TÉCNICO LISBOA

Improving Object Detection through Contextual Rescoring

Lourenço Maria Casella Vaz Pato

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Pedro Manuel Quintas Aguiar
Prof. Renato Manuel Pereira Negrinho

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Pedro Manuel Quintas Aguiar
Member of the Committee: Prof. André Filipe Torres Martins

November 2019

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Declaração

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

Agradecimentos

Aos meus orientadores, Pedro Aguiar e Renato Negrinho, pela disponibilidade, pelas valiosas ajudas e conselhos, pelas longas discussões e pelo desafio que sempre colocaram em mim.

Aos meus colegas que me acompanharam e ajudaram durante o percurso de cinco anos no Técnico. Em particular ao Ismael Trindade, Afonso Castela, Diogo Ferreira, Tomás Matias, Duarte Dias e José Diogo Peres, pela amizade, companheirismo, e pelos conselhos que ofereceram durante a escrita desta tese.

Um agradecimento especial à minha família, pais, avós e irmãos que sempre me apoiaram durante todo o meu percurso académico e aos quais dedico esta tese.

Resumo

Detectores de objectos actuais baseiam-se numa abordagem de dois passos: primeiro, identificar regiões na imagem que contenham objectos, e depois, prever a classe de objecto dentro das regiões. As previsões de classe são feitas independentemente para cada região, não utilizando informação contextual, que pode ser inferida pela presença de outros objectos. A partilha de informação entre detecções claramente contribui para melhorar os resultados no problema de reconhecimento, dado que existem fortes dependências entre co-ocorrências de objectos na mesma imagem (ex. é improvável encontrar um sofá e um cavalo na mesma imagem).

Neste trabalho, abordamos o problema da utilização do contexto na detecção de objectos. Fazemos uma análise dos principais erros feitos pelos detectores actuais e que fontes de contexto podem ser usadas para mitigar esses erros. Para incorporar informação relativa a co-ocorrências, propomos um modelo que utiliza redes neuronais recorrentes bidireccionais com mecanismo de atenção treinadas para reavaliar o conjunto de detecções produzidas por uma arquitectura de detecção existente. Propomos um objectivo de treino como sendo o conjunto de confianças reavaliadas que maximiza a Average Precision para o conjunto de detecções em causa. Resultados experimentais no MS COCO dataset demonstram que o modelo proposto obtém melhorias consistentes na Average Precision na ordem de 0.5 a 1, utilizando diferentes detectores (Cascade R-CNN e Faster R-CNN) e diferentes backbones convolucionais (ResNet-50 e ResNet-101).

Palavras-chave: Aprendizagem Profunda, Visão Computacional, Detecção de Objectos, Contexto, Redes Neuronais Recorrentes, Mecanismos de Atenção.

Abstract

Current state-of-the-art object detectors rely on a two-stage approach: first, identify regions in the image that are likely to contain objects, then predict the object class inside the regions. Class predictions are made independently from other regions, thus having an insufficient use of context that can be inferred from the presence of other objects. Sharing this information would clearly improve the results of the recognition problem since there are strong dependencies between co-occurrences of objects in the same image (e.g., an image is unlikely to contain both a couch and a horse).

We tackle the problem of incorporating context in object detection. We analyse the errors that current detectors make and what sources of context can be used to mitigate these errors. To incorporate the information relative to object co-occurrences, we propose a bidirectional recurrent neural network with attention model that learns a rescoring rule, given a set of object detections from an existing detection architecture. The training target we propose is the set of rescored confidences that maximises Average Precision for the given set of detections. Through experiments in the MS COCO dataset, our model obtains consistent improvements in Average Precision that range from 0.5 to 1, across different convolutional backbones (ResNet-50 and ResNet-101) and different architectures (Cascade R-CNN and Faster R-CNN).

Keywords: Deep Learning, Computer Vision, Object Detection, Context, Recurrent Neural Networks, Attention Mechanisms

Contents

List of Tables

List of Figures

List of Acronyms

CNN Convolutional neural network.

ILSVRC ImageNet Large Scale Visual Recognition Challenge.

RPN Region proposal network.

DPM Deformable parts-based model.

RNN Recurrent neural network.

GRU Gated recurrent unit.

LSTM Long short-term memory.

AP Average Precision.

IoU Intersection over Union.

RNN Recurrent neural network.

MS COCO Microsoft Common Objects in Context.

PASCAL VOC PASCAL Visual Object Classes.

mAP Mean Average Precision.

R-CNN Region convolutional neural network.

SVM Support vector machine.

RoI Region of Interest.

NMS Non-maximum suppression.

YOLO You Only Look Once.

ION Inside-Outside Net.

SIN Structure Inference Net.

NLP Natural language processing.

MLP Multi-layer perceptron.

ReLU Rectified linear unit.

Chapter 1

Introduction

1.1 Motivation

In computer vision, object classification usually refers to scenarios that deal with images taken under somewhat controlled environments, where typically only the candidate object is depicted. In contrast, when the goal is to simultaneously locate and identify multiple objects in a single image, the task is referred to as *object detection*. In this case, a successful detection is represented by a bounding box that tightly encloses an object in the image and the corresponding object class label.

Object detection has attracted the attention of the scientific community in the recent past because being able to understand the surrounding environment in an automatic way is crucial for applications in areas such as autonomous vehicles, social robots, surveillance or other industrial tasks. In controlled environments, the objects can be captured under a constrained setting, e.g., using plain background and a fixed pose. In this scenario, the recognition problem is simplified and good results have been demonstrated for object classification.

Object detection is much harder than object classification because it requires the system to deal with multiple objects, typically of distinct sizes and appearances, imaged against uncontrolled backgrounds, instead of just classifying one instance per image. In fact, several real-world applications involve a multitude of challenging situations - occlusions, multiple object instances, background clutter, scale and rotational variance, and the presence of small objects - that make the detection problem particularly difficult to solve. While object detection seems trivial for humans, it is far from being solved by machines.

Current state-of-the-art object detection methods base their decisions on local features, i.e., on spatial characteristics of local image patches [1–4]. This strongly contrasts with the human approach to object detection, which is based not only on local evidence but also on a global understanding of the context within which the object is present. This has been made clear by studies that show how humans are worse at classifying local image patches but outperform automatic systems when the surrounding context is given [5]. Figure 1.1 illustrates this point with a compelling example for the use of contextual information in object detection - a human is able to identify the object from the perceived environment.

Naturally, incorporating context in the detection pipeline is expected to improve performance in sev-

Figure 1.1: Patches in the images are composed of similar pixel patterns, differing by a 90° rotation. Contextual information from the scene and object orientation allow us to recognise it as a car in the left image, and a pedestrian in the right (extracted from [6]).

eral scenarios. Using the surrounding background and the presence of other objects in the image, a detector can more confidently infer the class and the location of each object being analysed.

Moreover, in the more global problem of visual perception, object detection is only a part of the solution. When the goal is a system able to infer the global understanding of the environment, context is a key element, linking the task of identifying visual patterns with the one of understanding the environment.

1.2 Object Detection Overview

Early work on object detection relied on extracting hand-crafted low-level features such as edges or key-points and using these features to train machine learning models to classify the objects into the respective category [7]. More advanced features include SIFT features [8] and histograms of oriented gradients [9]. An example of these approaches is the bag-of-visual-words [10].

Currently, the most advanced object detectors rely on convolutional neural networks (CNNs). After AlexNet's [11] success in the popular image classification challenge ILSVRC 2012 [12], deep CNNs have dominated almost every sub-field of computer vision and the first approaches to object detection using CNNs followed shortly after [1]. These CNN models were applied to a selected set of "regions of interest" in the image (generated by a region proposal method), followed by a set of class-specific classifiers. Later approaches shared computation between regions of interest [2], and included the identification of those regions in the detection pipeline, using a Region Proposal Network (RPN) [3]. These changes substantially reduced processing time for each image and improved detection performance. Current state-of-the-art detectors use a cascade of bounding box regression/classification sub-networks that are progressively trained to refine higher quality predictions [4].

More recently, researchers started to develop methods to include context in the detection process, coming up with two typical approaches that differ by including context as a post-processing phase (as a rescoring or refinement step) [13–15] or including it directly in the detection pipeline [16–19].

These studies have used a wide range of contextual cues such as local visual features in the image, the geometric layout of the scene, the semantic context from other objects in the image, geographic location, and cultural context to inform detections [16].

Initial approaches incorporating context in object detection used logistic regression classifiers to predict object presence, location and size from contextual descriptors [16]. Later approaches have experimented with complex models that explore the relationships between object classes, using deformable parts-based models (DPM) [13, 17], binary trees [15] or graphical models [19]. More recently, deep neural networks have also been used to model contextual information using spatial recurrent neural networks [14, 18].

Our work contributes to the field by exploring the use of recurrent neural networks with attention to rescore sequences of detections. These models have implicit and explicit mechanisms that are able to capture contextual information in sequences, having achieved state-of-the-art results in tasks such as neural machine translation [20, 21] and image captioning [22].

1.3 Proposed Approach

In this work, we propose a rescoring approach to object detection (see Figure 1.2). The input of our model is a set of detections made by an existing object detection architecture (baseline). From this input, our contextual rescoring model predicts a new confidence value for each detection.

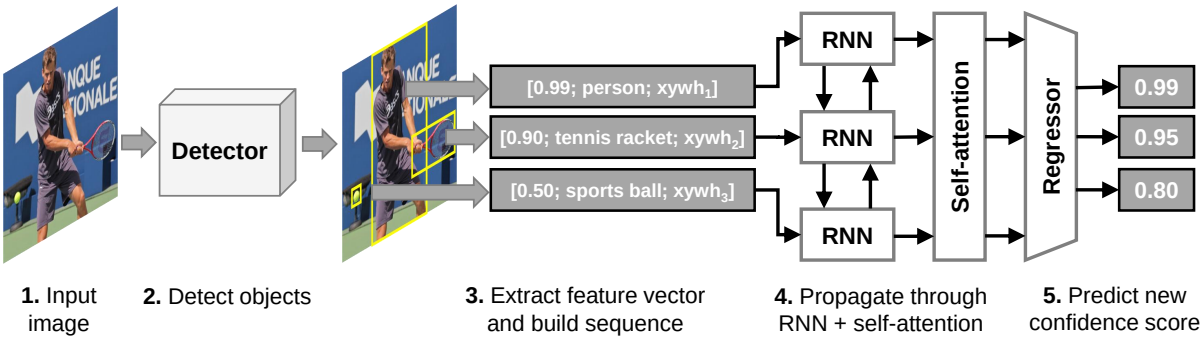


Figure 1.2: Overview of the proposed approach. **1-2.** A set of detections is collected by an object detection model. **3.** A vector of high-level features is extracted from each detection (by concatenating the confidence score, object category, and bounding box coordinates) and the sequence of detections is created. **4.** The sequence is processed by a RNN with self-attention mechanism. **5.** A regressor predicts a new confidence score for each object in the sequence. The target confidence values for the sequence of detections are the confidence values that maximise AP for the given set of detections.

Our approach starts by converting a set of detections into a *sequence* of detections, which is ordered by descending confidence. Then, our model processes the sequence of detections with the goal of predicting the new sequence of rescored confidences. When making a new prediction, the model takes into account the *whole sequence* of objects detected in the image. Given the annotations and the set of detections we compute the optimal set of confidences with respect to AP and use it as the training targets. Context is incorporated via the exploitation of the RNN's internal memory and the self-attention mechanism, which can be interpreted as creating an explicit representation of the sequence context.

An interesting point that deserves to be emphasised is that the proposed model does not use visual information to exploit context. In fact, each detection is represented by a feature vector that only includes the original detection confidence, the detected object category, and the bounding box coordinates. While

the baseline architecture only uses visual information, our model focus instead on exploiting the usage of what can be called high-level context, such as class co-occurrence information, relative positioning between objects and the object relative size in the image. Naturally, one immediately thinks of simply feeding the image visual features to our models, in addition to the already provided features. However, our approaches with this scenario (reported in Appendix A) have produced little to no AP improvements with the addition of significant computational costs.

The proposed model consists of a bidirectional 3-layer gated recurrent unit (GRU) with self-attention. We chose a GRU as a variant of recurrent neural networks (RNNs) to process sequential data and capture long-range dependencies. By using a bidirectional GRU, we make sure that every prediction is informed by all other objects in the sequence. Stacking three GRU layers improves performance by allowing the model to learn more complex dependencies. The self-attention mechanism addresses long-range dependencies and improves performance by using a context vector from all the elements in the sequence. Our experiments also show that the GRU models outperform long short-term memory (LSTM) models in the rescoring task by a small margin.

To train the model, we formulate the problem as a regression problem and introduce a training target that is aligned with the desired performance metric: Average Precision (AP). During the computation of AP, detections are sorted by confidence, assigning higher priority to high confidence detections, while detections with better localisation have more weight in the final result due to the increasing levels of interception over union (IoU) threshold. The optimal rescoring value must sort detections in such a way that correct detections have higher confidence than incorrect ones and prioritise detections with better localisation. The proposed training target addresses this mismatch by assigning a higher target confidence to detections with better localisation and thus optimising for the AP metric. The training target is defined to be the IoU overlap with the associated ground-truth instance for correct detections; on the other hand, false positives get assigned a target of zero.

Results on the MS COCO dataset show that the proposed model consistently improves Average Precision by 0.5 to 1 across different state-of-the-art baseline detectors and different backbone networks. An analysis of the rescored detections shows that the model is able to decrease confidence for confusions with background, increase confidence for correct detections, decrease localisation errors and confusions with different classes, and single out objects that have duplicate detections. Models are also generalisable: a detector trained on the detections from a detection architecture is able to improve performance on the detections of other detection architectures. We may then conclude that the proposed model learns to exploit context from detections (e.g. correlations between classes and object locations) and to use it to effectively rescore detections and improve performance on top of very strong baseline detectors.

Original Contributions

We identify the original contributions of this work as:

- the error analysis on current state-of-the-art detectors;
- the context rescoring approach applied to sequences of detections, which is agnostic to the detector used to produce the set of detections;
- the use of RNNs with attention to process sequences of detections in an image;
- the use of the IoU with the ground-truth as the training target that maximises Average Precision.

Although other authors have used RNNs to process visual feature maps (also known as spatial RNNs) [18], our approach is the first to use them to process sets of detections. While the traditional training target used by other approaches optimises for the log loss or for a multi-task loss [2–4, 23], ours optimises directly for the optimal rescoring values.

1.4 Thesis Outline

Chapter 2 introduces object detection and overviews previous literature in the field. We start by introducing the main concepts, metrics and the most popular datasets. The literature review starts by describing the approaches that have achieved state-of-the-art results in recent years, covering both multi-stage region-based architectures and single-stage architectures. Finally, we describe the previous work that incorporates context in object detection.

In Chapter 3, we describe recurrent neural networks and attention mechanisms. In our proposed approach, these models are used to process a sequence of detections in an image.

Chapter 4 analyses the main errors made by state-of-the-art object detection architectures and discusses how context can be used to mitigate these errors. This chapter also describes an exploratory analysis on how to generate the training targets that maximise Average Precision. We propose and compare three matching algorithms and three target values.

Chapter 5 details the proposed approach. First, we formulate the problem of rescoring candidate detections. Then, we describe the proposed model architecture and its implementation, which uses recurrent neural networks with self-attention to rescore detections with the contextual information from other objects in the image.

In Chapter 6, we report and discuss the results of the proposed approach. We show performance improvements on Average Precision for different detectors. We analyse the context learnt by the model and discuss its limitations. The chapter also includes a study of the choices of model parameters.

Chapter 7 concludes the thesis, summarising the approach and outlining the main results and limitations to be addressed in future work.

In Appendix A, we report the results from experiments that add visual features to the proposed model and compare them to rescored results without visual features.

In Appendix B, we include rescored image samples from the validation set, sorted by a selection metric.

Chapter 2

Background in Object Detection

This chapter starts by introducing metrics and the most popular datasets in the literature. The next section describes state-of-the-art methods for object detection, covering both multi-stage architectures and single-stage architectures. We then overview previous work that proposes models that incorporate context in object detection.

2.1 Metrics

Intersection over Union

Intersection over union (IoU) measures the overlap between two bounding boxes. In object detection, it measures the overlap between a predicted bounding box B_p with a ground-truth bounding box B_{gt}. IoU is given by the area of their intersection divided by the area of the union of the two boxes:

$$\text{IoU}(B_p, B_{gt}) = \frac{\text{Area}(B_p \cap B_{gt})}{\text{Area}(B_p \cup B_{gt})}. \tag{2.1}$$

This metric is used to evaluate localisation performance. Performance is higher if predictions have a high IoU overlap with the ground-truth. IoU is invariant to bounding box scalings.

Types of detections

In object detection, there are four possible types of detections:

- True Positive (TP): a correct detection with the correct class (IoU $\geq$ *threshold*).
- False Positive (FP):
  - confusion with background: detected an object where there is none (IoU $<$ *threshold*);
  - misclassification: detected an object but assigned the wrong class (IoU $\geq$ *threshold*);
  - duplicate detection: correctly detected an object that was already detected (IoU $\geq$ *threshold*);
- True Negative (TN): no detection made where there is no object.
- False Negative (FN): failed to detect a ground-truth bounding box.

The *threshold* specifies how accurate the locations must be for the detections to be associated to a ground-truth. The standard value for *threshold* is 0.5.

Precision and Recall

Precision and recall are common metrics for binary classification. Precision measures how accurate the predictions are. It is given by the fraction of the predictions that are correct:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{all detections}}. \tag{2.2}$$

Recall measures the ability to find all relevant instances (objects). It is given by the fraction of the ground-truth bounding boxes that are correctly found:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{all ground truths}}. \tag{2.3}$$

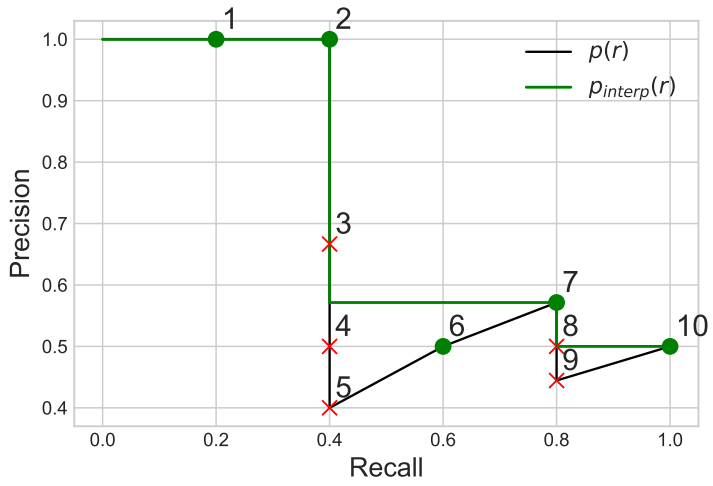
2.1.1 Average Precision

The main metric of interest in this work is Average Precision (AP), which is used in the MS COCO dataset [24]. It measures both the detection accuracy and the localisation performance. The first step in the computation of AP is determining true positives and false positives. This is done by matching each predicted bounding box with the ground-truth that it refers to (see Algorithm 1). The procedure matches each prediction with the ground-truth from the same category and the highest IoU overlap (Lines 3, 6). If a match is found, and the ground-truth was not already assigned, the detection is labelled as a true positive (Lines 7-9). If no match is found or if the matched ground-truth is already assigned to another detection (duplicate detection), it is labelled as a false positive (Lines 10-11).

Algorithm 1 COCO matching algorithm

```
 1: for all classes do
 2:     for all images do
 3:         compute IoU for all detections and ground-truths from the same class
 4:         sort dets by confidence score
 5:         for det in dets do
 6:             matched_gt ← arg max IoU(det, gts)
 7:             if not covered(matched_gt) then
 8:                 det ← true positive
 9:                 covered(matched_gt) ← True
10:             else
11:                 det ← false positive
12:             end if
13:         end for
14:     end for
15: end for
```

The following step is tracing the interpolated precision-recall curve, as shown in Figure 2.1. The curve $P(r)$ is drawn starting from the highest confidence detections and filling in the graph the point that corresponds to the current level of precision P at recall r. This curve is then made monotonically



#	Score	Type	Precision	Recall
1	0.95	**TP**	1	0.2
2	0.9	**TP**	1	0.4
3	0.85	FP	0.66	0.4
4	0.8	FP	0.5	0.4
5	0.75	FP	0.4	0.4
6	0.5	**TP**	0.5	0.6
7	0.4	**TP**	0.57	0.8
8	0.3	FP	0.5	0.8
9	0.2	FP	0.44	0.8
10	0.1	**TP**	0.5	1.0

Figure 2.1: Precision-recall curve for a class with five objects. The table shows ten detections made by the detector sorted by confidence score. For each detection, precision and recall are computed for the running set of detections and drawn in the graph: '$\times$' denotes false positives and '$\bullet$' denotes true positives.

decreasing by re-assigning the precision at each point as the maximum precision of any point with higher recall, by the equation:

$$P_{\text{interp}}(r) = \max_{\tilde{r} \geq r} P(\tilde{r}). \tag{2.4}$$

AP approximates the area under the interpolated precision-recall curve by averaging the interpolated precision at 101 equally spaced recall levels. It is computed for each class separately and at different levels of IoU threshold. The IoU threshold defines how close (i.e., how much overlap) a detection must be to the ground-truth to be considered true positive. For a given class C and IoU threshold t, it is given by

$$\text{AP}_C^t = \frac{1}{101} \sum_{r \in [0:.01:1]} P_{\text{interp}}(r, C, t). \tag{2.5}$$

The final metric for Average Precision is the average AP across the 80 object classes and at 10 different IoU levels,

$$\text{AP} = \frac{1}{10} \sum_{t \in [.50:.05:.95]} \frac{1}{80} \sum_{C \in \text{classes}} \text{AP}_C^t. \tag{2.6}$$

Averaging APs with increasing IoU thresholds rewards detectors with better localisation.

Mean Average Precision

The main metric of interest in this work is the Average Precision described before. Older datasets such as PASCAL VOC [25] use mean Average Precision (mAP)[1] as their metric of interest. The computation of mAP is similar to AP with small differences. The precision-recall curve is computed as in AP (Figure 2.1) with the difference that the IoU threshold is fixed at 0.5. Contrarily to AP, the approximated area under the precision-recall curve (Equation 2.5) is computed at 11 recall levels, instead of 101 levels. For a

[1] The COCO dataset omitted the mean from the mAP denomination.

given class C, it is given by

$$\text{AP}_C = \frac{1}{11} \sum_{r \in [0:.1:1]} P_{\text{interp}}(r, C). \tag{2.7}$$

The final mAP is the average AP across the 20 object classes of PASCAL VOC.

$$\text{mAP} = \frac{1}{20} \sum_{C \in \text{classes}} \text{AP}_C. \tag{2.8}$$

2.2 Datasets

The two main benchmarks in the object detection literature are the PASCAL Visual Objects Classes (VOC) dataset [25] and the Microsoft Common Objects in Context (COCO) [24] dataset. Both of these datasets feature pictures from objects found in "real life" scenarios, taken with consumer cameras and published online. Each one of these images was manually annotated with bounding boxes surrounding the relevant objects.

There are also other specialised datasets for specific domains or sub-fields of object detection, for example, pedestrian, traffic sign and aerial imagery datasets. General datasets such as VOC and COCO are the most appropriate for the study of context, given the diversity of environments in their database.

PASCAL Visual Object Classes

The PASCAL Visual Object Classes (VOC) [25] was one of the first standard datasets for object detection containing a substantial number of images. It has evolved through the years by adding new classes and samples but is now deprecated. The last updated version from 2012 contains 11,530 images with 27,450 annotated objects.

The annotations are divided into 20 object classes grouped in four main categories:

- **Person**: person;
- **Animal**: bird, cat, cow, dog, horse, sheep;
- **Vehicle**: aeroplane, bicycle, boat, bus, car, motorbike, train;
- **Indoor**: bottle, chair, dining table, potted plant, sofa, tv/monitor.

Microsoft COCO: Common Objects in Context

The Microsoft COCO dataset [24] is the current standard benchmark for object detection. This dataset focus on covering "non-iconic views" of objects, with objects photographed from diverse perspectives. The images are more crowded and objects are found off-center, in uncommon poses and with several occlusions. These properties make this a challenging dataset, but also the ideal dataset for exploring the use of context.

There are 80 object categories split into 12 supercategories:

- **Person**: person;
- **Vehicle**: bicycle, car, motorcycle, airplane, bus, train, truck, boat;
- **Outdoor**: traffic light, fire hydrant, stop sign, parking meter, bench;
- **Animal**: bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe;
- **Accessory**: backpack, umbrella, handbag, tie, suitcase;
- **Sports**: frisbee, skis, snowboard, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket;
- **Kitchen**: bottle, wine glass, cup, fork, knife, spoon bowl;
- **Food**: banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake;
- **Furniture**: chair, couch, potted plant, bed, dining table, toilet;
- **Electronic**: tv, laptop, mouse, remote, keyboard, cell phone;
- **Appliance**: microwave, oven, toaster, sink, refrigerator;
- **Indoor**: book, clock, vase, scissors, teddy bear, hair drier, toothbrush.

The 2017 version of the dataset contains 118,000 training images (`train2017`), 5,000 validation images (`val2017`) and 41,000 testing images (`test-dev2017` and `test-challenge2017`). `train2017` is used to train models, `val2017` is used to do early stopping and model selection, `test-dev2017` is used for reporting results, and `test-challenge2017` is used for challenge submissions. The main metric is AP, but there are 11 other metrics of interest: Average Precision and Average Recall at three different IoU thresholds and for three different object sizes.

2.3 Architectures

2.3.1 Multi-stage Architectures

In this section, we present an overview of recent multi-stage detectors that have been state-of-the-art in object detection for the past years. In particular, we focus on region-based convolutional neural network (R-CNNs). These detectors base their detections on a two-staged process: collecting region proposals and then classifying the object inside the regions with the use of a convolutional neural network. This multi-stage approach is slower than single-stage methods (Section 2.3.2) but produces detectors that are more accurate at classification and localisation.

Regions with CNN Features - R-CNN

Regions with CNN features or R-CNN [1] was one of the first approaches to use deep CNNs for object detection. These models showed a dramatic increase over traditional methods: they improved the best result on the PASCAL VOC 2012 challenge by more than 30%, achieving a mAP of 53.3%. These improvements confirm the power of deep CNNs to generate rich feature descriptors for image regions.

Architecture The architecture is composed of three steps: select class-agnostic region proposals through a region proposal method [26–28]; warp each image proposal to a fixed dimension and use a deep CNN (ConvNet) to extract a fixed-length feature vector; predict the object class and bounding box from the feature vector. Figure 2.2 illustrates the architecture of the detector.

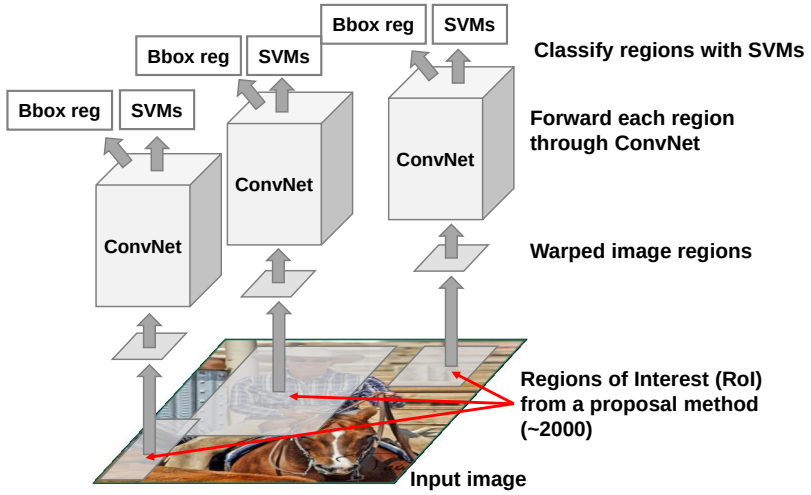


Figure 2.2: R-CNN detection architecture (extracted from [1]).

Regions of Interest Region of interest are extracted using a region proposal method, in this case, selective search [27], which is a fixed algorithm (no learning at this stage). Selective search starts by generating a sub-segmentation of the image, and then recursively groups regions with similar textures. This method extracts a set of approximately 2000 image patches that are interpreted as class-agnostic region proposals.

Feature Extraction Each region proposal is then warped into a 227×227 RGB image and passed through a deep CNN (backbone). The backbone network for the proposed implementation was a variant of AlexNet [11] followed by five convolutional layers and two fully connected layers to extract a 4096-dimensional feature vector for each proposal.

Object Classifiers and Bounding Box Regression The feature vectors are then classified using a set of class-specific linear SVM classifiers (Figure 2.2). To improve localisation performance, a bounding box regression stage is used. After classifying each proposal, a new bounding box is generated using a class-specific bounding box regressor. Given the feature vector and the original proposal coordinates, the regressor predicts a new location for the bounding box.

Training Transfer learning is performed during training. The backbone network is pre-trained in a large auxiliary dataset for image classification (ILSVRC2012 [12]). The network is then fine-tuned for the task of object detection during training using stochastic gradient descent.

Limitations:

- The CNN backbone needs to be evaluated for each of the 2000 proposals, therefore detection and training are very slow (using the author's setup, about 47s per image and 84h training[2]).
- The region proposal method is a fixed algorithm, i.e., no learning at that stage.
- Regions of interest are warped, distorting the objects in the image.
- The model is not trained end-to-end - the CNN, SVMs and bounding box regressors are trained separately using different loss functions.
- No use of context to improve detection - region proposals are classified independently.

Fast R-CNN

Fast R-CNN [2] builds on R-CNN to more efficiently classify object proposals. By sharing the costly CNN computation between proposals, execution becomes much faster, hence the name Fast R-CNN. Compared to R-CNN, it not only trains 9x faster and is 213x faster in test-time but also improves detection accuracy. It also introduces a new multi-task loss function, allowing for the network to be trained end-to-end.

Architecture A Fast R-CNN network takes as input an image and a set of object proposals, called regions of interest (RoI), from a region proposal method. The network first processes the whole image through a deep CNN to produce a convolutional feature map. Combining the regions of interest and the feature map, the RoI pooling layer extracts a fixed-length feature vector for each proposal.

This feature vector is then processed by a sequence of fully connected layers that branch into two output layers: a softmax classification layer (replaces R-CNN's SVM classifiers) that produces $K+1$ estimate class probabilities (K object classes plus the "background" class), and a bounding box regression layer that predicts the bounding box coordinates for each of the K classes. The architecture of this network is illustrated in Figure 2.3.

Regions of Interest Regions of interest are extracted from the superposition of the region proposals with the convolutional feature map. The RoI pooling layer is used to warp the variable size RoIs into a small feature map of a fixed size of $H \times W$ (in this case 7×7). Pooling is performed by dividing each $h \times w$ RoI window into a $H \times W$ grid and then max-pooling the values for each grid cell.

Training As in R-CNN, the backbone network is pre-trained for image classification using the ImageNet dataset. Fast R-CNN introduces a multi-task loss function that accounts both for classification and localisation that enables to jointly fine-tune the softmax layer and the bounding box regressor.

The output layer is composed of two networks: the first outputs a probability distribution (per RoI) over the $K+1$ object classes, $p = (p_0, p_1, \ldots, p_K)$, the second outputs a bounding box parameterisation

[2]These numbers were reported using hardware from the time the article was published. Using current hardware these times should be shorter.

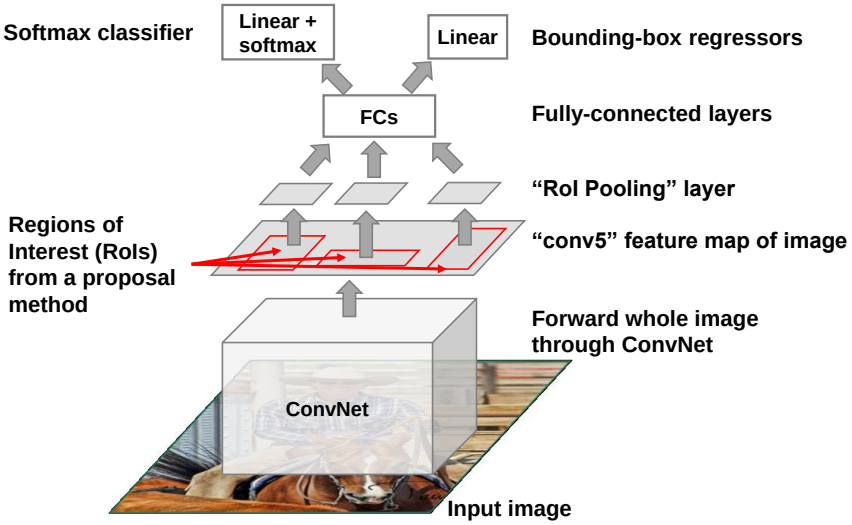


Figure 2.3: Fast R-CNN detection architecture (adapted from [2]).

for each of the K classes given by $t^k = (t^k_x, t^k_y, t^k_w, t^k_h)$. Each training RoI is associated to a ground-truth class u and bounding box v. The multi task loss is defined as:

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v), \tag{2.9}$$

in which $L_{\text{cls}} = -\log p_u$ is the log loss for the ground-truth class u and $L_{\text{loc}}(t^u, v)$ is the localisation loss given by

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t^u_i - v_i), \tag{2.10}$$

where

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise.} \end{cases} \tag{2.11}$$

The smooth_{L_1} function or Huber loss is a L_1 loss function that is less sensitive to outliers than the L_2 loss. The Iverson bracket function in Equation 2.9, $[u \geq 1]$, is used to activate the localisation loss: it evaluates to 1 when the target class is an object and 0 when it corresponds to no object/background. The hyperparameter λ weights the two loss functions and is set to $\lambda = 1$.

Limitations:

- Detection is still slow: even though inference testing takes only 0.3 seconds, the region proposal method takes 2 seconds per image and is the bottleneck of the operation.
- The region proposal method is a fixed algorithm, i.e., no learning at this stage.
- No use of context to improve detection - region proposals are classified independently.

Faster R-CNN

Faster R-CNN [3] makes Fast R-CNN even faster by the introduction of a Region Proposal Network (RPN) that sits on top of the convolutional feature map and learns to detect object regions. With this RPN, the use of an external region proposal method is avoided, which was the bottleneck of Fast R-CNN. This way, computing proposals becomes nearly free (about 10 ms per image). By merging RPN with Fast R-CNN we get Faster R-CNN, a faster and more accurate detector.

Architecture The architecture of Faster R-CNN is similar to the one of Fast R-CNN (see Figure 2.4). The network takes as input an image and propagates it through a CNN to produce a feature map of the image. At this stage, the feature map is passed to the RPN that generates the region proposals. Using the feature map and the generated proposals, RoI pooling is performed to produce a fixed-length feature vector that is then passed to the softmax classifier and bounding box regressors as described for Fast R-CNN.

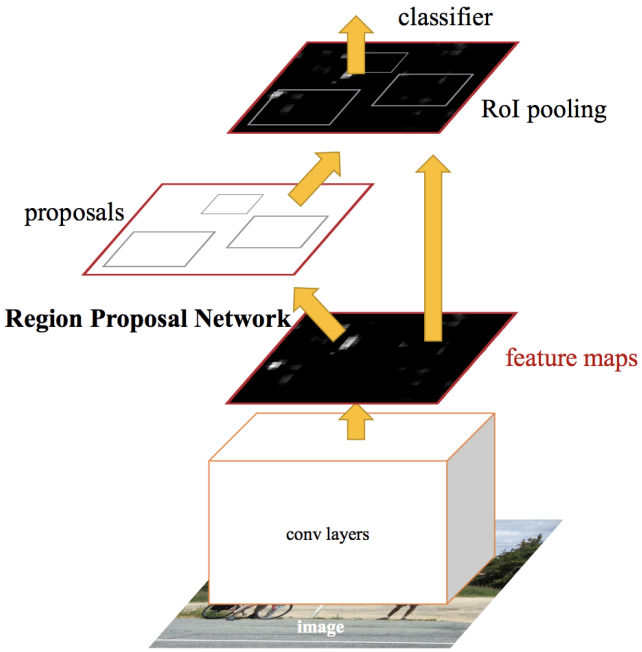


Figure 2.4: Faster R-CNN detection pipeline (extracted from [3]).

Regions of Interest Regions of Interest (RoI) are generated by the Region Proposal Network (RPN). RoIs are generated by sliding the RPN over the convolutional feature map, where the input of the RPN is a $n \times n$ window of the feature map. At each sliding-window position, the RPN it proposes k *anchors*. *Anchors* are reference bounding boxes that are centered at the current sliding window position. In this case there are $k = 9$ *anchors* at three different scales and three aspect ratios as shown in Figure 2.5.

Each anchor is passed to two sibling fully-connected layers: a classification (cls) layer that predicts $2k$ "objectness" scores and a regression (reg) layer that predicts $4k$ bounding box coordinates. For a feature map of size $W \times H$ there are in total $W \times H \times k$ anchors.

In order to reduce the number of high overlap proposals, non-maximum suppression (NMS) is applied: a region is removed if any other region with high overlap (IoU ≥ 0.7) has a higher score. Then, the top 2000 proposals are selected from the total region proposals.

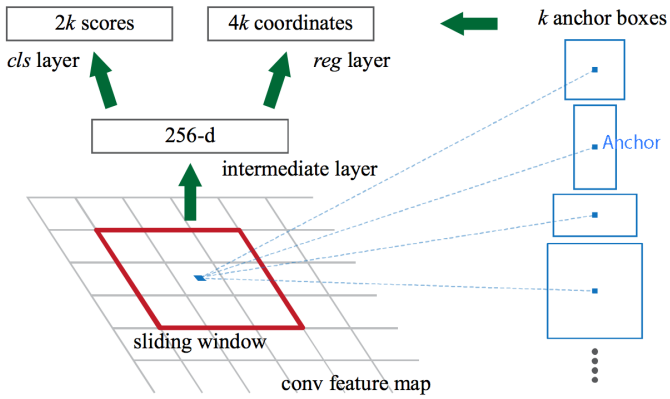


Figure 2.5: Architecture of the Region Proposal Network (extracted from [3]).

Training The training of the whole network can be done in a unified way. The article proposes different methods for the training of the network, but highlight the 4-Step Alternating Training method:

1. Randomly initialise the layers of the RPN and initialise the shared convolutional layers with a pre-trained model for ImageNet classification and then fine-tune the RPN for region proposal.
2. Train a separate Fast R-CNN network using the proposals from the step-1 RPN. At this point the two networks do not share convolutional layers.
3. Join the two networks by sharing the convolutional network. Freeze the shared convolutional layers and fine-tune the RPN.
4. Fine-tune the unique layers of Fast R-CNN while keeping the other layers frozen.

The RPN is trained to minimise the multi-task loss function,

$$L(\boldsymbol{p}, \boldsymbol{t}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*). \tag{2.12}$$

The index i is the index of the anchor in the mini-batch and p_i is the predicted probability of the anchor i being an object and t_i are the coordinates of the predicted box. Each anchor is matched with a ground-truth box, $p_i^* \in \{0, 1\}$ indicates whether there is an association between the anchor and ground-truth and t_i^* are the coordinates of the associated ground-truth.

The classification loss L_{cls} is the log loss over two classes (object vs. not object) and the regression loss uses the smooth$_{L_1}(t_i - t_i^*)$ (Equation 2.11). The p_i^* term activates or deactivates the regression depending if the predicted is associated to a ground-truth. Each sum is normalised by the mini-batch size, N_{cls}, and the number of anchor locations, N_{reg}, and balanced by the hyper-parameter $\lambda = 10$.

Limitations:

- No use of context to improve detection - region proposals are classified independently.

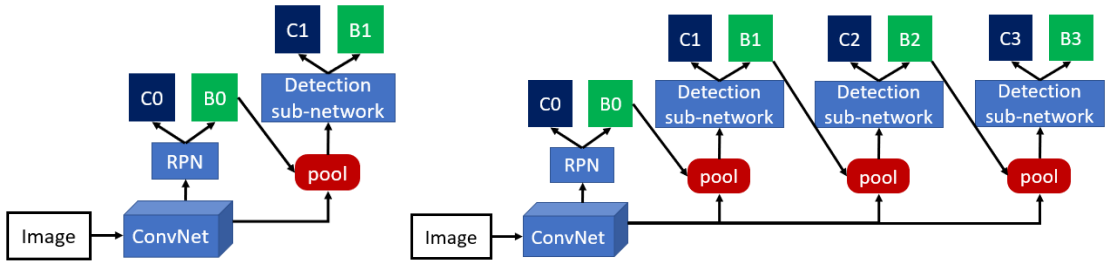
Cascade R-CNN

Cascade R-CNN [4] proposes a multi-stage solution to tackle the detection of "close" false positives, corresponding to bounding boxes whose location is close to the object but not correct. A detector must find the true positives and reject the false negatives. According to the article, simply increasing the detection threshold degrades performance. The proposed solution addresses this by using a sequence of detectors that iteratively refine region proposals, where each cascade stage specialises in bounding boxes in a certain IoU threshold.

Cascade R-CNN can be implemented on top of any two-stage detector based on R-CNN and consistently improves AP by 2 to 4 points, while only adding a marginal computational cost - the bottleneck of the operation is still the CNN feature extraction.

Architecture The proposed detector extends the two-stage architecture of Faster R-CNN as shown in Figure 2.6. In the first stage, region proposals ("B0") are generated using a Region Proposal Network. In the second stage, RoI pooling is done using "B0" and the convolutional feature map and then processed by a detection sub-network to produce a class ("C1") and bounding box ("B1") for each proposal. This would complete the architecture of Faster R-CNN. To extend it, two new detection stages are added where each additional detection stage is specialised in the IoU distribution of the previous stage. For the case where the cascade was trained with IoU thresholds of $U = \{0.5, 0.6, 0.7\}$, the first stage is trained to refine bounding boxes with bad localisation IoU ≥ 0.5 (regions may contain a lot of background), the second specialises in bounding boxes with IoU ≥ 0.6 and the third stage with IoU ≥ 0.7 (regions contain mostly the object).

Proposals are progressively refined with specialised regressors, improving localisation. The same refinement is done for class scores where the latter stages specialise in regions with better localisation. The paper shows that this cascaded refinement is superior to an approach that uses a single bounding box regression stage. The main difference is that each stage specialises at an IoU level, whereas in the previous case a single stage is used to refine regions at all levels of IoU threshold.



(a) Faster R-CNN (b) Cascade R-CNN

Figure 2.6: Cascade R-CNN adds a series of new detection sub-networks specialised at increasing IoU thresholds. "C" is classification, "B" is bounding box regression, "pool" is RoI pooling (extracted from [4])

2.3.2 Single-stage Architectures

The previously discussed multi-stage approaches have limitations in terms of execution time. These detectors come short in providing a real-time solution capable of detecting at a real-time frame-rate of 30 fps or higher. Single-stage detectors use simpler approaches that require less computation and therefore can be used in applications where real-time detection is needed.

In this section, we describe a popular real-time detector, You Only Look Once (YOLO).

You Only Look Once - YOLO

You Only Look Once (YOLO) [23] reframes the object detection problem as a single regression problem from image pixels to bounding box coordinates and the corresponding class probabilities. It is much faster than R-CNNs because it directly predicts bounding boxes and class labels with a single forward-pass of the network. Because of this simple implementation, it can be optimised end-to-end for detection performance. This architecture is extremely fast, capable of processing images at45 fps on the base implementation, with a lighter backbone network it is capable of up to 155 fps.

YOLO implicitly includes context in the detection, because each prediction is a function of the whole image pixels. This contrasts with R-CNNs, that look only to local image patches. Although this low-level context in incorporated in this architecture, we believe that the system can benefit from the use of higher-level contextual information by explicitly adding a context mechanism to capture this context.

Architecture All the components of the detector are unified into a single neural network. The first stage of the detector divides the input image into a $S \times S$ grid. If the center of an object falls into a cell, that cell is responsible for detecting the object. Each grid cell predicts $B = 2$ bounding box coordinates and their corresponding confidence score. The confidence score is interpreted as the probability that the box contains any object and how accurate is the box (see Figure 2.7).

Each bounding box regression consists of five predictions: x, y, w, h, and the confidence score. The coordinates $(x, y) \in [0, 1]$ are the coordinates of the center of the box, relative to the bounds of the grid cell. Predictions $w, h \in [0, 1]$ are the width and height of the box relative to dimensions of the whole image. Confidence is formally defined as $P(\text{Object} = 1) \times \text{IoU}(\hat{B}, B^*)$: in case there is no object, the confidence should be 0; in case there is an object, the confidence score should be IoU between the ground-truth B^* and the prediction $\hat{B}$. Each grid cell also predicts a set of C class probabilities, $P(\text{Class}_i|\text{Object} = 1)$.

During inference, the class-specific scores for each box is given by

$$P(\text{Class}_i|\text{Object} = 1) \times P(\text{Object} = 1) \times \text{IoU}(\hat{B}, B^*) = P(\text{Class}_i, \text{Object} = 1) \times \text{IoU}(\hat{B}, B^*). \quad (2.13)$$

That is, multiplying the confidence score for each box by the class probability of their corresponding grid cell. The result measures the probability of the box containing an object from each class and how well localised is the bounding box. The final product is a $S \times S \times (B \times 5 + C)$ tensor.

By thresholding and applying non-maximum suppression, only the best detections are selected.

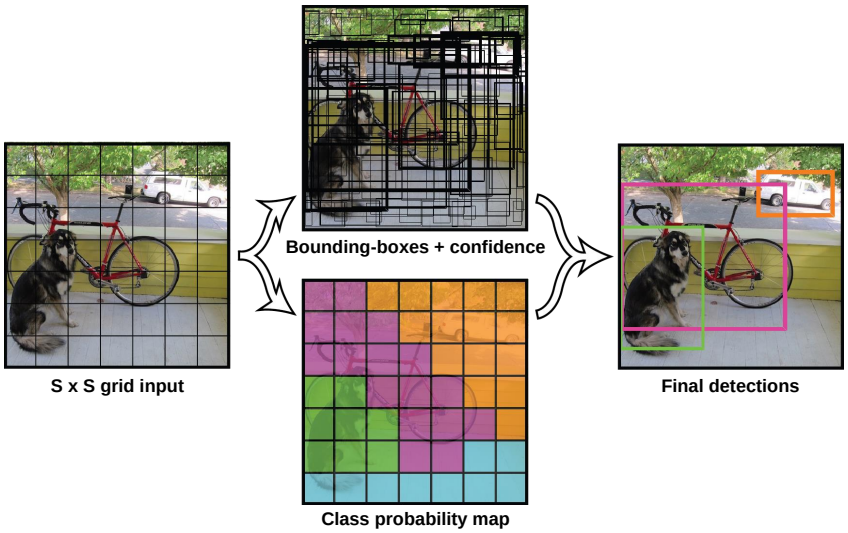


Figure 2.7: YOLO detection model. For the published PASCAL-VOC implementation: grid size $S = 7$, number of boxes per cell $B = 2$, number of classes $C = 20$. The final prediction is a $7 \times 7 \times 30$ tensor, containing 98 proposed boxes. In comparison, the implementations of R-CNN described previously use a total of 2000 region proposals (extracted from [23]).

Training The backbone network is pre-trained for ImageNet classification. After that, the fully connected layers are added and the network is trained to optimise the multi-task loss function, that accounts for localisation, confidence, and classification:

$$L = L_{\text{loc}} + L_{\text{conf}} + L_{\text{cls}}, \tag{2.14}$$

where

$$L_{\text{loc}} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + \left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right], \tag{2.15}$$

$$L_{\text{obj}} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \left[\mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 + \lambda_{\text{noobj}} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \right], \tag{2.16}$$

$$L_{\text{cls}} = \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left[(p_i(c) - \hat{p}_i(c))^2 \right]. \tag{2.17}$$

The terms $\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i, \hat{C}_i$, and $\hat{p}_i(c)$ are the predictions and x_i, y_i, w_i, h_i, C_i and $p_i(c)$ are the corresponding ground-truth. The term $\mathbb{1}_i^{\text{obj}}$ denotes if an object appears in cell i and $\mathbb{1}_{ij}^{\text{obj}}$ denotes that the jth bounding box of cell i is responsible for that prediction, therefore only activating the localisation, confidence and classification errors if there is an object to detect. The weights $\lambda_{\text{coord}} = 5$ and $\lambda_{\text{noobj}} = .5$ balance the loss function by increasing the loss for localisation and decreasing the loss for the confidence of non objects.

Error analysis The article publishes an error analysis [29] and compares it with Fast R-CNN in Figure 2.8. The analysis shows that YOLO's main source of error is localisation - using region proposals and proposal refinement shows improvement. On the other hand, Fast R-CNN is more prone to make more confusions with background.

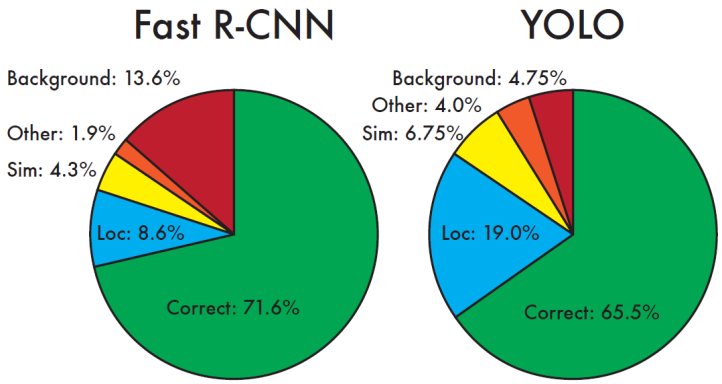


Figure 2.8: Error analysis: Fast R-CNN vs. YOLO (extracted from [23]).

Limitations:

- Because each grid cell only predicts two bounding boxes ($B = 2$), the model is limited on the number of nearby objects in crowded images.
- Struggles to precisely locate objects, especially small ones.
- Struggles to generalise to objects in new unusual aspect ratios.

2.4 Context in Object Detection

There is not a clear definition for context in computer vision. In [30] it is defined as "any and all information that may influence the way a scene is perceived". Under this definition, none of the previously described approaches explicitly incorporates context in their detection pipeline, other than through visual information. During the classification stage of the detection, each region proposal is evaluated independently, not accounting for any correlation between object classes, object location or any other potential source of context.

In this section, we describe recent work that lists sources of context for object detection and the proposed approaches to incorporate these sources of context.

2.4.1 An Empirical Study of Context in Object Detection

In [16] the authors summarise the sources of context that have been used for object detection and propose an approach that learns to predict object presence, location and size from these contextual cues.

The listed sources of context are:

- Local Pixel Context: pixel information surrounding the object.
- 2D Scene Gist Context: global image statistics that capture the gist of the image.
- 3D Geometric Context: information from the 3D scene layout, occlusions, orientations, support surfaces, and contact points.
- Semantic Context: the type of scene/environment being depicted, the presence of other objects and their location.
- Photogrammetric Context: the specific characteristics of the camera used (height, orientation, focal length, etc).
- Illumination Context: sun direction, sky colour, cloud cover, indoor illumination.
- Weather Context: weather conditions, temperature, season, etc.
- Geographic Context: GPS location, terrain type, landscape, population density.
- Temporal Context: time of capture, nearby frames (in video).
- Cultural Context: photographer bias, framing and focus.

Approach

The authors propose an approach that chooses an off-the-shelf local detector and augments it by adding three contextual classifiers. The three classifiers are trained to predict the object presence, location and object size from the contextual cues listed above. The classifiers are then combined with the local detector to predict a score for each detection.

Object Presence Objects tend to appear in typical environments: object presence can be estimated using 2D scene context (using low-level image features), geometric context (using a geometric class score - ground, left, right, center, sky, solid, porous), semantic context (co-occurrence with other objects), geographic context and semantic context.

To include geographic and semantic context, the authors use an auxiliary dataset with densely annotated images. Annotations contain keywords describing the geographic location (urban/rural, zoo, seaside, etc) and object presence (animals, vehicles, monuments, etc). When classifying an image, they query this dataset for the 80 closest images and extract their keywords. These keywords are then used to build a geographic and semantic descriptor of the image.

For each of these contextual descriptors, a logistic regression classifier is trained to predict the likelihood of the presence of each object class.

Object Location Object location can be inferred from the whole scene gist and the geometric context. Using the two descriptors, the authors train logistic regression classifiers to predict where the object(s) are likely to be in a 5×5 grid.

Object Size The idea is to predict the object height in the image, given its location. Using scene gist and geometric context descriptors, five different classifiers are trained to predict the probability of the

object belonging to one of five size classes. From the predicted object size, the bounding box width and height can be estimated.

Combining Contexts The different sources of context are then combined using a logistic regression classifier to predict the confidence of an object detection in the image.

Results

This contextual approach, the performance on the PASCAL VOC08 test set is boosted from 18.2 to 22.0 mAP. Not only the performance is improved, but also the errors made by the detector are more reasonable (most object confusions occur with objects that share similar contexts - e.g. cats and dogs, airplanes and birds). It is also shown that context can help in situations where object appearance is degraded (small objects, unusual poses or occlusions).

A comparison between the sources of context shows that, for this approach, the object size context is the strongest source, while the object location is the weakest. It must be noted that the results cannot be directly compared to the official results because geographic and semantic context use information from an external augmented dataset.

2.4.2 Other Work on Context

Hierarchical Context [15] Hierarchical Context proposes a rescoring approach that assumes a hierarchical relationship between object classes. Object co-occurrences and spatial dependencies are modelled with a tree graphical model, where each node represents the presence of an object class in the image. The model learns co-occurrence and spatial priors between object classes. Predictions are made by performing inference in the tree graphical model.

Inside-Outside Net [18] The proposed Inside-Outside Net (ION) expands the typical RoI approach of Fast R-CNN to include two additional sources of information: skip-layer connections and spatial recurrent neural networks (RNNs). Skip-layer connections extract features from deeper layers in the CNN and combine them with the RoI pooling feature map. Spatial RNNs aggregate contextual information by sweeping the feature map from the four directions (left/right/top/bottom), to create a new feature map that aggregates information from the vicinities of each cell.

Structure Inference Net [19] This approach extends a Faster R-CNN architecture to include a Structure Inference Net (SIN). A SIN models object co-occurrences with a graphical model, in which region proposals are the nodes and the directed edges represent the influence an object class has on another. The SIN is placed on top of the RPN of Faster R-CNN and extracts region proposals from the RoI pooling layer to build the graphical model.

Context Refinement [14] This approach extends a Faster R-CNN architecture with a context module to refine classification and bounding box regression. Contrarily to the Fast R-CNN approach that uses only the local feature map, Context Refinement identifies surrounding regions that may carry useful contextual information and aggregates them into a context vector. The context vector is combined with the convolutional feature vector and used to produce an object class score and a bounding box prediction.

2.4.3 Our Contributions

Our work contributes to the field by exploring the use of recurrent neural networks with attention to rescore sequences of detections (see Chapter 3). These models have implicit and explicit mechanisms that are able to capture contextual information in sequences, having achieved state-of-the-art results in tasks such as neural machine translation [20, 21] and image captioning [22]. We use these models to process sequences of detections in an image and use these mechanisms to capture the image's semantic (object presence) and geometric context (object size and location).

Chapter 3

Recurrent Neural Networks and Attention Mechanisms

Our proposed model (see Section 5.1) uses a recurrent neural network encoder to rescore sequences of detections. Recurrent neural networks (RNNs) are a class of neural networks for processing sequential data. In this chapter, we describe RNNs and introduce two popular RNN variants: the long short-term memory (LSTM) and the gated recurrent unit (GRU). In the following section, we describe attention mechanisms that capture long range dependencies in sequences.

3.1 Recurrent Neural Networks

Recurrent neural networks (RNN) [31] are a class of neural networks used to process sequences of inputs. Unlike feedforward neural networks, RNNs are stateful models, they have an internal state - that can be interpreted as their internal memory - that is updated at every step. Figure 3.1 illustrates the state transition model that can be "unfolded" into a directed graph. This internal state representation can then be used to make a prediction about the processed sequence.

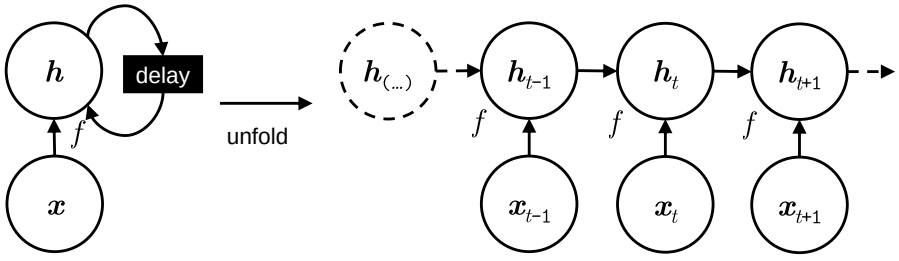


Figure 3.1: The recurrent graph and the unfolded graph of a RNN. The delay block represents the persistence of the state $\boldsymbol{h}_t$ from one element to the following (adapted from [32]).

The RNN's hidden state of element t is a fixed length vector $\boldsymbol{h}_t$ and is a function f of previous state $\boldsymbol{h}_{t-1}$ and current input $\boldsymbol{x}_t$,

$$\boldsymbol{h}_t = f\left(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t\right). \tag{3.1}$$

We can derive that $\boldsymbol{h}_t$ is a function of initial state $\boldsymbol{h}_0$ and the past sequence of inputs $\boldsymbol{x}_{1:t}$:

$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t) = f(f(\boldsymbol{h}_{t-2}, \boldsymbol{x}_{t-1}), \boldsymbol{x}_t) = \cdots = f^{(n)}(\boldsymbol{h}_0, \boldsymbol{x}_{1:t}). \tag{3.2}$$

A simple RNN model has

$$f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t) = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t. \tag{3.3}$$

During training, the network learns the set of parameters of function f and, optionally, the initial state $\boldsymbol{h}_0$. The learnt parameters are shared between all the elements in arbitrarily sized sequences, instead of learning a separate model for each element. This recurrent structure results in a very deep computational graph from a simple model f.

Bidirectional RNNs

The described RNNs model only causal dependencies, where the current output does not depend on future inputs. In some cases, sequences that do not have a causal dependency and where the full sequence is available, we can use the whole sequence to predict the current output. A bidirectional RNN [33] achieves this by modeling the sequence both from the beginning to the end and from the end to the beginning. Figure 3.2 illustrates the computational graph of a bidirectional RNN composed of two RNNs: a forward RNN and a backward RNN. The resulting output $\boldsymbol{y}_t$ is then a function of past and future inputs,

$$\boldsymbol{y}_t = o(h(\boldsymbol{x}_{1:t}),\ g(\boldsymbol{x}_{t:L})) \tag{3.4}$$

where L is the sequence length, h is the forward update function, g is the backward update function, and o is the output function that combines both hidden states.

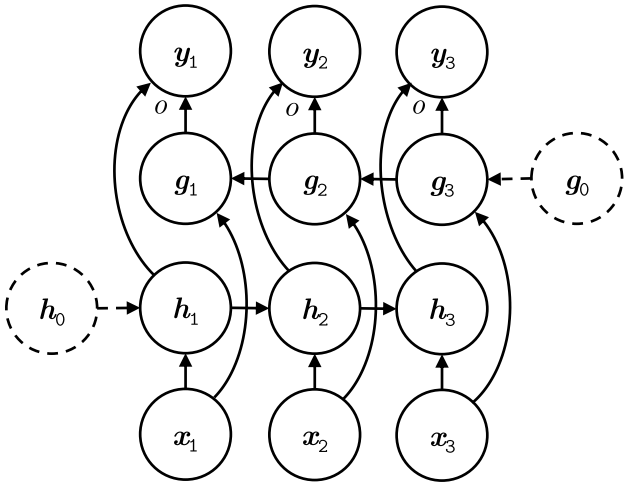


Figure 3.2: Computational graph of a bidirectional RNN processing a sequence of three inputs. The output sequence $\boldsymbol{y}$ combines both the forward RNN $\boldsymbol{h}$ and the backward RNN $\boldsymbol{g}$ (adapted from [32]).

Stacked RNNs

A limitation of RNNs is the difficulty in modelling complex dependencies. The hidden vector limits the amount of information that can be stored in the cell's state. One way of increasing model complexity is increasing the dimensionality of the feature vector. Another way is to add a new layer to the model by stacking another RNN [34] on top of the existing one. Stacking is performed by connecting the RNN's hidden layers to the inputs of another RNN, as illustrated in Figure 3.3.

The intuition behind stacking is that different layers of the RNN learn different patterns of the input at different levels of abstraction. Formally, element $\boldsymbol{y}_t$ of the output sequence is given by

$$\boldsymbol{y}_t = o\left(\boldsymbol{h}_t^{(2)}\right) = o\left(f_2\left(\boldsymbol{h}_{t-1}^{(2)},\, \boldsymbol{h}_t^{(1)}\right)\right) = o\left(f_2\left(\boldsymbol{h}_{t-1}^{(2)},\, f_1\left(\boldsymbol{h}_{t-1}^{(1)},\, \boldsymbol{x}_t\right)\right)\right) = f\left(\boldsymbol{x}_{1:t}\right). \tag{3.5}$$

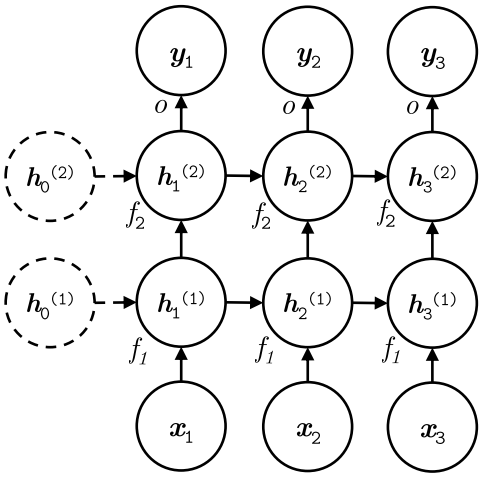


Figure 3.3: Computational graph of two stacked RNNs. The input of the second RNN is the hidden layer of the first RNN.

3.1.1 Long Short-Term Memory

Simple RNNs like the ones described earlier suffer from the vanishing/exploding gradient problem when training for long sequences. This problem occurs when applying the chain rule over a sequence of products. If gradients are small, when multiplied they tend to 0. Likewise, exploding gradients occur when gradients are large. This problem makes traditional RNNs inadequate for processing long sequences.

The long short-term memory (LSTM) [35] model is a RNN variant that solves this problem with a gated RNN architecture that makes this model resilient to vanishing gradients and suitable for processing long sequences. Results in NLP tasks such as speech recognition [36], machine translation [37] and image captioning [38] have shown that LSTMs are successful at in modelling long sequences.

The LSTM propagates two internal states: the cell state $\boldsymbol{c}$ and the output vector $\boldsymbol{h}$. At each point in the sequence, the LSTM cell has a set of gates with a value between 0 and 1 that control what information of the cell state is kept, updated or forgotten. These gates are a function of the current input $\boldsymbol{x}_t$, previous output vector $\boldsymbol{h}_{t-1}$, and the network's learnt parameters.

The model's equations are:

- **Candidate values:** creates a vector of candidate cell state values for the next state

$$\tilde{\boldsymbol{c}}_t = \tanh\left(\boldsymbol{W}_c \cdot [\boldsymbol{x}_t, \boldsymbol{h}_{t-1}] + \boldsymbol{b}_c\right). \tag{3.6}$$

- **Forget gate:** controls what information of the state to forget

$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}_f \cdot [\boldsymbol{x}_t, \boldsymbol{h}_{t-1}] + \boldsymbol{b}_f\right). \tag{3.7}$$

- **Input gate:** controls what information moves from the candidate values to the next cell state

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}_i \cdot [\boldsymbol{x}_t, \boldsymbol{h}_{t-1}] + \boldsymbol{b}_i\right). \tag{3.8}$$

- **Output gate:** controls what information outputs from the current cell state to the hidden vector

$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{W}_o \cdot [\boldsymbol{x}_t, \boldsymbol{h}_{t-1}] + \boldsymbol{b}_o\right). \tag{3.9}$$

- **Update cell state:**

$$\boldsymbol{c}_t = \boldsymbol{f}_t * \boldsymbol{c}_{t-1} + \boldsymbol{i}_t * \tilde{\boldsymbol{c}}_t. \tag{3.10}$$

- **Compute new output vector:**

$$\boldsymbol{h}_t = \boldsymbol{o}_t * \tanh(\boldsymbol{c}_t). \tag{3.11}$$

Where σ is a sigmoid function that squeezes the inputs to a value between 0 and 1, and the weights $\boldsymbol{W}_f, \boldsymbol{b}_f, \boldsymbol{W}_i, \boldsymbol{b}_i, \boldsymbol{W}_o, \boldsymbol{b}_o, \boldsymbol{W}_c, \boldsymbol{b}_c$ are the LSTM's parameters that are learned during training. Figure 3.4 illustrates the information flow from one state to the following.

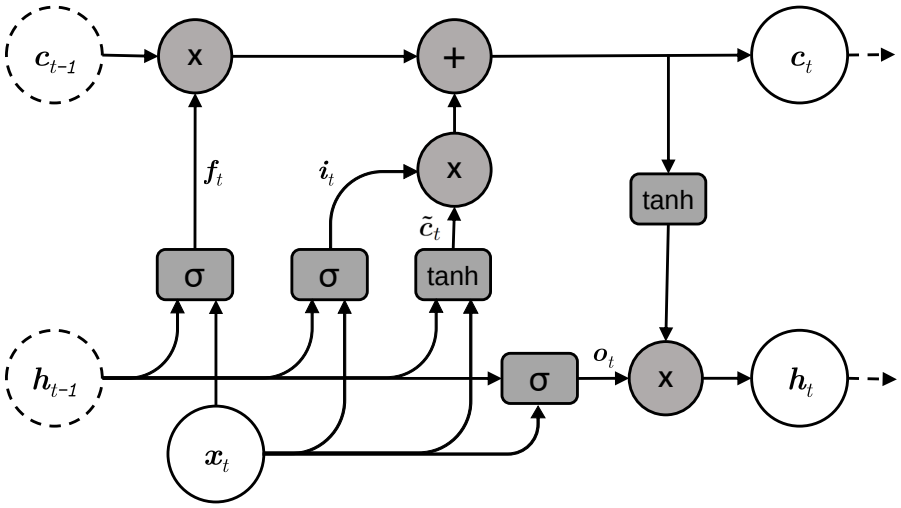


Figure 3.4: The computational graph of a LSTM cell (adapted from [39]).

3.1.2 Gated Recurrent Unit

Another popular variant of gated RNNs is the gated recurrent unit (GRU) [40], illustrated in Figure 3.5. The model is simpler than LSTM in the way that it uses only one hidden vector $\boldsymbol{h}$ and only two gates:

- **Update gate:** controls what information to update on the hidden state vector

$$\boldsymbol{z}_t = \sigma\left(\boldsymbol{W}_z \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t]\right). \tag{3.12}$$

- **Reset gate:** this gate controls what information to keep from the previous state

$$\boldsymbol{r}_t = \sigma\left(\boldsymbol{W}_r \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t]\right). \tag{3.13}$$

The candidate vector is given by

$$\tilde{\boldsymbol{h}}_t = \tanh\left(\boldsymbol{W}_h \cdot [\boldsymbol{r}_t * \boldsymbol{h}_{t-1}, \boldsymbol{x}_t]\right), \tag{3.14}$$

and the hidden state is updated by

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) * \boldsymbol{h}_{t-1} + \boldsymbol{z}_t * \tilde{\boldsymbol{h}}_t. \tag{3.15}$$

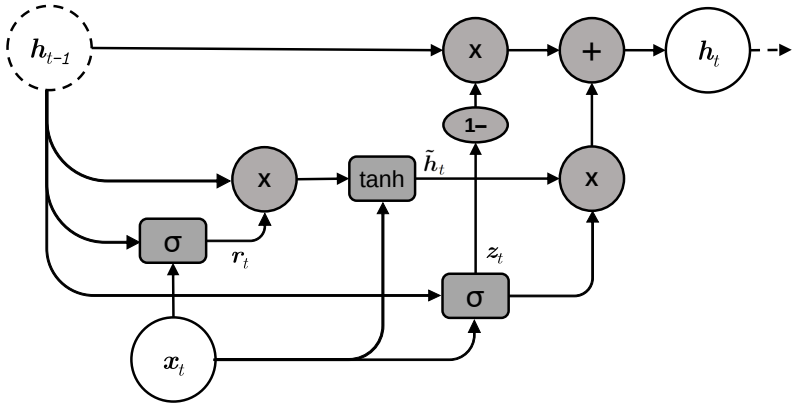


Figure 3.5: The computational graph of a GRU cell (adapted from [39]).

3.2 Attention Mechanisms

The performance of RNNs in modeling long term dependencies is still limited. It is difficult for a RNN to remember the first elements of the sequence after dozens have been processed. When making a prediction, certain distant elements in the sequence may be more relevant than the latest ones and useful state information may have been overwritten. Attention mechanisms focus on learning the correlations between elements in the sequence, regardless of their distance. Tasks such as machine translation [20, 41], language modelling [21] and image captioning [22] have benefited from these attention mechanisms to capture long term dependencies.

3.2.1 Self-Attention

The idea behind self-attention is to enrich the representation of the hidden state of the network with a context vector that encompasses the context from the whole sequence.

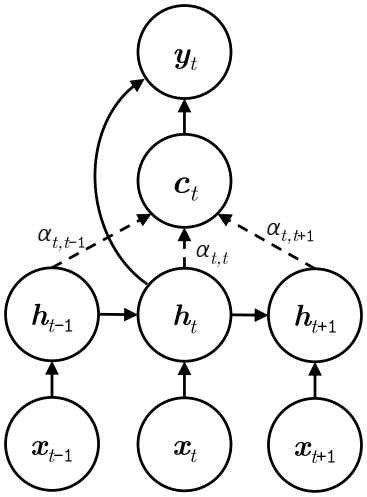


Figure 3.6: RNN with self-attention. The context vector $\boldsymbol{c}_i$ is the weighted average of the hidden vectors and the weights $\alpha_{t,j}$ are the alignment scores between $\boldsymbol{h}_t$ and $\boldsymbol{h}_j$.

For each element i in the sequence, the model computes an alignment score between each element and i. This alignment value represents how the two elements influence each other. As shown in Figure 3.6, the context vector $\boldsymbol{c}_i$ is given by the weighted average of all the hidden vectors in the sequence, weighted by the alignment score:

$$\boldsymbol{c}_i = \sum_{j=1}^{L} \alpha_{ij} \boldsymbol{h}_j, \tag{3.16}$$

where L is the sequence length, $\boldsymbol{h}_j$ is the hidden vector of element j, and α_{ij} measures the alignment between i and j. The weights α_{ij} result from a normalised scoring function:

$$\alpha_{ij} = \frac{\exp(\text{score}(\boldsymbol{h}_i, \boldsymbol{h}_j))}{\sum_{k=1}^{L} \exp(\text{score}(\boldsymbol{h}_i, \boldsymbol{h}_k))}, \tag{3.17}$$

where score$(\boldsymbol{h}_i, \boldsymbol{h}_j)$ is a scoring function that measures the alignment between $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$. Several alignment scoring functions have been proposed:

- dot product [41]: score$(\boldsymbol{h}_i, \boldsymbol{h}_j) = \boldsymbol{h}_i^\top \boldsymbol{h}_j$,
- scaled dot product [21]: score$(\boldsymbol{h}_i, \boldsymbol{h}_j) = \frac{\boldsymbol{h}_i^\top \boldsymbol{h}_j}{\sqrt{L}}$,
- general [41]: score$(\boldsymbol{h}_i, \boldsymbol{h}_j) = \boldsymbol{h}_i^\top \boldsymbol{W_a} \boldsymbol{h}_j$,
- additive [20]: score$(\boldsymbol{h}_i, \boldsymbol{h}_j) = \boldsymbol{v}_a^\top \tanh(\boldsymbol{W_a}[\boldsymbol{h}_i; \boldsymbol{h}_j])$.

The resulting context vector $\boldsymbol{c}_t$ is a representation of the context of the whole sequence and can be combined with the hidden vector $\boldsymbol{h}_t$ to predict the current output $\boldsymbol{y}_t$.

Chapter 4

Analysis

In the first section of this chapter, we analyse the most common errors made by object detectors, motivated by a quantitative error analysis of a state-of-the-art detector. Afterwards we motivate the use of context to address and mitigate these errors, namely the semantic context inferred by the presence of other objects.

In the second section, we explore how we can optimise Average Precision. Specifically, we propose three rescoring algorithms and compare them on AP. These algorithms take in a set of detections and produce the target score for each detection to be used during training.

4.1 Error Analysis

For this analysis we used an off-the-shelf implementation of Cascade R-CNN with a ResNet-101 [42] backbone from OpenMMLab's MMDetection toolbox [43], trained on COCO `train2017`[1]. All the presented detections along with their confidences are generated by running this detection architecture on COCO `val2017`.

4.1.1 Types of Errors

We identified the main types of errors: confusion with background, misclassifications, and duplicate detections.

Confusion with Background An object is incorrectly detected where there is no annotated object (see Figure 4.1). Often the object is correctly identified, but the annotation is missing because the object is in the background (annotations are inconsistent regarding background objects). The detections can also result from background clutter that resembles some object class.

[1] For more information about the implementation, please refer to the project's github page `https://github.com/open-mmlab/mmdetection/`

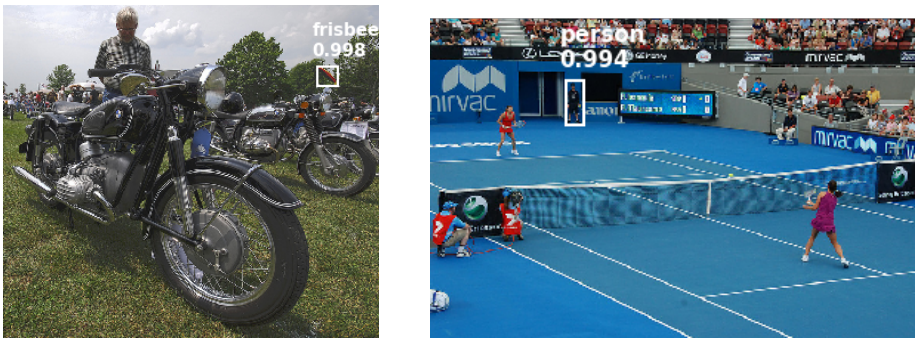


Figure 4.1: Top false positives - confusion with background. Left: a flag is confused with a frisbee. Right: the person in the background is correctly identified but is not on the annotations.

Misclassifications An object is classified in the wrong class (see Figure 4.2). The confusions often occur when the bounding box has poor localisation and there is another object close to the detected one (e.g., a person sitting on a couch).

Figure 4.2: Top false positives - misclassifications. Left: a surfer is confused with a dog. Right: an animal that is annotated as cow is confused with a sheep

Duplicate Detections Detected an object instance that was already detected. Detectors often predict several bounding boxes for the same object (see Figure 4.3). AP prioritises the detection with the highest confidence, all other detections are false positives. This occurs frequently even after applying non-maximum suppression.

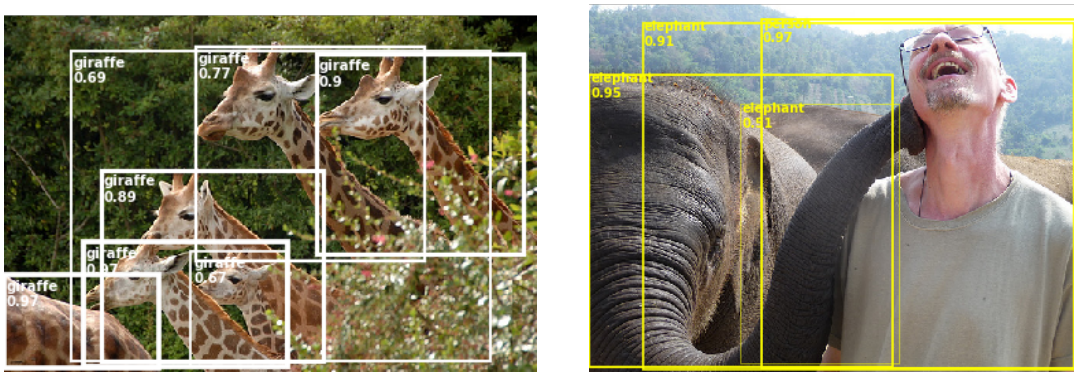


Figure 4.3: Top false positives - duplicate detections. Left: there are six giraffes but the detector finds seven. Right: the detector finds multiple instances of the same elephant.

Annotations are imperfect. The detector frequently recognises objects in the background that that were not annotated. Also, annotations are sometimes inconsistent: on a picture of a tennis match (Figure 4.4) with a crowd on the background, some spectators are annotated as 'person', others are not; on pictures with bookshelves in the background, some are densely annotated with 'book' and others ignore them. This inconsistent labelling on background object results in many wrong false positives due to insufficient annotations, and consequently degrading AP.

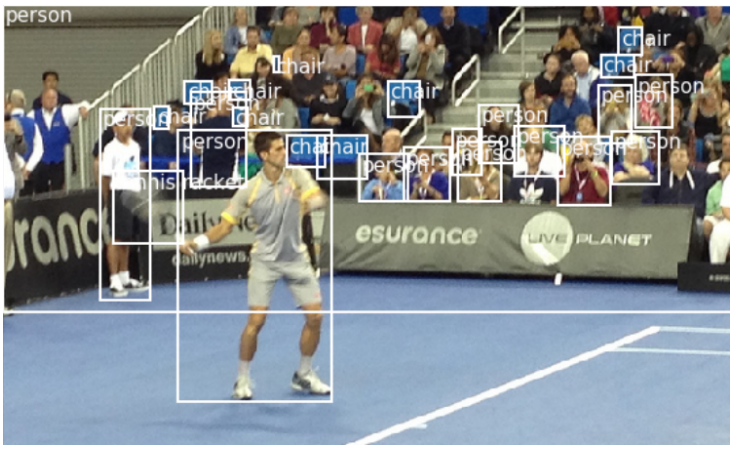


Figure 4.4: Missing annotations. The annotations for the spectators in the background are inconsistent.

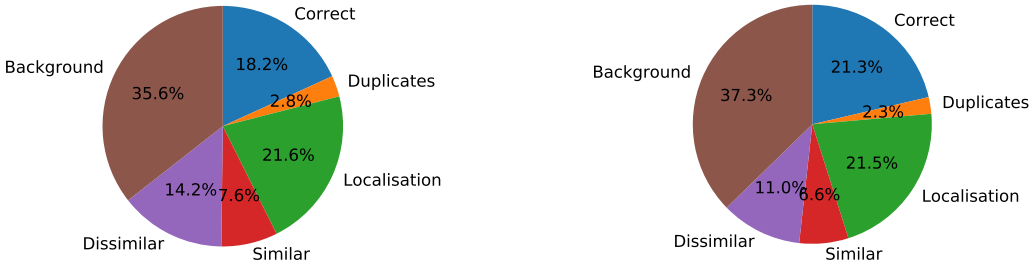
4.1.2 Quantitative Error Analysis

To diagnose the types of errors made by detectors we follow a similar approach to Hoiem et al., 2012 [29] that splits detection errors into four types. To this procedure, we add a fifth type: duplicate detections. In our analysis, to determine false positives, detections are matched with the ground-truth with the highest overlap, regardless of their category (unlike the matching done in the computation of AP that matches each class independently) and are evaluated (matched to a ground-truth) by descending confidence. The types of false positives are:

- **Duplicate detection:** the detection has the correct label and IoU ≥ 0.5, but the corresponding ground-truth was already assigned to another detection.
- **Localisation error:** the detection has the correct label but the bounding box is misaligned ($0.1 \leq$ IoU < 0.5). Many localisation errors are also duplicate detections with a small overlap.
- **Confusion with similar category:** the detection has a similar category and IoU ≥ 0.1. We assume that two category are similar if they belong to the same supercategory.
- **Confusion with dissimilar category:** the detection has a dissimilar category and IoU ≥ 0.1.
- **Confusion with background:** the remaining false positives with IoU < 0.1.

We count the number of false positives in each category and compare them in the pie charts of Figure 4.5. Most detections are false positives. Our rescoring approach aims at reducing the amount of confidence placed on erroneous detections and is unable to change the total number of correct/incorrect detections. AP can be greatly improved by reducing the amount of confidence placed on false positives,

as shown in Section 4.2.4. In Section 6.1.2, we do a similar analysis that compares the total accumulated confidence by error type instead of counting the number of detections.



(a) Faster R-CNN. (b) Cascade R-CNN.

Figure 4.5: Error distribution of Faster R-CNN and Cascade R-CNN. Background confusion and localisation errors are the most common errors made by both detectors.

Confusion Matrix

To better understand the common confusions made by the detector, we display in Figure 4.6 the confusion matrix. For better visualisation, we remove the diagonal of the matrix (correct classifications) because it would overshadow the misclassifications in the image. Each cell $C_{i,j}$ in the confusion matrix represents the probability of predicting the class i given that the object is from class j and the prediction is incorrect. Formally,

$$C_{i,j} = \begin{cases} P(X = i \mid Y = j) = \frac{P(X=i,Y=j)}{P(Y=j)} = \frac{\text{\#confusions between } i,j}{\text{\#instances of } j}, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases} \tag{4.1}$$

where X is the predicted object category and Y is the ground-truth object category.

4.1.3 How context can help with these errors?

Misclassifications/Confusion with Background - Class Co-occurrences

Some objects tend to appear in the same images as others. Using context, we can leverage the knowledge from these co-occurrences and decrease confidence in predictions of object classes that do not fit the depicted environment. We visualise these co-occurrences in the co-occurrence matrix in Figure 4.7. This diagram provides motivation to why even the integration of context based on class co-occurrences might help.

The matrix illustrates the expected number of instances of the co-occurrent class to be encountered in an image given that an instance from the observed class has been found. Formally, it is given by:

$$M_{i,j} = \mathbf{E}\left[X_i \mid Y_j \geq 1\right] = \frac{\text{\#instances of } i \text{ in images with } Y_j \geq 1}{\text{\#images with } Y_j \geq 1}. \tag{4.2}$$

Where X_i is the number of instances of the co-occurrent class i and Y_j is the number of instances of the observed class j in one image.

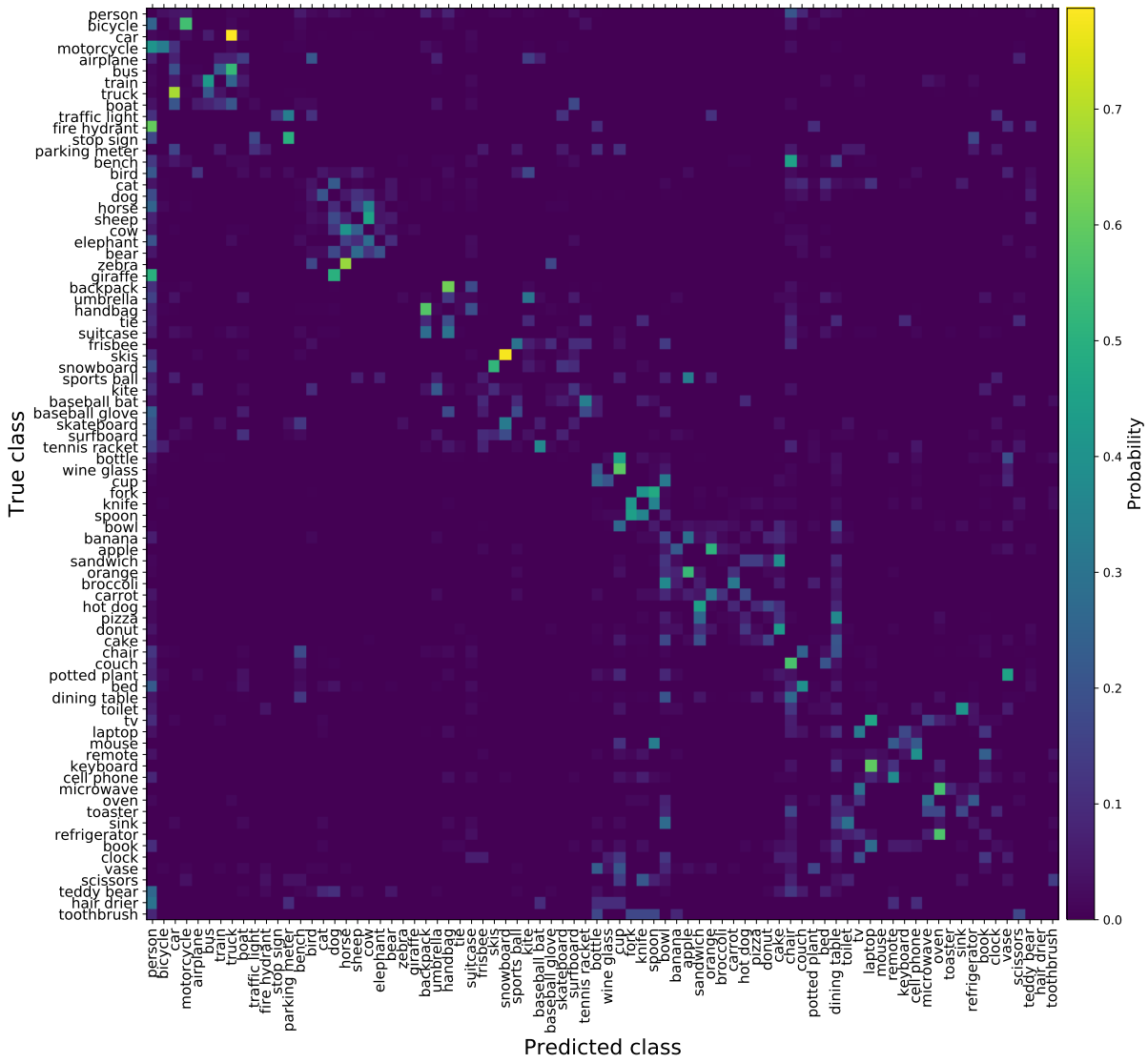


Figure 4.6: Confusion matrix (for ease of visualisation the diagonal is omitted). Confusions often happen between objects in the same supercategory - car/truck, bicycle/motorcycle, fork/knife/spoon, microwave/oven, bench/chair, vase/potted plant, cup/wine glass, etc - but there are also some unexpected confusions - fire hydrant/person, giraffe/person, apple/sports ball. Person is often wrongfully predicted, due to being the most common class and that it is often found in close proximity to other objects (e.g. a person sitting on a chair), where a poorly localised detection from the other object results in confusion with person.

Duplicate Detections/Localisation Errors - Non-Maximum Suppression

The typical mechanism to remove duplicate detections is to use non-maximum suppression. The procedure is a simple fixed algorithm, remove the bounding boxes whose confidence is lower than any of its "neighbours". A "neighbour" is any bounding box from the same object category that has an IoU above a certain threshold (typically 0.5).

Choosing the bounding box with the highest confidence is not always optimal with respect to AP. Figure 4.8 illustrates an example where the highest confidence detection is not the best option. Our rescoring model's target should not select the highest confidence detection but the one that has a higher IoU overlap. Section 4.2 explores what rescoring approach maximises AP.

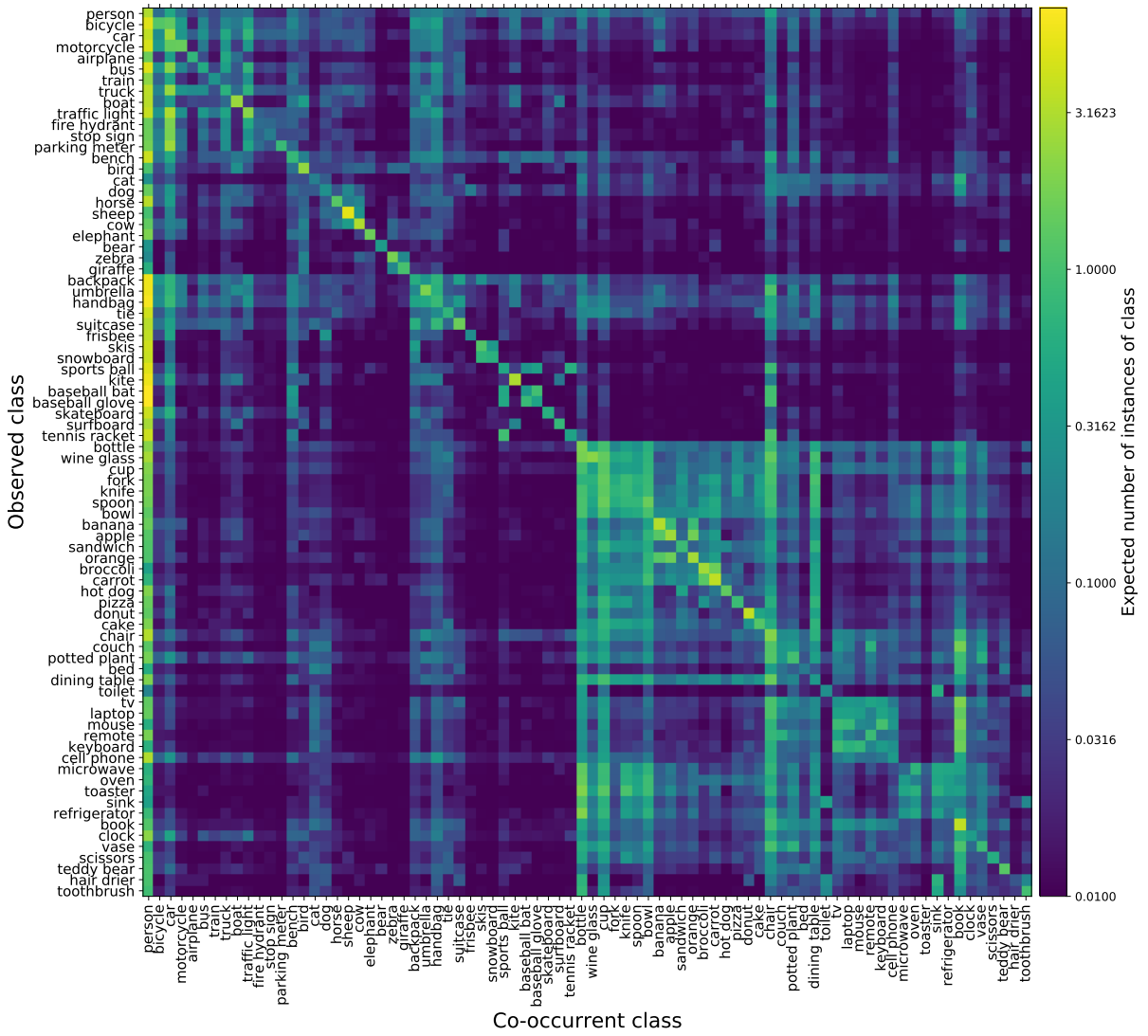


Figure 4.7: Co-occurrence matrix. Brighter spots indicate common co-occurrences, while darker spots indicate rare co-occurrences. Natural correlations stand out: cutlery, vehicles, tennis racket and sports ball, etc. Also negative correlations: sports objects do not co-occur with kitchen objects, outdoor objects, and food. The diagonal indicates the co-occurrence with objects from the same class - it is possible to identify which objects tend to appear in flocks (birds, sheep, bananas, books, etc) and which tend to have a single instance (fire hydrant, toilet, refrigerator). From the light columns, we can identify the most common classes (person) and darker columns indicate the rarest (hair drier and toaster).

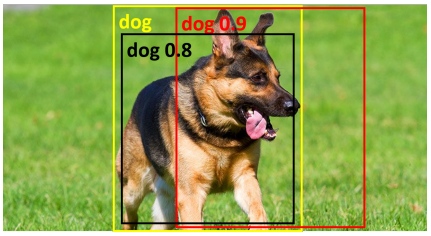


Figure 4.8: The highest confidence detection (red) has the poorest localisation (ground-truth in yellow). Once a detection has been made for the object, the others are false positives. For maximising AP, the detection with the highest IoU should be kept and duplicates should be driven to zero.

4.2 Optimal Rescoring

In this section, we explore the problem of optimal rescoring. The problem is formulated as: given a set of detections and a set of annotations (ground-truths), we ought to find the confidence score to assign to each detection in order to maximise Average Precision, i.e. finding the combination(s) of confidences that yield the best possible improvement to AP.

Before discussing how to obtain the optimal rescoring, we recall the computation of Average Precision and how it is affected by changes in detection confidence.

Then we divide the rescoring problem into two steps: matching each detection with the corresponding ground-truth (unmatched detections are false positives) and determining the optimal target value for their confidence. We propose three methods for each of the two steps and compare the results.

4.2.1 How Average Precision is affected by changes in ordering

As introduced in Section 2.1.1, Average Precision is a complex metric to optimise. Summarily, AP is computed in the following way: starting from the highest confidence detections, the true positives are determined by matching each detection with the ground-truth from the same category and highest overlap (with several different IoU thresholds); then, the precision-recall curve is traced and interpolated; AP is given by the average of the interpolated precision at 101 recall levels.

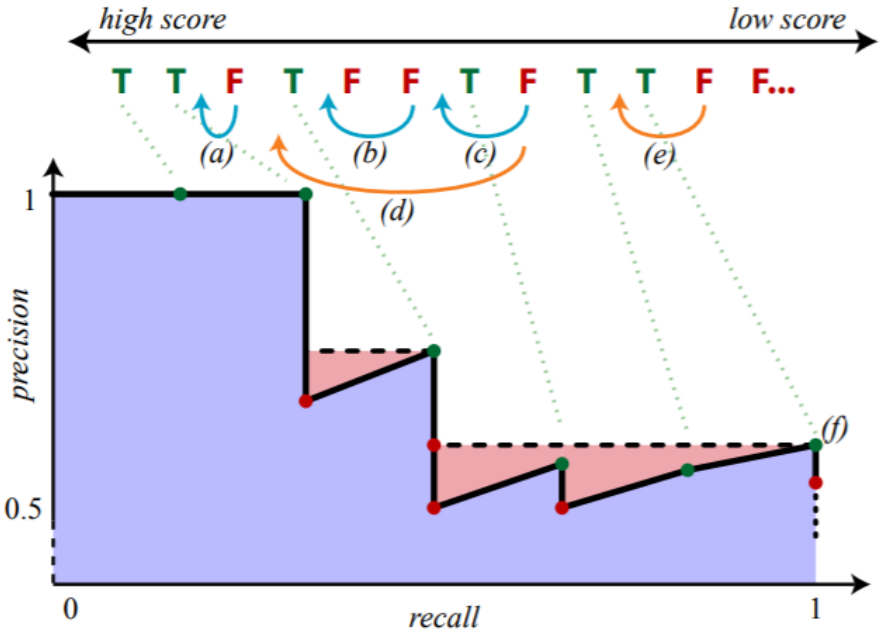


Figure 4.9: Precision-recall curve for an object category with six ground-truth instances. The pink area illustrates the result of the interpolated precision-recall curve. AP measures the total area of the pink and blue regions. **Blue arrows** (a-c) show perturbations with no effect on AP: (a) the order of detections does not change; (b) a FP swaps places with another FP; (c) a FP swaps places with a TP, but a higher-recall TP (f) has higher precision so there is no change to area under the interpolated curve. **Orange arrows** (d-e) show perturbations have an effect on AP: (d) the FP is moved beyond a TP that appears on the interpolated curve; (e) the FP moves past a single TP, lowering the precision level at recall $= 1$ (extracted from [44]).

As discussed in Figure 4.9, AP is a function of the ordering of the bounding box confidences and not all changes in this ordering have an impact on AP. We can draw the following conclusions:

- changing the absolute value of the confidences with a linear scaling does not change AP - what influences AP is the ordering induced by the confidences (with the exception of detections with confidence < 0.05 that are discarded in the AP evaluation for the COCO dataset);
- removing the lowest confidence detections by a threshold degrades AP - although most of the removed detections will be false positives, removing a lower confidence true positive that sits on the interpolated curve will degrade AP;
- the optimal rescoring rule reorders detections to make all the true positives have higher confidence than the false positives - the non-trivial challenge is finding which detections should be kept (true positives) and which should be removed (false positives);
- detections with higher IoU are preferred over detections with higher confidence - AP rewards localisation by computing AP at different IoU thresholds; detections with higher IoU will survive the longest at various IoU thresholds.

4.2.2 Matching Bounding Boxes to Annotations

When training a model to rescore detections, we compute the error with respect to the optimal rescoring for the given sequence. The computation of the optimal rescoring is divided into two steps: matching the detection to the corresponding ground-truth and then re-assigning its confidence to the optimal value.

Matching a detection with a ground-truth determines if it is a true positive or a false positive. This problem is non-trivial because several detections can correspond to the same ground-truth instance (Figure 4.8). We have to determine which detection to prioritise when making the matches. The matching algorithm used for evaluation prioritises detections with higher confidence. As discussed before, this is not the optimal solution if we are trying to maximise AP in the cases where the highest confidence detections are not the detections with the highest overlap.

In order to understand whether to prioritise confidence versus overlap, we propose three algorithms for matching detections to ground-truths. Algorithms 2 and 4 prioritise detection confidence and Algorithm 3 prioritises overlap:

- **Greedy matching by detection confidence** (Algorithm 2): **go through all detections** starting from highest confidence (line 3) and match each detection with the unmatched ground-truth from the same class with the **highest overlap** (line 13) above the IoU threshold. This is similar to how the detections are matched during evaluation.
- **Greedy matching by ground-truth overlap** (Algorithm 3): **go through all ground-truths** (line 3) and match each ground-truth with the unmatched detection from the same class with the **highest overlap** (line 13) above IoU threshold.
- **Greedy matching by ground-truth and confidence** (Algorithm 4): **go through all ground-truths** (line 3) and match each ground-truth with the unmatched detection from the same class with the **highest confidence** (line 13) and with overlap above the IoU threshold.

In all cases, the algorithm starts with a high IoU threshold and gradually reduces it (line 2), matching first the best localised detections. All the unmatched detections are marked as false positives.

Algorithm 2 Greedy matching by detection confidence

```
1: procedure GREEDYCONFIDENCE(dets, gts)
2:     for threshold ∈ 0.95 : 0.5 : 0.05 do        ▷ Start at higher thresholds and gradually reduce
3:         for det ∈ dets do                        ▷ ordered by descending confidence
4:             if covered(det) then
5:                 continue
6:             end if
7:             for gt ∈ gts do
8:                 if same_category(det, gt) and not assigned(gt) and IoU(det, gt) ≥ threshold then
9:                     add gt to set
10:                end if
11:            end for
12:            if not empty(set) then
13:                matched_gt ← arg max_{gt∈set} IoU(det, gt)
14:                det ← True Positive
15:                assigned(matched_gt) ← True
16:                covered(det) ← True
17:            end if
18:        end for
19:    end for
20: end procedure
```

Algorithm 3 Greedy matching by ground-truth overlap

```
1: procedure GREEDYGROUNDTRUTHOVERLAP(dets, gts)
2:     for threshold ∈ 0.95 : 0.5 : 0.05 do        ▷ Start at higher thresholds and gradually reduce
3:         for gt ∈ gts do
4:             if covered(gt) then
5:                 continue
6:             end if
7:             for det ∈ dets do
8:                 if same_category(det, gt) and not assigned(det) and IoU(det, gt) ≥ threshold then
9:                     add det to set
10:                end if
11:            end for
12:            if not empty(set) then
13:                matched_det ← arg max_{det∈set} IoU(det, gt)
14:                matched_det ← True Positive
15:                assigned(matched_det) ← True
16:                covered(gt) ← True
17:            end if
18:        end for
19:    end for
20:    All unmatched in dets are False Positives
21: end procedure
```

Algorithm 4 Greedy matching by ground-truth and confidence

1: **procedure** GREEDYGROUNDTRUTH+CONFIDENCE(dets, gts)
2: **for** $threshold \in 0.95 : 0.5 : 0.05$ **do** ▷ Start at higher thresholds and gradually reduce
3: **for** $gt \in gts$ **do**
4: **if** covered(gt) **then**
5: continue
6: **end if**
7: **for** $det \in dets$ **do**
8: **if** same_category(det, gt) **and not** assigned(det) **and** IoU(det, gt) $\geq threshold$ **then**
9: **add** det to set
10: **end if**
11: **end for**
12: **if not** empty(set) **then**
13: $matched\_det \leftarrow \arg\max_{det \in set}$ confidence(det)
14: $matched\_det \leftarrow$ True Positive
15: assigned($matched\_det$) $\leftarrow$ True
16: covered(gt) $\leftarrow$ True
17: **end if**
18: **end for**
19: **end for**
20: All unmatched in $dets$ are False Positives
21: **end procedure**

4.2.3 Target Confidence

After matching detections to ground-truths, the following step is to assign a new confidence to the matched detections (true positives). We propose three targets:

- **Binary target:** simply choose to keep true positives and discard false positives - matched detections get a target of 1, unmatched detections get a target confidence of 0.
- **Confidence target:** keep original confidence for true positives and discard false positives - matched detections keep the original confidence, unmatched detections get a target confidence of 0.
- **IoU target:** reorder true positives so that bounding boxes with better localisation have higher confidence - matched detections get a target equal to IoU with the matched ground-truth bounding box ($\in [0.5, 1]$), unmatched detections get a target confidence of 0.

4.2.4 Comparison of Rescoring Algorithms

To compare the different algorithms and target metrics we apply the algorithm to the detections of the Cascade R-CNN model on the COCO `val2017` dataset. Given the set of detections and the ground-truth annotations, we run the different rescoring rules and aggregate the results in Table 4.1. The table shows the best possible improvement that can be achieved by rescoring the set of detections from the detector, i.e., no detections are created or removed, their confidences are simply changed.

The differences between the matching algorithms are marginal: although they prioritise different metrics, using a gradually decreasing IoU threshold (line 2) already prioritises detections by their location accuracy and the algorithms produce similar matches.

The rescoring targets, however, show clear differences. The *Binary* target uses the same value for

Table 4.1: Average Precision for different rescoring algorithms and targets

Matching algorithm	Rescoring target		
	Binary	Confidence	IoU
Detection confidence	48.553	51.546	55.816
Ground-truth overlap	48.548	51.595	**55.819**
Ground-truth + confidence	48.554	51.547	55.817
Baseline		42.11	

all true positives, while the *Confidence* target keeps the same confidence for all ground-truths. Both these targets do not differentiate detections with good localisation from the ones with bad localisation. The *IoU* target reorders the detections into the optimal target ordering by assigning higher confidence to the detections with the highest IoU and thus prioritising detections with better localisation.

Chapter 5

Proposed Approach

5.1 Problem Formulation

We incorporate context in object detection by rescoring a set of detections from a previous object detector (also referred to as candidate detections). The model's input is a sequence of candidate detections $\boldsymbol{x}$ and the output is a sequence of rescored detections $\boldsymbol{y}$. Each element of the output sequence is a function of the whole input sequence,

$$y_i = f\left(\boldsymbol{x}; \boldsymbol{\theta}\right), \tag{5.1}$$

where f is the model predictive function and $\boldsymbol{\theta}$ are the model parameters.

The goal of the model is to predict the sequence of confidences to assign to the input sequence that maximises Average Precision. The discussion of finding the optimal rescoring target for a set of detections is done in Section 4.2. The chosen matching algorithm is "Ground-truth overlap" (Algorithm 3) with the "IoU" target. Thus, the optimal target output is given by

$$\boldsymbol{y}_i^* = \begin{cases} \text{IoU}\left(y_i, g_i\right), & \text{if matched}(y_i) \\ 0, & \text{otherwise,} \end{cases} \tag{5.2}$$

where matched(y_i) returns true if the matching algorithm finds a match for candidate y_i and g_i is the ground-truth object associated to the match.

The problem is formulated as a regression to the optimal rescoring target. We use the mean squared error as the loss function:

$$L(\boldsymbol{y}, \boldsymbol{y}^*) = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - y_i^*\right)^2, \tag{5.3}$$

where $\boldsymbol{y}$ are the rescored confidences, $\boldsymbol{y}^*$ is the target sequence, and N is the sequence length.

5.2 Feature Extraction

Candidate detections are pre-processed to extract a vector of high-level features for each candidate. These features determine the model's sources of context. Therefore, our model is trained to incorporate semantic context (the presence of other objects in the image) and geometric context (object size and location).

The extracted feature vector for candidate i is given by

$$\boldsymbol{x}_i = [score_i] \oplus [\text{one\_hot}\,(class_i)] \oplus \left[\frac{x_i}{W}, \frac{y_i}{H}, \frac{w_i}{W}, \frac{h_i}{H}\right], \tag{5.4}$$

where $\oplus$ denotes vector concatenation, $x_i,\ y_i$ are the coordinates of the top left corner of the candidate bounding box, $w_i,\ h_i$ are its width and height, and $W,\ H$ are the width and height of the image. Features $score_i$ and $class_i$ are the confidence score and object class from the candidate detection, resulting from the output of the detector we are building on top of. The function one_hot creates a one-hot encoded vector for the object class, a vector of zeros of length equal to the number of classes with a one on the index of the corresponding object class. An example of the feature extraction is illustrated in Figure 5.1. For the proposed model in the COCO dataset, the feature vector has a length of 85.

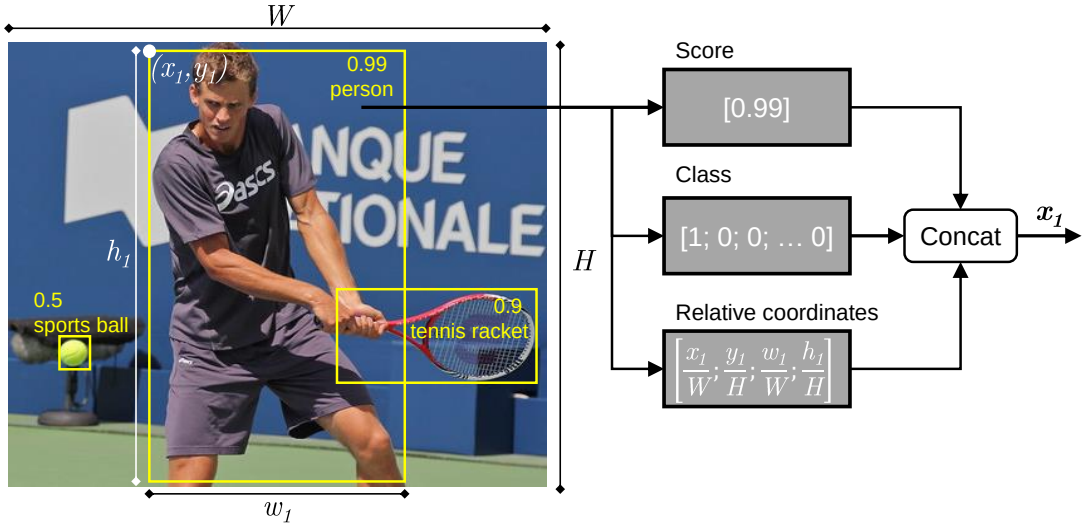


Figure 5.1: The input of our model is a set of candidate detections.

Sequences are generated by extracting the feature vector for every detection, then grouped by image, and ordered by decreasing confidence. The maximum length of a sequence is 100 given that the detectors have a limit of 100 objects per image. Images containing fewer than 100 detections are padded to have length of 100.

Notice that we opt not to use the visual features in this approach. The experiments we performed (see Appendix A) show that directly including visual features into the feature vector degrades performance with the proposed model.

5.3 Model Architecture

Our proposed model consists of a bidirectional gated recurrent unit (GRU) with a self-attention mechanism followed by a regression layer as shown in Figure 5.2.

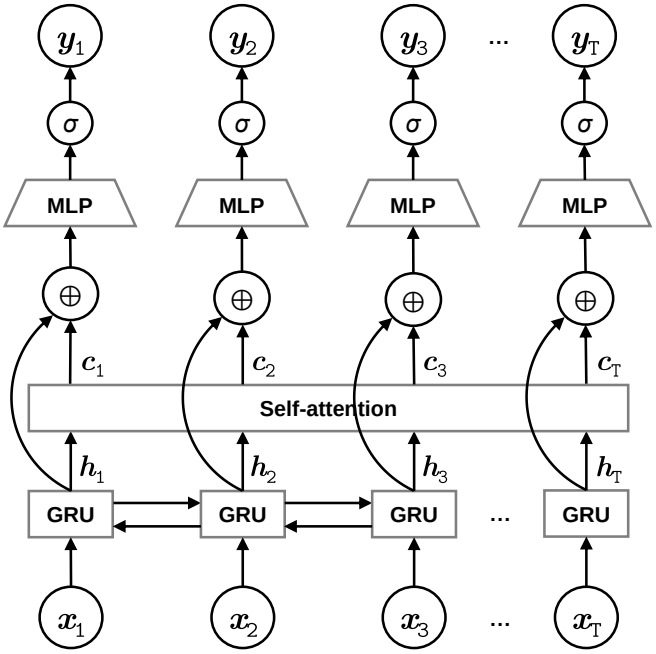


Figure 5.2: Model architecture. The input sequence $\boldsymbol{x}$ is sent to a recurrent neural network (GRU) with a self-attention mechanism. The symbol $\oplus$ denotes the vector concatenation of hidden state $\boldsymbol{h}$ with the context vector $\boldsymbol{c}$. From this concatenated feature vector, a multilayer perceptron (MLP) followed by a sigmoid activation (σ) predicts the rescored confidence $\boldsymbol{y}$.

RNN The model uses a bidirectional GRU as the RNN variant to process sequential data. The model computes two hidden states $\overrightarrow{\boldsymbol{h}_t}$ and $\overleftarrow{\boldsymbol{h}_t}$ of size n_h, corresponding to the forward and backward sequences, that are concatenated to produce the state vector $\boldsymbol{h}_t$ of size $2n_h$. We also stack n_L GRU layers to produce a deeper model. The use of a bidirectional model allows the internal representation of state to be a function of both previous and future objects in the sequence.

Self-Attention The self-attention mechanism creates a context vector that summarises important elements in the sequence. The computation of the context vector described in Section 3.2.1. The proposed implementation uses the scaled dot-product attention as the alignment scoring function:

$$\text{score}(\boldsymbol{h}_i, \boldsymbol{h}_j) = \frac{\boldsymbol{h}_i^\top \boldsymbol{h}_j}{\sqrt{L}}. \tag{5.5}$$

MLP The last component of our model is a multi-layer perceptron (MLP) that predicts a value for the rescored confidence from the hidden state and context vector. This module consists of a linear layer of size $4n_h \times 80$ with ReLU activation and a linear layer of size 80×1 with sigmoid activation. An hyperparameter search shows that the choice of architecture has little influence on Average Precision.

5.4 Implementation Details

The model was implemented and trained in Python using the PyTorch framework [45].

Pre-processing An object detection model is run on the whole COCO dataset (`train2017` + `val2017`) to generate candidate detections that are stored in memory in JSON format (300 MB + 15 MB). Features are extracted from the candidate detections, aggregated into the input and target sequences as PyTorch tensors, and stored in memory (1000 MB + 50 MB). Given that images contain at most 100 detections, shorter sequences are padded with zeros to have a length of 100.

Ordering When the model is trained with input sequences ordered by descending confidence, it struggles to predict a rescored sequence that changes the ordering of the confidences. The model is biased to predict the scores in the same decreasing order, making no changes to AP. We force the model to predict sequences that permute the ordering of the input sequence by shuffling the input sequences during training. We introduce a parameter `shuffle_rate` that is the probability of returning a random permutation of (i.e. shuffling) the input batch.

Optimiser We use the Adam optimiser [46] with batch size of 256 and learning rate of 0.003.

Learning Rate Decay A learning rate decay mechanism was implemented to maximise AP and avoid overfitting. When AP plateaus for more than `patience` number of epochs on `val2017`, the learning rate is multiplied by a factor of 0.2 and the model parameters are reverted to the parameters of the epoch that achieved the best AP so far. Reverting the parameters to the best epoch prevents the model from overfitting too early and reducing the learning rate allows the model to do a better fine-tuning of the parameters around the best epoch. Our implementation uses an initial learning rate of 0.003 and $\texttt{patience} = 4$ epochs (reduce learning rate if AP does not improve for 4 epochs) with 1 epoch cooldown (the first epoch after a learning rate reduction does not count towards `patience`).

Early Stopping Training is stopped if AP does not improve for 20 consecutive epochs on `val2017`.

Chapter 6

Experimental Results

In this chapter, we report the main results of our experiments. We start by evaluating the performance, in terms of the Average Precision (AP), focusing on demonstrating the improvements obtained by using the proposed method. Then, we illustrate the context learnt by the model by presenting examples that show significant changes in confidence. Finally, in the last section, we present ablation studies of the proposed model, covering the relevance of the features, the detection ordering, the choice of architecture, and model hyperparameters.

6.1 Performance Comparison

6.1.1 Results with Different Baselines

To evaluate the robustness of our method, we performed experiments on the detections from two baselines: Faster R-CNN and Cascade R-CNN implemented by OpenMMLab's MMDetection [43]. The models were tested with two different backbone networks, ResNet-50 and ResNet-101 [42], pre-trained on ImageNet [12] from PyTorch's model zoo and adapted for object detection by MMDetection's training.

Table 6.1: Performance results with multiple architectures and backbones on COCO `val2017` and `test-dev2017` datasets, before and after rescoring. AP^{50} and AP^{75} correspond to AP at IoU $= 0.5$ and IoU $= 0.75$, respectively. AP^S, AP^M and AP^L refer to small (area $< 32^2$), medium ($32^2 <$ area $< 96^2$) and large objects (area $> 96^2$).

Base model (backbone)	rescored	`val2017` (5k)						`test-dev2017` (20k)					
		AP	AP^{50}	AP^{75}	AP^S	AP^M	AP^L	AP	AP^{50}	AP^{75}	AP^S	AP^M	AP^L
Faster R-CNN	✗	36.41	58.41	39.07	21.56	40.08	46.60	36.7	58.8	39.6	21.6	39.8	44.9
(ResNet-50)	✓	**37.41**	59.96	40.07	21.84	40.67	48.73	**37.4**	60.2	40.3	21.8	40.4	46.1
Faster R-CNN	✗	39.37	60.67	43.01	22.10	43.59	52.01	39.7	61.4	43.2	22.1	43.1	50.2
(ResNet-101)	✓	**39.91**	61.56	43.47	22.36	43.84	53.02	**40.1**	62.2	43.5	22.1	43.4	50.8
Cascade R-CNN	✗	41.13	59.27	44.81	22.59	44.52	54.84	41.5	60.0	45.2	23.3	44.0	53.1
(ResNet-50)	✓	**41.76**	60.25	45.34	23.05	45.14	55.97	**42.0**	60.7	45.5	23.5	44.7	54.2
Cascade R-CNN	✗	42.11	60.31	45.93	23.16	45.96	56.29	42.4	61.2	46.2	23.7	45.5	54.1
(ResNet-101)	✓	**42.81**	61.51	46.45	23.86	46.67	57.52	**42.9**	62.1	46.6	23.9	46.1	55.3

Table 6.1 shows the comparison of the detector results before and after rescoring. Rescored detections always perform better, with consistent improvements of $0.4 \sim 1$ in AP for different baselines and backbones. Results at different object scales also show that the model produces larger improvements for larger objects. Poorly localised detections also have a larger AP improvement ($AP^{50} > AP^{75}$). As shown in Table 6.2, the rescored detections still have a large room for improvement when compared to the target AP.

Table 6.2: Average Precision on `val2017` before and after rescoring, and the target AP.

model	Baseline AP	Rescored AP	Target AP
Faster R-50	36.41	37.41	51.67
Faster R-101	39.37	39.91	53.35
Cascade R-50	41.13	41.76	54.88
Cascade R-101	42.11	42.81	55.82

6.1.2 Confidence Distribution

In a similar analysis to the one in Section 4.1.2, we explore the types of errors after rescoring. The previous analysis counts the number of occurrences of each error type. The same analysis is not suitable for a rescoring approach because rescoring does not change the number of true positives or false positives, they are simply assigned different confidences.

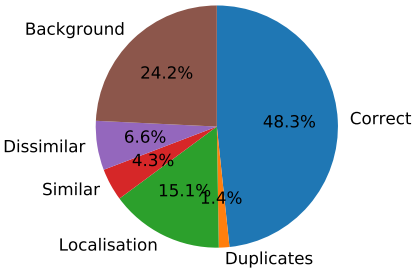
We compare the total accumulated confidence for each error type, before and after rescoring, in the pie charts of Figure 6.1, where each slice represents the fraction of the total accumulated confidence split by error type/correct classification. The fraction of confidence for type t is given by

$$p_t = \frac{\sum_{j \in \mathcal{D}_t} c_j}{\sum_{i \in \mathcal{D}} c_i}, \tag{6.1}$$

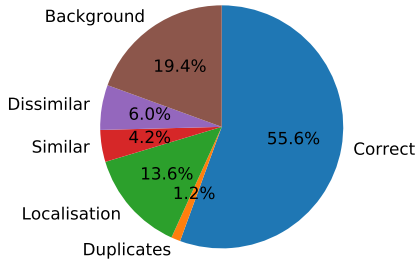
where c_i is the confidence for detection i, $\mathcal{D}$ is the full set of detections, and $\mathcal{D}_t$ is the set of detections with error of type t.

We observe a substantial increase in the relative amount of confidence assigned to correct classifications, and a decrease in all types of errors, that is consistent with other baselines. Confusions with background, localisation errors are the types of errors that show a higher decrease in confidence, also consistent with other baselines.

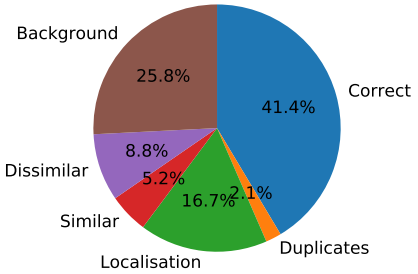
Notice that these charts compare only the relative value of the summed confidences, normalised by the total confidence, before and after rescoring, and not their absolute value. In fact, the absolute value of the accumulated confidences decreases. This is caused by the negative bias in the training data (there are more negative samples than positives). Although the metric of interest does not directly measure this distribution of confidences, this analysis gives us an interesting insight into the types of errors that are mostly addressed by the rescoring rule.



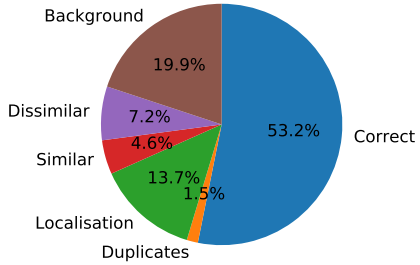
(a) Cascade R-CNN before rescoring



(b) Cascade R-CNN after rescoring



(c) Faster R-CNN before rescoring



(d) Faster R-CNN after rescoring

Figure 6.1: Distribution of confidence before and after rescoring on `val2017`. The slices represent the fraction of the total accumulated confidence assigned to each error type. Correct detections have the highest increase in relative confidence, while all types of errors have decreased relative confidence. The main types of errors addressed are confusions with background and localisation errors.

6.1.3 Class-wise AP Comparison

The COCO dataset provides metrics for the class-wise Average Precision. Table 6.3 shows the object categories that have the highest AP change after rescoring. The AP difference for class c was computed by averaging the AP differences (ΔAP) for four different baselines (Cascade R-CNN and Faster with ResNet-101 and ResNet-50 backbones):

$$\Delta\text{AP}_c = \frac{\Delta\text{AP}_c^{\text{Cascade101}} + \Delta\text{AP}_c^{\text{Cascade50}} + \Delta\text{AP}_c^{\text{Faster101}} + \Delta\text{AP}_c^{\text{Faster50}}}{4}. \tag{6.2}$$

Most classes show a significant and consistent AP increase. Although the overall change in AP is positive, there are classes that show an AP decrease. There is not a clear pattern as to what types of classes benefit the most with this rescoring.

Table 6.3: Object classes with the highest changes in AP.

top positives		top negatives	
class	ΔAP	class	ΔAP
couch	+ 1.9	clock	- 0.5
toaster	+ 1.7	elephant	- 0.5
frisbee	+ 1.6	giraffe	- 0.2
cake	+ 1.4	wine glass	- 0.2
pizza	+ 1.3	person	- 0.2
donut	+ 1.2	bicycle	- 0.2
sandwich	+ 1.1	sink	- 0.2
orange	+ 1.1	zebra	- 0.1
toilet	+ 1.1	traffic light	- 0.1
bed	+ 1.1	tennis racket	- 0.1

6.1.4 Does the Model Generalise for Different Baselines?

Different architectures have different detection profiles, their distribution on detections is different (e.g., Cascade R-CNN produces more background detections). A model that has learned context from one detector should be able to generalise the learned knowledge and improve performance with different detectors. Table 6.4 compares the AP increase obtained by using a model trained on one detector and evaluated on a different detector. Although AP improvements are not as large when tested with different baselines, all models are still able to generalise well and provide significant and consistent improvements.

Table 6.4: AP increase for models trained with different detectors. Lines refer to the detector where the model was trained and columns refer to the detector where the model was evaluated.

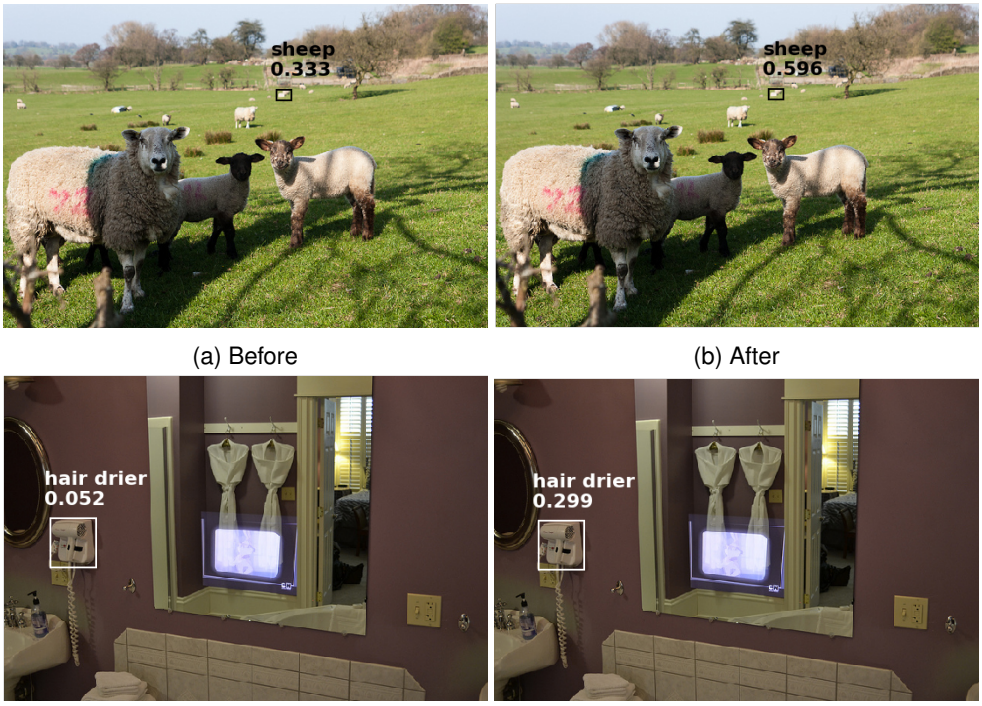
Trained on (`train2017`)	Evaluated on (`val2017`)			
Detector (ResNet)	Faster (R-50)	Faster (R-101)	Cascade (R-50)	Cascade (R-101)
Faster (R-50)	**+ 1.00**	**+ 0.60**	+ 0.56	+ 0.54
Faster (R-101)	+ 0.82	+ 0.54	+ 0.51	+ 0.48
Cascade (R-50)	+ 0.51	+ 0.13	**+ 0.63**	+ 0.61
Cascade (R-101)	+ 0.50	+ 0.26	+ 0.53	**+ 0.70**

6.2 Visualising Rescoring

In this section, we present the samples that lead to the highest differences in confidence after rescoring. The examples were obtained from COCO `val2017` with a model trained on the detections of Cascade R-CNN with ResNet-101 backbone that reported the highest AP after rescoring of 42.81. It is worth noticing that the model is biased towards negative changes in confidence, given that there are more negative training samples. This bias is not a problem because we are not necessarily interested in the absolute value of the confidence, but on their ordering.

6.2.1 Increasing Confidence

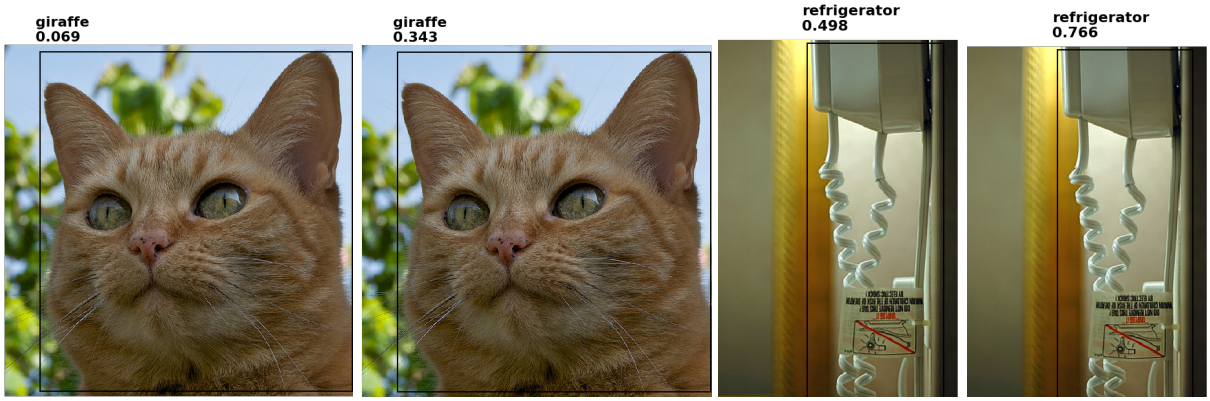
Figure 6.2 shows examples of objects that increase their confidence after rescoring. This increase in confidence occurs mostly for objects that appear in flocks (e.g., 'sheep' or 'cow') and for rare objects (e.g., 'hair drier'). In images that contain fewer objects, i.e., fewer sources of context, the model does not have enough information to make an informed decision and false positives have increased confidence (Figure 6.3). This is a limitation of our approach – the lack of contextual information in images with fewer objects degrades performance.



(a) Before (b) After

(c) Before (d) After

Figure 6.2: Confidence increase - the model learned the co-occurrence of 'sheep', even in the background. 'Hair drier' is a rare class, other objects such as 'sink' and 'bottle' help the detector induce the image context and increase confidence.



(a) Before (b) After (c) Before (d) After

Figure 6.3: Mistakes made by the model - images with few detections have no context. On (a) and (b), the detector produces two overlapping detections, 'cat' and 'giraffe', and increased the confidence for 'giraffe'. On (c) and (d), the detector produces only one detection, 'refrigerator' (ground-truth is 'hair drier'), our model has no other source of context and wrongfully increases confidence.

6.2.2 Decreasing Confidence

Figure 6.4 shows the capability of the model to identify out of context false positives and reduce their confidence. This effect is more evident when the image contains several objects from the same class and an object from an unrelated class is found.

(a) Before

(b) After

(c) Before

(d) After

Figure 6.4: Confidence decrease - the model reduces confidence for objects out of context. 'Frisbee' in a picture of 'motorcycles', and 'baseball bat' in an image with lots of 'bananas'. For better visualisation, we display only the relevant detection and omit all others.

Non-Maximum Suppression Figure 6.5 shows an example where the model appears to have learned to apply non-maximum suppression, one detection is kept while others have their confidence substantially decreased.

(a) Before

(b) After

Figure 6.5: The model suppresses duplicate detections. 'Skateboard' is detected multiple times, but the model is able to maintain the best located detection and decrease the confidence for duplicates.

6.3 Ablation studies

Which models are better at capturing context?

Models capture context in different ways: the GRU encodes it the hidden state and self-attention creates a context vector for each element. But how much does each model contribute to AP improvements?

Table 6.5 answers the question by comparing the improvements obtained by the use of a bidirectional model and by the addition of a self-attention mechanism. The base model uses a unidirectional RNN with $n_L = 3$ stacked layers and a hidden state of size $n_h = 256$ trained with `shuffle_rate` $= 0.75$. We compare the performance improvement of a bidirectional model and the addition of a self-attention mechanism both with a GRU and a LSTM.

We also compare a model that does not use RNNs. This model is built by replacing the RNN layers by a linear layer (Linear(85,128) + ReLU) followed by the self-attention mechanism (using "general" attention [41]) and the regressor (Linear(256,128) + ReLU + Linear(128,80) + ReLU + Linear(80,1) + Sigmoid).

Table 6.5: Ablation study of model components comparison.

	RNN	attention	bidirectional	# parameters	`val2017` AP
baseline					42.11
Linear + self-attention		✓		0.1 M	42.64
LSTM	LSTM			1.4 M	42.58
Bi-LSTM	LSTM		✓	3.9 M	42.77
LSTM + self-attention	LSTM	✓		1.5 M	42.65
Bi-LSTM + self-attention	LSTM	✓	✓	4.0 M	42.75
GRU	GRU			1.1 M	42.58
Bi-GRU	GRU		✓	2.9 M	42.78
GRU + self-attention	GRU	✓		1.2 M	42.73
Bi-GRU + self-attention	GRU	✓	✓	3.0 M	**42.81**

The choice of LSTM or GRU seems to have little impact on performance. The argument for the choice of GRU in the final model is the higher performance achieved with a smaller model. The limitation of a unidirectional model is that the model is causal: each prediction is a function of previous input, while bidirectional and attention models relax this constraint. The results using a linear layer + self-attention demonstrate the attention mechanism's ability to capture context without the use of a RNN.

How important is each feature?

Table 6.6 compares the importance of the features used to train the model. The results were obtained with a leave-one-out methodology, where each result is obtained by training the same model with all other features but the one under analysis. We observe that the most important feature is the original confidence while the least important are the bounding box coordinates. Not using the original confidence

degrades AP by 2.25 while a model trained using solely the original confidences is able to slightly improve performance.

Table 6.6: Feature importance: leave-one-out comparison.

	confidence	category	coordinates	`val2017` AP
baseline				42.11
all features	✓	✓	✓	42.81
no coordinates	✓	✓		42.44
no category	✓		✓	42.28
no confidence		✓	✓	39.88
just confidence	✓			42.16

Detection ordering

If the candidate detections are sorted by descending confidence during training, this bias is learned by the model that ends up predicting rescorings that maintain the input's ordering. To force the model to produce a reordering in the output predictions we introduce the `shuffle_rate` parameter as described in Section 5.4. Table 6.7 compares the impact of this hyperparameter in the model's performance. A `shuffle_rate` $= 0$ means that detections are always sorted by confidence (easy first) and a `shuffle_rate` $= 1$ means that detections are always in random order.

To compare this ordering with a structured order, we introduce the location-based method that sorts candidates based on their location in the image. This method tries to maintain bounding boxes that are close in the image also close in the input sequence. The intuition behind this sorting is that the information from near bounding boxes is more important and make use of the GRU short-term memory. Detections are sorted as follows: first, divide the image in a 3×3 grid, start the sequence with the candidates whose centre is inside the middle cell, then add the remaining cells clockwise. Inside each cell, the detections are sorted by their confidence.

Table 6.7: Effect of the bounding box ordering on AP results, obtained on `val2017`.

`shuffle_rate`	0 (easy-first)	0.15	0.30	0.45	0.60	**0.75**	0.90	1 (random)
`val2017` AP	42.56	42.53	42.64	42.74	42.78	**42.81**	42.68	42.67
location-based ordering				42.34				
baseline AP				42.11				

The proposed location-based ordering is not as effective as a randomly sorted sequence. Choosing a high `shuffle_rate` helps the model learn to reorder detections. Having the order completely randomised (`shuffle_rate`) degrades performance in comparison to `shuffle_rate` $= 0.75$. This is related to the fact that at evaluation, the detections are sorted by confidence. A `shuffle_rate` $= 0.75$ achieved the best trade-off between sorting randomly and sorting by confidence.

Hyperparameter tuning

Table 6.8 shows the impact of the GRU hyperparameters, the number of GRU layers n_L and the size of the hidden units n_h. The base model uses a bidirectional GRU with self-attention and the `shuffle_rate` $= 0.75$. The increasing number of layers and the size of hidden units increases performance with diminishing returns and also makes the model more prone to overfitting. A choice of $n_h = 256$ and $n_L = 3$ achieved the best trade-off.

Table 6.8: AP results on `val2017`: comparison on the size of the hidden units n_h and number of stacked GRU layers n_L.

	n_h					
	16	32	64	128	256	512
$n_L = 1$						
# parameters	0.01 M	0.04 M	0.09 M	0.27 M	0.87 M	3.06 M
`val2017` AP	42.34	42.44	42.59	42.63	42.67	42.69
$n_L = 2$						
# parameters	0.02 M	0.06 M	0.17 M	0.57 M	2.06 M	7.78 M
`val2017` AP	42.52	42.55	42.70	42.71	42.72	42.65
$n_L = 3$						
# parameters	0.03 M	0.08 M	0.25 M	0.87 M	3.24 M	12.51 M
`val2017` AP	42.51	42.63	42.71	42.64	**42.81**	42.70
baseline	42.11					

Self-Attention Layer

Table 6.9 compares the choice of self-attention layer. Some of these approaches use additional parameters to learn context, but, interestingly, the scaled dot product, which has no additional trainable parameters, is the type of attention that achieved the best results. The learning of the scoring function is made by the GRU parameters.

Table 6.9: Attention type comparison

Attention type	`val2017` AP
Dot product [41]	42.72
Scaled dot product [21]	**42.81**
General [41]	42.63
Additive [20]	42.74

Chapter 7

Conclusions

In this chapter, we summarise the contributions and achievements of the thesis and suggest research directions for future work that builds on our approach and addresses some of its limitations.

7.1 Achievements

In this thesis, we discuss the incorporation of context in object detection. We started by reviewing the state-of-the-art literature in object detection and previous work that has explored the use of context. State-of-the-art methods have improved performance through the last years but still rely on an approach that evaluates regions of interest independently from each other. Recent research has shown that the incorporation of context can substantially improve performance of object detectors. Our work contributes to this line of work by exploring the use of recurrent neural networks with attention, which are models that include implicit and explicit mechanisms that capture context in sequential data.

We performed a comprehensive error analysis on a state-of-the-art detector and identified the most common errors and the sources of context that can help mitigate these errors. Confusion with background, localisation errors, and confusion with dissimilar objects are the most prevalent errors. To address these issues, we identified the sources of context to be used: the geometric context from the object size and location and the semantic context from object co-occurrences.

We studied the metric of interest, Average Precision, and how to optimise it through a rescoring approach. The metric is sensitive only to permutations in the ordering of confidence scores. We proposed an optimal rescoring algorithm that, given a set of detections and ground-truths, computes the new confidence scores that maximise AP by assigning higher confidences to detections that have higher overlap with their corresponding ground-truth.

We proposed a model to rescore sequences of detections made by a previous detector. The proposed model consists of a bidirectional GRU with self-attention followed by a classifier. We train this model on the COCO dataset to predict the optimal rescoring targets.

Finally, we validate the proposed model with experimental results. The experiments show that the model is able to consistently improve AP and effectively perform a better redistribution of confidence

between true and false positives. Results are consistent and generalisable across different baseline detectors. Our proposed model can effectively improve the performance by exploiting contextual information on object co-occurrences, the relative positioning between objects and the object's relative size. The most common success examples show that the model increases confidence for objects that typically appear in flocks, decrease confidence for false positives in unexpected scenarios and single out objects that have duplicate detections.

7.2 Future Work

In this thesis, we explore context as a rescoring problem. The rescoring approach is limited, still: although it improves precision, it does not help to detect objects that were not detected by the baseline detector in the first place (i.e., it has no effect on recall). To improve in this respect, it is necessary to include context reasoning inside the detection architecture.

The main limitation of the proposed approach is the lack of visual information - most of the model's mistakes could be avoided with a visual understanding of the image. Future work should focus on how to incorporate visual information along with contextual information to (re)score detections. A suggested approach is to unify the detector and the contextual rescoring model into a single architecture, trained end-to-end, where the predictions are generated from visual features (using the visual feature map extracted by the CNN) and from contextual features.

Experiments with attention-only models have shown promising results. These models achieve comparable results to RNNs while using a fraction of the parameters. Because our research mainly focused on RNNs, few efforts have been devoted to investigating more complex attention models. Motivated by recent work in natural language processing, sustaining that "Attention is All You Need" [21, 47], where RNN architectures were replaced by transformer models that use attention-only architectures instead of recurrent networks, we suggest that these models be explored in the scope of context rescoring.

Bibliography

[1] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. URL http://arxiv.org/abs/1311.2524.

[2] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. URL http://arxiv.org/abs/1504.08083.

[3] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

[4] Z. Cai and N. Vasconcelos. Cascade R-CNN: delving into high quality object detection. *CoRR*, abs/1712.00726, 2017. URL http://arxiv.org/abs/1712.00726.

[5] R. Mottaghi, S. Fidler, J. Yao, R. Urtasun, and D. Parikh. Analyzing semantic segmentation using hybrid human-machine crfs. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3143–3150, 2013.

[6] A. Torralba and P. Sinha. Contextual priming for object detection. *International Journal of Computer Vision*, 53, 07 2003. doi: 10.1023/A:1023052124951.

[7] S. Agarwal, J. O. du Terrail, and F. Jurie. Recent advances in object detection in the age of deep convolutional neural networks. *CoRR*, abs/1809.03193, 2018. URL http://arxiv.org/abs/1809.03193.

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004. ISSN 1573-1405. doi: 10.1023/B:VISI.0000029664.99615.94. URL https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[9] 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)*, volume 1, pages 886–893 vol. 1, June 2005. doi: 10.1109/CVPR.2005.177.

[10] C. Lampert, M. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 01 2008.

[11] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012. doi: 10.1145/3065386.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[13] 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*, 32(9):1627–1645, Sep. 2010. doi: 10.1109/TPAMI.2009.167.

[14] Z. Chen, S. Huang, and D. Tao. Context refinement for object detection. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[15] M. J. Choi, J. J. Lim, A. Torralba, and A. S. Willsky. Exploiting hierarchical context on a large database of object categories. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 129–136, June 2010. doi: 10.1109/CVPR.2010.5540221.

[16] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert. An empirical study of context in object detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1271–1278, June 2009. doi: 10.1109/CVPR.2009.5206532.

[17] R. Mottaghi, X. Chen, X. Liu, N. Cho, S. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 891–898, June 2014. doi: 10.1109/CVPR.2014.119.

[18] S. Bell, C. L. Zitnick, K. Bala, and R. B. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. *CoRR*, abs/1512.04143, 2015. URL `http://arxiv.org/abs/1512.04143`.

[19] Y. Liu, R. Wang, S. Shan, and X. Chen. Structure inference net: Object detection using scene-level context and instance-level relationships. *CoRR*, abs/1807.00119, 2018. URL `http://arxiv.org/abs/1807.00119`.

[20] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

[22] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015. URL `http://arxiv.org/abs/1502.03044`.

[23] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL http://arxiv.org/abs/1506.02640.

[24] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[25] 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*, 88(2):303–338, June 2010.

[26] I. Endres and D. Hoiem. Category independent object proposals. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, pages 575–588, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15555-0.

[27] J. R. Uijlings, K. E. Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *Int. J. Comput. Vision*, 104(2):154–171, Sept. 2013. ISSN 0920-5691. doi: 10.1007/s11263-013-0620-5. URL http://dx.doi.org/10.1007/s11263-013-0620-5.

[28] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, Nov 2012. doi: 10.1109/TPAMI.2012.28.

[29] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision – ECCV 2012*, pages 340–353, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33712-3.

[30] T. M. Strat. Employing contextual information in computer vision. In *In Proceedings of ARPA Image Understanding Workshop*, pages 217–229, 1993.

[31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, Oct. 1986. doi: 10.1038/323533a0.

[32] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[33] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681, 1997.

[34] S. E. Hihi and Y. Bengio. Hierarchical recurrent neural networks for long-term dependencies. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 493–499. MIT Press, 1996. URL http://papers.nips.cc/paper/1102-hierarchical-recurrent-neural-networks-for-long-term-dependencies.pdf.

[35] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[36] A. Graves, A. Mohamed, and G. E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013. URL http://arxiv.org/abs/1303.5778.

[37] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL http://arxiv.org/abs/1409.3215.

[38] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014. URL http://arxiv.org/abs/1411.4555.

[39] C. Olah. Understanding LSTM networks, 2015. URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Consulted in September 2019.

[40] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078.

[41] M. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015. URL http://arxiv.org/abs/1508.04025.

[42] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[43] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. Mmdetection: Open mmlab detection toolbox and benchmark. *CoRR*, abs/1906.07155, 2019. URL http://arxiv.org/abs/1906.07155.

[44] P. Henderson and V. Ferrari. End-to-end training of object class detectors for mean average precision. *CoRR*, abs/1607.03476, 2016. URL http://arxiv.org/abs/1607.03476.

[45] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[46] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[47] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

Appendix A

Visual Features

Our proposed model does not exploit visual features to focus instead on categorical and location features. This approach assumes that the processing of visual information is handled by the base detector while our model exploits what can be called high-level context. In this appendix, we report the experiments performed that attempted to include visual features in the rescoring model.

Extracting Visual Features

During the pre-processing step, in addition to extracting the categorical and location features, we run each image patch through a pre-trained ResNet-152 [42] from PyTorch's `torchvision` package. This choice was motivated by this model's superior performance in ImageNet. We modify this backbone network by removing the last fully-connected classification layer and use it as a visual feature extractor, pre-trained for ImageNet classification.

Image patches are normalised to have mean `mean = [0.485, 0.456, 0.406]` and `std = [0.229, 0.224, 0.225]` in order to match the training distribution, and reshaped to a size of `(224, 224)` to fit input size requirements.

The network extracts a 2048-dimensional feature vector for each image patch that is stored in disk along with the categorical and location features. These additional features largely increase the dataset size. The dataset that previously used approximately 1.1 GB of memory now uses over 12.8 GB of memory.

Incorporating Visual Features in the Proposed Model

We explored three approaches to the incorporation of visual features in the proposed model:

(a) Increase GRU input size (to a size of 2048+85) to include the visual features and the original vector (confidence, category, and coordinates).

(b) Add a MLP to process visual features and concatenate the MLP processed visual features with the

other inputs before sending them to the GRU. The MLP is composed of a followed by Linear(2048, 256) + ReLU, Linear(256, 80) + ReLU.

(c) Add a MLP, parallel to the GRU to process the vector of visual features. The MLP is composed of the following layers: Linear(2048, 512) + ReLU, Linear(512, 256) + ReLU, Linear(256, 80) + ReLU. The extracted vector of length 80 is concatenated with the resulting vector from the GRU + MLP, also of length 80, and processed by a Linear(160, 1) + sigmoid layer.

Table A.1: Comparison of rescoring approaches with visual features. 'Ours' refers to the proposed model without visual features, n_L refers to the GRU's number of layers. (a), (b) and (c) refer to the previously described experiments. Approach (a) with $n_L = 3$ is omitted due to numerical instability during training.

model	`val2017` AP
baseline	42.11
$n_L = 2$	
Ours	42.72
(a)	40.01
(b)	42.64
(c)	**42.76**
$n_L = 3$	
Ours	**42.81**
(b)	42.64
(c)	42.66

Table A.1 compares the approaches on Average Precision and the number of parameters. The experiments use a base model that uses a bidirectional GRU with $n_h = 256$ and `shuffle_rate` $= 0.75$. A simple approach such as (a) is not suitable for a GRU model. The input features are treated as contextual information, shared with other detections. The GRU cannot hold categorical information when it is fed with so many visual features.

The results from approaches (b) and (c) show that the use of visual features yields little or no improvements to the proposed approach. If a small improvement is obtained, it does not justify the additional computational costs from the approach.

Conclusions

This approach to the inclusion of visual features is sub-optimal. Features were extracted from isolated image patches using a network pre-trained for image classification. The computational costs from extracting each of these feature vectors do not justify the possible improvements reported by our experiments.

The errors made by this rescoring approach suggest that the inclusion of visual features can produce better results. A more appropriate approach is to extract visual features from the baseline detector's convolutional feature map (that are fine-tuned for object detection) by applying RoI pooling. We suggest this approach as a future research direction.

Appendix B

Rescored Samples

In this appendix, we display some samples from `val2017`. The metric used for selection was the cosine distance between the candidate confidences $\mathbf{c}$ and the rescored confidences $\mathbf{c}'$, computed by

$$1 - \frac{\mathbf{c} \cdot \mathbf{c}'}{\|\mathbf{c}\|_2 \|\mathbf{c}'\|_2}. \tag{B.1}$$

The selected set of samples displays the ones with the highest cosine distance and containing less than 16 detections (for better readability), sorted by descending cosine distance. Each sample contains three images: candidate detections, rescored detections, and annotations. We recommend viewing the examples digitally (with zoom).

Figure B.1

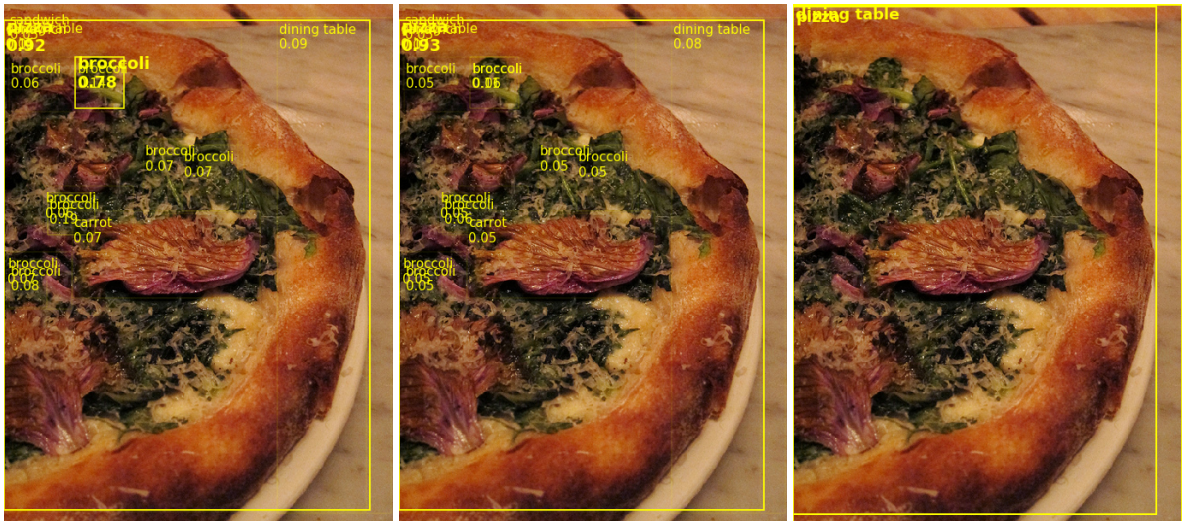


Figure B.2

Figure B.3

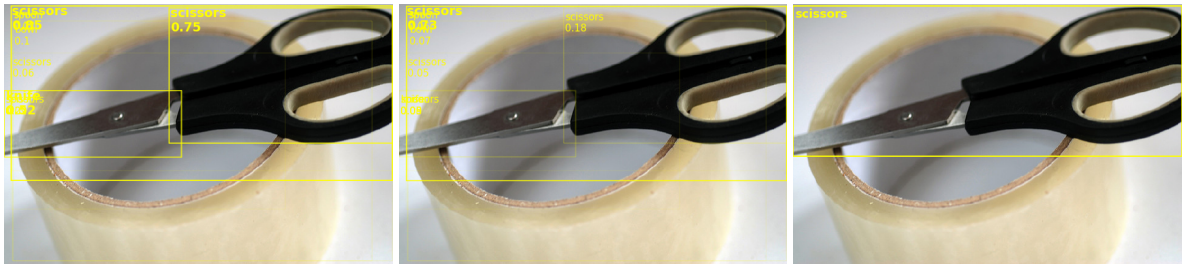


Figure B.4

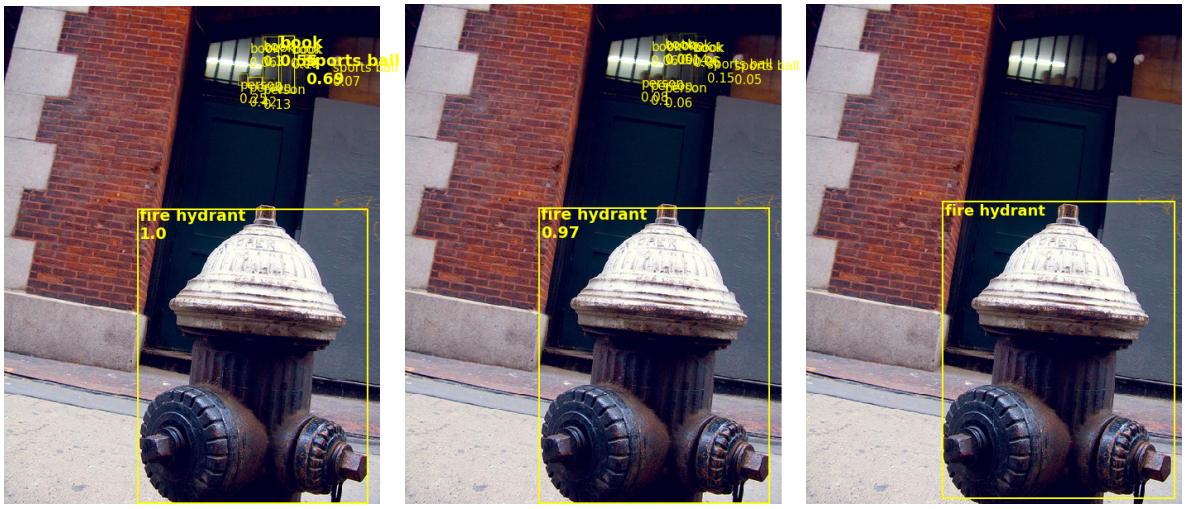


Figure B.5

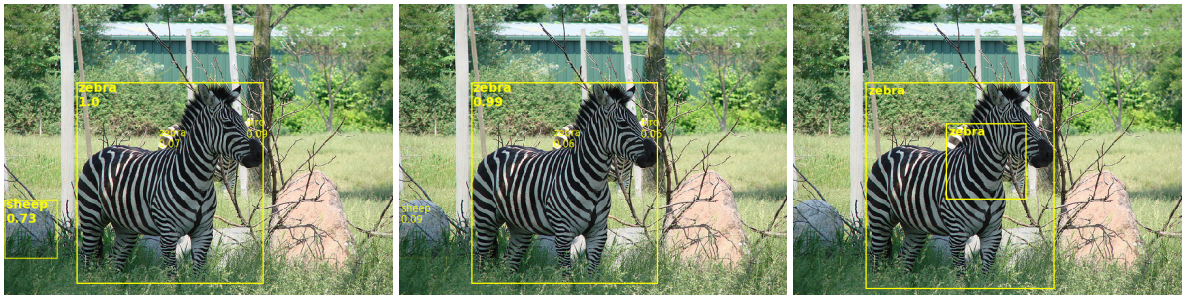


Figure B.6

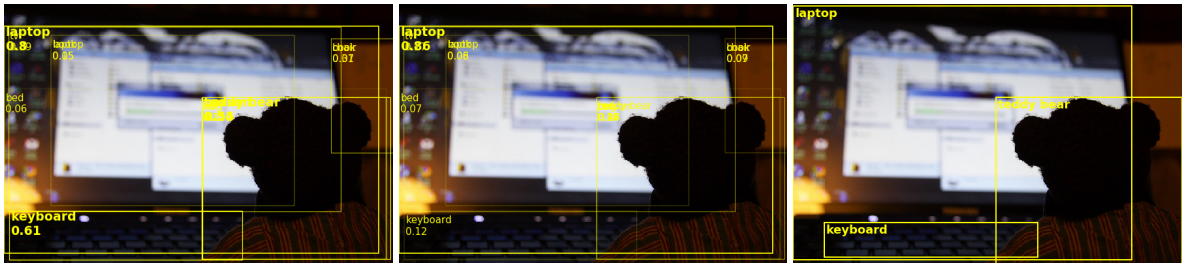


Figure B.7

Figure B.8

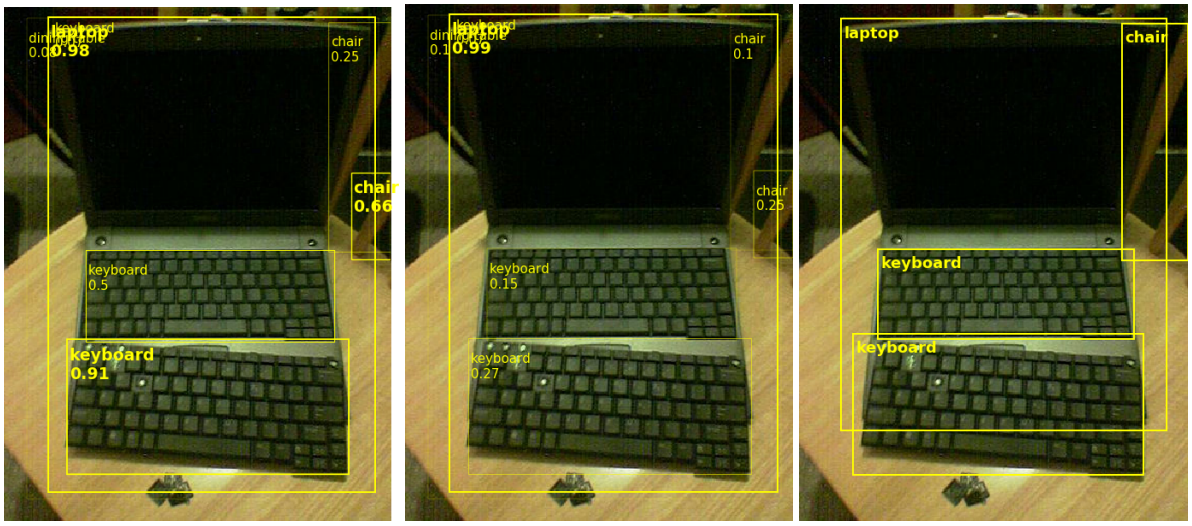


Figure B.9

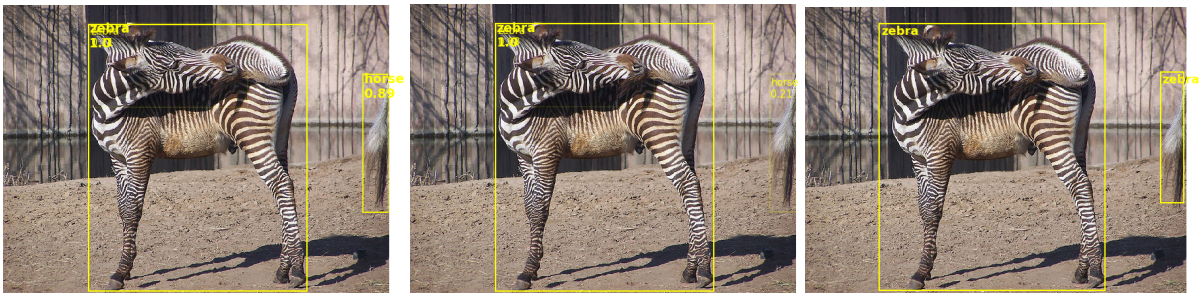


Figure B.10

Figure B.11

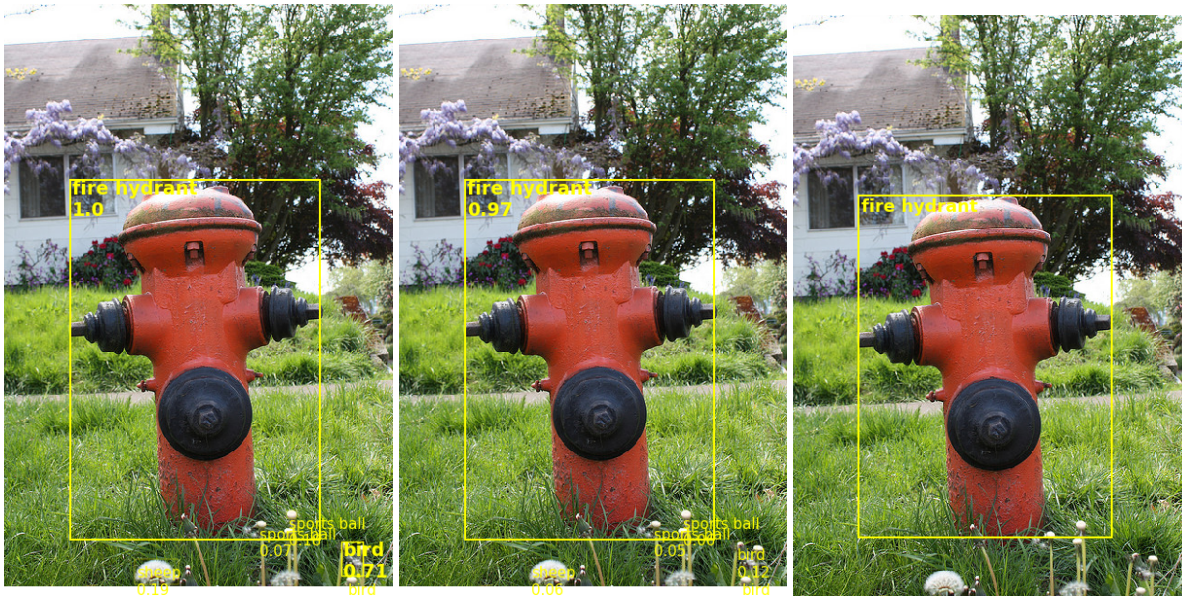


Figure B.12

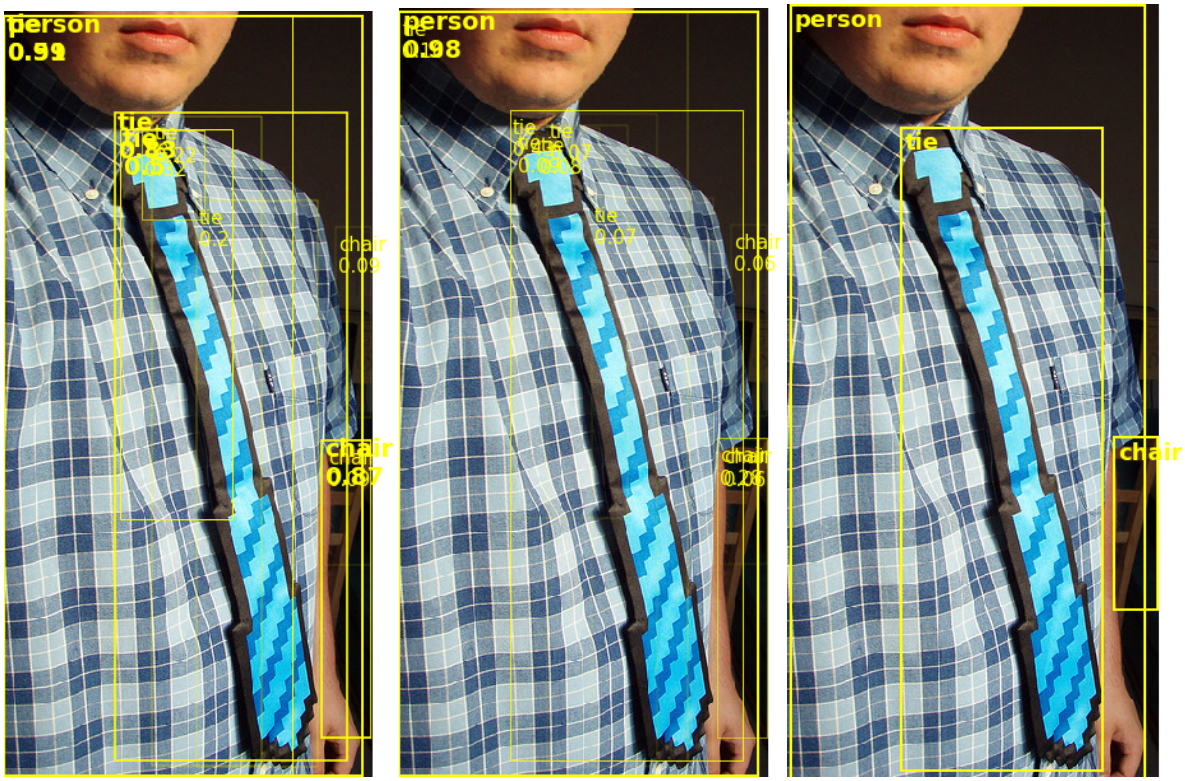


Figure B.13

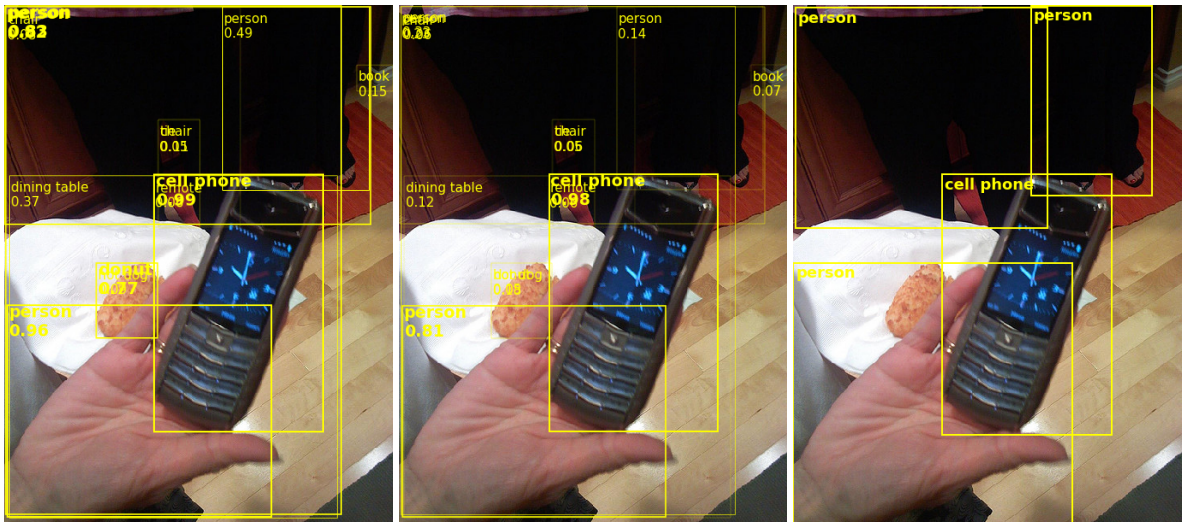


Figure B.14

Figure B.15

Figure B.16

Figure B.17

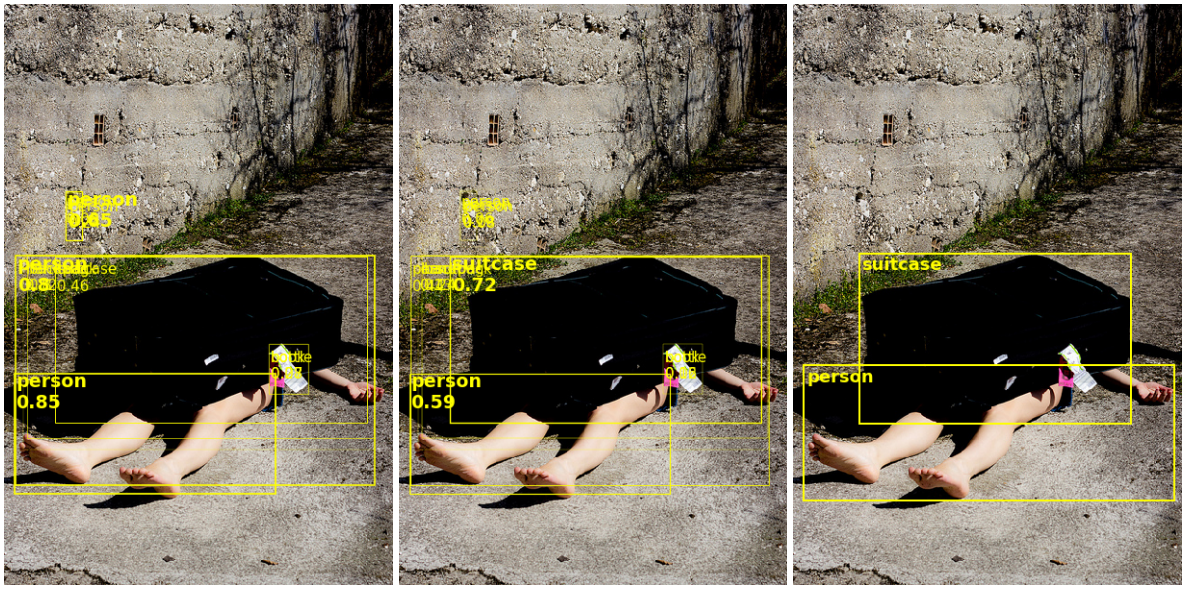


Figure B.18

Figure B.19