequence by its nature. However, information about the positions of grid regions in the whole image is valuable and should be taken into account. To model the order of image regions, positional encoding is used. There are different ways of positional encoding, fixed or learned. Some researchers tested both fixed and learned positional encoding, and concluded that the results of both of the methods were nearly the same [49].

In this thesis, sinusoidal positional encoding [50] is used:

$$PE_{(i,\ 2j)} = \sin(\frac{i}{10000^{2j/d_{feature}}}) \tag{3.1}$$

$$PE_{(i,\ 2j+1)} = \cos(\frac{i}{10000^{2j/d_{feature}}}) \tag{3.2}$$

where *i* is the position of the grid region in the sequence, *j* is the feature depth index, and $d_{feature}$ is the same with the feature depth. At first, generated values by sine and cosine functions described above were directly concatenated and used. For a sample image region with position *i* in the sequence, the positional encoding vector is calculated as:

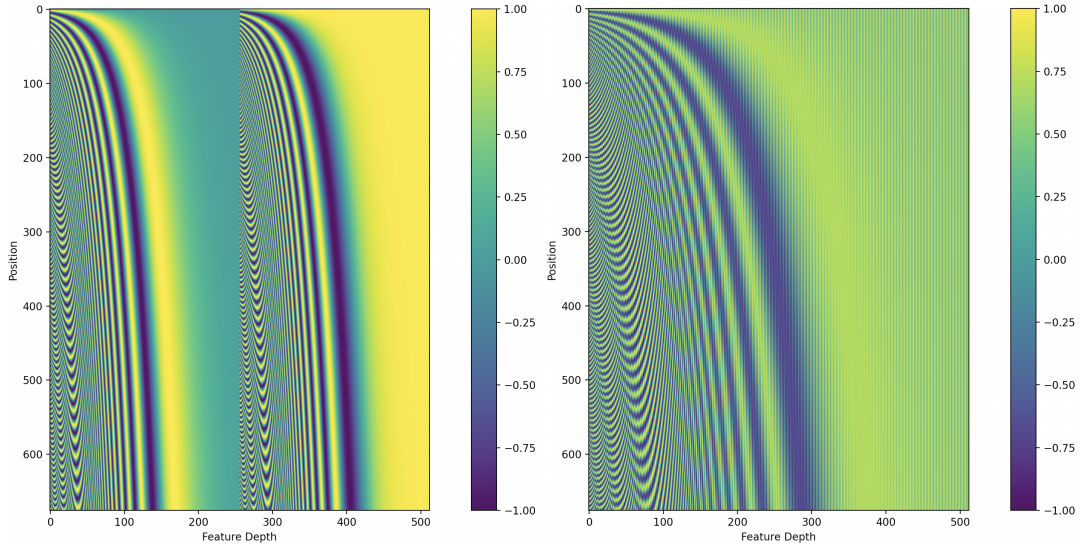
$$\vec{PE}_i = [\sin(w_1), \sin(w_2), ..., \sin(w_{d_{feature}/2}), \cos(w_1), \cos(w_1), ..., \cos(w_{d_{feature}/2})] \tag{3.3}$$

For 676 (26x26, scale2 output of the YOLOv3 baseline) grid region positions and 512 feature depth, it is shown in Figure 3.4a. Then, it was changed a little. Instead of directly concatenating, sine and cosine functions are concatenated pairwise:

$$\vec{PE}_i = [\sin(w_1), \cos(w_1), \sin(w_2), \cos(w_2)..., \sin(w_{d_{feature}/2}), \cos(w_{d_{feature}/2})] \tag{3.4}$$

For 676 (26x26, scale2 output of the YOLOv3 baseline) grid positions and 512 feature depth, it is shown in Figure 3.4b.

Calculated unique information about the position of a grid region (i.e., positional encoding) is added to the convolutional feature embedding of this grid region. In this way, the multi-head attention layer could deduce a distance between different grid (i.e., image) regions from added positional encodings.



(a) Direct concatenation.

(b) Pairwise concatenation.

Figure 3.4: Positional Encoding with different concatenations

3.3.3 Multi-Head Attention

Multi-head attention layer is the main layer where reasoning between grid cells which represent image regions takes place. Reasoning between different cells of the input sequence is modeled by using self-attention, which is based on three main concepts: query, key, and value. In high-level abstraction, the query of a single grid cell in the sequence searches potential relationships and tries to associate this cell with other cells, i.e., image regions, in the sequence through keys. The comparison between query and key pairs gives us the attention weight for the value. Interaction between attention weights and values determines how much focus to place other grid cells, i.e., image regions, of the input sequence while representing the current cell. While the "Attention is all you need" [20] paper used self-attention for linguistic reasoning, it is utilized for visual reasoning in this thesis.

In self-attention, the query, key, and value matrices are calculated first. These matrices are calculated by multiplying the input sequence X with three different weight matrices: W^Q, W^K, and W^V:

$$Q = XW^Q \tag{3.5}$$

$$K = XW^K \tag{3.6}$$

$$V = XW^V \tag{3.7}$$

This multiplication for a sample input sequence in size 9x16 (number of grid cells, feature depth) and matrices in size 16x16 (feature depth, embedding size of the attention model) is shown in Figure 3.5, Figure 3.6 and Figure 3.7. In X, each row represents one grid cell, i.e. image region. The number of rows in weight matrices should be equal to feature depth in X. The output matrix size is in the form of a number of grid cells times the embedding size of the attention model, i.e., 9x16 for this sample.

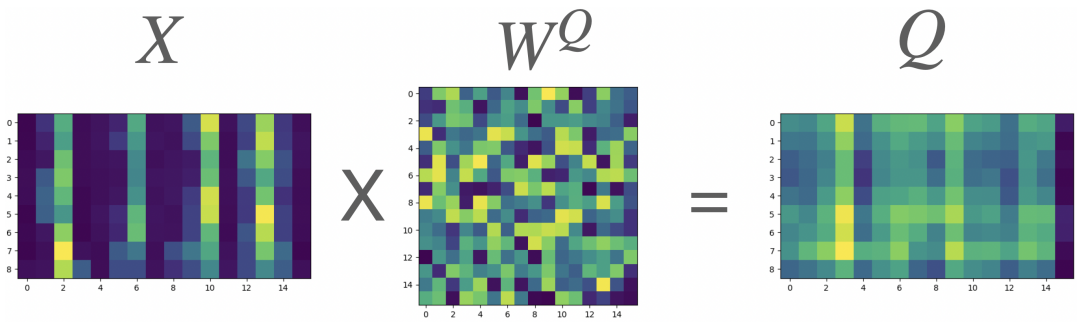


Figure 3.5: Calculation of query.

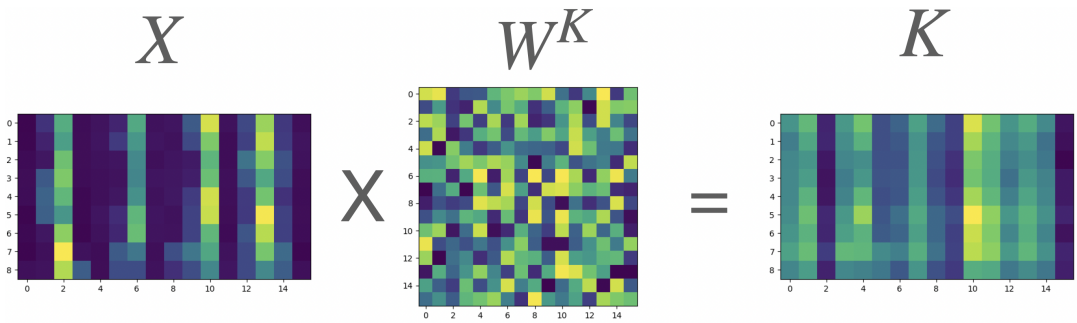


Figure 3.6: Calculation of key.

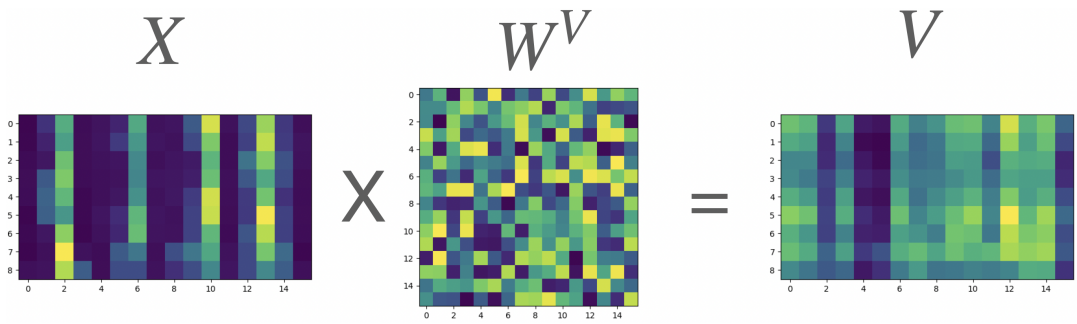


Figure 3.7: Calculation of value.

To compare query and key matrices, the scaled dot-product attention is used [20]:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.8}$$

where the relation between the query of a grid and keys of all the grids is calculated by the dot-product of Q and K^T. Then, the result of this dot-product is scaled by the square root of the d_k, which is the dimension of query and key matrices. When the magnitude of the dot-product of the query and key matrices grow large, this negatively affects the softmax operation. Thus, the scaling factor $\sqrt{d_k}$ is used to handle this problem. After the scaling operation, attention weights are normalized by using the softmax function. At the final stage, each grid cell (i.e., image region) is encoded by taking a summation of weighted value matrix columns, which are weighted by normalized attention weights. The attention weights tell where to look in the value matrix. In other words, they tell which parts of the image are valuable, informative, and relevant while encoding the current grid. Calculation of attention matrix for previously defined sample sequence is shown in Figure 3.8, Figure 3.9 and Figure 3.10.

The self-attention mechanism was further improved in a multi-headed manner [20]. In multi-head attention, self-attention computation is performed for a defined number of heads in parallel. Each of these heads is independent of the others. The major superiority of multi-head attention over single attention is that it enables the model to work on different representation subspaces. Multi-head attention uses multiple sets of query, key, and value matrices. Each head has a different query, key, and value matrices since each of these sets are obtained by multiplying the input grid sequence

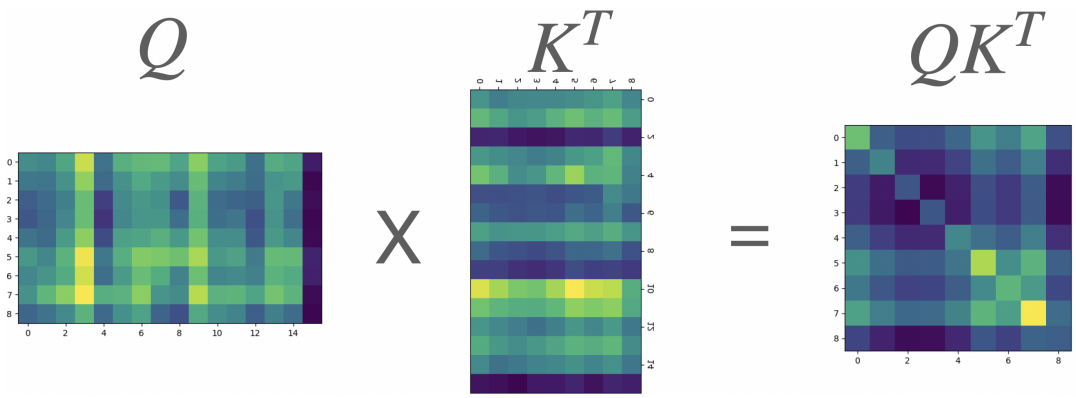


Figure 3.8: Calculation of self-attention.

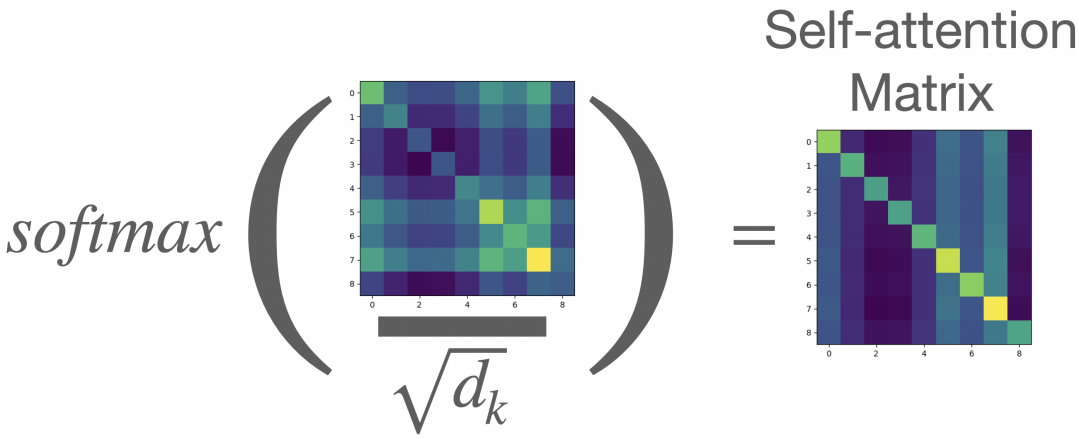


Figure 3.9: Normalization of self-attention.

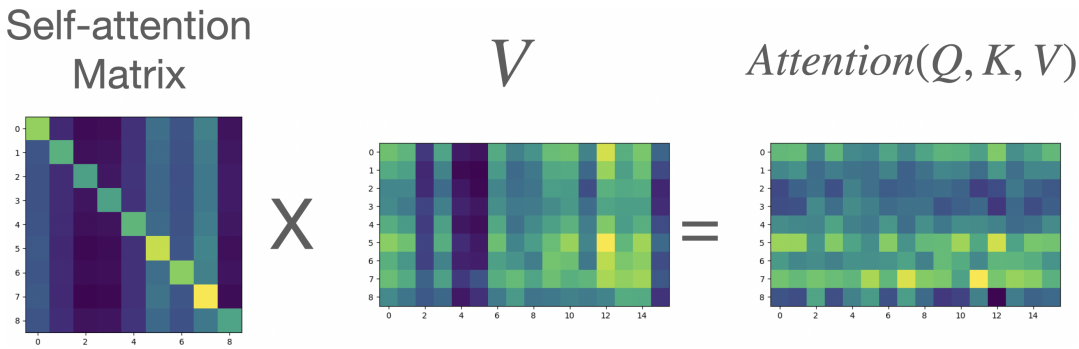


Figure 3.10: Calculation of attention.

with a separate and randomly initialized weight matrices denoted as W_i^Q, W_i^K and W_i^V.

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{3.9}$$

where $head_i$ is the attention calculated in head i. In this way, different relationships between grid cells (i.e., image regions) are learned in each head. Then, each head's attentions are concatenated and transformed using a weight matrix W^O:

$$MultiHead(Q, K, V) = Concat(head_1, head_2, ..., head_h)W^O \tag{3.10}$$

While improving the reasoning capability of the multi-head attention layer, this strategy does not increase computational complexity. The embedding size of the attention model in a single head case is now divided by the number of heads. In this way, extra complexity is compensated [51].

Accessing different grid cells (i.e., image regions) takes an important place in the reasoning ability of the multi-head attention layer. Alexey Dosovitskiy et al. have computed "attention distance" to understand how far the Vision Transformer (ViT) has accessed the entire image to integrate semantically relevant information for classification [45]. Even in the early layers, the multi-head attention layer of the ViT-L/16 model has the ability to reach the entire image nearly. The authors have made an analogy between the receptive field in convolutional neural networks and the attention distance. To attend a 120 pixels receptive field, nearly 60 convolutional layers are required with a 3x3 filter size. This clearly shows the advantage of a multi-head attention layer for global reasoning.

3.3.4 Skip Connections

There are two skip connections in the reasoning layer: around the multi-head attention layer and around the MLP layer. Around the multi-head attention layer, the input from flatten layer is added to the output of the multi-head attention layer. Around the MLP layer, the input to the Norm layer is added to the output of the MLP layer.

These skip connections provide two main advantages. The first one is that backpropagation is improved as stated in ResNet [29] paper with the help of these residual

skip connections. A smoother loss surface is obtained when residual skip connections are utilized. It can be difficult to converge the model without using residual skip connections. The second advantage of the residual skip connections is that original information is propagated to the following layers. There is a possibility that the multi-head attention layer could discard some part of the input sequence. With the help of the residual skip connection around the multi-head attention layer, the original input sequence is added back to the output of the multi-head attention layer, and information loss is prevented [52].

3.3.5 Normalization

Like residual skip connections, normalization operation is applied in two places in the reasoning layer: before the MLP layer and before Rearrange layer. Besides residual skip connection, normalization is another key factor in converging the model easily. Sergey Ioffe and Christian Szegedy stated that internal covariate shift could be a big problem in training deep neural networks. They define this phenomenon as the change of the distribution of each layer's input with the change of parameters of the previous layers [53]. They proposed batch normalization as a solution to this problem. In batch normalization, some statistics of the batch are calculated, some statistics are trained for each of the channels of the input. Moving average of the mean and variance are stored to use in inference. Gamma and beta parameters are trained to learn the required scale and shift. However, batch normalization is not suitable for sequence models. Thus, layer normalization [54] is employed in the reasoning layer as in transformer encoder [20]. Difference between batch and layer normalization is shown in Figure 3.11.

In layer normalization, the mean and variance values are calculated across channels:

$$\mu_i = \frac{1}{K}\sum_{k=1}^{K} x_{i,k} \tag{3.11}$$

$$\sigma_i^2 = \frac{1}{K}\sum_{k=1}^{K} (x_{i,k} - \mu_i)^2 \tag{3.12}$$

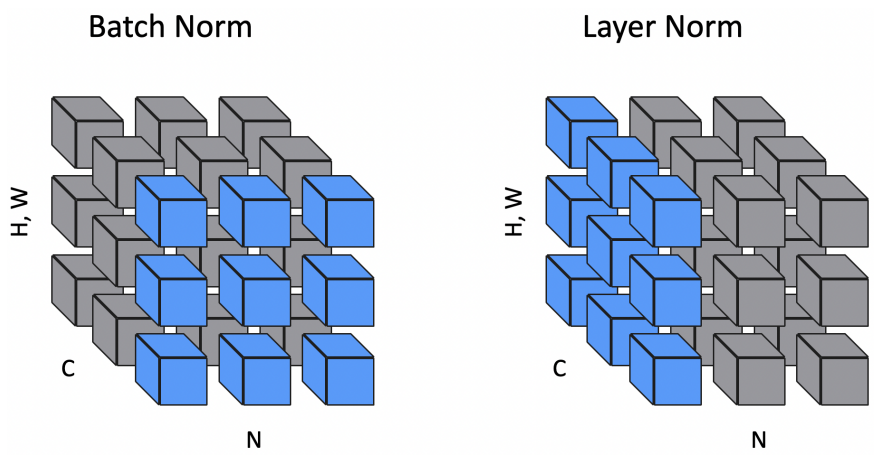


Figure 3.11: Difference between batch and layer normalization.

$$\hat{x}_{i,k} = \frac{x_{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \tag{3.13}$$

$$LN_{\gamma,\beta}(x_i) = \gamma \hat{x}_i + \beta \tag{3.14}$$

where μ_i is the mean value, σ_i^2 is the variance, $\hat{x}_{i,k}$ is normalized feature value, γ and β are the learnable parameters. The normalization layer guarantees to deal with reasonable input distributions and improves performance.

3.3.6 MLP

The output of the multi-head attention layers is fed to the multilayer perceptron (MLP) layer after normalization. MLP layer is composed of two linear layers and ReLU non-linearity in between:

$$MLP(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{3.15}$$

MLP layer creates a new representation of the reasoning information coming from the multi-head attention layer. The main difference between multi-head attention and MLP layer is that multi-head attention layer weights are input dependent while the

same MLP layer is applied to each of the inputs identically [51]. In other words, the weights of the MLP layer do not change with the input.

3.3.7 Rearrange

Previously, the three-dimensional (HxWxC) feature map was converted to a sequence in the Flatten sublayer. Rearrange is the last sublayer of the reasoning layer. After the Rearrange, the data will be fed to the YOLO layer to detect objects in the image. However, the YOLO layer expects a three-dimensional (HxWxC) input (batch dimension ignored). Thus, the two-dimensional (HWxC) sequence data is converted to a three-dimensional (HxWxC) form by rearranging in this sublayer. Rearranging operation is visualized in Figure 3.12.

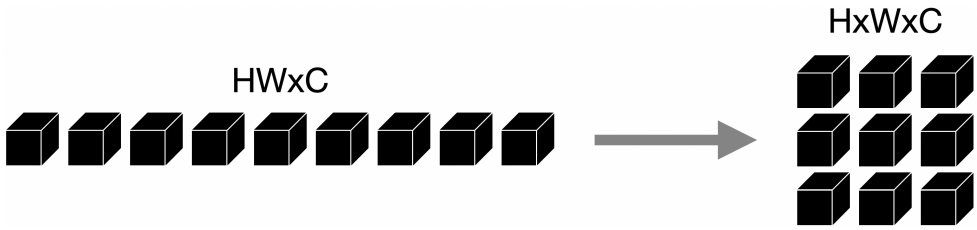


Figure 3.12: Visualization of rearranging.

3.4 Network Configurations

Six different network configurations are trained and tested on COCO [4] dataset. In the first four configurations, YOLOv3 is used as a baseline. Four different YOLOv3-Reasoner configurations are tested. Then, the most promising one is adapted to the YOLOv4-P6 and YOLOR-P6 models. Only reasoning features are fed to the detection head in the first two YOLOv3-Reasoner models. It reminds the approach used in DETR [19]. In DETR, the authors fed a set of image features to the transformer encoder and tried to predict bounding boxes on the decoder side using the output features of the encoder. In the other two YOLOv3-Reasoner configurations, neck output convolutional features are concatenated with reasoning features, then this new feature set is fused in a 1x1 convolutional layer and fed to the detection head. By this concatenation strategy, image features given to the detection head are further improved.

YOLOv4-P6-Reasoner and YOLOR-P6-Reasoner models follow the same idea as YOLOv3-Reasoner4. Details related to all of these reasoner network configurations are given in the following subsections.

3.4.1 YOLOv3-Reasoner

Four different configurations of the YOLOv3-Reasoner are examined: YOLOv3-Reasoner1, YOLOv3-Reasoner2, YOLOv3-Reasoner3, and YOLOv3-Reasoner4.

3.4.1.1 YOLOv3-Reasoner1

In this architecture, Darknet-53 output is fed to a 1x1 convolutional layer for dimension reduction of the feature depth. Then, the output of the 1x1 convolutional layer is fed to the reasoning layer on three different scales. Since the feature depth is the same for these three different scales, the same embedding size and number of heads are used in all of the scales. Embedding size is required to be divisible by the number of heads, so the number of heads is chosen to be 5 for all of the scales. For all of the scales, one reasoning layer is used, and the expansion rate for the MLP layer is chosen one.

Not using a high capacity reasoner is meaningful here since it is not needed to use the reasoning layer to extract features from a raw image, unlike ViT [45]. Darknet-53 has already extracted convolutional features from the image. The purpose of the reasoning layer in this configuration is to model visual reasoning between image regions and enhance features semantically. In this YOLOv3-Reasoner1 configuration, the reasoning layer does not include a positional encoding layer. The output of the reasoning layer is fed to the YOLO layer to predict bounding boxes and class probabilities for each grid region. The dimensions of the input given to the YOLO layer do not change since the reasoning layer keeps the dimensions of its output the same as its input. The whole architecture of YOLOv3-Reasoner1 is shown in Figure 3.13.

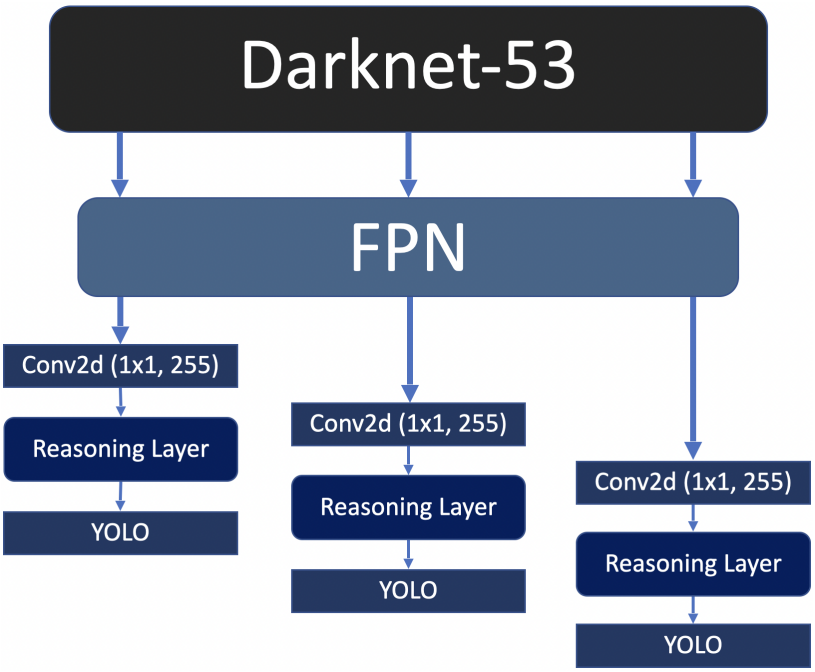


Figure 3.13: The architecture of YOLOv3-Reasoner1.

3.4.1.2 YOLOv3-Reasoner2

Unlike YOLOv3-Reasoner1, Darknet-53 output is directly fed to the reasoning layer in this configuration. Since input feature depth is different for different scales, embedding size and number of heads are also chosen differently for different scales.

In the first scale, where mostly large objects are detected, the number of heads is chosen 16. Thus, the embedding size for each head has become 64. In the second scale, where mostly medium objects are detected, the number of heads is chosen 8. Again, the embedding size for each head has become 64 on this scale. For the last scale, where mostly small objects are detected, the number of heads is chosen 4, and the embedding size for each head has become 64. Number of heads are chosen in a way that previous works in transformers are taken into consideration [20], [45], [19]. Like the YOLOv3-Reasoner1 configuration, one reasoning layer is used in each scale, and the expansion rate is set to one for MLP layers. In this YOLOv3-Reasoner2 configuration, positional encoding is utilized in the reasoning layer. The reasoning layer output is fed to the 1x1 convolutional layer. Then, fused features are fed to the YOLO layer to predict bounding boxes and class probabilities for each grid region.

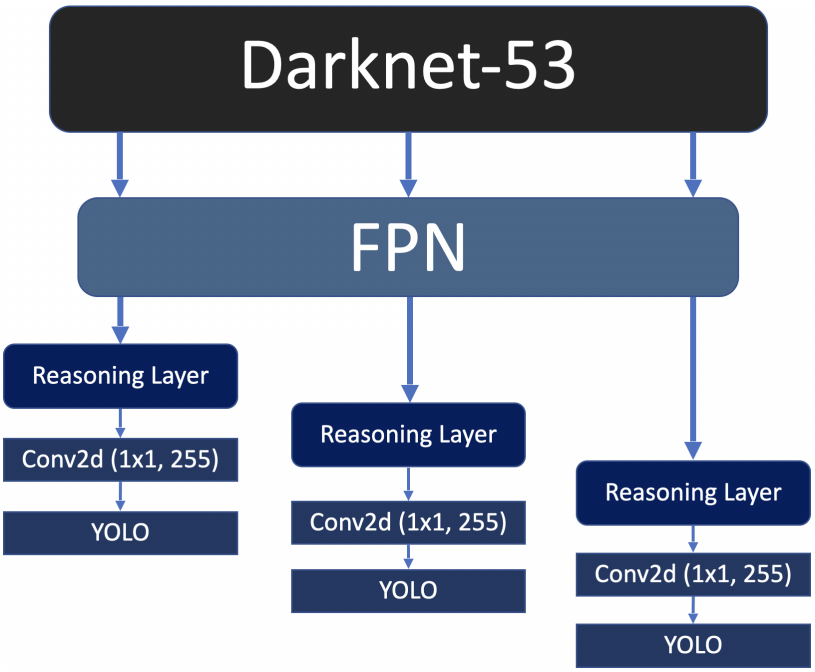


Figure 3.14: The architecture of YOLOv3-Reasoner2.

The whole architecture of YOLOv3-Reasoner2 is shown in Figure 3.14.

3.4.1.3 YOLOv3-Reasoner3

In this configuration, the previous architecture is a little bit changed. Still, Darknet-53 output is directly fed to the reasoning layer instead of the 1x1 convolutional layer. The output of the reasoning layer is concatenated with Darknet-53 output through a shortcut connection. Then, the output of the concatenation layer is fed to the 1x1 convolutional layer to fuse the information coming from feature maps composed of reasoning features between image regions and original only-convolutional Darknet-53 output features.

The concatenation layer has a very crucial function here. It ensures the reusability of the original only-convolutional Darknet-53 output features. There is a possibility that some of the features have been weakened in the reasoning layer. With the help of the DenseNet [55] like strategy, all of the valuable Darknet-53 output features are utilized for object detection. At the last stage, fused features are fed to the YOLO layer to predict each grid region's bounding boxes and class probabilities. Positional

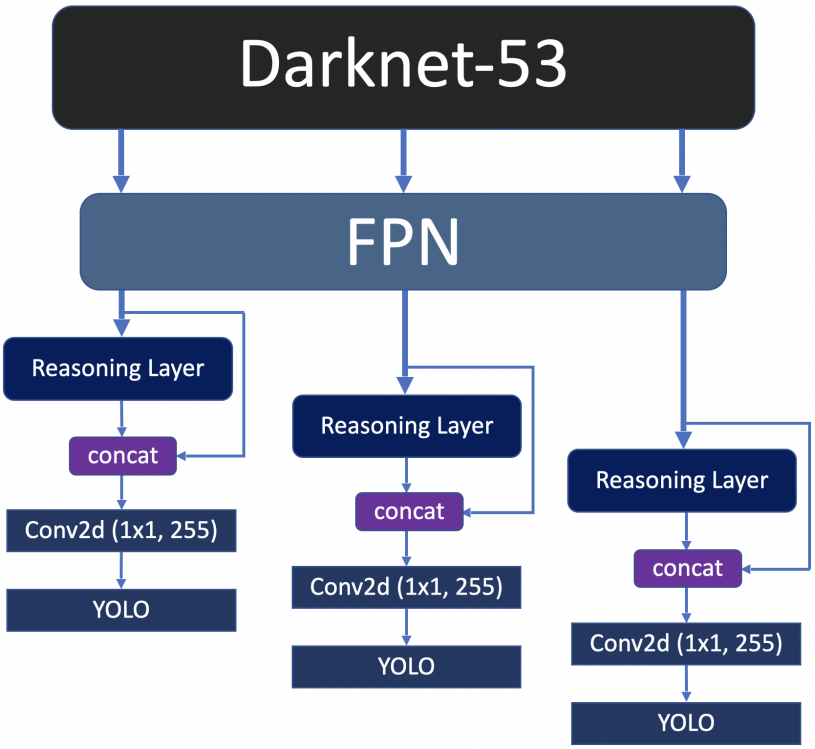


Figure 3.15: The architecture of YOLOv3-Reasoner3.

encoding is not utilized in this configuration. Like YOLOv3-Reasoner2, the number of heads is chosen as 16, 8, and 4, respectively, for each of the scales. The embedding size for each head is set to 64. Like the previous configurations, one reasoning layer is used in each scale, and the expansion rate is chosen one for MLP layers. The architecture of YOLOv3-Reasoner3 is shown in Figure 3.15.

3.4.1.4 YOLOv3-Reasoner4

The configuration of the YOLOv3-Reasoner4 is nearly identical to the previous one, as it is shown in Figure 3.16. Like YOLOv3-Reasoner3, the output of the reasoning layer is concatenated with Darknet-53 output through a shortcut connection. Then, the output of the concatenation layer is fed to the 1x1 convolutional layer to fuse the information. At the last stage, fused features are fed to the YOLO layer to predict bounding boxes and class probabilities for each grid region. The number of heads is chosen as 16, 8, and 4, respectively, for each of the scales. The embedding size for each head is set to 64. Like the previous configuration, one reasoning layer is used in

each scale, and the expansion rate is chosen one for MLP layers. The only change is that positional encoding is utilized in this configuration.

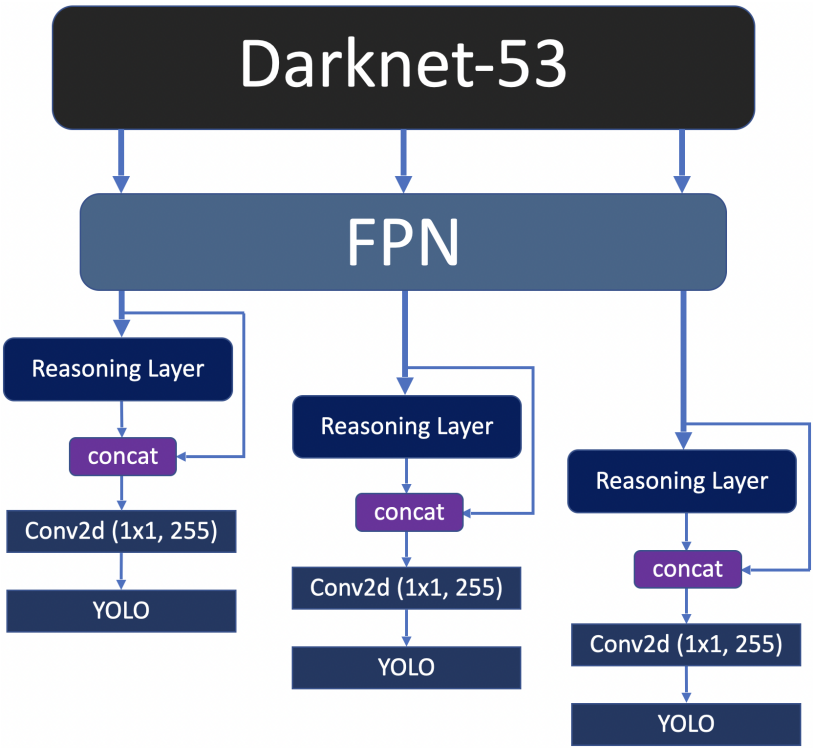


Figure 3.16: The architecture of YOLOv3-Reasoner4.

3.4.2 YOLOv4-P6-Reasoner

In this architecture, CPANSPP neck output is fed to the reasoning layer. Since input feature depth is different for different scales, embedding size and number of heads are also chosen differently. The embedding size for all of the heads is fixed to 64. The number of heads is chosen as 4, 8, 16, and 16 for each of the scales, respectively. Like the YOLOv3-Reasoner4 configuration, one reasoning layer is used in each scale, and the expansion rate is set to one for MLP layers. The output of the reasoning layer is concatenated with CPANSPP output through a shortcut connection. Then, the concatenated features are fed to 1x1 convolutional layers to fuse the reasoning features with neck outputs. In the implementation of YOLOv4-P6, a 1x1 convolutional layer with 255 filters is embedded in the YOLO head. Thus, an additional 1x1 convolutional layer is used to fuse reasoning and convolutional features with a compatible

number of filters at each scale. At the last stage, feature depth reduced features are fed to the YOLO detection head to predict bounding boxes and class probabilities. The architecture of the YOLOv4-P6-Reasoner is shown in Figure 3.17.

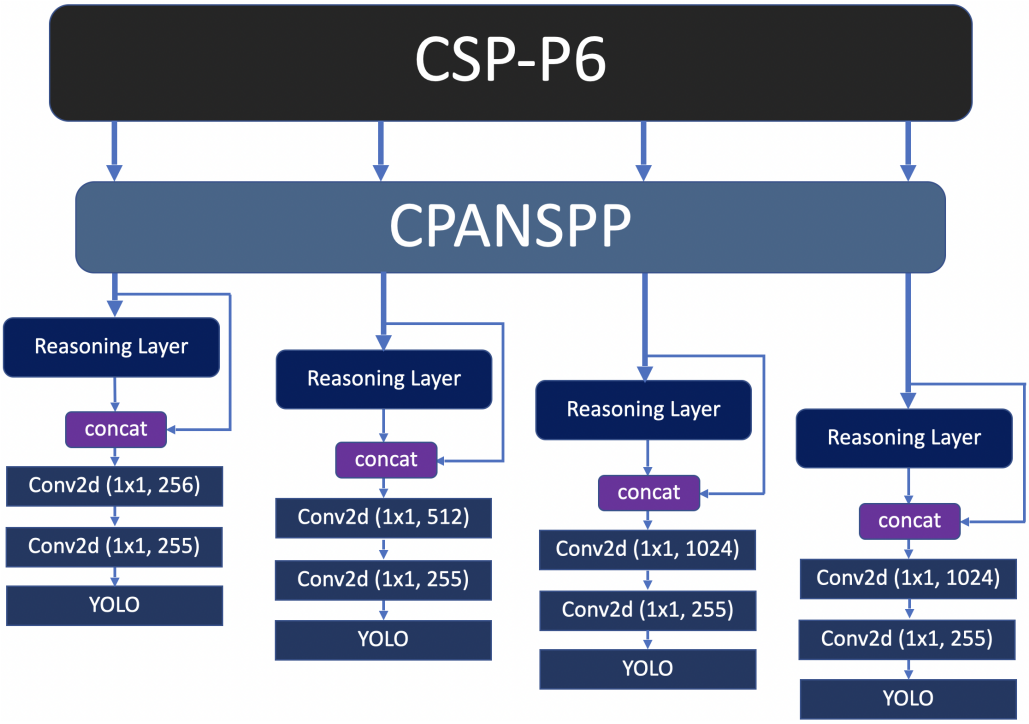


Figure 3.17: The architecture of YOLOv4-P6-Reasoner.

3.4.3 YOLOR-P6-Reasoner

The architecture of the YOLOR-P6-Reasoner is very similar to the previously explained YOLOv4-P6-Reasoner since the baseline networks YOLOv4-P6 and YOLOR-P6 are similar. Like YOLOv4-P6-Reasoner, embedding size and number of heads are chosen differently for different scales. The embedding size for all of the heads is fixed to 64. The number of heads is chosen as 4, 6, 8, and 10 for each of the scales, respectively. Since the baseline YOLOR-P6 is lighter than the YOLOv4-P6, the number of heads for each scale is reduced. Like the previous reasoner model, one reasoning layer is used in each scale, and the expansion rate is set to one for MLP layers. The output of the reasoning layer is concatenated with CPANSPP-Light-Implicit output through a shortcut connection. Then, the concatenated features are fed to 1x1 convolutional layers to fuse the reasoning features with neck outputs. At the last stage, fused fea-

tures are fed to the YOLO layer to predict bounding boxes and class probabilities for each grid region. The neck of the baseline YOLOR-P6 is different from the YOLOv4-P6. It is lighter and improved by implicit knowledge. Thus, YOLOR-P6-Reasoner uses features that are improved by the reasoning layer and implicit knowledge-driven neck. The architecture of the YOLOR-P6-Reasoner is shown in Figure 3.18.

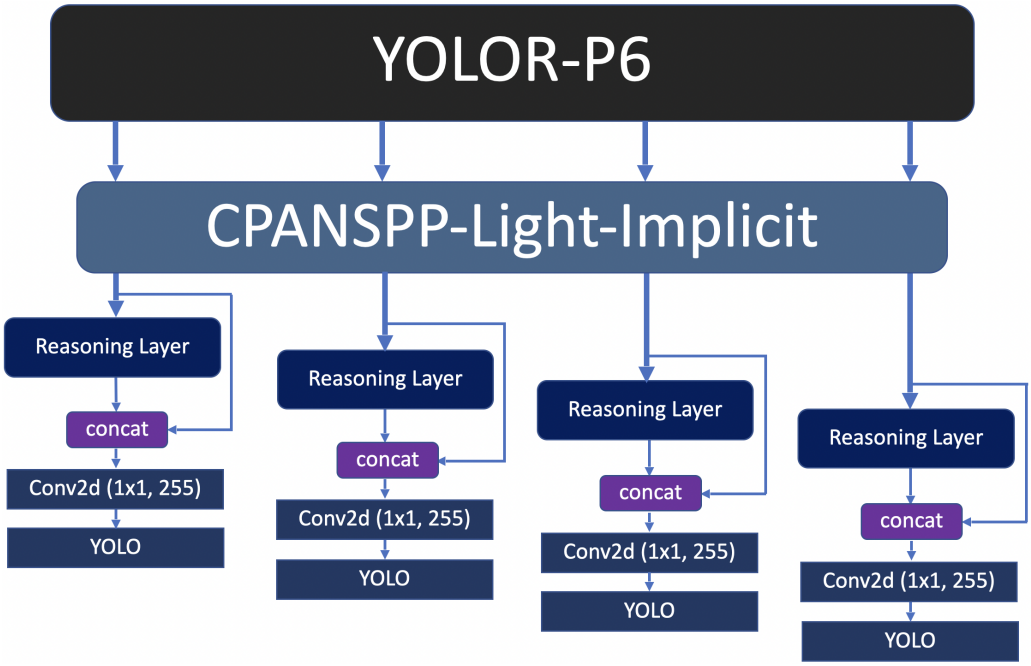


Figure 3.18: The architecture of YOLOR-P6-Reasoner.

CHAPTER 4

EXPERIMENTAL RESULTS

In this chapter, experimental results are presented. Firstly, the dataset is introduced. Implementation details and evaluation metrics are also given. Then, early evaluation results of the YOLOv3-Reasoner models are presented. Finally, YOLOv3-Reasoner2, YOLOv3-Reasoner4, YOLOv4-P6-Reasoner, and YOLOR-P6-Reasoner models are compared with state-of-the-art in the quantitative evaluation part. Some sample predictions of these networks are also given in the qualitative evaluation.

4.1 Dataset

All the experiments are performed on the MS COCO (Microsoft Common Objects in Context) [4] large-scale object detection dataset. The 2017 configuration of the dataset consists of 118K training and 5K validation images from 80 different object categories. All the object categories are shown in Figure 4.1. It contains over 500K object instances. Besides object detection, the COCO dataset could be used for many computer vision tasks such as image captioning and semantic segmentation.

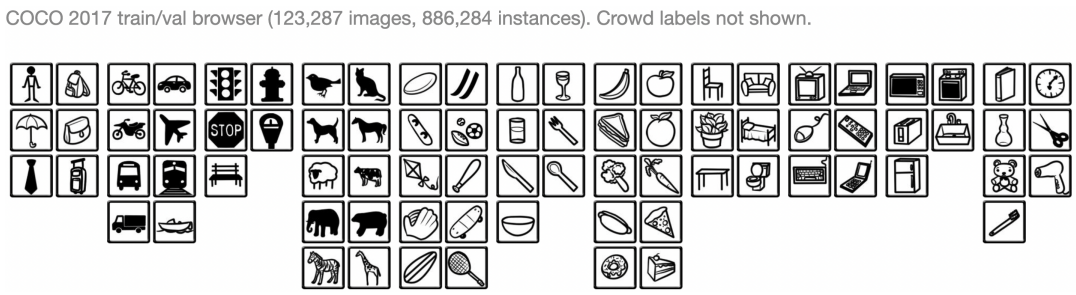


Figure 4.1: Object categories in COCO dataset [4].

4.2 Implementation Details

Implementation details related to all the reasoner networks are given in the following parts.

4.2.1 YOLOv3-Reasoner Models

In the first stage, all the YOLOv3-Reasoner configurations are trained for 100 epochs from scratch on the COCO dataset. Data Parallelism which is implemented by *DataParallel* function of the Pytorch [56] library is used. This function splits the input across the specified GPUs. In the forward pass, the same model runs the computation in each GPU for each of the different data portions (mini-batches) in parallel. In the backward pass, gradients from each device are gathered together to update the model. Hyperparameter settings of the YOLOv3 are used as it is, except for some minor modifications. The initial learning rate was set to 0.001. Adam optimizer is used with parameters set to β_1 = 0.9, β_2 = 0.999 and ϵ = 1e-08. As stated previously, all the network configurations were trained from scratch, and no pre-trained weights were used. The training time of different network configurations for 100 epochs varied from three to five days on 4 Quadro RTX 8000. In the second stage, the most promising models, YOLOv3-Reasoner2, YOLOv3-Reasoner4, and the baseline YOLOv3, are trained until they converge.

4.2.2 YOLOv4-P6-Reasoner

The YOLOv4-P6-Reasoner model is trained from scratch until it converges. Hyperparameter settings of the baseline YOLOv4-P6 are used as it is.

4.2.3 YOLOR-P6-Reasoner

The YOLOR-P6-Reasoner model is trained from scratch until it converges. Hyperparameter settings of the baseline YOLOR-P6 are used as it is.

4.3 Evaluation Metrics

As an evaluation metric, "mean average precision" (mAP) is used. To understand what the mAP is, it is required to define IoU, true positive (TP), false positive (FP), and false negative (FN) terms firstly. IoU is defined as the ratio of the overlapping area of two boxes to their union area. It is visualized in Figure 4.2.

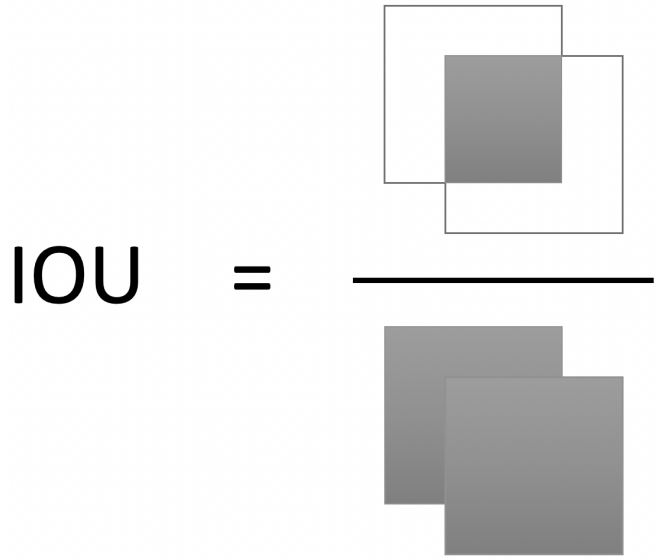


Figure 4.2: IoU (Intersection over Union).

In object detection, it is the ratio between predicted and ground truth bounding boxes. If the prediction has an IoU greater than or equal to the defined threshold, it is classified as TP. If IoU is less than the threshold, it is classified as FP. Only one prediction is matched with the ground truth. If there is more than one prediction for an object, the highest score is considered TP, and the rest of them are considered FP. If there is an object in the image and the model fails to detect it, this case is classified as FN and also contributes to FP. Precision and recall terms are defined in the following equations.

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

Average precision is defined as the area under the precision-recall curve. Its calculation was changed in 2010 to measure the performance of the methods more precisely [15]. New average precision is computed in two stages. A version of precision-recall curve is computed by setting the precision for recall *i+1* to the maximum precision for any recall $j \geq i+1$ [15, 57]:

$$\hat{precision}(recall_{i+1}) = \max_{recall_j \geq recall_{i+1}} precison(recall_j) \tag{4.3}$$

Then, AP (for a class) is computed as the area under this newly computed precision-recall curve [57]:

$$AP = \sum (recall_{i+1} - recall_i)\hat{precision}(recall_{i+1}) \tag{4.4}$$

Mean average precision is computed by taking the average over all the classes that the COCO dataset contains.

In the early evaluation part, we compare the models using mAP computed at a .50 IoU rate. In the quantitative evaluation, we use six different versions of the mean average precision computed for different conditions: AP, AP_{50}, AP_{75}, AP_S, AP_M, and AP_L. AP is the primary metric of the COCO Object Detection Task and is computed by taking the average for ten different IoU thresholds from .50 to .95 with .05 intervals. Taking an average of over ten different IoU thresholds highlights the detectors whose localization performance is better. AP_{50} is a predefined metric that is used in the early evaluation part. AP_{75} is the mean average precision computed for the .75 IoU threshold. We also compare the performance of the models in quantitative evaluation across different scales. AP_S is computed for small objects whose area is less than 32^2. The area is measured with the number of pixels in the segmentation mask of the object. AP_M is computed for medium objects whose area is less than 96^2 and greater than 32^2. AP_L is computed for large objects whose area is greater than 96^2.

4.4 Results and Analysis

In this section, experimental results are shared. Firstly, early evaluation results of the YOLOv3-Reasoner models are given. Proposed reasoner network configurations are summarized. Then, the proposed networks are compared with the baseline YOLOv3 in terms of mAP and inference time. In the quantitative evaluation, YOLOv3-Reasoner2, YOLOv3-Reasoner4, YOLOv4-P6-Reasoner, and YOLOR-P6-Reasoner models are compared with the state-of-the-art. Then, these reasoner models are compared with the baselines qualitatively.

4.4.1 Early Evaluation of YOLOv3-Reasoner Models

In the early evaluation stage, proposed reasoner models which use YOLOv3 as a baseline network are compared in terms of a number of parameters, mAP, and inference time.

4.4.1.1 Comparison of Yolov3-Base and Proposed Reasoner Models

The reasoner network configurations are explained in detail in section 3. A summary of these configurations can be seen in Table 4.1. A number of reasoning layers, number of heads in each scale, MLP size, and the total number of trainable parameters are given in the table. The difference in the total number of trainable parameters between baseline YOLOv3 and reasoner models is at an acceptable level when the total number of trainable parameters in the baseline model is considered. Even in the most complicated model, YOLOv3-Reasoner4, this difference is less than 5 million, which is nearly 7 percent of the total number of trainable parameters used in base YOLOv3. Thus, utilization of the reasoner models seems applicable in the environments where the baseline YOLOv3 could run.

The proposed YOLOv3-Reasoner networks and the baseline YOLOv3 are compared in terms of previously defined mAP and inference time. The results are shown in Figure 4.3. The performance of the YOLOv3-Reasoner1 is worse than the baseline YOLOv3. Also, its inference time is nearly 2 ms greater than the baseline. In this

Table 4.1: Details of Reasoner Networks

Model	Layers	Heads	MLP Size	Params
YOLOv3-Base	-	-	-	61,949,149
YOLOv3-Reasoner1	1	5-5-5	255-255-255	62,539,729
YOLOv3-Reasoner2	1	16-8-4	1024-512-256	65,612,736
YOLOv3-Reasoner3	1	16-8-4	1024-512-256	66,547,421
YOLOv3-Reasoner4	1	16-8-4	1024-512-256	66,547,421

configuration, the feature space that the reasoning layer works on is limited when compared with the other reasoner networks. This could be the main underlying reason for the degraded performance.

The YOLOv3-Reasoner2 achieves better performance than the baseline YOLOv3. In this configuration, the feature space that the reasoning layer works on is increased by switching the place of the 1x1 convolutional layer and the reasoning layer. Also, the number of heads used in the first two scales is increased. With the help of these improvements, YOLOv3-Reasoner2 achieves around 5% relative improvement in mAP with respect to the baseline YOLOv3.

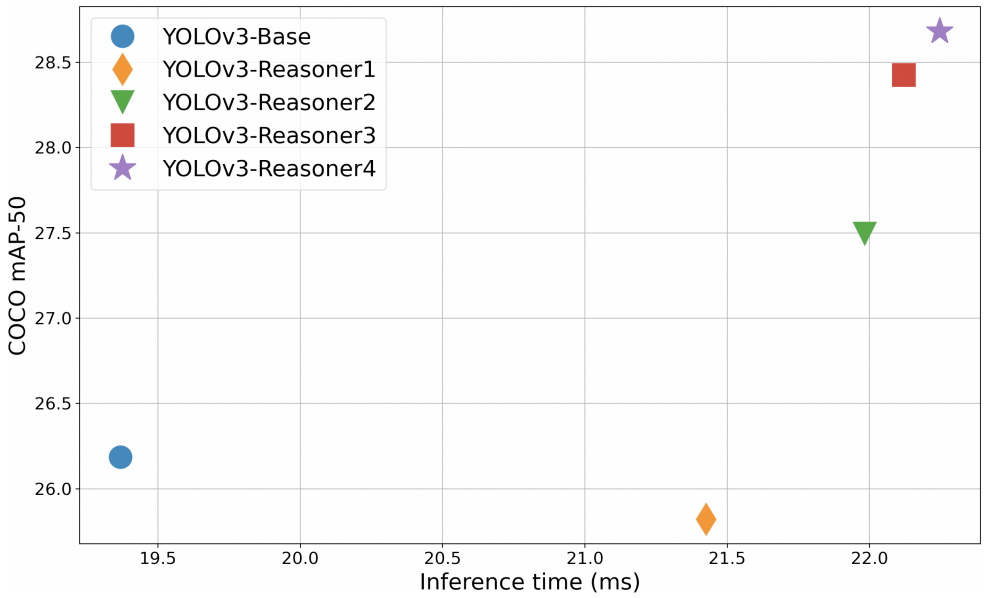
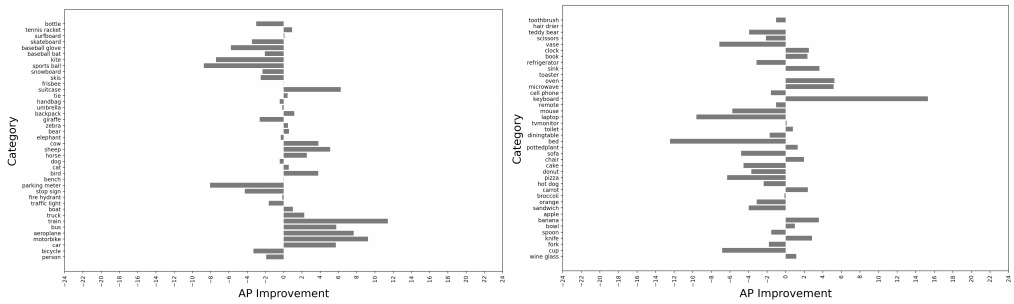


Figure 4.3: Comparison of the proposed YOLOv3-Reasoner networks and the baseline YOLOv3

The fusing strategy that is applied in the architecture of the YOLOv3-Reasoner3 further improves the performance. Using both the backbone output only-convolutional and reasoning features brings around 4% additional relative performance improvement to the YOLOv3-Reasoner3 over YOLOv3-Reasoner2. By the utilization of positional encoding, YOLOv3-Reasoner4 achieves the best result among all of the models. The YOLOv3-Reasoner4 model achieves greater than 9.5% relative improvement with respect to the baseline YOLOv3 in terms of mAP. Besides a minor increase in the inference time, the proposed YOLOv3-Reasoner4 model can still run in real-time with a speed of nearly 45 frames per second (FPS).

Differences in average precision for each category of the COCO dataset between the YOLOv3-Reasoner configurations and the baseline YOLOv3 are examined. Figure 4.4a and Figure 4.4b show the difference in average precision for 80 categories in COCO between the YOLOv3-Reasoner1 and the baseline YOLOv3. As it can be deduced from the figure, performance is improved in some of the categories while it degrades in the rest of them. However, YOLOv3-Reasoner1 performance is worse than the baseline YOLOv3 in overall.

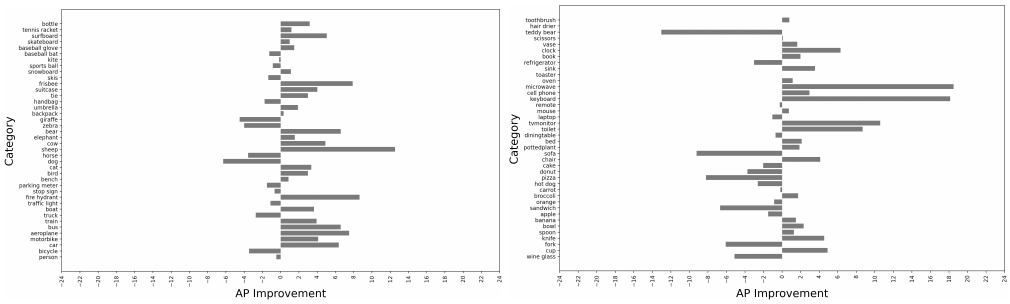


(a) First 40 categories

(b) Last 40 categories

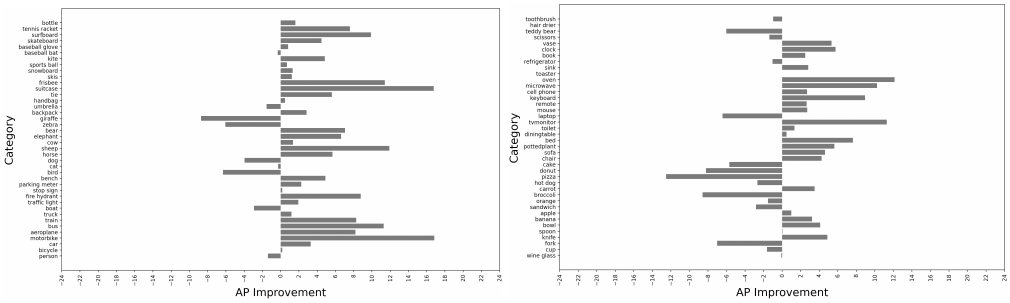
Figure 4.4: AP improvement achieved by YOLOv3-Reasoner1 over the baseline YOLOv3 on each category of the COCO

The performance is improved on much of the categories with YOLOv3-Reasoner2 and YOLOv3-Reasoner3 models as it is shown in Figure 4.5a, Figure 4.5b, Figure 4.6a and Figure 4.6b. There is still performance degrade in some of the categories, but both the YOLOv3-Reasoner2 and YOLOv3-Reasoner3 configurations could improve performance in overall.



(a) First 40 categories (b) Last 40 categories

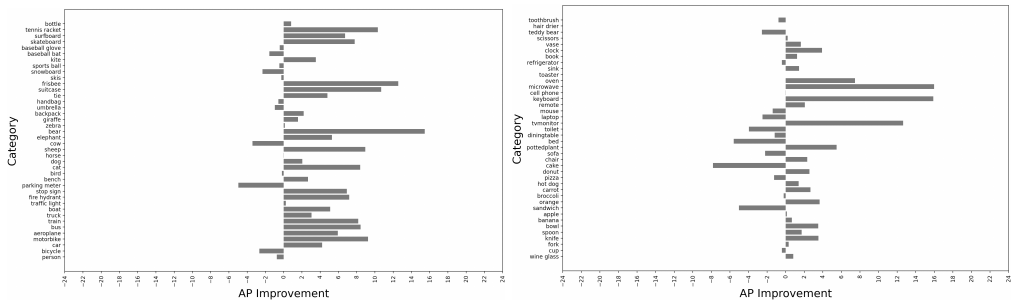
Figure 4.5: AP improvement achieved by YOLOv3-Reasoner2 over the baseline YOLOv3 on each category of the COCO



(a) First 40 categories (b) Last 40 categories

Figure 4.6: AP improvement achieved by YOLOv3-Reasoner3 over the baseline YOLOv3 on each category of the COCO

Differences in average precision for 80 categories of the COCO between the best reasoner model YOLOv3-Reasoner4 and the baseline YOLOv3 are shown in Figure 4.7a and Figure 4.7b. Improvements in the YOLOv3-Reasoner4 model are much better than in the YOLOv3-Reasoner3 model. It seems that usage of the positional encoding lowers the performance degrade in some of the categories. Added spatial information about the positions of the image regions has improved the performance of the YOLOv3-Reasoner4 model overall.



(a) First 40 categories

(b) Last 40 categories

Figure 4.7: AP improvement achieved by YOLOv3-Reasoner4 over the baseline YOLOv3 on each category of the COCO

4.4.1.2 Comparison of Yolov3-Reasoner4 Variants

Different variants of the best reasoner model YOLOv3-Reasoner4 are obtained by changing the hyperparameters. Capacity is increased in the models Yolov3-Reasoner4-Medium and YOLOv3-Reasoner4-Large by increasing the number of reasoning layers and heads in each scale. Details related to these YOLOv3-Reasoner4 variants can be seen in Table 4.2. The total number of trainable parameters for these models is also given. In the YOLOv3-Reasoner4-Medium variant, only the number of reasoning layers in each of the scales is increased to two. In the YOLOv3-Reasoner4-Large model, the number of reasoning layers in each scale is set to four. Moreover, the number of used heads in 3 different scales is increased to 32, 16, and 8, respectively. This configuration brings nearly 12 million extra parameters over the YOLOv3-Reasoner4-Base model.

Table 4.2: Details of YOLOv3-Reasoner4 Variants

Model	**Layers**	**Heads**	**MLP Size**	**Params**
YOLOv3-Reasoner4-Base	1	16-8-4	1024-512-256	66,547,421
YOLOv3-Reasoner4-Medium	2	16-8-4	1024-512-256	70,688,733
YOLOv3-Reasoner4-Large	4	32-16-8	1024-512-256	78,971,357

All the proposed YOLOv3-Reasoner4 variants and the baseline YOLOv3 are compared in terms of mAP and inference time. The results are shown in Figure 4.8.

YOLOv3-Reasoner4-Medium and YOLOv3-Reasoner4-Large variants achieve better performance than the baseline YOLOv3. Although these two variants are higher capacity models, their performance is worse than the YOLOv3-Reasoner4-Base. It is not easy to converge a high-capacity model. In Vision Transformer [45] paper, authors stated that they used extensive in-house dataset named JFT-300M [58] which contains more than 300 million high-resolution images. Also, they extensively trained their models on this JFT-300M dataset. Thus, a large-scale dataset and extensive training are crucial to converge a high-capacity transformer-based model. Also, the necessity of using a high-capacity model is debatable here. Convolutional backbone Darknet-53 is already in use for feature extraction. The reasoning layer is only responsible for modeling high-level semantic relationships between image regions. As it can be seen from the results, using a model with a reasonable capacity like the YOLOv3-Reasoner4-Base seems enough for performance improvement in this problem.

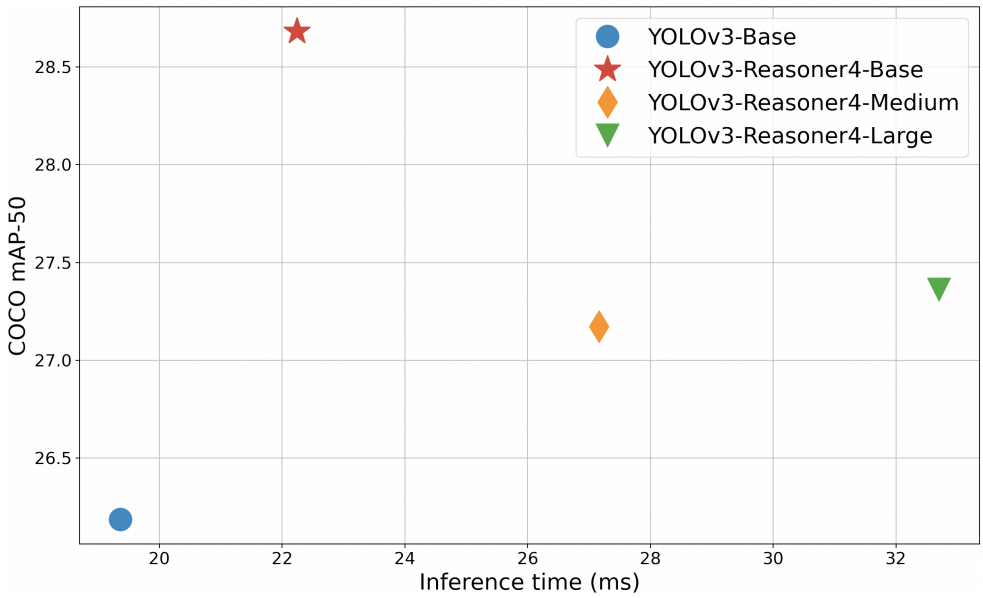
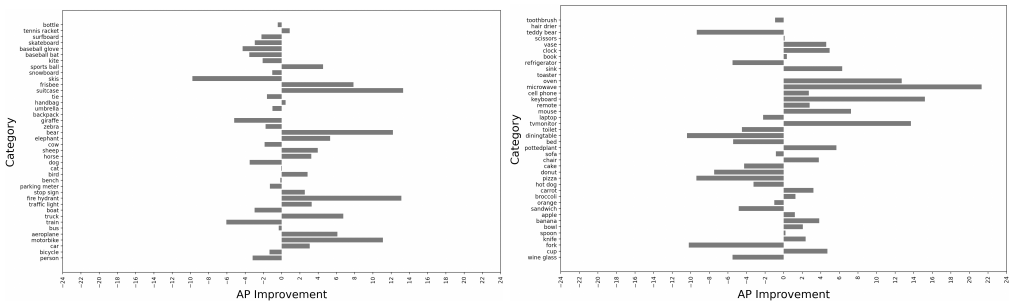


Figure 4.8: Comparison of the proposed YOLOv3-Reasoner4 variants and the baseline YOLOv3.

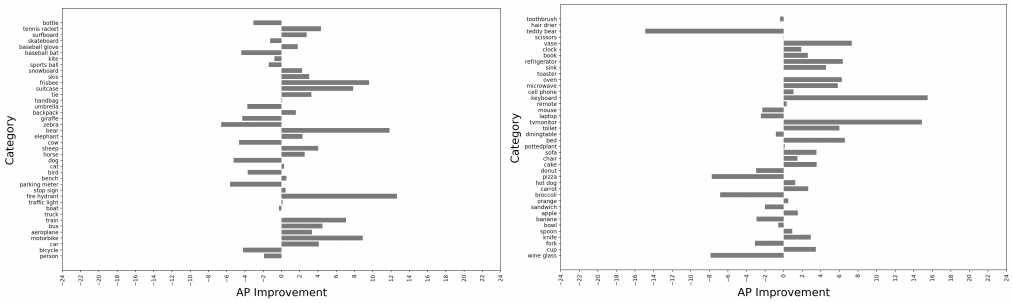
Differences in average precision for each category of the COCO dataset between the YOLOv3-Reasoner4 variants and the baseline YOLOv3 are shown in Figure 4.9a, Figure 4.9b, Figure 4.10a, and Figure 4.10b. As can be understood from the figures, the performance of the YOLOv3-Reasoner4-Medium and YOLOv3-Reasoner4-Large models is improved in some of the categories while degrading in the rest of them.

Despite a decrease in some categories, the overall performance of these two models is improved compared with the baseline YOLOv3.



(a) First 40 categories (b) Last 40 categories

Figure 4.9: AP improvement achieved by YOLOv3-Reasoner4-Medium over the baseline YOLOv3 on each category of the COCO



(a) First 40 categories (b) Last 40 categories

Figure 4.10: AP improvement achieved by YOLOv3-Reasoner4-Large over the baseline YOLOv3 on each category of the COCO

4.4.2 Quantitative Evaluation

The most promising reasoner models YOLOv3-Reasoner2 and YOLOv3-Reasoner4 are trained until they converge. As stated previously, we also apply the Reasoner4 method to the recent state-of-the-art one-stage object detectors: YOLOv4-P6 and YOLOR-P6. We train them until they converge with their baselines and compare their performance with other state-of-the-art object detectors. The results are shown in Table 4.3.

Table 4.3: Comparison with state-of-the-art object detectors.

Method	**Size**	**FPS**	**# params**	**AP**	**AP**$_{50}$	**AP**$_{75}$	**AP**$_S$	**AP**$_M$	**AP**$_L$
YOLOv3*	416	110	61M	21.6%	35.3%	22.8%	8.7%	23.7%	30.0%
YOLOv3-Reasoner2*	416	75	65M	27.1%	42.8%	28.6%	11.6%	29.5%	38.7%
YOLOv3-Reasoner4*	416	73	66M	27.9%	44.0%	29.6%	11.7%	30.1%	39.3%
YOLOv4-P6*	1280	27	128M	53.3%	71.1%	58.2%	35.6%	56.8%	64.9%
YOLOv4-P6-Reasoner*	1280	23	147M	53.8%	71.5%	58.8%	35.7%	57.4%	65.8%
YOLOv4-P7	1536	16	287M	55.5%	73.4%	60.8%	38.4%	59.4%	67.7%
YOLOR-P6*	1280	31	37M	52.0%	69.9%	56.9%	33.9%	55.9%	63.6%
YOLOR-P6-Reasoner*	1280	28	43M	52.5%	70.5%	57.6%	34.6%	56.3%	64.0%
YOLOR-D6	1280	-	152M	55.4%	73.3%	60.6%	38.0%	59.2%	67.1%
YOLOX-Darknet53 [37]	640	90	63M	47.4%	67.3%	52.1%	27.5%	51.5%	60.9%
YOLOX-M [37]	640	81	25M	46.4%	65.4%	50.6%	26.3%	51.0%	59.9%
YOLOX-L [37]	640	69	54M	50.0%	68.5%	54.5%	29.8%	54.5%	64.4%
YOLOX-X [37]	640	57	99M	51.2%	69.6%	55.7%	31.2%	56.1%	66.1%
EfficientDet-D0 [12]	512	97	4M	34.6%	53.0%	37.1%	12.4%	39.0%	52.7%
EfficientDet-D1 [12]	640	74	7M	40.5%	59.1%	43.7%	18.3%	45.0%	57.5%
EfficientDet-D2 [12]	768	57	8M	43.0%	62.3%	46.2%	22.5%	47.0%	58.4%
EfficientDet-D3 [12]	896	36	12M	47.5%	66.2%	51.5%	27.9%	51.4%	62.0%
EfficientDet-D4 [12]	1024	23	21M	49.7%	68.4%	53.9%	30.7%	53.2%	63.2%
EfficientDet-D5 [12]	1280	14	34M	51.5%	70.5%	56.7%	33.9%	54.7%	64.1%
EfficientDet-D6 [12]	1408	11	52M	52.6%	71.5%	57.2%	34.9%	56.0%	65.4%
DETR [19]	-	28	41M	42.0%	62.4%	44.2%	20.5%	45.8%	61.1%
DETR-DC5 [19]	-	12	41M	43.3%	63.1%	45.9%	22.5%	47.3%	61.1%
DETR-R101 [19]	-	20	60M	43.5%	63.8%	46.4%	21.9%	48.0%	61.8%
DETR-DC5-R101 [19]	-	10	60M	44.9%	64.7%	47.7%	23.7%	49.5%	62.3%
Swin-L (HTC++) [44, 59]	-	-	284M	57.7%	-	-	-	-	-

* Trained from scratch on COCO and no pre-trained weights used.

Their inference speeds are measured on a single Tesla V100 with batch=1. Measurements related to the other methods given in the table are also taken on Tesla V100, as stated in related papers. The performance is significantly improved in models YOLOv3-Reasoner2 and YOLOv3-Reasoner4 compared with the baseline YOLOv3. The mAP at 0.5 IoU is improved to 42.8% from 35.3% in YOLOv3-Reasoner2. It is further improved to 44.0% when neck output features are fused with the reason-

ing layer's output features in the YOLOv3-Reasoner4 model. The method used in YOLOv3-Reasoner4 is applied to the YOLOv4-P6 and YOLOR-P6 baselines. The performance is improved in the YOLOv4-P6-Reasoner model when compared with the baseline network YOLOv4-P6. The improvement is limited when compared with the previous baseline YOLOv3. However, it is increased in all object size categories: small, medium, and large. YOLOv4-P6-Reasoner is also still running in real-time with 23 frames per second (FPS). One possible explanation for this limited improvement is that the YOLOv4-P6 model may be over-optimized, as stated in YOLOX paper [37]. As a result of this over-optimization over the COCO dataset, the performance gap filled by the reasoning approach may not be as good as in the YOLOv3 baseline.

Like YOLOv4-P6-Reasoner, performance improvement achieved by YOLOR-P6-Reasoner is limited over the baseline network. The mAP at 0.5 IoU is improved to 70.5% from 69.9%. As previously stated for YOLOv4-P6-Reasoner, a possible reason for this limited improvement could be again related to the over-optimization of YOLOR-P6 on the COCO. When compared with the other methods shown in Table 4.3, YOLOR-P6-Reasoner seems an optimum solution for real-time one-stage object detection with low resource usage, high FPS, and precision.

4.4.3 Qualitative Evaluation

Qualitative examples from COCO test-dev2017 are shared in this part. In Figure 4.11, reasoner models YOLOv3-Reasoner2 and YOLOv3-Reasoner4 are compared with the baseline YOLOv3. The top left of Figure 4.11 shows that YOLOv3 could not detect the red-colored bus, which is partially occluded. YOLOv3-Reasoner2 and YOLOv3-Reasoner4 successfully detect it. YOLOv3-Reasoner4 also detects the driver of the gray-colored bus.

The top right of Figure 4.11 shows the result of YOLOv3 for a street view image. It could detect a lower number of cars and people in the image. Reasoner models, especially the YOLOv3-Reasoner4, could detect cars and people in challenging conditions. The traffic light is also detected by the YOLOv3-Reasoner4 while ignored by the YOLOv3-Reasoner2. One possible argument for this situation is that YOLOv3-

Reasoner2 may not be expecting a traffic light of this size compared with the cars and people in the image.

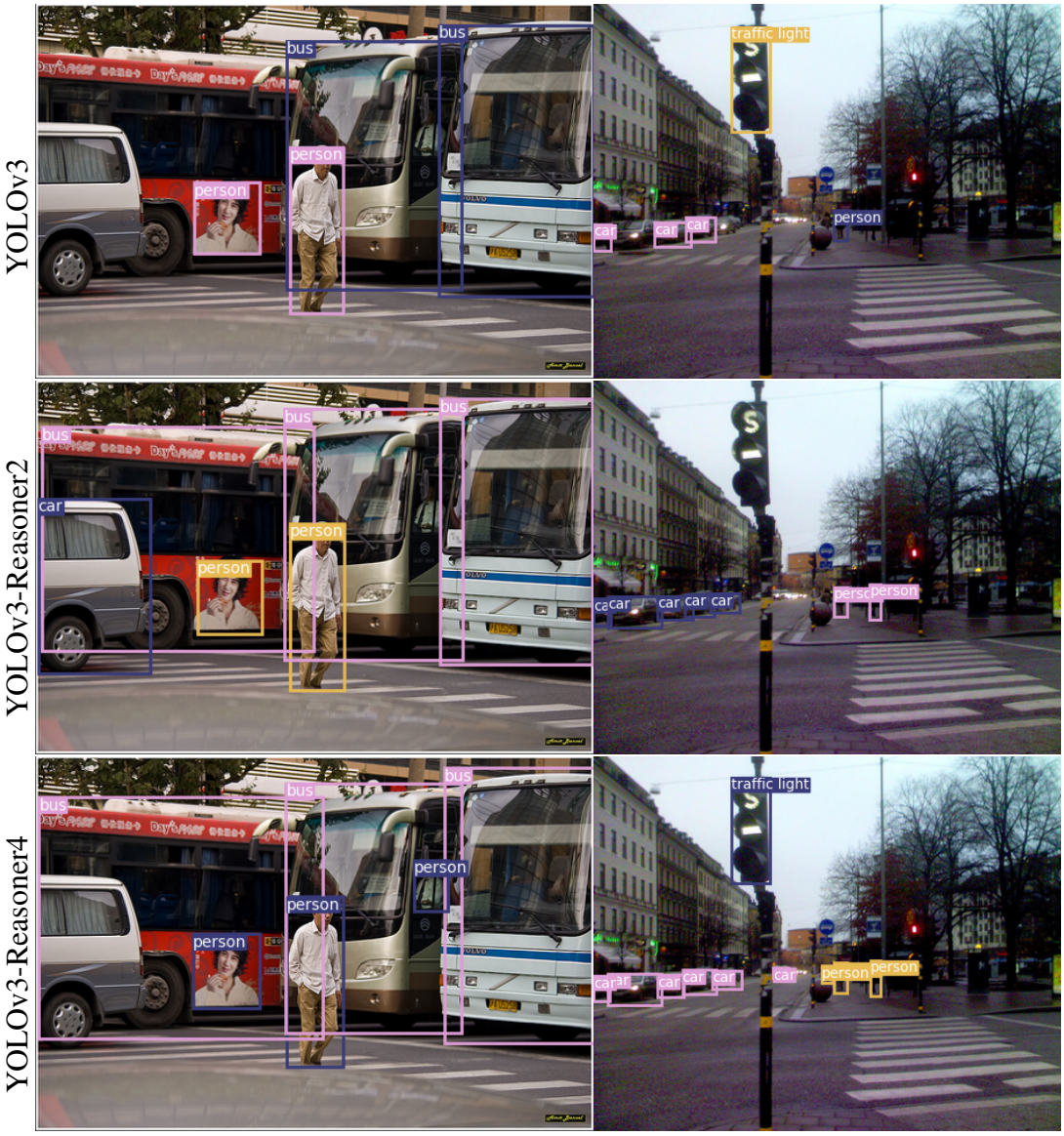


Figure 4.11: Qualitative result comparison between YOLOv3, YOLOv3-Reasoner2, and YOLOv3-Reasoner4

In Figure 4.12, YOLOv4-P6-Reasoner is compared with the baseline YOLO4-P6. The top left of Figure 4.12 shows that YOLOv4-P6 could detect the dog figurine in front of the window while ignoring the microwave. However, YOLOv4-P6-Reasoner ignores the dog figurine and successfully detects the microwave. One possible reason is that YOLOv4-P6-Reasoner may not be expecting a dog whose size is smaller than the cup or bottle in an image.

The top right of Figure 4.12 shows the result of YOLOv4-P6 for a street view image that contains different objects such as a train, bus, car, person, and bicycle. YOLOv4-P6 can not detect the far away truck and the bicycle on the person's right side on the street. It can be difficult for even a human to detect this bicycle at first glance. However, YOLOv4-P6-Reasoner could successfully detect the far away truck and the bicycle.

Figure 4.12: Qualitative result comparison between YOLOv4-P6 and YOLOv4-P6-Reasoner

In Figure 4.13, YOLOR-P6-Reasoner is compared with the baseline network YOLOR-P6. When the top left and bottom left figures are compared, it can be seen that YOLOR-P6-Reasoner detects more people and the chair close to them. Generally, reasoner models are successful in detecting groups of related objects despite challenging conditions.

The top right of Figure 4.13 shows the result of YOLOR-P6 for an image that contains kitchen-related objects such as a bowl, knife, and orange. YOLOR-P6 detects the haft of the bottom knife as a carrot because of the illumination. There is a possibility even for a human to detect it as a carrot if the whole context is not available. YOLOR-P6-

Reasoner does not detect it as a carrot. One possible explanation is that YOLOR-P6-Reasoner uses its reasoning ability to recognize the knife as a whole and not detect the haft as a carrot.

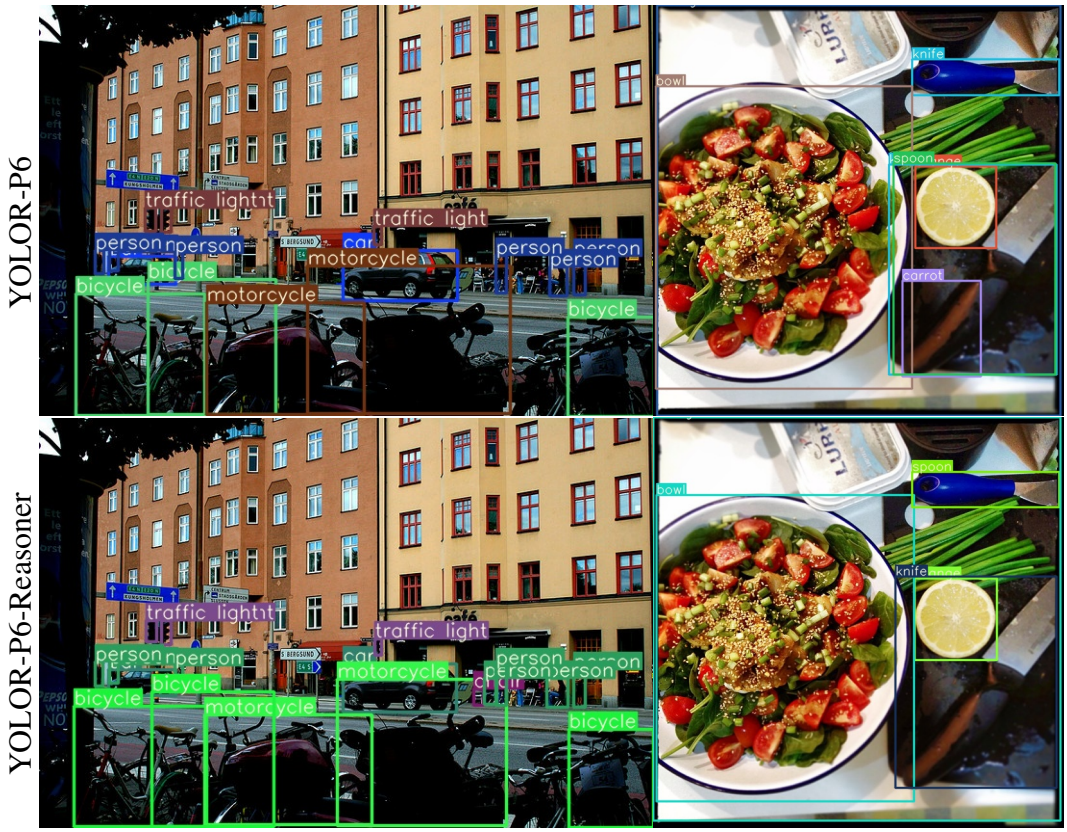


Figure 4.13: Qualitative result comparison between YOLOR-P6 and YOLOR-P6-Reasoner

We can deduce from the qualitative results that reasoner models could understand the semantic relationships between objects in an image. They are possibly using the reasoning knowledge to improve performance in challenging conditions such as partial occlusion and small-sized objects. As could be understood from the dog figurine result, their ignorance of objects may also be grounded on reasoning.

CHAPTER 5

CONCLUSIONS

In this thesis, a new architecture is presented as a way of using visual reasoning to improve one-stage object detection. This new architecture aims to bring the ability of reasoning between image regions to one-stage object detection. The current state-of-the-art one-stage object detection methods treat each image region locally and individually without considering semantic relationships between objects. This prevents them from detecting objects in an image region if high-quality convolutional features cannot be extracted. This is highly possible in challenging conditions such as deformation and occlusion.

The proposed method enhances features of the image regions by considering semantic relationships. The semantic relationships between image regions are extracted by utilizing the reasoning layer. The reasoning layer takes a bulk of convolutional features from the neck and processes them after converting to a sequence through flattening. The most critical part of this process is modeling visual reasoning, which takes place in multi-head attention. Even if a convolutional network is very deep, it has limited access to the entire image. The limited receptive field prevents convolutional networks from accessing the whole context of the image. On the other hand, multi-head attention has the ability to access the entire image. This ability provides a great advantage to reason between image regions globally. In this way, multi-head attention uses the whole context while encoding different relationships of an image region with others.

In this work, the proposed approach is adapted to three different state-of-the-art baseline one-stage object detectors: YOLOv3, YOLOv4-P6, and YOLOR-P6. In early evaluation, four different reasoner network configurations which use YOLOv3 as

a baseline are trained and tested on COCO [4]. In the YOLOv3-Reasoner1 and YOLOv3-Reasoner2 configurations, only reasoning layer output is fed to the detection head. In the YOLOv3-Reasoner3 and YOLOv3-Reasoner4 configurations, backbone output convolutional features are concatenated with reasoning features. Then, this new feature set is fed to the detection head after fusing in the 1x1 convolutional layer. YOLOv3-Reasoner4 configuration achieves the best performance among all the models. Although direct usage of reasoning features in the detection head achieves better performance than the baseline in the YOLOv3-Reasoner2 configuration, fusing only-convolutional and reasoning features gives the best result by ensuring the reusability of the original backbone output features. In the second part of the early evaluation, different variants of the best reasoner model YOLOv3-Reasoner4 are compared in terms of mAP and inference time. These variants are obtained by increasing the capacity of the YOLOv3-Reasoner4 model. The necessity of using a high-capacity model is debatable here. A convolutional backbone is good at extracting object-specific features. The role of the reasoning layer is to extract high-level semantic relationships between image regions. Using a convolutional backbone for feature extraction and a reasoning layer for modeling high-level semantic relationships seems to be the best practice.

In the quantitative evaluation, we compare the proposed reasoner networks YOLOv3-Reasoner2, YOLOv3-Reasoner4, YOLOv4-P6-Reasoner, and YOLOR-P6-Reasoner with their baselines and other state-of-the-art object detectors. YOLOv3-Reasoner2 and YOLOv3-Reasoner4 achieve significant improvement over the baseline, while YOLOv4-P6-Reasoner and YOLOR-P6-Reasoner achieve limited improvement. It may be related to over-optimized baselines YOLOv4-P6 and YOLOR-P6. However, YOLOv4-P6-Reasoner and YOLOR-P6-Reasoner still achieve improvement in all average precision metrics.

As can be deduced from the qualitative comparison of the reasoner models with their baseline networks, the reasoning layer possibly affects the predictions by taking into account the semantic integrity in a given scene. Object sizes and locations could be thought of as a part of semantic integrity. Reasoner models expect objects in reasonable sizes and locations compared with the other ones in a given scene. If the application requires the detection of objects with inconsistent size, like a dog fig-

urine example, preferring only convolutional models may be more suitable. However, reasoner models will be the right choice if the application requires the detection of objects in challenging conditions such as partial occlusion, small-sized objects, and illumination.

The results and analysis indicate that visual reasoning is promising for advancing one-stage object detection.

REFERENCES

[1] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[2] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," 2019.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[4] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

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

[6] P. Viola and M. Jones, "Robust real-time face detection," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, pp. 747–747, 2001.

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

[8] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[9] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21–37, 2016.

[12] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," 2020.

[13] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[14] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-yolov4: Scaling cross stage partial network. arxiv," *arXiv preprint arXiv:2011.08036*, 2020.

[15] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010.

[16] X. Chen and A. Gupta, "Spatial memory for context reasoning in object detection," 2017.

[17] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," 2018.

[18] X. Chen, L.-J. Li, L. Fei-Fei, and A. Gupta, "Iterative visual reasoning beyond convolutions," 2018.

[19] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," 2020.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[21] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "You only learn one representation: Unified network for multiple tasks," *arXiv preprint arXiv:2105.04206*, 2021.

[22] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," 2015.

[23] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2018.

[24] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[27] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2017.

[28] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 2016.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[30] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8759–8768, 2018.

[31] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Nas-fpn: Learning scalable feature pyramid architecture for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7036–7045, 2019.

[32] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[33] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of*

the IEEE/CVF international conference on computer vision, pp. 6023–6032, 2019.

[34] Z. Zheng, P. Wang, W. Liu, J. Li, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression. arxiv," *arXiv preprint arXiv:1911.08287*, 2019.

[35] D. Misra, "Mish: A self regularized non-monotonic activation function," *arXiv preprint arXiv:1908.08681*, 2019.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional nets for visual recog," ECCV, 2014.

[37] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[38] G. Song, Y. Liu, and X. Wang, "Revisiting the sibling head in object detector," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11563–11572, 2020.

[39] Y. Wu, Y. Chen, L. Yuan, Z. Liu, L. Wang, H. Li, and Y. Fu, "Rethinking classification and localization for object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10186–10195, 2020.

[40] K. Marino, R. Salakhutdinov, and A. Gupta, "The more you know: Using knowledge graphs for image classification," 2017.

[41] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[42] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[43] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014.

[44] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

[45] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[46] Z. Zhang, X. Lu, G. Cao, Y. Yang, L. Jiao, and F. Liu, "Vit-yolo:transformer-based yolo for object detection," in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 2799–2808, 2021.

[47] D. Du, P. Zhu, L. Wen, X. Bian, H. Lin, Q. Hu, T. Peng, J. Zheng, X. Wang, Y. Zhang, *et al.*, "Visdrone-det2019: The vision meets drone object detection in image challenge results," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

[48] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 390–391, 2020.

[49] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "Opennmt: Open-source toolkit for neural machine translation," in *Proc. ACL*, 2017.

[50] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," 2017.

[51] N. Adaloglou, "Transformers in computer vision," *https://theaisummer.com/*, 2021.

[52] N. Adaloglou, "Intuitive explanation of skip connections in deep learning," *https://theaisummer.com/*, 2020.

[53] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[54] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016.

[55] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2018.

[56] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[57] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results." http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html.

[58] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," 2017.

[59] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, *et al.*, "Hybrid task cascade for instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4974–4983, 2019.